НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи і мережі”

спеціальності 123 “Комп’ютерна інженерія ”

на тему: Транзакційний бот для спотової торгівлі на криптобіржі

Виконав : студент __4__ курсу, групи _ІО-83_
(шифр групи)

_____Бурлака Данило Сергійович_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____асистент Калюжний О.О._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) професор, д.т.н., Сімоненко В. П._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи і мережі”

спеціальності 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Бурлаки Данила Сергійовича

1. Тема проєкту *Транзакційний бот для спотової торгівлі на криптобіржі*
керівник проєкту Калюжний О.О., асистент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 11 травня 2022 року №1139-с

2. Термін здачі студентом закінченого проєкту *11 червня 2022 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Аналіз предметної області.

Розділ 2. Вибір стратегії арбітражу та інструментів.

Розділ 3. Розробка системи.

Розділ 4. Результати роботи бота.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, функціональна схема (діаграма класів), алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П.		

7. Дата видачі завдання «30» серпня 2021 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2021-15.12.2021*	
2.	*Вивчення та аналіз завдання*	*15.12.2021-15.03.2022*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2022-25.03.2022*	
4.	*Розробка структур окремих підсистем*	*25.03.2022-5.04.2022*	
5.	*Програмна реалізація системи*	*5.04.2022-15.04.2022*	
6.	*Оформлення пояснювальної записки*	*15.04.2022-20.05.2022*	
7.	*Захист програмного продукту*	*25.04.2022*	
8.	*Передзахист*	*12.06.2022*	
9.	*Захист*	*20.06.2022*	

Студент-дипломник _____ Данило БУРЛАКА
(підпис)

Керівник проєкту _____ Олександр КАЛЮЖНИЙ
(підпис)

АНОТАЦІЯ

У цій роботі був створений транзакційний бот для спотової торгівлі на криптобіржі Binance. Були розглянуті наявні рішення автоматизації арбітражу на різних ринках, перспективи розвитку криптовалютного ринку та використання ботів для більш швидкої і прибуткової торгівлі. Аналізуючи ринок та працюючи з волантивними монета, розроблений бот може приносити прибуток згідно з введеними стратегіями. Розроблена программа довела свою ефективність незалежно від нещодавнього падіння цін криптовалютних активів. Програмний продукт був розроблений на мові Python.

Ключові слова: транзакційний бот, криптобіржа Binance, автоматизація трейдингу, Python.

ANNOTATION

In this bachelor's degree project, a transaction bot for the spot trading on the Binance cryptocurrency exchange was created. Existing solutions for arbitration automation in different markets, prospects for cryptocurrency market development and the use of bots for more efficient and profitable trading were discussed. By analyzing the market and working with volatile coins, the developed bot can make a profit according to the introduced strategies. The developed program has proven its independence from the recent fall in cryptocurrency asset prices. The software product is developed in Python.

Keywords: transaction bot, Binance cryptocurrency exchange, trading automation, Python.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Транзакційний бот	3		
			для спотової торгівлі			
			на криптобіржі			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Транзакційний бот	55		
			для спотової торгівлі			
			на криптобіржі			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	Транзакційний бот	1		
			для спотової торгівлі			
			на криптобіржі			
			Структурна схема			
			системи			
	А4	***ІАЛЦ.4672008.005 Д2***	Транзакційний бот	1		
			для спотової торгівлі			
			на криптобіржі			
			Функціональна схема			
			(діаграма класів)			
	А4	***ІАЛЦ.467200.006 Д3***	Транзакційний бот	1		
			для спотової торгівлі			
			на криптобіржі			
			Алгоритм дій програмного			
			забезпечення			
	А4	***ІАЛЦ.467200.007 Д4***	Транзакційний бот	28		
			для спотової торгівлі			
			на криптобіржі			
			Текст програмного коду			

					ІАЛЦ.467200.001 ОА		
Зм	*Лист*	*№ докум.*	*Підп*	*Дата*			
Розроб		БурлакаД.С.			**Транзакційний бот для спотової торгівлі на криптобіржі** Опис альбому	Літ.	Аркуш 1 / Аркушів 1
Перев		КалюжнийО.О.					
						НТУУ "КПІ" ФІОТ ІО-83	

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Транзакційний бот для спотової торгівлі на криптобіржі*»

Київ – 2022

ЗМІСТ

НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ 2

ПІДСТАВИ ДЛЯ РОЗРОБКИ 2

МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ 2

ДЖЕРЕЛА РОЗРОБКИ 2

ТЕХНІЧНІ ВИМОГИ 2

Вимоги до розробленого продукту 2

Вимоги до програмного забезпечення 3

Вимоги до апаратної частини 3

ЕТАПИ РОЗРОБКИ 3

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку програми бота для автоматизованного арбітражу волатильних монет на криптовалютному ринку.

Область застосування: криптовалютні ринки.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка боту для автоматизації процесу торгівлі на криптобіржі волатильними монетами згідно розроблених стратегій.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Проста та гручка кастомізація конфігурації роботи бота.
- Автономна робота боту та ведення відповідного логу змін.
- Швидка реакція бота за зміни у ринку згідно розробленій стратегії.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- PyCharm IDE

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel® Core (TM) i3-2100T.
- ROM не менше ніж 64 ГБ.
- RAM не менше ніж 4 ГБ.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.12.2021-15.12.2021
Вивчення та аналіз завдання	15.12.2021-15.03.2022
Розробка архітектури та загальної структури системи	15.03.2022-25.03.2022
Розробка структур окремих частин системи	25.03.2022-5.04.2022
Програмна реалізація системи	5.04.2022-15.04.2022
Виправлення помилок	15.04.2022-20.05.2022
Оформлення пояснювальної записки	25.04.2022

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Транзакційний бот для спотової торгівлі на криптобіржі*»

Київ – 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 4

ВСТУП 5

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7

1.1 Огляд криптовалюти 7

1.2 Потенціал криптовалюти 9

1.3 Огляд валютного арбітражу 13

1.4 Види валютного арбітражу 14

1.5 Технічній аналіз фінансових ринків 15

1.6 Фундаментальний аналіз фінансових ринків 19

1.7 Поточні впровадження та проблеми арбітражу 21

ВИСНОВОК ДО РОЗДІЛУ 1 25

РОЗДІЛ 2. ВИБІР СТРАТЕГІЇ АРБІТРАЖУ ТА ІНСТРУМЕНТІВ 26

2.1 Визначення торгової стратегії 26

2.2 Розробка торгової стратегії 26

2.3 Технічний індикатор 27

2.4 У довгу чи у коротку 28

2.5 Частота торгівлі 29

2.6 Вибір криптобіржі 30

2.7 Мова програмування 32

2.8 Технології та бібліотеки 34

ВИСНОВОК ДО РОЗДІЛУ 2 35

РОЗДІЛ 3 ДЕТАЛІ РОЗРОБКИ СИСТЕМИ 36

3.1 Опис взаємодії компонентів 36
3.2 Файл налаштувань «Конфіг.yml» 36
3.3 Функція get_price() 36
3.4 Функція wait_for_price() 36
3.5 Функція pause_bot() 40
3.6 Функція buy() 41
3.7 Процедура update_portfolio() 42
3.8 Функція sell_coins() 43
ВИСНОВОК ДО РОЗДІЛУ 3 45
РОЗДІЛ 4 РЕЗУЛЬТАТИ РОБОТИ БОТА 46
4.1 Обрана конфігурація 46
4.2 Лог арбітражу 46
4.3 Лог торгівлі 48
ВИСНОВОК ДО РОЗДІЛУ 4 49
ВИСНОВКИ 50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 52
ДОДАТОК 1 3
ДОДАТОК 2 5
ДОДАТОК 3 7
ДОДАТОК 4 9

ПЕРЕЛІК СКОРОЧЕНЬ

SL	(Stop-loss) Стратегія «стоп-лосс»
TP	(Take-Profit) Стратегія «тейк-профіт»
MACD	(Moving average convergence divergence) Розбіжність ковзної середньої конвергенції
RSI	(Relative Strength Index) Індекс відносної сили
CBDC	(Central Bank Digital Currency) Цифрова валюта Центрального банку

ВСТУП

У сучасних економічних реаліях торгівля на криптовалютних ринках стає все більш непередбачуваною, а отже і ризикованою. Торгівля на фінансових маркетплейсах потребує від трейдера глибших знань, ніж раніше, а саме вміння використовувати велику кількість аналітичних інструментів, швидкість реагування на зміни та прийняття зважених рішень.

Управління соціально-економічними системами та прийняття в них рішень є центром сильної дисципліни і є однією з найбільш актуальних тем дослідження в сучасних науках про управління. Оскільки фінансові ринки є яскравим прикладом організаційних систем та існує високий рівень інтересу до вивчення їх функціонування та управління. Особливо враховуючи, що розміщення тимчасово вільних грошей на фінансових ринках стало дуже популярним в останні десятиліття. При правильному підході це приносить більше прибутку, ніж банківські вклади або зберігання грошей в іноземній валюті.

Інтуїтивно зрозумілий досвід торгівлі часто є найпоширенішою практикою для інвестування для широкої аудиторії. Це ще більш парадоксально, оскільки більшість гравців ринку добре знайомі з основами аналізу і теоретично можуть добре заробляти на біржовій грі. Однак статистика свідчить, що переважна більшість трейдерів програє.

Звичайно, прийняття інвестиційного рішення на фінансовому ринку на основі фундаментального та технічного аналізу є більш раціональним і вигіднішим, ніж інтуїтивні припущення. Однак, оскільки фінансовий ринок є динамічною системою, що швидко розвивається, закономірності та тенденції, які були визначені з часом методи аналізу та прогнозування з часом стають менш ефективними для застосування в реальній торгівлі.

Зростання темпів еволюції біржової торгівлі, безсумнівно, є поштовхом до розвитку все більш досконалих методів аналізу даних, які з'явилися на фінансових ринках і класифікуються як складні організаційні системи. Щоб проаналізувати багатовимірний набір даних, який становить поточний стан ринку, аналітичній команді поставлено завдання підтримати рішення про використання найбільш прибуткових, відомих, але водночас високонадійних та ефективних інструментів. з фінансовим інструментом.

Інтуїтивну торгівлю, засновану на власних припущеннях, замінюють потужні інструменти для системного аналізу стану ринкових систем на основі останніх розробок у сфері штучного інтелекту. Все більша кількість професійних трейдерів стає на бік технічного аналізу, який означає, що котирування одного фінансового інструменту не залежать від часових рядів інших. З появою нейромережевого апарату прихильники цих ідей отримали реальну можливість самоствердитися в своїх ідеях і застосувати нейромережі на практиці для виявлення внутрішніх невизначеностей розвитку динамічної системи – конкретного інструменту фінансового ринку.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд криптовалюти

Загалом, криптовалюти це цифрові форми платіжної системи, які не буде покладатися на банки і не вимагати часу перевірки транзакцій. Криптовалюти передбачають, що будь-хто матиме право отримувати та відправляти платежі в цифровому вигляді через функціональну однорангову систему. У зв'язку з цим криптовалюта припускає існуючі цифрові гроші, які функціонують як протилежність фізичним грошей. Цифрові транзакції в основному існують в онлайнових базах даних.

Під час здійснення цифрових транзакційних операцій будуть записані в публічні книги, а операції будуть зберігатися в цифрових гаманцях. Етимологія криптовалюті потрібна тому, що шифрування буде вбудоване у форму оплати під час перевірки та обробки транзакцій. Іншими словами, крипто-транзакції спираються на форми розширеного кодування під час передачі даних між гаманцями. У цьому відшенні шифрування в першу чергу служить для забезпечення безпечних і швидких транзакцій між користувачами. Історично біткойн був першою формою і типом криптовалюти з моменту її створення в 2009 році.

З моменту появи криптовалюти уряди не сильно звертали уваги на ці транзакції. Однак із все більшим зростанням цифровізації, роль криптовалюти також зросла і стала проблемою для уряду, оскільки фінансова система повинна створити правові нормативні рамки для регулювання транзакцій. А саме аспект фінансової прозорості став критичним для криптовалюти з урядової точки зору. Деякі користувачі почали посилатися на криптовалюту під час виконання непрозорих фінансових операцій, тому державні органи повинні були

відповісти або інтеграцією криптовалюти у свої фінансові системи, або встановленням додаткових механізмів перевірки.

На рисунку 1.1 зображена регуляція криптовалюних транзакцій у різних країнах світу. У блакитних країнах вона регулюється урядом, у червоних – повністю визнанна нелегальною. У помаранчевих існують часткові заборони, тобто коли банкам чи іншим фінансовим установам не дозволено займатися криптовалютами або пропонувати послуги людям або підприємствам, які використовують криптовалюту. Однак все більше і більше країн переглядають ці регуляції.

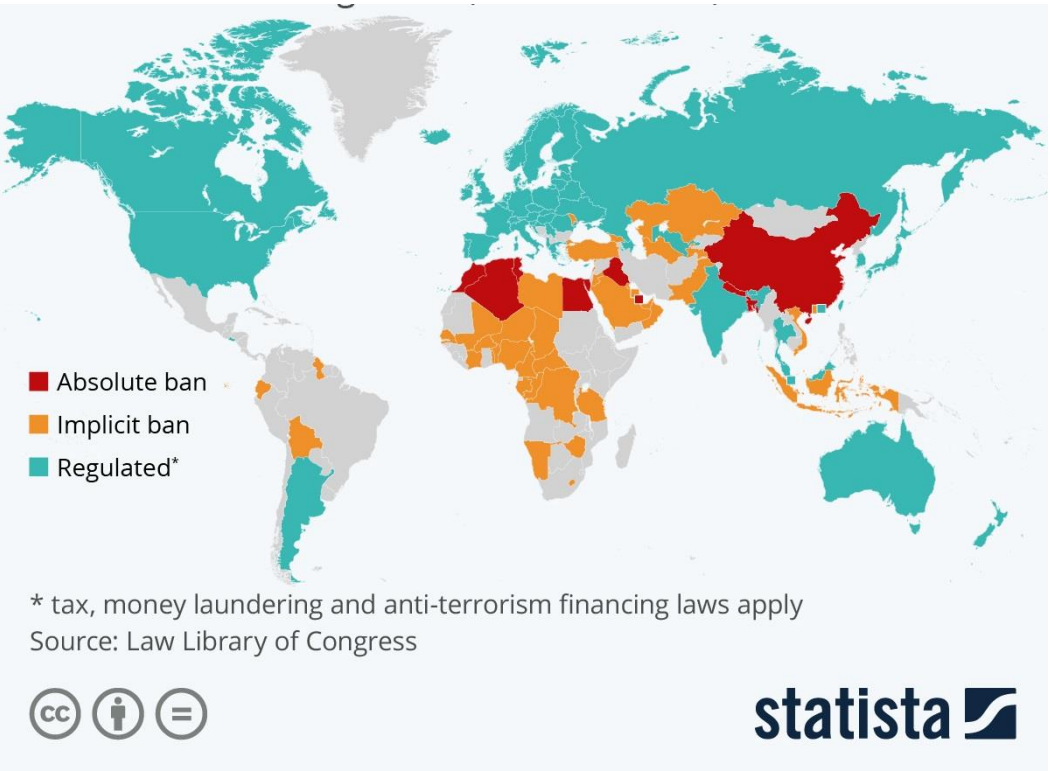


Рисунок 1.1. – Регуляція криптовалют у світі від Statista [1]

Особливе місце у розвитку криптовалюти мав процес майнінгу, який передбачав використання комп'ютерів під час розв'язування складних математичних транзакцій та генерування монет. Із зростаючою популярністю криптовалют уряди також почали активніше реєструвати та вимірювати ці транзакції заради прозорості.

Оскільки крипто-транзакції включають акції, облігації, а також можна торгувати з використанням різних форм технологій, очікування щодо їх зростання ролі на фінансових ринках є особливо високими.

1.2 Потенціал криптовалюти

За словами Марра [2], першою справжньою криптовалютою був біткойн, який з'явився на заголовках газет у 2009 році. Сьогодні біткойн є цілком встановленою і можна з впевненістю сказати основною кривовалютою, але така слава не прийшла одразу коли він вперше з'явився. Насправді знадобилося близько року, щоб ця нова валюта була використана й здобула якусь цінність, коли в 2010 році хтось вирішив придбати дві піци за 10 000 біткойнів [2]. Всього через рік після цього з'явилися дві нові криптовалюти, Litecoin та Namecoin, що ознаменувало початок «ери криптовалют», яка в кінцевому підсумку привела до криптовалютного буму, що призвело до припливу нових криптовалют протягом десятиліття (сьогодні існує понад тисячі криптовалют [2].

Протягом багатьох років у біткойна була своя частка злетів і падінь — він різко зріс у ціні в 2013 році, перш ніж зазнати краху в тому ж році, і те ж саме відбулося в 2017/2018 роках — але криптовалютна «екосистема» в цілому зростає. Таким чином, у 2021 році вони офіційно стали частиною мейнстріму [2]. Ці тренди виразно зображені на рисунку 1.2.

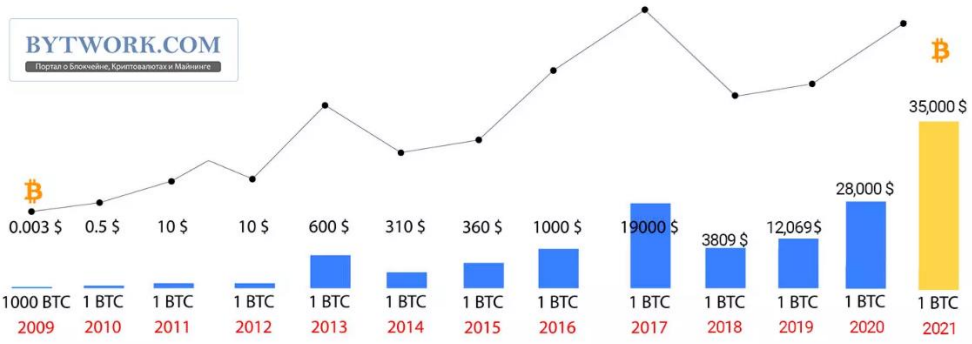


Рисунок 1.2. – Історія курсу біткоїна від Bytwork [3]

З появою криптовалюти люди почали говоряти про загибель готівки. За словами Майклза і Вігни [4], є велика вірогідність, що у майбутньому гроші та валюти будуть просто цифровою версією тієї ж самої фіатної валюти, яка є у нас зараз, і в такому сценарії фізичні готівкові гроші, ймовірно, будуть повністю виведені з виробництва.

Як зазначає Берсетче, згідно з опитуванням, проведеним Банком міжнародних розрахунків на початку 2021 року, із 65 опитаних центральних банків світу майже 86% з них роблять активні кроки до розуміння та, можливо, створення центральної цифрової валюти своїх банків, або шляхом простого дослідження концепції, перегляду доказів концепції та навіть експериментальних розробок [5]. Насправді автор стверджує, що близько 15% опитаних банків зробили активні кроки до дослідження для пілотів.

Далі пояснює, як під час інтерв'ю, даного CNBC П'єро Чиполлоне, заступника голови Банку Італії, італійський чиновник розповів про те, як він і його колеги у фінансовому світі бачать, що готівка стає все менш популярною серед торговців і споживачів, та як глобальна фінансова система поступово рухається до повної цифровізації [5]. Згідно з Берсетче, багато центральних банків світу починають створювати CBDC, які є цифровими валютами такими як криптовалюти, але замість децентралізаційного керування ними керує центральний банк.

Бенуа Кер, який колись працював у Європейському центральному банку, а зараз очолює BIS Innovation Hub, стверджує, що цифрові валюти центрального банку – це в основному цифрові банкноти [5]. У цьому сенсі існує просто версія банкнот двадцять першого 21-го століття.

Багато країн у всьому світі стають все більш безготівковими, наприклад, Китай і Швеція, які є лідерами у цьому процессі. Їх логіка у тому, що готівка практично вимирає, особливо у великих містах цих країн. Оскільки децентралізовані криптовалюти по суті є проблематичними для національних урядів для регулювання та контролю, створення централізованих цифрових

валют зробить роботу з валютою зручною та ефективною. У той же час це дозволить національним урядам зберегти свої повноваження в таких фінансових питаннях.

В епоху посилення централізації та глобалізації має сенс лише те, що національні уряди виберуть цей шлях. Підкріплює цю логіку і стрімкий ріст капіталізації криптовалютного ринку за останні декілька років (див. рисунок 1.3)

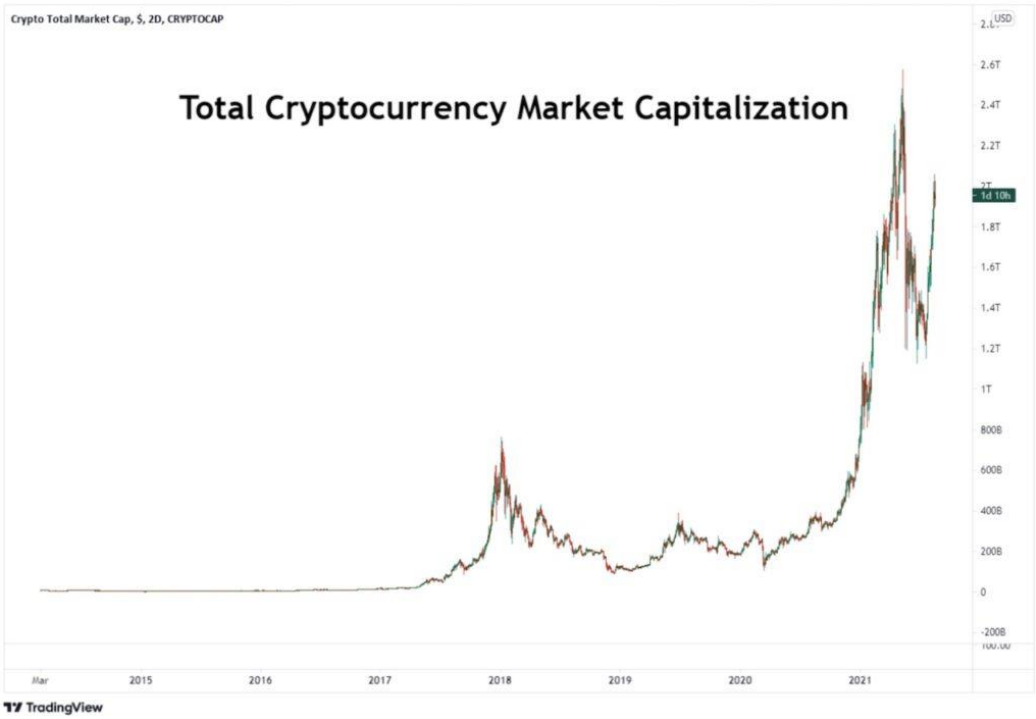


Рисунок 1.3. – Капіталізація криптовалютного ринку [6]

Китай уже почав готуватися до того, що здається неминучою еволюцією фінансів, оцифрувавши свою валюту - юань. За словами Дівакара, в той час як решта світу все ще знаходиться в процесі створення власних цифрових валют центрального банку, китайці вже пішли вперед і зробили це, оцифрувавши свій юань. Все це є частиною проекту уряду Китаю DCEP (Цифрові валюти/електронні платежі), щоб перенести економіку Китаю в майбутнє, водночас кидаючи виклик тому, що досі було беззаперечним домінуванням долара США [7].

Як повідомляє автор, китайський уряд/центр розпочав роботу над оцифруванням юаня 7 років тому (у 2014 році), а в квітні 2020 року перша пілотна програма DCEP була запущена в кількох китайських містах. Цього року програму було розширено на найбільші мегаполіси Китаю - Пекін і Шанхай, та нещодавно в Ченду було виділено близько 6 мільйонів доларів цифрових юанів для пробних цілей [7]. Швидше за все, Китай почне перехід на цифрові юані найближчим часом.

Важливо зазначити, що Берсетче також пояснює, що тим хто каже що цифрові валюти центрального банку не можуть забезпечити деякі інші переваги приватних криптовалют, а саме миттєві платежі та недорогі транзакційні витрати, треба звернути увагу на той факт, що цифрові валюти центрального банку надають ті самі переваги, а також швидші розрахунки [5].

Таким чином, маючи практично ті ж переваги що й децентралізовані валюти та будучи централізованими, цілком імовірно, що цифрові валюти центрального банку — це майбутнє фінансів, а не приватні криптовалюти. Оскільки винахід фіатної валюти йшов пліч-о-пліч зі створенням і централізацією надійного національного уряду (зрештою, фіатна валюта є інструментом національного уряду), а світ ставав все більш глобалізованим та централізованим, це лише підбивало до висновків що цифрові валюти центрального банку є наступним кроком в еволюції валюти.

З точки зору того, що чекає на готівку в майбутньому, тенденція до цього часу була низхідною і готівка стає все менш популярною у світі. Це особливо актуально для розвиненого світу, де поява цифрової валюти, як у формі цифрових валют центрального банку, так і у формі різних криптовалют, почала штовхати готівку все далі й далі вниз, коли справа доходить до фінансових транзакцій. Готівка, ймовірно, буде використовуватися протягом цього десятиліття, але, ймовірно, буде повністю припинена в більшості частин світу (наприклад, у Швеції та Китаї) найближчим часом. Враховуючи, що Китай уже розпочав тестування своєї цифрової валюти центрального банку (оцифрований

юань) — уже значною мірою відмовившись від готівки, особливо у великих мегаполісах — і з багатьма іншими країнами, які вже потенційно розробляють власні цифрові валюти центрального банку, схоже, що цифрова фіатна валюта – це майбутнє фінансів.

Таким чином, готівка навряд чи залишиться надовго в цьому світі, що постійно змінюється, який уже прийняв цифрові гроші та валюту.

1.3 Огляд валютного арбітражу

Валютний арбітраж передбачає купівлю та продаж валютних пар у різних брокерів, щоб скористатися перевагами їх різних спредів. Валютний спред визначається як різниця між цінами «пропозиція» та «запит» на актив. «Ставка» означає те, що покупець пропонує заплатити за конкретний актив, а «Запит» стосується ціни продажу активу, запропонованої продавцем. Розмір валютного спреду часто використовується для вимірювання ліквідності та ефективності ринку. Більший валютний спред означає, що ринок менш ефективний, а нижчий валютний спред означає більш ефективний ринок.

На ринку фіатної валюти банки часто використовують арбітраж, щоб скористатися перевагами розбіжностей цін між кількома валютними парами, щоб отримати прибуток. Боти, які торгують на звичайних традиційних ринках, такі як Bloomberg і NASDAQ, існують, але доступні виключно для інвестиційних будинків і брокерів. Ці звичайні ринкові боти потребують доступу для обміну даними з ринку. Ці дані обміну зазвичай недоступні для звичайних користувачів [8].

Прозора природа блокчейну надає трейдерам криптовалюту доступ до логу замовлень біржі та стимулює розробляє торгові боти, які діють на основі цих даних. Блокчейн є розподіленою книгою з усієї історії транзакцій, яку ведуть користувачі та майнери мережі. У результаті цього вже існує багато ботів, які

орієнтуються на криптовалютні біржі, такі як "Crypto Trader", "Haasbot", "Zenbot", "Gekko" та "BTCRobot". Багато з цих ботів мають чималу місячну абонентську плату. Більшість використовує стратегію міжбіржового арбітражу [9].

1.4 Види валютного абрітражу

У результаті великої кількості бірж криптовалют можна виконувати два типи арбітражів, щоб отримати вигоду з неефективності ринку. Перший — це «міжбіржовий» арбітраж, а другий — «внутрішньобіржовий» арбітраж. Перший відноситься до виконання арбітражних послідовностей на біржі, а другий означає виконання арбітражів на різних біржах. Кожен з них має свої переваги та проблеми, які будуть розглянуті нижче.

Першу техніку, міжбіржовий арбітраж, дослідив Норрі [9] у своїй статті про торгових ботів біткойн. Цей тип арбітражу досліджує ціни криптовалюти на багатьох різних біржах і виявляє розбіжності в цінах між ними. Коли буде виявлено розбіжність цін, криптовалюта з нижчою ціною буде куплена на одній біржі, потім переведена та продана на іншій. Здійснюючи такі арбітражі на багатьох різних біржах, ціна криптовалют має стабілізуватися до постійної ринкової вартості, а виконавець арбітражів може отримати прибуток, безпосередньо пов’язаний із знайденими відмінностями в цінах [8].

Другий тип арбітражної техніки, схожий на перший, також використовує переваги цінових спредів між двома біржами. У торговому боті «Blackbird», коли виявлено спред, який достатньо великий щоб покрити комісію за короткі продажі та покупку на двох відповідних біржах, «короткі» продажі виконуються на біржі де ціна вища, а «довгі» виконується на біржі, де ціна нижча. Коли обидві ціни в кінцевому підсумку зближуються, позиції розраховуються (продаються для довгої позиції і купуються в позиції короткого

продажу). Чистий прибуток в основному становить половину спреду за вирахуванням торгових зборів, помножених на обсяг торгівлі криптовалютою [10]. Ця техніка стверджує, що вона ринко-нейтральна, оскільки не наражає користувача на будь-які ризики, пов’язані з коливаннями ринку. Це також усуває необхідність передавати активи між різними біржами.

Третій метод був відкритий під час вивчення рішень набору домашніх завдань Масачусетського технологічного інституту Деменом і Голдвассером [11]. Ця третя техніка, «внутрішньобіржовий» арбітраж, по суті є в повсякденній нашій торгівлі, що розвивається і закінчується певною валютою, але в кінцевому підсумку ви отримуєте більше початкової валюти. Алгоритм Беллмана-Форда [12] може бути використаний для виявлення розбіжностей ціни на біржі, а також можливості арбітражу потім можуть бути виконані ботом. [11]

1.5 Технічний аналіз фінансових ринків

Технічний аналіз – це метод аналізу та прогнозування змін майбутніх цін та інструментів фінансового ринку на основі аналізу змін у минулому. Він заснований на аналізі часових рядів цін, відображених на графіках зміни цін. Технічний аналіз також використовує інформацію про кількість покупок / продажів конкретного фінансового інструменту за певний період часу, обсяг торгів та інші дані зі статистикою. Все більше і більше методів технічного аналізу використовуються на фондових біржах і фінансових ринках для аналізу цін, які змінюються з часом.

До появи технічного аналізу протягом кількох століть на різних фінансових ринках спостерігалися коливання цін. Наприкінці 19 століття американський журналіст Чарльз Доу написав серію статей про ринок цінних паперів, які згодом стали основою його теорій. Передбачається, що тенденція

результатів часового ряду буде наявною або значущою, поки не буде досягнуто її поворотного моменту. Теорія лише показує напрямок тренду і не передбачає тривалість тренду чи величину та поворотні моменти змін всередині неї.

Згідно з теорією Чарльза Доу, майже завжди більше акцій слідує певній тенденції внутрішнього ринку. Щоб оцінити цю «ринкову ситуацію», Dow створив два індекси, один з яких зараз називається Dow Jones Industrial Average, і вперше розрахував ціну у 12 акцій великих компаній на той час.

Другий індекс, який називається «блакитними фішками», індекс Dow Jones Railway Index, базується на цінах 12 акцій залізничних компаній. Ця теорія, заснована на співвідношенні продажів і цін, показує напрямок тенденції при різних рівні підтримки Промисловий індекс і індекс залізниці слід розглядати разом, тобто зміни в одному індексі повинні підтверджуватися змінами в іншому.

Ще одна теорія, хвильова теорія Еліота, справила великий вплив на розвиток технічного аналізу. Основа принципу - зміни у фінансовому ланцюжку мають циклічний характер. Концепція заснована на групі чисел Фібоначчі. Використовуючи природу природних циклів, Елліот показав 80-річний стабільний період із 8 хвиль: 5 з яких підвищують ринок, а потім опускають 3 (рисунок 1.4).

Найдовшим періодом у теорії Елліота є велике надколо, складається з восьми хвиль якого суперцикла, кожен з яких поділяється на 8 хв меншого періоду і так далі рекурсивно. Процес розщеплення триває, виявляється основні хвилини (головні), проміжні (побічні) тані хвилі. Розглянемо основні відмінності генетичних алгоритмів від традиційних.

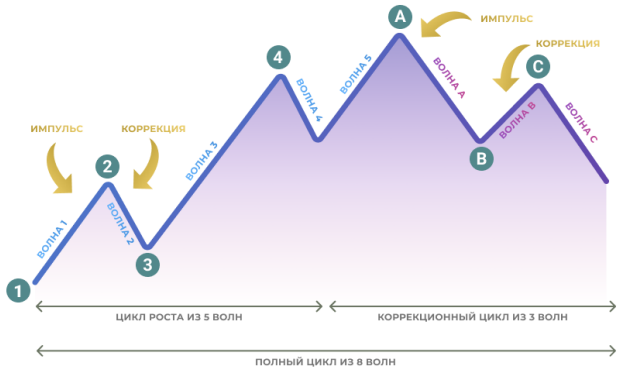


Рисунок 1.4. – Періоди Еліота [13]

Згідно з теорією Елліота, період повного підйому («бика-ведмедя») пов’язаний між западинами «ведмежих» фаз і складається з восьми хвиль. Виділяють п'ять хвиль висхідної фази: три висхідні хвилі (перша, третя і п'ята - пульсові) і дві низхідні хвилі консолідації або корекції (друга і четверта - корекційні). Висхідна фаза складається з трьох хвиль: двох пульсових і однієї корекційної.

Еволюція комп'ютерної техніки в другій половині 20-го століття дала поштовх до розвитку більш досконалих засобів і методів аналізу, а також появі нових методів, що використовують можливості комп'ютерної техніки. В даний момент з'являються нові ринкові показники, розробляються методи визначення залежності від зміни цін, формуються нові теорії аналізу і прогнозування фінансового ринку, удосконалюються стратегії торгівлі фінансовими інструментами. Проте всі сучасні методи технічного аналізу базуються на логіці та принципах поведінки учасників фінансового ринку, вже встановлених у класичних теоріях Доу та Еліота.

У цей період торгові системи надають трейдерам велику кількість технічних індикаторів. Технічний показник - це математична залежність на основі статистичної функції та торгові індикаторів, технічні індикатори можуть аналізувати зміни ринкових тенденцій, аналізуючи поведінку.

Існує кілька груп технічних індикаторів:

- індикатори тренду - для підтвердження або спростування ринкової тенденції;

- осцилятори - вказують точки повороту;

- індикатори каналу - показують межі поточного тренда.

Найпопулярнішими серед трендових індикаторів є класичні ковзні середні, а також різні доповнення до методів перетину різних багатьох ковзних середніх та індексу збіжності ковзних середніх (MACD).

Використані осцилятори включають в себе стохасцилятор або Williams% R, а також показники показників потужності (RSI), які можуть показувати позицію купівлі або продаж фінансового інструменту за допомогою певних статистичних моделей. Найпоширенішим індикатором каналу є смуги Болінджера. У цьому ж рамках виходить лінія тренду, лінія підтримує та опору, що вказує на психологічні рівні.

Переважна фінансово-професійних трейдерів мають власні технічні індикатори, які трейдер використовують для прогнозування конкретного ринку або інструменту. Отриманий це ринковий аналіз шляхом визначення кількох технічних показників із використанням різних груп, що може дати найкращі результати аналізу ринкових моментів та укладення торгових контрактів.

До недоліків технічного аналізу ринків можна віднести несподівану реакцію, яка спостерігається при публікаціях різноманітних економічних звітів і в тих чи інших мірах, які представляють різні політичні чи економічні підходи, які під час аналізу фінансового ринку. У зв’язку з цими великими технічними аналітиками в тій чи іншій мірі використовують у своїй роботі та фундаментальні аналітичні інструменти.

1.6 Фундаментальний аналіз фінансових ринків

Фундаментальний аналіз був розроблений разом з іншим важливим напрямком аналізу фінансового ринку - технічним аналізом. Його послідовники дають прогноз, що реальні причини руху цін на ті чи інші фінансові інструменти можна обґрунтувати та передбачити на основі вивчення різних макроекономічних показників та реакцій ринку, схожих на певні дані.

Фундаментальний аналіз використовується як для прогнозування довгострокового (місяць чи рік) розвитку ринку, так і для аналізу коротких позицій з деякими змінами, що рідше використовується і викликає певні суперечки серед аналітиків.

Класичною роботою по фундаментального аналізу є відома праця Б. Грема і Д. Додда 1934 р. під назвою «Аналіз цінних паперів». Автори спираються на макроекономічні показники та індекси ринкової активності. Спочатку фундаментальний аналіз використовувався лише для оцінки вартості компаній-емітентів та їх акцій відповідно, а також для аналізу фінансових показників компанії, таких як чистий прибуток, чистий капітал та грошові потоки.

На основі цих показників інвестори вирішили знизити або підвищити курс акцій компаній. Нині поняття фундаментального аналізу значно розширилося, і тепер воно охоплює аналіз економічних, політичних подій і навіть стихійних лих.

З розвитком комп'ютерних технологій та Інтернет-технологій економічні події стали доступними для інвесторів і трейдерів у режимі реального часу, що прискорює реакцію ринку на подію.

Основні фактори, які впливають на зміну ціни фінансового інструменту, можна розділити на три основні групи:

- політичні події, природні явища та кризи;

- економічні події;

-.фінансова політика.

До першої групи входять такі заходи, як зустрічі провідних політичних лідерів, різноманітні саміти, аналіз ринку, переговори щодо основних міжнародних угод. Падіння ринків, крах відомих банків, військова ситуація, заворушення, революції, терористичні атаки і навіть повені та землетруси можуть мати серйозний вплив на економіку досліджуваних країн, як фінансову, соціальну, так і природну.

До другої групи входить, насамперед, найважливіший показник – основна процентна ставка країни, є й інші складові фіскальної політики: бюджетна, податкова, інвестиційна та монетарна політика.

До третьої групи можна віднести публікації макроекономічних показників. Фундаментальний макроекономічний аналіз спрямований на виявлення загальних тенденцій, що характеризують економіку країни в цілому та її основні галузі. Існують основні макроекономічні показники в таких сферах:

- валовий внутрішній продукт, послуги та виробництво;
- фондові ринки.
- інфляція;
- ринок праці, доходи та зайнятість;

Фундаменталісти розрізняють очікувані новини (наприклад, публікація запланованих макроекономічних показників) і випадкові (наприклад, стихійні лиха, терористичні акти), тривалі (місяць, рік) і короткі (тиждень, день, добу) фактори життєвого циклу, світові новини, новини окремої держави і навіть окремого суб'єкта.

На основі довгої та успішної історії фундаментального аналізу, багато ринкових аналітиків вивчають це як остаточне твердження про те, що всі макроекономічні показники базуються на змінах фінансових котирувань. інструменти. Також критикується функція фундаментального аналізу оцінки першого результату, якщо фундаментальний прогноз з’являється, це можна пояснити з точки зору технічного аналізу, це може бути збігом. У відомому

дослідженні трейдера Ларрі Вільямса, прогнози аналітиків Wall Street щодо зміни цін на фінансові інструменти справджуються лише в 34% випадків, що менше випадкових прогнозів.

Але навіть найбільш скептично налаштований критик не може не погодитися, що основні економічні події та новини мають певний вплив на котирування цих фінансових інструментів. Тому в сучасних реаліях мало хто з трейдерів не використовує інформацію про економічні новини та події в щоденній торгівлі, особливо з появою онлайн-економічних календарів ця можливість стала дуже доступною.

1.7 Поточні впроваждення та проблеми криптовалютного арбітражу

Звичайно на ринку є і деяки складнощі. У статті Осипович і Чон [14] обговорюють справу Стефана Ціна, 21-річного австралійця з Каліфорнії, який побудував бізнес на основі криптовалютного арбітражу. У 2016 році він заснував Virgil Capital, хедж-фонд, який спеціалізується на криптовалютному арбітражі. Він призупиняє навчання в школі Міневра в Сан-Франциско, щоб керувати фондом, який минулого року повернув понад 400% після зборів і зараз керує 23,5 мільйонами доларів. [13]

Цинь дає інший міжбіржовий арбітраж або купує на одному біржі. Ця практика ставить перед собою кілька проблем. Найважливіша проблема – це час зняття коштів. Тому що, коли хтось купує валюту на біржі і хоче вивести її на іншу біржу, її потрібно обробити блокчейном і видобути. Хіба що комісія майнера висока, імовірно, що транзакція займе значний час, після чого можливість арбітражу зникне.

Інша проблема є тому, що часто потрібні сумі капіталу, щоб дійсно вплинути на вирівнювання ставок на двох цільових біржах.

У статті одного користувача біржі він говорить, що внутрішньобіржовий арбітраж створює свої власні проблеми. По-перше, багато інших має обмеження швидкості для свого API, яке обмежує кількість запитів API, які можна зробити в цей період часу. Ці обмеження курсів запитів відрізняються від біржі, але вони, як правило, починаються приблизно від 60 запитів на хвилину і може досягати 120 запитів на хвилину [15].

Іншою проблемою, яка представляє цю техніку, є дуже маленьким вікном, у якій ви повинні мати ці арбітражі. На ринку арбітражних можливостей FOREX є можливість щонайбільше одну секунду, до того моменту, коли можливість вже використана.

Третя проблема є в тому, що на криптовалютах за бір фіксованих угод тягнеться комісія. Ці можуть відповідати5 від біржі до біржі, але, як правило, коливаються від 0,0% до 0,5% за угоду. Усі це фактори, які необхідні при створенні арбітражного бота [9].

Зі статті Hoppi про ботів для торгівлі криптовалютою в даний час на ринку існує багато реалізацій торгових ботів. «Blackbird», «Haasbot», «Zenbot», «Gekko», «CryptoTrader» та «3Commas» — це кілька доступних торгових ботів криптовалютою, доступних на ринку в даний час. Розглянемо їх по детальніше.

«Blackbird» — біржовий, нейтральний на ринку арбітражний бот, написаний на C++, оскільки він насправді не продається, а продає валюту на одному з бірж, коли спостерігається значну різницю в ціні. Коли ціна в кінцевому підсумку вирівнюється, він встановлює позицію і отримує прибуток. Крім того, операції купівлі/продажу та/купівлі відбуваються паралельно на двох різних біржах, незалежно. Перевага: немає необхідності мати справу з перенесенням проблем із затримкою [10].

«Hassbot» —пропонує 3 різні тарифні плани: один для початківців за ціною 0,073 BTC, один для просунутих трейдерів за 0,208 BTC і один для інвесторів, що розвиваються, за ціною 0,127 BTC. Ціни на один рік. Haasbot, написаний на

C#, і пропонує безліч налаштувань для стратегічної торгівлі ботами. Haasbot може використовувати технічні індикатори, такі як RSI, MACD і Фібоначі.

Присутні заходи безпеки та страхування для збереження інвестицій користувачів. Існує також функція автоматичного налаштування, яку можна використовувати для оптимізації торгової стратегії. Також доступна історія тестування та тестування в реальному часі, розширені сповіщення та звіти.

Платформа зручна для розробників: це означає, що кожен може написати свій власний код для розробки та налаштувати власних ботів. Найважливіше те, що платформа пропонує безліч технічних індикаторів, які можна використовувати для своїх переваг при розробці торгової стратегії [16].

«Zenbot» - бот для обміну криптовалютами з командного рядка, який використовує Node.js та MongoDB. До його можливостей належать: Повністю автоматизований підхід до торгівлі на основі технічного аналізу, повна підтримка GDAX, Poloniex, Kraken, Bittrex, Quadriga, Gemini, Bitfinex, CEX.IO та Bitstamp; архітектура плагінів для реалізації підтримки обміну або написання нових стратегій; симулятор для тестування стратегій на основі історичних даних; «Паперовий» режим торгівлі, який працює на змодельованому балансі під час перегляду реального ринку; настроювані зупинки продажу, зупинки купівлі та стопи (трейлінг) прибутку; і гнучкий період вибірки та частота торгів – у середньому 1-2 операції/день з періодом 1 год, 15-50/день з періодом 5м [17].

«Gekko» - безкоштовна платформа для торгівлі та тестування на біржі Bitcoin TA з відкритим кодом, яка підключається до популярних бірж Bitcoin. Він написаний на JavaScript і працює на Node.js. Він пропонує Trader «Paper» і «Tradebot», який здійснює фактичні операції. Gekko пропонує підтримку плагінів і можливість розробляти власні торгові стратегії. Gekko підтримує 3 різні біржі (включаючи Bitfinex, Bitstamp і Poloniex) [18].

«CryptoTrader» - хмарний міжбіржовий арбітражний бот, який не вимагає встановлення програмного забезпечення. Усі основні біржі криптовалют підтримуються як для тестування, так і для реальної торгівлі. Він також

пропонує ринок стратегій, де стратегії можна купувати та продавати. Він пропонує тестування стратегій торгівлі, щоб побачити, як вибрана стратегія працюватиме в різних ринкових умовах [19].

«3Commas» - торгова платформа, яка пропонує кілька цікавих функцій, включаючи одночасні операції з стоп-лоссом і тейк-профітом. Він також включає функцію під назвою Trailing Features, яка дозволяє отримувати прибуток на більш високому порозі, ніж ви встановили спочатку. У ньому також доступні довгі (довші часові рамки) і короткі (коротші терміни, як правило, денна торгівля) боти.

Він також пропонує бота QFL, який добре працює на стабільних ринках. Торгова стратегія QFL – це ще один варіант торгівлі, заснований на «підтримці ціни». Він також пропонує комбінований бот, який дає вам список монет, якими ви хочете, щоб бот торгував, а потім автоматично керуватиме вашим балансом. Ця опція дозволяє оптимально використовувати баланс і набагато легше використовувати. кілька звичайних ботів [20].

Невідомо, чи можуть ці торговельні боти виявляти чи виконувати арбітражні можливості. Більшість ботів мають можливість розробляти власні торгові стратегії та навіть надавати тестові середовища для перевірки вашої стратегії торгового бота. Однак Норрі зазначив, що деякі торгові боти з кращими оглядами вимагатимуть до 0,32 BTC щомісячної ліцензійної плати [12, 17].

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі роботи була розглянута сфера криптовалютного ринку та арбітражу. Також були розглянуті види аналізу цього ринку. Є два основних видів аналізу ринку: технічний та фундаментальний, кожен зі своїми перевагами та недоліками.

Переважна більшість професійних трейдерів мають власні технічні індикатори, які вони використовує для прогнозування конкретного фінансового ринку або інструменту. Найчастіше це ринковий аналіз шляхом визначення кількох технічних показників з використанням різних груп, що може дати найкращі результати при аналізі ринкових моментів та укладанні торгових контрактів.

Але вони не забувають і про зовнішній вплив на ринок та користуються фундаментальним аналізом у своїх міркуваннях. Фундаменталісти розрізняють очікувані новини (наприклад, публікація запланованих макроекономічних показників), випадкові (наприклад, стихійні лиха, терористичні акти), тривалі (місяць, рік) та короткі (тиждень, день, добу) фактори.

Також у першому розділі було розглянуто поточні впровадження та проблеми криптовалютного ринку. Серед декількох чинників, одними з найсутевіших є великі проценти на згоди та необхідність великого капіталу для помітного прибутку.

РОЗДІЛ 2. Вибір стратегії арбітражу

2.1 Визначення торгової стратегії

Торгова стратегія — це набір заздалегідь визначених правил, які застосовуються для прийняття рішень щодо купівлі чи продажу цінного паперу або інших активів [21]. Зазвичай вона включає в себе план з конкретними цілями інвестування, управління ризиками, часовий горизонт і податкові наслідки.

Як правило, у стратегії є три етапи, кожен з яких включає в себе метрики вимірювання для оцінки та зміни плану відповідно до ринкових змін. Ці етапи включають планування, розміщення торгів та їх виконання [21].

По-перше, необхідно оцінити статус біржі і на основі цього, поєднуючи свої наперед визначені правила, визначити продаж і рішення про покупку. Згодом ці рішення надсилаються брокеру для їх виконання. Щоб отримати доступ до ефективності стратегії, її можна перевірити або, іншими словами, помістити в симуляційне середовище з історичними ринковими даними для вимірювання його ефективності.

2.2 Розробка торгівельної стратегії

По суті, існує два шляхи розробки стратегії, які полягають у використанні технічних або фундаментальних даних [21]. Трейдери, які використовують перший тип, зазвичай спостерігають за ціною цінного папару або активу та її рухом, щоб визначити чи є резон їх купляти і скільки слід у кількості. Тоді як останній, як випливає з назви, враховує основні кількісні та якісні дані, такі як фінансовий стан та репутація компанії, щоб оцінити стабільність та здоров’я її безпеки.

Можна стверджувати, що кількісний трейдинг можна вважати третім типом [21], але він схожий на технічну торгівлю в тому сенсі, що використовує числові дані, але значно більшої складності з використанням багатьох точок даних.

У нашому випадку ми сконцентруємося на принципу SL та TP, оскільки він добре працює навіть коли ціна активів нестабільна. Останні декілька місяців ціна біткоїну та інших криптовалют значно знизилася, тому застосовувати деякі стратегії немає сенсу.

Оскільки предбачити зміну цін у цей час доволі важко, кращим варіантом є покупка та продаж волатильних монет – ціна яких стрімко змінюється. У такому випадку не обов'язково проводити фундаментальний аналіз для подальшого розподілу активів, достатньо уважно слідкувати за змінами цін та робити швидкі рішення. Як как цю потребу автоматизований бот може задовільти набагато краще будь якої людини.

2.3 Технічний індикатор

Розробка торгової стратегії з технічного підходу в значній мірі покладається на спостереження за набором сигналів даних, які називаються технічними індикаторами. Вони створюються шляхом агрегування даних про безпеку за допомогою певних математичних формул, так що стає набагато швидшим і всеосяжнішим, щоб розпізнати тенденцію та рух ціни криптовалюти від швидкого перегляду до швидкої зміни ринкової ціни [21].

Деякі з найпоширеніших включають індекс відносної сили (RSI) і ковзне середнє (КС, англ. Moving Average). Трейдери зазвичай встановлюють свій вибір індикаторів залежно від мети та використовуваних стратегій.

Експерименти, проведені під час цього проекту, використовують індикатор КС. Як це можна обчислити, так це те, що в будь-який торговий день

він спочатку визначає певний період часу для отримання історичних даних, наприклад, 20 днів, потім обчислює середню ціну закриття цього цінного паперу протягом попередніх 19 днів і вибраний і продовжує робити це рекурсивно до теперішнього часу. Циклування протягом тривалого періоду призводить до більш гладкої лінії, що допомагає інвесторам визначити загальну тенденцію, на яку не впливає щоденна волатильність ціни акцій.

2.4 У довгу чи у коротку

Метою будь-якої розумної торгової стратегії є отримання прибутку. Підхід, який найбільш широко відомий навіть трейдерам-початківцям, це «у довгу». Це коли інвестори купують певну кількість криптовалюти за ціною, то сподіваються продати їх за вищою ціною в майбутньому, отримуючи таким чином прибуток за рахунок збільшення ціни за акцію.

Хоча це здається здоровим глуздом, є ряд інвесторів, як правило, професійних з досвідом, що роблять прямо протилежне, тобто продають акції за високою ціною і чекають, коли їх ціна впаде. Ця стратегія називається «у коротку». Механіка, як це працює, досить цікава: інвестор, який дотримується цієї стратегії, позичить певну кількість криптовалюти у брокера чи інших інвесторів і продасть їх за вищою ціною. У цей момент цей інвестор матиме певну суму грошового прибутку від продажу криптовалюти і боргу цієї кількості валюти. Коли ціна впаде, він викупить їх, цього разу за нижчою ціною і поверне ту кількість валюти, яку він позичив. Таким чином, інвестор отримує прибуток від різниці в ціні між продажем і покупкою.

Це вводить концепцію ордерів на тейк-профіт (TP) і стоп-лосс (SL). Загалом їх можна розуміти як верхній і нижній пороги ціни акції для замовлення на продаж, які визначаються індивідуально трейдерами. Позиція буде закрита, або, іншими словами, всі акції в цьому порядку будуть продані, якщо ціна цієї

акції досягне значення тейк-профіту. Це для трейдерів, щоб запобігти зберігати позицію занадто довго і обійти найвищу ціну за цей період [23].

Механіка стоп-лоссу використовуються у зворотньому до того, що було описано вище: позиція буде закрита, коли ціна акцій впаде до нижнього порогу, щоб пом’якшити втрати [24]. Однак ці граничні значення використовуються більш універсально в реальних ситуаціях. Наприклад, коли інвестор знаходиться в короткій позиції, значення тейк-профіту може використовуватися як засіб для того, щоб він дізнався, що настав час викупити ті акції, які він позичив, до того, як їх ціна піде ще вище, і він отримає більш значні втрати.

2.5 Частота торгівлі

Це термін, що відноситься до кількості ордерів, розміщених трейдером за встановлений проміжок часу. Це може варіюватися від кожні два тижні, щодня до кожні 0,003 секунди. Експериментами в цьому проекті є внутрішньоденна торгівля, або денна торгівля, яка полягає в тому, щоб діяти на зміну ціни щосекунди і отримувати прибуток від дуже малих змін ціни; і торгівля позиціями, яка полягає в тому, щоб щодня перевіряти ціну акцій, щоб вирішити, утримувати позицію чи закривати, таким чином, замовлення в цьому виді торгівлі може тривати місяцями або роками.

Планом цього проекту є експериментувати зі стратегіями, що належать до обох кінців спектру, що означає, що програму можна виконувати один раз на день лише за короткий час до моменту закриття ринку для торгівлі, а потім дочекатися зростання курсу акцій або вниз на кілька днів. Він також може працювати доти, поки ринок відкритий, перевіряючи ціну щосекунди та намагаючись обчислити майбутню ціну, щоб вирішити, купувати чи продавати. Цей метод торгівлі називається скальпінгом.

2.6 Вибір криптобіржі

Хоча більшість відомих криптобірж мають схожий функціонал та принципи роботи, важливо обрати біржу з якою буде зручно, а головне швидко працювати. В залежності від стратегії бот повинен найнайшвидше реагувати на зміни на ринку, щоб забезпечити більший прибуток або мінімальні втрати.

У виборі криптобіржі мною було виділено два головних критерія: загальний об`єм усіх транзакцій на спотовому ринці та функціональність API для обробки сигналів.

В загальному випадку, чим більше об`єм транзакцій певного виду на криптобіржі, тим скоріше можна купити вигідну валюту або продати наявну. Також це свідчить про кількість активних користувачів у біржі.

На рисунку 2.1 наведено порівняння статистики реального об`єму транзакцій на спотовому ринці серед 23 найбільших криптобірж минулого року. Як видно з графіку, Binance випереджає усіх з великим відривом, особливо в середені 2021 року під час буму нових монет.

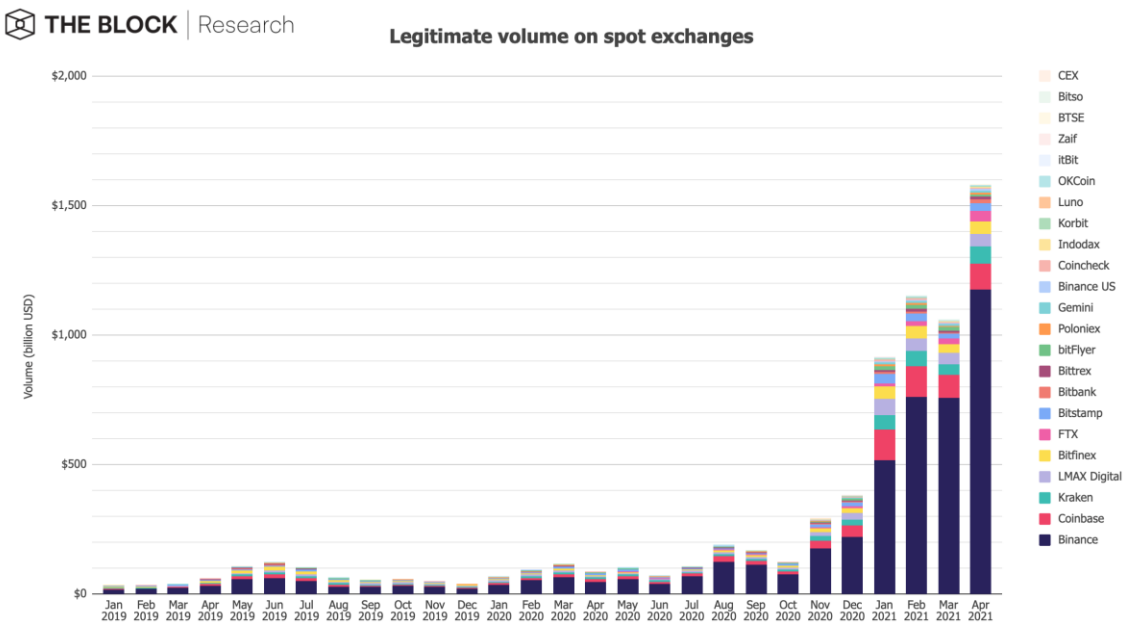
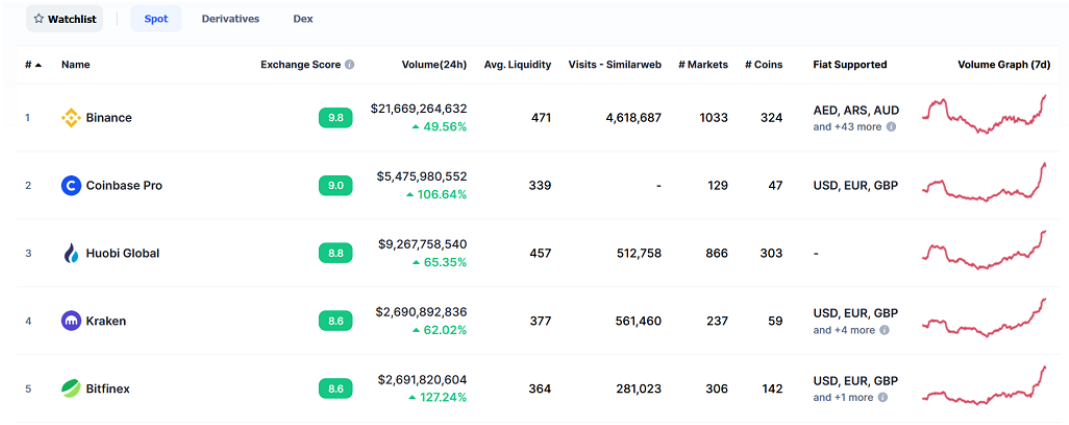


Рисунок 2.1 – Об`єм транзакцій на спотовому ринці за даними THE BLOCK [25]

Щоб переконатися, що об`єм транзакцій не був тільки результатом торгівлі так званих «китів», користувачами з великими статками, а і звичайними трейдерами, проаналізуєм кількість користувачів у найбільших біржах.



☆ Watchlist | Spot | Derivatives | Dex

#	Name	Exchange Score	Volume(24h)	Avg. Liquidity	Visits - Similarweb	# Markets	# Coins	Fiat Supported	Volume Graph (7d)
1	Binance	9.8	$21,669,264,632 ▲ 49.56%	471	4,618,687	1033	324	AED, ARS, AUD and +43 more	
2	Coinbase Pro	9.0	$5,475,980,552 ▲ 106.64%	339	-	129	47	USD, EUR, GBP	
3	Huobi Global	8.8	$9,267,758,540 ▲ 65.35%	457	512,758	866	303	-	
4	Kraken	8.6	$2,690,892,836 ▲ 62.02%	377	561,460	237	59	USD, EUR, GBP and +4 more	
5	Bitfinex	8.6	$2,691,820,604 ▲ 127.24%	364	281,023	306	142	USD, EUR, GBP and +1 more	

Рисунок 2.2 – Статистика найбільших криптобірж від CoinMarketCap [26]

З рисунку 2.2 порівняльної таблички ми бачимо, що кількість щоденних користувачів Binance також випереджає конкурентів. Тільки на веб-платформу кожен день заходять міліони користувачів. Також криптобіржа має найбільшу кількість доступних фіатних валют та монет.

Всі ці переваги зробити вибір не складним, але є ще один чималий фактор – функціональність API платформи. На щастя, у Binance виявилося з цим все добре і API можна використовувати універсально.

Інтерфейсом платформи є RESTful API, який використовує HTTP-запити для надсилання та отримання даних. Крім того, доступний WebSocket, який дозволяє передавати такі дані, як ціни та оновлення облікового запису.

Binance API також забезпечує доступ до середовища для тестувальних торгів, що допомогає поліпшити стратегії без втрати реальних активів. [27]

2.7 Мова програмування

Для розробки даного боту мною було обрано мову програмування Python, як найзручнішою з роботою Binance та перспективною у подальшому користуванні.

Python — це інтерпретована, об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою, розроблена Гвідо ван Россумом. Спочатку вона була випущена у 1991 році. Python має репутацію мови, зручної для початківців, замінюючи Java як найбільш широко використовувану вступну мову, оскільки вона справляється з більшою частиною складнощів для користувача, дозволяючи новачкам зосередитися на повному осягненні концепцій програмування, а не на дрібних деталях. [28]

Вбудований робочий процес Python дозволяє вирішувати проблеми з програмами під час розробки коду бота максимально просто і ефективно. Цикл редагування/тестування/налагодження робить Python надійним та зручним.

Як видно з рисунку 2.3, мова впевнено вийшла на перше місце у рейтингу найпопулярніших мов на початку цього року.

Pos. Jan 2022	Pos. Jan 2021	Programming Language	Ratings	Chart Ratings	Variations
1	3	Python	13.58%		+1.86%
2	1	C	12.44%		-4.94%
3	2	Java	10.66%		-1.30%
4	4	C++	8.29%		+0.73%
5	5	C#	5.68%		+1.73%
6	6	Visual Basic	4.74%		+0.90%
7	7	JavaScript	2.09%		-0.11%
8	11	Assembly language	1.85%		+0.21%
9	12	SQL	1.80%		+0.19%
10	13	Swift	1.41%		-0.02%

Рисунок 2.3 – Рейтинг мов програмування за даними Tiobe [29]

Завдяки простоті навчання та використання коди Python можна легко писати та виконувати набагато швидше, ніж інші мови програмування. Одна з головних причин, чому популярність Python експоненціально зросла, пов'язана з його простотою синтаксису, щоб його було легко читати та розвивати.

На рисунку 2.4 представлена динаміка розвитку мов, але чітко видно ріст у кількості програмістів, які надають перевагу Python.

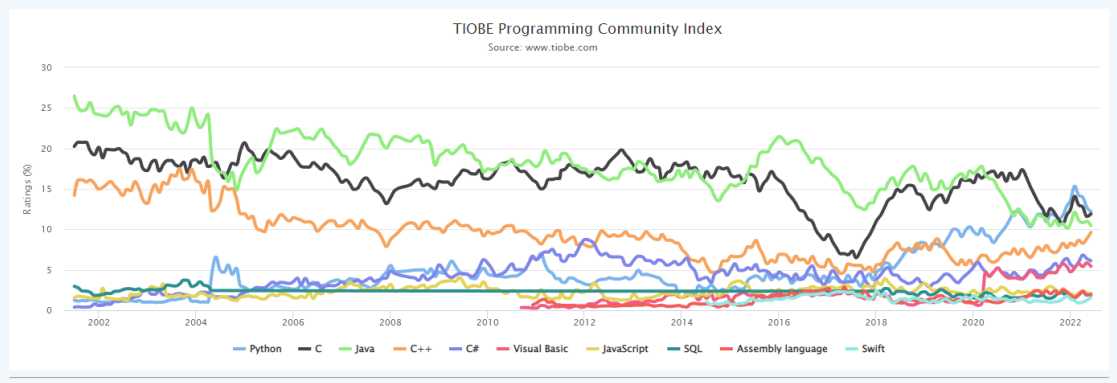


Рисунок 2.4 – Динаміка зміни рейтингу мов за даними Tiobe [29]

Іноді розробник, який спеціалізується на іншій мові програмування, може запитати: «Чому Python повільний?» І так, порівняно з деякими іншими мовами, такими як Java, C#, Go, JavaScript або C++, Python часто має дещо нижчу швидкість виконання. Однак у сучасному світі час розробки набагато важливіше, ніж час роботи комп'ютера. А з точки зору часу виходу на ринок, Python просто не можна обіграти.

Але ще досить сутевим фактором у виборі стала бібліотека Python, яка була розроблена спеціально для ефективної роботи з Binance – "python-binance".

Ця бібліотека стала популярною після конкурсу від Binance у 2017 році. Ціль конкурсу була розробка бібліотеки для роботи з криптобіржею на різних мовах програмування. Кожен переможець отримав по 1000 монет Binance (BNB), яка є внутрішньою валютою.

Під час конкурсу BNB торгувався нижче 2 доларів, але в першій половині 2021 року він піднявся до високого рівня в 691 доларів. Для категорії Python було кілька хороших заявок, але python-binance врешті переміг. Це був розумний підхід від біржі, бо гарантував, що найкращі розробники наполегливо працювали над створенням хорошої бібліотеки.

Отже зваживши всі "за" та "проти" використання Python у проекті, ми можемо зробити однозначний висновок, що мова відповідає усім вимогам

проекту. Також слід додати, що кількість розглянутих переваг, що надаються при її використанні, набагато перевищує усі недоліків.

2.8 Технології та бібліотеки

Для написання та роботи транзакційного боту були використані наступні засоби та технології:

1) Ключ Binance API. Це унікальний код, який генеруються для роботи з інтерфейсом Binance. Він не надає доступ до вводу/виводу рахунків через міркування безпеки, але дозволяє проводити майже усі інші операції через інші програми.
2) Бібліотека Python-binance. Являє собою бібліотеку мови Python, яка використовується для автоматизації взаємодії з біржою криптовалют Binance.
3) Бібліотека Colorama. Один із вбудованих модулів Python для відображення тексту різними кольорами для поліпшеної читабельності виводу.
4) Бібліотека JSON. Дозволяє Python працювати з файлами з розширенням json, які використовуються для структурованих данних.
5) Бібліотека PyYAML. Дозволяє Python працювати з файлами з розширенням yml, які використовуються для серіалізації даних.
6) Бібліотека tradingview-ta. Бібліотека для отримання технічного аналізу із сервісу TradingView, який обробляє та відображає зміну ринку.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі роботи були розглянуті різні стратегії та важливі фактори для арбітражу ринку. Була обрана загальна стратегія для реалізації програми та можливісті її подальшого корректування.

Було порівняні найбільші криптобіржі за загальним об'ємом транзакцій, кількістю активних користувачів, наявністю та функціональністю API. По результатам порівняння була обрана біржа Binance.

Далі було розглянуто питання вибору мови програмування для написання проекту. Найбільш підходящим виявився Python, про що йдеться у розділі. Зокрема, було розглянуто загальні відомості про цю мову програмування, її численні переваги та декілька недоліків. На основі цього було прийнято рішення в її користь, так як мова відповідає всім вимогам проекту.

Останнім були розглянуті технології та бібліотеки використанні у ході написання робочого коду транзакційного боту.

РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ

3.1 Опис взаємодії програмних компонентів

Згідно з дослідженими матеріалами та обраного методу реалізації переходимо до частини розробки програмного продукту.

Можна виділити декілька умов, які я намагався задовільнити у процессі розробки програми:

1) Бот повинен працювати повністю автономно згідно з користувацькими налаштуваннями, які представляють стратегію арбітражу.
2) Повинна бути реалізація тестового моду, у якому можна створити та корегувати стратегію в залежності від останніх трендів без страху втратити активи.
3) Оскільки бот буде працювати без постійного нагляду, уся інформація робота бота має записуватися у логи.
4) Скрипт повинен зупинятися, коли індикатори виявляють «ведмежий тренд» на ринку із-за значного падіння в ціні валюти. Такі періоди небезпечні через значний ризик продавати більшість куплених монет у стоп-лосс.

Схема роботи програми представлена у додатках. Уся конфігурація стрaгегії, а саме налаштування SL/TP, режим роботи, часові інтервали, валюта та дані облікової запису Binance (API Ключ) задаються та зчитуються в yml файлах. Інформація про портфель куплених монет зберігається у json файлі. Тикери та сигнальні зразки задаються та зчитуються у txt файлах, у такий же файл йде лог подій.

3.2 Файл налаштувань «Конфіг.yml»

Yml-файл у якому заздалегідь корегується стратегія відповідно до стану ринку та попередніх трендів. Представлений у двух частинах:

1) Параметри для налаштування скрипта:
   - TEST_MODE (Чи підключати тестовий маркетплейс)
   - LOG_TRADES (Чи зберігати логи роботи боту)
   - LOG_FILE (У якому файлі зберігаються логи)
   - TLD_USER (Чи працює бот з США або Японії, включаючи можливу роботу з VPN - запити з цих країн оброблюються по іншому згідно з політикою Binance)
2) Параметри для налаштування арбітражу:
   - PAIR_WITH (Валюта яка буде використовуватися при купівлі та продажу)
   - QUANTITY (Загальна сума за одну згоду у долларах США)
   - FIATS (Список виключених торгових пар)
   - MAX_COINS (Кількість одночасно відстежуванних монет у портфелі)
   - TIME_DIFFERENCE (Кількість часу для обчислення різниці від поточної ціни)
   - RECHECK_INTERVAL (Кількість разів перевірки SL/TP протягом проміжку TIME_DIFFERENCE)
   - CHANGE_IN_PRICE (Різниця у % між першою та другою перевірками ціни, після якої бот буде продавати з порфелю)
   - STOP_LOSS (Визначення % зменшення ціни, при якому буде продаватися монета, яка не приносить прибутку)
   - TAKE_PROFIT (Визначення % збільшення ціни, при якому буде продаватися монета, яка приносить прибуток)
   - CUSTOM_LIST (Чи використовувати кастомний лист тикерів для фільтрації пар)

• TICKERS_LIST (У якому файлі збережені тикери)

• USE_TRAILING_STOP_LOSS (Чи використовувати трейлінг SL)

• TRAILING_STOP_LOSS (Значення у відсотках, на які буде переміщено SL при досягнені TP)

• TRAILING_TAKE_PROFIT (Значення у відсотках, на які буде переміщено TP при досягнені TP)

• TRADING_FEE (Процент торгова комісії при арбітражу)

• SIGNALING_MODULES (Назви модулів проекту, в яких оброблюються сигнали)

3.3 Функція get_price()

Після того бот зчитав задані налаштування у файлах, він починає роботу з аналізу ринку. Для цього потрібно дізнатись поточну ціну монет на біржі, що і робиться у функції get_price (див. рисунок 3.1)

```
def get_price(add_to_historical=True):
    """Повертає поточну ціну всіх монет на Binance"""

    global historical_prices, hsp_head

    initial_price = {}
    prices = client.get_all_tickers()

    for coin in prices:

        if CUSTOM_LIST:
            if any(item + PAIR_WITH == coin['symbol'] for item in tickers) and all(item not in coin['symbol'] for item in FIATS):
                initial_price[coin['symbol']] = {'price': coin['price'], 'time': datetime.now()}
        else:
            if PAIR_WITH in coin['symbol'] and all(item not in coin['symbol'] for item in FIATS):
                initial_price[coin['symbol']] = {'price': coin['price'], 'time': datetime.now()}

    if add_to_historical:
        hsp_head += 1

        if hsp_head == RECHECK_INTERVAL:
            hsp_head = 0

        historical_prices[hsp_head] = initial_price

    return initial_price
```

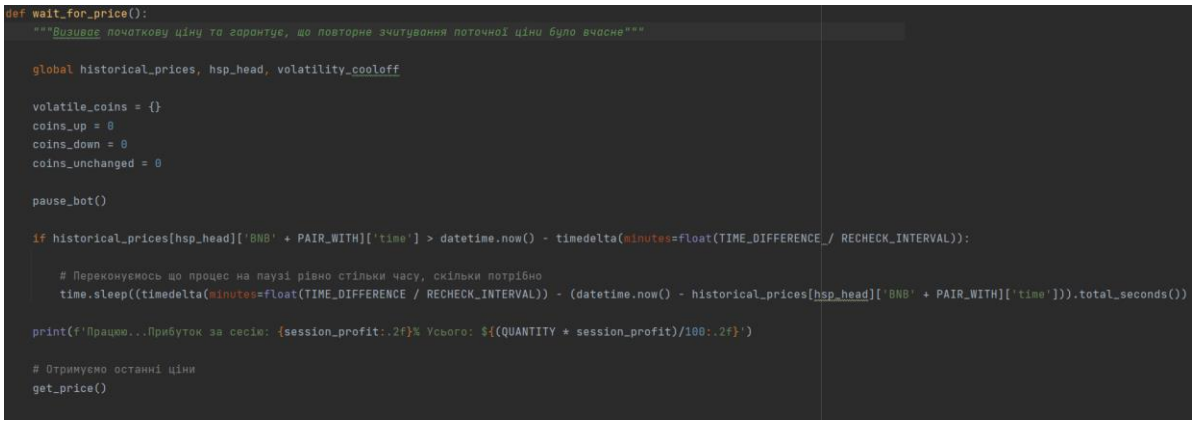
Рисунок 3.1 – Реалізація функції get_price()

Аналізуються лише монети передбачені в конфігурації, повертається назва монети, ціна та час.

3.4 Функція wait_for_price()

Після першого аналізу ціни бот перевіряє ринок на зміни у спробі знайти волатильні монети, які задовільняють певні умови для подальшої покупки.

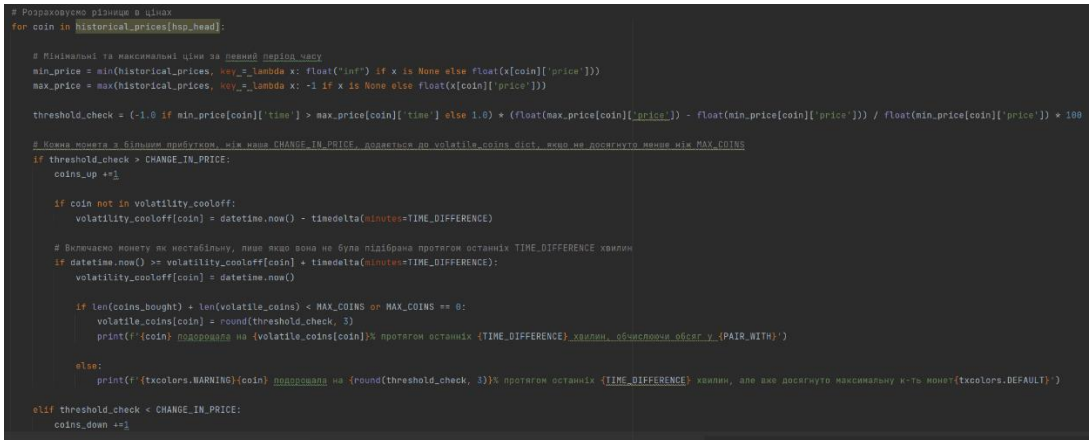
Періодичність перевірок залежить від конфігурації, також бот інформувати про прибуток за сесію.



```
def wait_for_price():
    """Визиває початкову ціну та гарантує, що повторне зчитування поточної ціни було вчасне"""

    global historical_prices, hsp_head, volatility_cooloff

    volatile_coins = {}
    coins_up = 0
    coins_down = 0
    coins_unchanged = 0

    pause_bot()

    if historical_prices[hsp_head]['BNB' + PAIR_WITH]['time'] > datetime.now() - timedelta(minutes=float(TIME_DIFFERENCE / RECHECK_INTERVAL)):

        # Переконуємось що процес на паузі рівно стільки часу, скільки потрібно
        time.sleep((timedelta(minutes=float(TIME_DIFFERENCE / RECHECK_INTERVAL)) - (datetime.now() - historical_prices[hsp_head]['BNB' + PAIR_WITH]['time'])).total_seconds())

    print(f'Працюю...Прибуток за сесію: {session_profit:.2f}% Усього: ${(QUANTITY * session_profit)/100:.2f}')

    # Отримуємо останні ціни
    get_price()
```

Рисунок 3.2 – Реалізація функції wait_for_price(), періодичний аналіз ринку

Після чергової перевірки скрипт порівнює ціну монет та шукає волатильні моменти, саме ті через яких можна отримати прибуток на фоні тренду. Результат порівняння виводиться.



```
# Розраховуємо різниця в цінах
for coin in historical_prices[hsp_head]:

    # Мінімальні та максимальні ціни за певний період часу
    min_price = min(historical_prices, key = lambda x: float("inf") if x is None else float(x[coin]['price']))
    max_price = max(historical_prices, key = lambda x: -1 if x is None else float(x[coin]['price']))

    threshold_check = (-1.0 if min_price[coin]['time'] > max_price[coin]['time'] else 1.0) * (float(max_price[coin]['price']) - float(min_price[coin]['price'])) / float(min_price[coin]['price']) * 100

    # Кожна монета з більшим прибутком, ніж наше CHANGE_IN_PRICE, додається до volatile_coins dict, якщо не досягнуто меньше ніж MAX_COINS
    if threshold_check > CHANGE_IN_PRICE:
        coins_up +=1

        if coin not in volatility_cooloff:
            volatility_cooloff[coin] = datetime.now() - timedelta(minutes=TIME_DIFFERENCE)

        # Включаємо монету як нестабільну, лише якщо вона не була підібрана протягом останніх TIME_DIFFERENCE хвилин
        if datetime.now() >= volatility_cooloff[coin] + timedelta(minutes=TIME_DIFFERENCE):
            volatility_cooloff[coin] = datetime.now()

            if len(coins_bought) + len(volatile_coins) < MAX_COINS or MAX_COINS == 0:
                volatile_coins[coin] = round(threshold_check, 3)
                print(f'{coin} подорожала на {volatile_coins[coin]}% протягом останніх {TIME_DIFFERENCE} хвилин, обчислюючи обєкт у {PAIR_WITH}')

            else:
                print(f'{txcolors.WARNING}{coin} подорожала на {round(threshold_check, 3)}% протягом останніх {TIME_DIFFERENCE} хвилин, але вже досягнуто максимальну к-ть монет{txcolors.DEFAULT}')

    elif threshold_check < CHANGE_IN_PRICE:
        coins_down +=1
```

Рисунок 3.3 – Реалізація функції wait_for_price(), розрахунок зміни цін

Після знаходження потрібних монет та співставлення з сигналами функція повертає отримані результати та виводить їх для логу.

```
    else:
        coins_unchanged +=1


# Зовнішні сигнали
externals = external_signals()
exnumber = 0

for excoin in externals:
    if excoin not in volatile_coins and excoin not in coins_bought and \
            (len(coins_bought) + exnumber + len(volatile_coins)) < MAX_COINS:
        volatile_coins[excoin] = 1
        exnumber +=1
        print(f'Зовнішній сигнал отримано {excoin}, обчислюючи обсяг у {PAIR_WITH}')

return volatile_coins, len(volatile_coins), historical_prices[hsp_head]
```

Рисунок 3.4 – Реалізація функції wait_for_price(), реєстрування волатильних монет

3.5 Функція pause_bot()

Дана функція забезпечує призупинення роботи пошуку монет для купівлі у моменти коли ринок стає «ведмежим». У такий час є великий ризик паніки серед трейдерів, так як ціна на деякі монети різко йде вниз і замість того, щоб урівноважити попит і пропозицію, користуваючі біржі не купляють монети за низкою ціною, що призводить до ще більшого падіння та пробиття так званої межі підтримки – граничної ціни, коли попит має стати більше пропозиції.

```
def pause_bot():
    """Призупиняємо скрипт, коли зовнішні індикатори виявлять ведмежий тренд на ринку"""
    global bot_paused, session_profit, hsp_head

    # Починаємо рахувати скільки часу бот буде призупинений
    start_time = time.perf_counter()

    while os.path.isfile("signals/paused.exc"):

        if bot_paused == False:
            print(f'{txcolors.WARNING}Призупиняю купівлю через зміну ринкових умов, стоп-лосс і тейк-профіт продовжать працювати...{txcolors.DEFAULT}')
            bot_paused = True

        # Функція продажу працює навіть під час призупинення
        coins_sold = sell_coins()
        remove_from_portfolio(coins_sold)
        get_price(True)

        # Зупиняємо тут
        if hsp_head == 1: print(f'Призупинено...Прибуток сесії: {session_profit:.2f}% Усього: ${(QUANTITY * session_profit)/100:.2f}')
        time.sleep((TIME_DIFFERENCE * 60) / RECHECK_INTERVAL)


    else:
        # Припиняємо відлік часу паузи
        stop_time = time.perf_counter()
        time_elapsed = timedelta(seconds=int(stop_time-start_time))

        # Відновляємо роботу бота і встановлюємо для pause_bot значення False
        if bot_paused == True:
            print(f'{txcolors.WARNING}Відновлюю покупку через зміну ринкових умов, загальний час сну: {time_elapsed}{txcolors.DEFAULT}')
            bot_paused = False

    return
```

Рисунок 3.5 – Реалізація функції pause_bot()

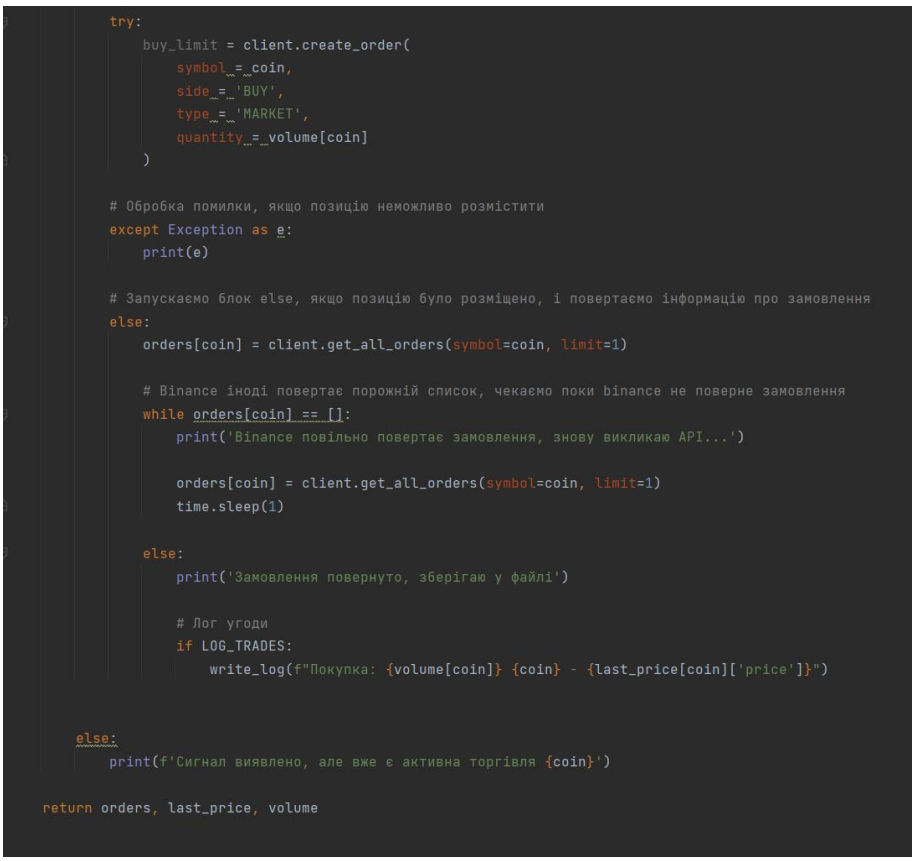
3.6 Функція buy()

Після того як потрібна волатильна монета знайдена, ми хочемо як найшвидше її купити. Бот перевіряє чи потрібно працювати з реальним маркетплейсом або тестовим та готується до купівлі перевіркою на поточні угоди з цією монетою. Також початок купівлі документується у лозі.

```
buy():
"""Розміщує ринкові ордери на покупку для кожної знайденої нестабільної монети"""
volume, last_price = convert_volume()
orders = {}

for coin in volume:

    # Купуємо лише тоді, коли немає активних торгів з монетою
    if coin not in coins_bought:
        print(f"{txcolors.BUY}Готуємося до покупки {volume[coin]} {coin}{txcolors.DEFAULT}")

        if TEST_MODE:
            orders[coin] = [{
                'symbol': coin,
                'orderId': 0,
                'time': datetime.now().timestamp()
            }]

            # Лог угоди
            if LOG_TRADES:
                write_log(f"Покупка: {volume[coin]} {coin} - {last_price[coin]['price']}")

            continue
```

Рисунок 3.6 – Реалізація функції buy(), підготовка до покупки

Далі ми намагаємось заключити угоду на купівлю монети згідно з конфігурацією та обробляємо ситуацію коли вже є поточні угоди. Йде запис у лог яку монету ми куплямо та за скільки.



```
            try:
                buy_limit = client.create_order(
                    symbol = coin,
                    side = 'BUY',
                    type = 'MARKET',
                    quantity = volume[coin]
                )

            # Обробка помилки, якщо позицію неможливо розмістити
            except Exception as e:
                print(e)

            # Запускаємо блок else, якщо позицію було розміщено, і повертаємо інформацію про замовлення
            else:
                orders[coin] = client.get_all_orders(symbol=coin, limit=1)

                # Binance іноді повертає порожній список, чекаємо поки binance не поверне замовлення
                while orders[coin] == []:
                    print('Binance повільно повертає замовлення, знову викликаю API...')

                    orders[coin] = client.get_all_orders(symbol=coin, limit=1)
                    time.sleep(1)

                else:
                    print('Замовлення повернуто, зберігаю у файлі')

                    # Лог угоди
                    if LOG_TRADES:
                        write_log(f"Покупка: {volume[coin]} {coin} - {last_price[coin]['price']}")


        else:
            print(f'Сигнал виявлено, але вже є активна торгівля {coin}')

    return orders, last_price, volume
```

Рисунок 3.7 – Реалізація функції buy(), розміщення угоди

3.7 Процедура update_portfolio()

Коли ми купили волатильну монету, з'являється необхідість за нею слідкувати, щоб потім продати зафіксувавши прибуток або збитки. Для цього ми додаєм її у порфоліо, яке зберігає усі куплені монети для подальшого продажу. Скрипт продовжує працювати додаючи куплені монети та може занести у портфоліо кількість монет обмежену початковою конфігурацією. Коли монета продається, монета вилучається з порфоліо іншою процедурою. При цьому усі зміни ведуться у лог.

```python
update_portfolio(orders, last_price, volume):
"""Додає кожну куплену монету до портфоліо для подальшого відстеження або продажу"""
if DEBUG: print(orders)
for coin in orders:

    coins_bought[coin] = {
        'symbol': orders[coin][0]['symbol'],
        'orderid': orders[coin][0]['orderId'],
        'timestamp': orders[coin][0]['time'],
        'bought_at': last_price[coin]['price'],
        'volume': volume[coin],
        'stop_loss': -STOP_LOSS,
        'take_profit': TAKE_PROFIT,
        }

    # Зберігаємо монети у json файл у тому ж каталозі
    with open(coins_bought_file_path, 'w') as file:
        json.dump(coins_bought, file, indent=4)

    print(f'Угода з ID {orders[coin][0]["orderId"]} розміщена та збережена у файлі')
```

Рисунок 3.8 – Реалізація процедури update_portfolio()

3.8 Функція sell_coins()

Ця функція перевіряє куплені монети у портфоліо на готовність їх продати та розміщує згоду на продаж.

Для початку для кожної монети у портфоліо обчислюється рівень SL та TP згідно з конфігурацією. Перший покаже як швидко монету треба продати при зменшенні ціни, а другий – коли продати, якщо ціна продовжує рости. Ці показники також корегуються в залежності від змін цін завдяки трейлінг механіці.

```python
def sell_coins():
    """продає монети, які досягли порогу стоп-лосс або тейк-профіт"""

    global hsp_head, session_profit

    last_price = get_price(False)
    coins_sold = {}

    for coin in list(coins_bought):
        # Визначаємо стоп-лосс і тейк-профіт
        TP = float(coins_bought[coin]['bought_at']) + (float(coins_bought[coin]['bought_at']) * coins_bought[coin]['take_profit']) / 100
        SL = float(coins_bought[coin]['bought_at']) + (float(coins_bought[coin]['bought_at']) * coins_bought[coin]['stop_loss']) / 100


        LastPrice = float(last_price[coin]['price'])
        BuyPrice = float(coins_bought[coin]['bought_at'])
        PriceChange = float((LastPrice - BuyPrice) / BuyPrice * 100)

        # Переконуємося, що ціна вище тейк-профіту і повторно відповідно відкоригуємо SL і TP, якщо використовується пробний стоп-лосс
        if LastPrice > TP and USE_TRAILING_STOP_LOSS:

            # Збільшуємо тейк-профіт на TRAILING_TAKE_PROFIT
            coins_bought[coin]['take_profit'] = PriceChange + TRAILING_TAKE_PROFIT
            coins_bought[coin]['stop_loss'] = coins_bought[coin]['take_profit'] - TRAILING_STOP_LOSS
            if DEBUG: print(f"{coin} Тейк-профіт досягнуто, коригуємо наступне його значення {coins_bought[coin]['take_profit']:.2f}  та стоп-лосс {coins_bought[coin]['stop_loss']:.2f} ві
            continue
```

Рисунок 3.9 – Реалізація функції sell_coins(), робота з SL/TP

Після обчислення необхідних параметрів проводиться перевірка, яка визначає чи монети у портфоліо готові бути продані. Відбуваться спроба розмістити згоду на продаж, а також обробка варіантів коли це зробити неможливо. Також запам'ятовуємо позицію, щоби ненароком не купити ту ж саму монету.

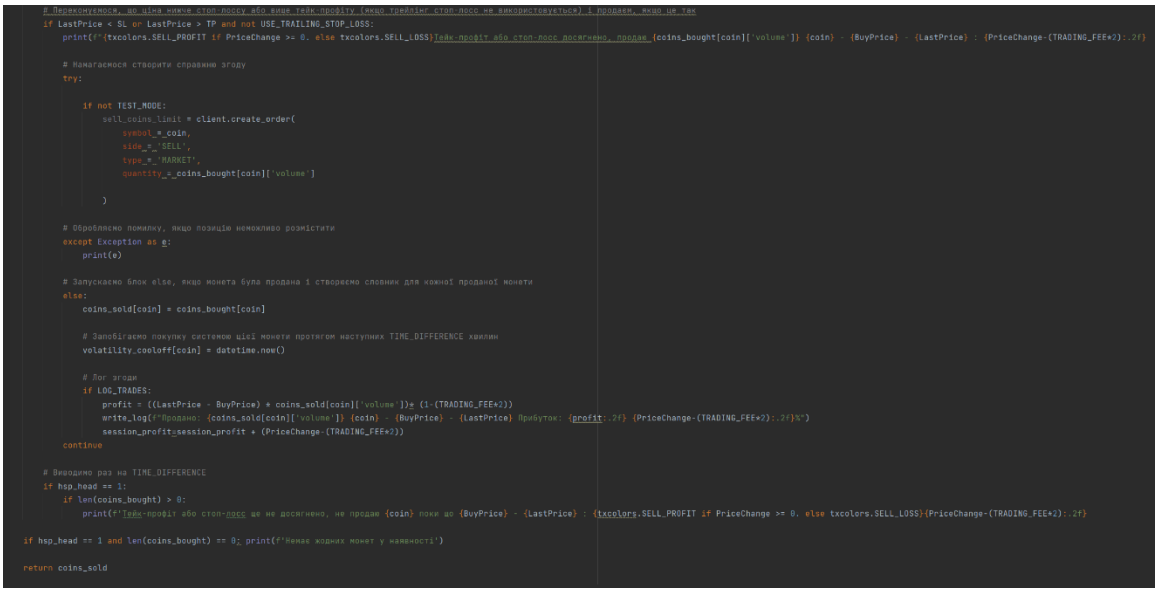


Рисунок 3.10 – Реалізація функції sell_coins(), продаж монет

ВИСНОВОК ДО РОЗДІЛУ 3

У результаті роботи з розробки програмного забезпечення для автоматизації торгівлі на криптобіржі Binance можна виділити наступне:

- Була реалізована раніше обгрунтована загальна стратегія торгів волантильними монетами
- Представлена та пояснена конфігурація роботи бота
- Детально описаний принцип роботи головних функцій та процедур
- Продемонстровано фрагменти коду для описаних процессів у вигляді скріншотів

РОЗДІЛ 4. РЕЗУЛЬТАТИ РОБОТИ БОТА

4.1 Обрана конфігурація

Ефективність роботи бота напряму залежить від конфігурації та стану поточного ринку. Оскільки в останній час ринок досі волатильний та ціна більшості криптовалют йде вниз, були вибрані не ризикові значення SL та TP (рисунок 4.1)

```
# Кількість разів для перевірки TP/SL протягом кожного TIME_DIFFERENCE (мінімум: 1).
# Binance API має обмеження щодо запитів(максимум 1200 запитів на хвилину на IP), якщо більше то прилетить бан
RECHECK_INTERVAL: 10

# Різниця у % між першою та другою перевірками ціни, після якої бот буде продавати
CHANGE_IN_PRICE: 10

# Визначення % зменшення ціни, при якому будемо продавати монету яка не приносить прибутку
STOP_LOSS: 5

# Визначення % збільшення ціни, при якому будемо продавати монету яка приносить прибуток
TAKE_PROFIT: 3

# Чи використовувати кастомний лист tickers.txt для фільтрації пар
CUSTOM_LIST: True

# Назва списку кастомних тікерів
TICKERS_LIST: 'Тикери.txt'

# Чи використовувати трейлінг SL
USE_TRAILING_STOP_LOSS: False

# При досягненні TAKE_PROFIT переміщаємо STOP_LOSS на TRAILING_STOP_LOSS відсотків нижче TAKE_PROFIT фіксуючи прибуток
# При досягненні TAKE_PROFIT переміщаємо TAKE_PROFIT вгору на TRAILING_TAKE_PROFIT відсотків
TRAILING_STOP_LOSS: .4
TRAILING_TAKE_PROFIT: .1
```

Рисунок 4.1 – Конфігурація роботи боту

4.2 Лог арбітражу

Бот виводить інформацію про свою роботу у консоль. Є також розширенний варіант конфігурації логу арбітражу, коли буде виводитися інформація про поточні налаштування, статус підключення до API та додаткові деталі про поточний ринок.

Початок роботи бота зображено на рисунку 4.2, бот працює повністю автоматично без потреби у вводі будь якої інформації. В залежності від

конфігурації у файлах, виводиться попередження, інформація про ринок та спроби пошуку волантивних монет.

```
C:\Users\DanyloB\AppData\Local\Programs\Python\Python310\python.exe C:\Users\DanyloB\Desktop\Crypto_trading_bot\Бінанс_бот.py
[2022-06-12 22:03:20] Натисніть Ctrl-C, щоб зупинити сценарій
[2022-06-12 22:03:20] ПОПЕРЕДЖЕННЯ: Використовується мережа Mainnet та реальний рахунок. Зачекайте 30 секунд як захід безпеки
[2022-06-12 22:03:50] Запуск Мод_паузи
[2022-06-12 22:03:50] Мод_паузи: Ринок виглядає не дуже добре, призупиняю купівлю у режимі 11/7. Очікую 1 хвилин для наступної перевірки ринку
[2022-06-12 22:03:52] Запуск Мод_сигнальних_семплів
[2022-06-12 22:03:52] Мод_сигнальних_семплів: Аналіз 16 монет
[2022-06-12 22:03:55] Призупиняю купівлю через зміну ринкових умов, стоп-лосс і тейк-профіт продовжать працювати...
[2022-06-12 22:03:56] Призупинено...Прибуток сесії: 0.00 Усього: $37.7
[2022-06-12 22:04:00] Мод_сигнальних_семплів: Максимальний сигнал у ICPUSDT у к-ті 8 (шорт)
[2022-06-12 22:04:00] Мод_сигнальних_семплів: Немає монет вище порогу 18 в обох таймфреймах. Очікую 4 хвилин для наступного аналізу
[2022-06-12 22:04:08] Немає жодних монет у наявності
[2022-06-12 22:04:51] Мод_паузи: Ринок виглядає не дуже добре, призупиняю купівлю у режимі 8/7. Очікую 1 хвилин для наступної перевірки ринку
```

Рисунок 4.2 – Вивід логу арбітражу бота

Якщо користувач бажає мати більш розгорнутий лог з інформацією про роботу з API, достатньо лише включити необхідний флаг у конфігурації. Тоді буде виводитися уся додаткова інформація про стан ринку та монет у портфоліо. На рисунку 4.3 зображений початок виводу логу при розгорнутій конфігурації.

```
[2022-06-13 22:25:12] Завантажена конфігурація нижче
{
    "script_options": {
        "TEST_MODE": false,
        "LOG_TRADES": true,
        "LOG_FILE": "\u0417\u0433\u043e\u0434\u0438.txt",
        "TLD_USER": false
    },
    "trading_options": {
        "PAIR_WITH": "USDT",
        "QUANTITY": 10,
        "FIATS": [
            "EURUSDT",
            "GBPUSDT",
            "JPYUSDT",
            "USDUSDT",
            "DOWN",
            "UP"
        ],
        "MAX_COINS": 5,
        "TIME_DIFFERENCE": 2,
        "RECHECK_INTERVAL": 10,
        "CHANGE_IN_PRICE": 10,
        "STOP_LOSS": 5,
        "TAKE_PROFIT": 3,
        "CUSTOM_LIST": true,
        "TICKERS_LIST": "\u0422\u0438\u043a\u0435\u0440\u0438.txt",
        "USE_TRAILING_STOP_LOSS": false,
        "TRAILING_STOP_LOSS": 0.4,
        "TRAILING_TAKE_PROFIT": 0.1,
        "TRADING_FEE": 0.075,
        "SIGNALLING_MODULES": [
            "\u041c\u043e\u0434_\u043f\u0430\u0443\u0437\u0438",
            "\u041c\u043e\u0434_\u0441\u0438\u0433\u043d\u0430\u043b\u044c\u043d\u0438\u0445_\u0441\u0435\u043c\u043f\u043b\u0456\u0432"
        ]
    }
}
[2022-06-13 22:25:12] Облікові дані завантажено з Креди.yml
[2022-06-13 22:25:12] Натисніть Ctrl-C, щоб зупинити сценарій
[2022-06-13 22:25:12] ПОПЕРЕДЖЕННЯ: Використовується мережа Mainnet та реальний рахунок. Зачекайте 30 секунд як захід безпеки
[2022-06-13 22:25:42] Запуск Мод_паузи
[2022-06-13 22:25:43] Мод_паузи: Ринок виглядає не дуже добре, призупиняю купівлю у режимі 14/7. Очікую 1 хвилин для наступної перевірки ринку
[2022-06-13 22:25:44] Запуск Мод_сигнальних_семплів
[2022-06-13 22:25:44] Мод_сигнальних_семплів: Аналіз 16 монет
```

Рисунок 4.3 – Конфігурація роботи боту

4.3 Лог торгівлі

Результати усіх успішних покупок та продаж виводяться у файл для зберігання та подальшої корректировки стратегії. Незважаючи на те, що ситуація на критовалютному ринку досить не стабільна, як видно з логу на рисунку 4.4, трохи більше ніж половина торгів були вдалими.

```
Покупка 0.56 BCH - 20.00 USDT
Покупка 3.23 VET - 20.00 USDT
Продаж 3.23 VET - 21.98 USDT
Покупка 0.09 LTC - 20.00 USDT
Продаж 0.09 LTC - 22.17 USDT
Покупка 0.84 UNI - 20.00 USDT
Покупка 0.008 BNB - 20.000 USDT
Продаж 0.56 BCH - 23.10 USDT
Продаж 0.84 UNI - 19.72 USDT
Продаж 0.008 BNB - 18.835 USDT
Покупка 0.61 XTZ - 20.00 USDT
Покупка 5.14 QTUM - 20.00 USDT
Продаж 0.61 XTZ - 22.43 USDT
Покупка 0.10 LTC - 20.00 USDT
Покупка 0.79 VTHO - 20.00 USDT
Продаж 5.14 QTUM - 19.58 USDT
Продаж 0.79 VTHO - 21.50 USDT
Покупка 0.010 BNB - 20.00 USDT
Продаж 0.10 LTC - 24.10 USDT
Продаж 0.010 BNB - 22.08 USDT
```

Рисунок 4.4 – Вивід логу торгівлі

ВИСНОВОК ДО РОЗДІЛУ 4

У цьому розділі була представлена робота транзакційного бота, його конфігурація та результат торгів. Зважаючи на поточний стан ринку, був обраний більш консервативний підхід до торгів та представлена обрана стратегія у конфігурації.

Результати торгів у нестабільному поточному ринку загалом позитивні. З подальшим корегуванням можна буде спостерігати більш послідовний зріст прибутку.

ВИСНОВКИ

У ході виконання дипломної роботи був створений бот для автоматизованного арбітражу волатильних монет на криптовалютному ринку. Ця програма дозволяє користувачам криптобіржі вести торги згідно їх стратегій без постійної особистої участі та усуває можливість прийняття рішень емоційного характеру.

У першому розділі роботи була розглянута сфера криптовалютного ринку, арбітражу та види його аналізу. Зазначені переваги та недоліки різних видів прогнузувань. Також описані поточні впроваждення та проблеми криптовалютного ринку помічені користувачами.

У другому розділі була пророблена робота з огляду різних стратегій, вибору криптобіржі та технологій для написання боту. Вибори були зроблені після детального порівняння серед найпопулярніших рішень. Описані бібліотеки Python для роботи з криптобіржою Binance.

У третьому розділі детально описуються реалізація раніше обгрунтованої загальної стратегії торгів волантильними монетами. Також представлена та пояснена конфігурація роботи бота. У розділі описаний принцип роботи головних функцій та процедур, а також наведені їх фрагменти коду з поясненнями.

В останньому розділі була представлена робота транзакційного бота, його конфігурація та результат торгів. Торги автоматизованим ботом навіть у поточному нестабільному ринку виявилися прибутковими та будуть продовжені з подальшим корегуванням стратегій.

Підводячи підсумок дипломної роботи, можна впевнено стверджувати, що усі поставлені задачі були виконані та мета дослідження досягнута.

Створенний бот працює згідно початкових цілей та дозволяє автоматизовувати обрані стратегії у найшвидший спосіб.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «Where the World Regulates Cryptocurrency», Katharina Buchholz in Statista [Електронний ресурс] – Режим доступу до ресурсу: https://cdn.statcdn.com/Infographic/images/normal/27069.jpeg (дата звернення 11.06.2022).
2. «A Short History Of Bitcoin And Crypto Currency Everyone Should Read», Марр Б. [Електронний ресурс] – Режим доступу до ресурсу: https://www.forbes.com/sites/bernardmarr/2017/12/06/a-short-history-of-bitcoin-and-crypto-currency-everyone-should-read/?sh=4665d0083f27 (дата звернення 11.06.2022).
3. «Bitcoin price chart for the entire history from 2008 to 2022», Bytwork.com [Електронний ресурс] – Режим доступу до ресурсу: https://bytwork.com/sites/default/files/styles/webp_dummy/public/images/faq/kursy-za-vsyu-istoriyu.webp?itok=Y021OqqJ (дата звернення 11.06.2022).
4. «The Coming Currency War: Digital Money vs. the Dollar», Майклз Д. і Вігна П. [Електронний ресурс] – Режим доступу до ресурсу: https://www.wsj.com/articles/the-coming-currency-war-digital-money-vs-the-dollar-11569204540 (дата звернення 11.06.2022).
5. «Dwindling cash use is pushing central banks to race toward digital currencies», Берсетче Д. [Електронний ресурс] – Режим доступу до ресурсу: https://www.cnbc.com/2021/02/12/cryptocurrency-dwindling-cash-use-is-pushing-central-banks.html (дата звернення 11.06.2022).
6. «Can the Buffett Indicator Effectively Measure Crypto Market Valuations», Bybit.com [Електронний ресурс] – Режим доступу до ресурсу: https://static.ffbbbdc6d3c353211fe2ba39c9f744cd.com/wp-content-

learn/uploads/2021/08/24134540/Total-crypto-market-cap-1024x710.jpg (дата звернення 11.06.2022).

7. «China is winning the digital currency race. Can it unseat the dollar?», Дівакар, А [Електронний ресурс] – Режим доступу до ресурсу: https://www.trtworld.com/magazine/china-is-winning-the-digital-currency-race-can-it-unseat-the-dollar-44689 (дата звернення 11.06.2022).
8. «Free lunch! arbitrage opportunities in the foreign exchange markets», K. Yamada and H. Takayasu [Електронний ресурс] – Режим доступу до ресурсу: https://www.nber.org/system/files/working_papers/w18541/w18541.pdf (дата звернення 11.06.2022).
9. «Beginner's guide to bitcoin trading bots», A. Norry [Електронний ресурс] – Режим доступу до ресурсу: https://blockonomi.com/bitcoin-trading-bots/ (дата звернення 11.06.2022).
10. «Advantage of short-selling and minimum budget», butor (Github user) [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/butor/blackbird/issues/100 (дата звернення 11.06.2022).
11. «Problem 7 set solutions», E. Demiane and S. Goldwasser [Електронний ресурс] – Режим доступу до ресурсу: https://courses.csail.mit.edu/6.046/spring04/handouts/ps7sol.pdf (дата звернення 11.06.2022).
12. «Bellman-ford algorithm», Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm (дата звернення 11.06.2022).
13. «Волновая теория Эллиотта» [Електронний ресурс] – Режим доступу до ресурсу: https://info.exmo.me/wp-content/uploads/2021/02/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F-

%D0%AD%D0%BB%D0%BB%D0%B8%D0%BE%D1%82%D1%82%D0%B0-%D0%B2%D0%BE%D0%BB%D0%BD%D1%8B.png (дата звернення 11.06.2022).

14. «Bitcoin’s crashing? that won’t stop arbitrage traders from raking in millions», A. Osipovich and E. Jeong [Електронний ресурс] – Режим доступу до ресурсу: https://www.wsj.com/articles/bitcoins-crashing-that-wont-stop-arbitrage-traders-from-raking-in-millions-1517749201 (дата звернення 11.06.2022).
15. «A brief look at crypto arbitrage trading», scrawl (username) [Електронний ресурс] – Режим доступу до ресурсу: https://steemit.com/cryptocurrency/@scrawl/a-brief-look-at-crypto-arbitrage-trading (дата звернення 11.06.2022).
16. «Our automated crypto trading platform features» [Електронний ресурс] – Режим доступу до ресурсу: https://www.haasonline.com/trade-bots/ (дата звернення 11.06.2022).
17. «Zenbot» [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/DeviaVir/zenbot (дата звернення 11.06.2022).
18. «Gekko» [Електронний ресурс] – Режим доступу до ресурсу: https://gekko.wizb.it/ (дата звернення 11.06.2022).
19. «Cryptotrader - build your own trading bot in minutes» [Електронний ресурс] – Режим доступу до ресурсу: https://cryptotrader.org/ (дата звернення 11.06.2022).
20. «3commas» [Електронний ресурс] – Режим доступу до ресурсу: https://3commas.io (дата звернення 11.06.2022).
21. «Trading Strategy», J. Chen [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/t/trading-strategy.asp (дата звернення 11.06.2022).

22. «Technical Indicator», J. Chen [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/t/technicalindicator.asp (дата звернення 11.06.2022).
23. «Take-Profit Order - T/P», J. Chen [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/t/take-profitorder.asp (дата звернення 11.06.2022).
24. «Stop-Loss Order», M. Kramer [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/s/stop-lossorder.asp (дата звернення 11.06.2022).
25. «Legitimate volume on spot exchanges», THE BLOCK [Електронний ресурс] – Режим доступу до ресурсу: https://www.tbstat.com/wp/uploads/2021/05/12.-Legitimate-volume-on-spot-exchanges-1.png (дата звернення 11.06.2022).
26. «Top Cryptocurrency Spot Exchanges», M. Kramer [Електронний ресурс] – Режим доступу до ресурсу: https://coinmarketcap.com (дата звернення 11.06.2022).
27. «Binance API» [Електронний ресурс] – Режим доступу до ресурсу: https://www.binance.com/en/binance-api (дата звернення 11.06.2022).
28. «What is Python?», Teradata [Електронний ресурс] – Режим доступу до ресурсу: https://www.teradata.com/Glossary/What-is-Python (дата звернення 11.06.2022).
29. «TIOBE Index», TIOBE organization [Електронний ресурс] – Режим доступу до ресурсу: https://www.tiobe.com/tiobe-index/ (дата звернення 11.06.2022).

ДОДАТОК 1

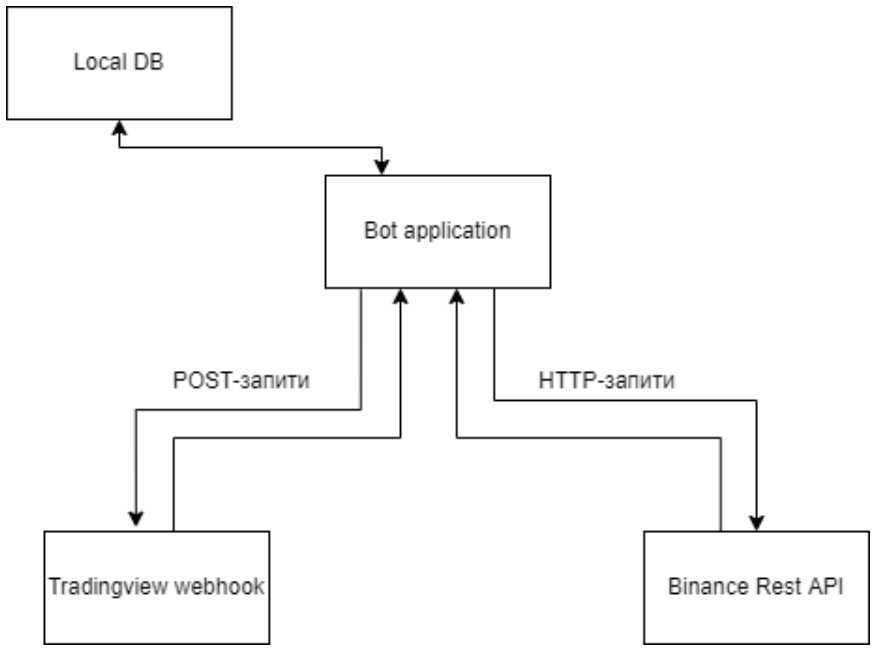
Еволюційні алгоритми глобальної пошукової оптимізації

Структурна схема системи

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2022 р



					ІАЛЦ.467200.004 Д1		
		№ докум.	Підпис	Дата			
Розробив		Бурлака Д. С.			**Транзакційний бот для спотової торгівлі на криптобіржі** Структурна схема системи	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Калюжний О.О.					
						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83	
Н. Контр.		Сімоненко В. П.					
Затвердив							

ДОДАТОК 2

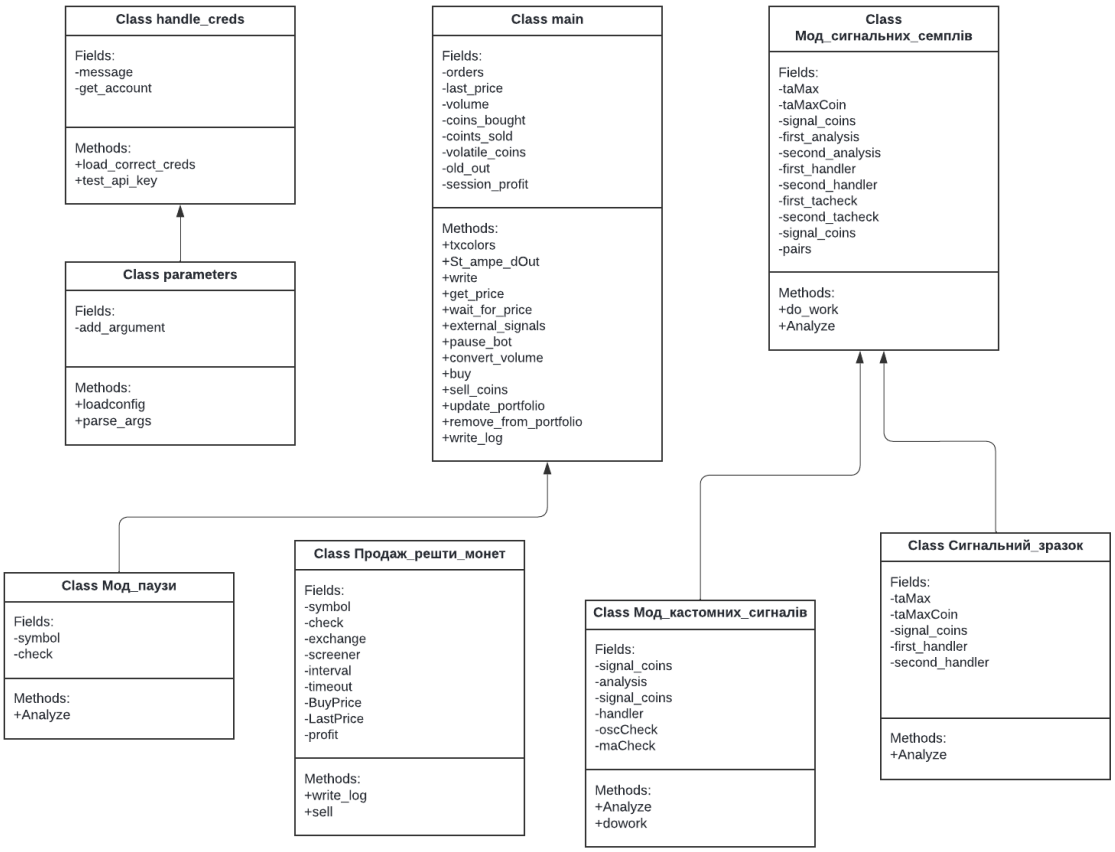
Еволюційні алгоритми глобальної пошукової оптимізації

Функціональна схема (діаграма класів)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2022 р



					ІАЛЦ.467200.005 Д2	
		№ докум.	Підпис	Дата		
Розробив		Бурлака Д. С.			**Транзакційний бот для спотової торгівлі на криптобіржі** Функціональна схема (діаграма класів)	Літ. / Аркуш / Аркушів: 1 / 1
Перевірив		Калюжний О. О.				
Н. Контр.		Сімоненко В. П.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83
Затвердив						

ДОДАТОК 3

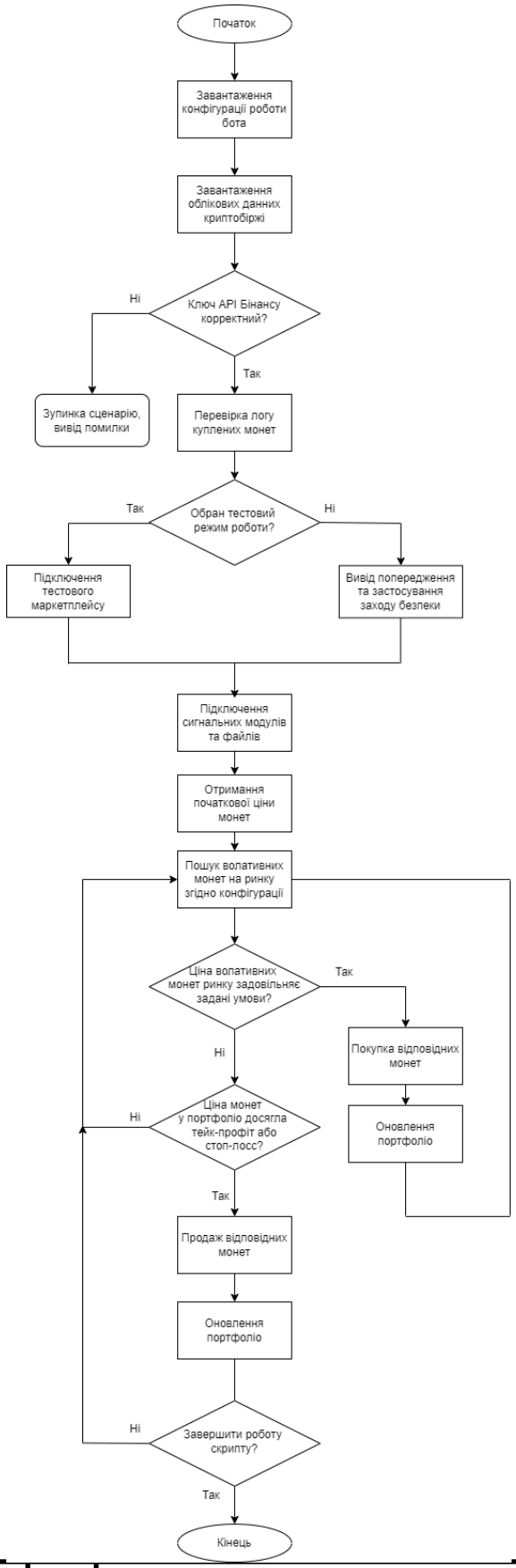
Еволюційні алгоритми глобальної пошукової оптимізації

Алгоритм дій програмного забезпечення

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2022 р



					ІАЛЦ.467200.006 Д3	
		№ докум.	Підпис	Дата		
Розробив		Бурлака Д. С.			**Транзакційний бот для спотової торгівлі на криптобіржі** Алгоритм дій програмного забезпечення	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Калюжний О.О.				
Н. Контр.		Сімоненко В. П.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83
Затвердив						

ДОДАТОК 4

Еволюційні алгоритми глобальної пошукової оптимізації

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 41

Київ 2022 р

```
import glob
import importlib
import os
import sys
import threading
import glob
import importlib
import os
import sys
import threading

from colorama import init

init()

from binance.client import Client
from binance.exceptions import BinanceAPIException
from requests.exceptions import ReadTimeout, ConnectionError

from datetime import datetime, timedelta
import time


import json

# Завантажуємо допоміжні модулі
from helpers.parameters import (
    parse_args, load_config
)

# Завантажуємо модулі кредів
from helpers.handle_creds import (
    load_correct_creds, test_api_key
)



class txcolors:
    BUY = '\033[92m'
    WARNING = '\033[93m'
    SELL_LOSS = '\033[91m'
    SELL_PROFIT = '\033[32m'
    DIM = '\033[2m\033[35m'
    DEFAULT = '\033[39m'


# Трекаємо tracks прибуток/збиток кожної сесії
global session_profit
session_profit = 0


# Вивід з мітками часу
old_out = sys.stdout
class St_ampe_dOut:
    nl = True
    def write(self, x):
        if x == '\n':
            old_out.write(x)
            self.nl = True
        elif self.nl:

old_out.write(f'{txcolors.DIM}[{str(datetime.now().replace(microsecond=0))}]{txcolors.DEFAUL
T} {x}')
            self.nl = False
```

```
        else:
            old_out.write(x)

    def flush(self):
        pass

sys.stdout = St_ampe_dOut()


def get_price(add_to_historical=True):
    """Повертає поточну ціну всіх монет на Binance"""

    global historical_prices, hsp_head

    initial_price = {}
    prices = client.get_all_tickers()

    for coin in prices:

        if CUSTOM_LIST:
            if any(item + PAIR_WITH == coin['symbol'] for item in tickers) and all(item not
in coin['symbol'] for item in FIATS):
                initial_price[coin['symbol']] = { 'price': coin['price'], 'time':
datetime.now()}
        else:
            if PAIR_WITH in coin['symbol'] and all(item not in coin['symbol'] for item in
FIATS):
                initial_price[coin['symbol']] = { 'price': coin['price'], 'time':
datetime.now()}

    if add_to_historical:
        hsp_head += 1

        if hsp_head == RECHECK_INTERVAL:
            hsp_head = 0

        historical_prices[hsp_head] = initial_price

    return initial_price


def wait_for_price():
    """Визиває початкову ціну та гарантує, що повторне зчитування поточної ціни було
вчасне"""

    global historical_prices, hsp_head, volatility_cooloff

    volatile_coins = {}
    coins_up = 0
    coins_down = 0
    coins_unchanged = 0

    pause_bot()

    if historical_prices[hsp_head]['BNB' + PAIR_WITH]['time'] > datetime.now() -
timedelta(minutes=float(TIME_DIFFERENCE / RECHECK_INTERVAL)):

        # Переконуємось що процес на паузі рівно стільки часу, скільки потрібно
        time.sleep((timedelta(minutes=float(TIME_DIFFERENCE / RECHECK_INTERVAL)) -
(datetime.now() - historical_prices[hsp_head]['BNB' + PAIR_WITH]['time'])).total_seconds())

    print(f'Працюю...Прибуток за сесію: {session_profit:.2f} Усього: ${(QUANTITY *
session_profit)/100:.2f}')

    # Отримуємо останні ціни
```

```
    get_price()

    # Розраховуємо різницю в цінах
    for coin in historical_prices[hsp_head]:

        # Мінімальні та максимальні ціни за певний період часу
        min_price = min(historical_prices, key = lambda x: float("inf") if x is None else
float(x[coin]['price']))
        max_price = max(historical_prices, key = lambda x: -1 if x is None else
float(x[coin]['price']))

        threshold_check = (-1.0 if min_price[coin]['time'] > max_price[coin]['time'] else
1.0) * (float(max_price[coin]['price']) - float(min_price[coin]['price'])) /
float(min_price[coin]['price']) * 100

        # Кожна монета з більшим прибутком, ніж наша CHANGE_IN_PRICE, додається до
volatile_coins dict, якщо не досягнуто менше ніж MAX_COINS
        if threshold_check > CHANGE_IN_PRICE:
            coins_up +=1

            if coin not in volatility_cooloff:
                volatility_cooloff[coin] = datetime.now() -
timedelta(minutes=TIME_DIFFERENCE)

            # Включаємо монету як нестабільну, лише якщо вона не була підібрана протягом
останніх TIME_DIFFERENCE хвилин
            if datetime.now() >= volatility_cooloff[coin] +
timedelta(minutes=TIME_DIFFERENCE):
                volatility_cooloff[coin] = datetime.now()

                if len(coins_bought) + len(volatile_coins) < MAX_COINS or MAX_COINS == 0:
                    volatile_coins[coin] = round(threshold_check, 3)
                    print(f'{coin} подорощала на {volatile_coins[coin]}% протягом останніх
{TIME_DIFFERENCE} хвилин, обчислюючи обсяг у {PAIR_WITH}')

                else:
                    print(f'{txcolors.WARNING}{coin} подорощала на {round(threshold_check,
3)}% протягом останніх {TIME_DIFFERENCE} хвилин, але вже досягнуто максимальну к-ть
монет{txcolors.DEFAULT}')

        elif threshold_check < CHANGE_IN_PRICE:
            coins_down +=1

        else:
            coins_unchanged +=1


    # Зовнішні сигнали
    externals = external_signals()
    exnumber = 0

    for excoin in externals:
        if excoin not in volatile_coins and excoin not in coins_bought and \
                (len(coins_bought) + exnumber + len(volatile_coins)) < MAX_COINS:
            volatile_coins[excoin] = 1
            exnumber +=1
            print(f'Зовнішній сигнал отримано {excoin}, обчислюючи обсяг у {PAIR_WITH}')

    return volatile_coins, len(volatile_coins), historical_prices[hsp_head]


def external_signals():
    external_list = {}
    signals = {}
```

```
    # Перевіряємо каталог і завантажуємо пари з файлів у external_list
    signals = glob.glob("signals/*.exs")
    for filename in signals:
        for line in open(filename):
            symbol = line.strip()
            external_list[symbol] = symbol
        try:
            os.remove(filename)
        except:
            if DEBUG: print(f'{txcolors.WARNING}Не вдалося видалити зовнішній сигнальний
файл{txcolors.DEFAULT}')

    return external_list


def pause_bot():
    """Призупиняємо скрипт, коли зовнішні індикатори виявлять ведмежий тренд на ринку"""
    global bot_paused, session_profit, hsp_head

    # Починаємо рахувати скільки часу бот буде призупинений
    start_time = time.perf_counter()

    while os.path.isfile("signals/paused.exc"):

        if bot_paused == False:
            print(f'{txcolors.WARNING}Призупиняю купівлю через зміну ринкових умов, стоп-
лосс і тейк-профіт продовжать працювати...{txcolors.DEFAULT}')
            bot_paused = True

        # Функція продажу працює навіть під час призупинення
        coins_sold = sell_coins()
        remove_from_portfolio(coins_sold)
        get_price(True)

        # Зупиняємо тут
        if hsp_head == 1: print(f'Призупинено...Прибуток сесії: {session_profit:.2f} Усього
${(QUANTITY * session_profit)/100:.2f}')
        time.sleep((TIME_DIFFERENCE * 60) / RECHECK_INTERVAL)

    else:
        # Припиняємо відлік часу паузи
        stop_time = time.perf_counter()
        time_elapsed = timedelta(seconds=int(stop_time-start_time))

        # Відновляємо роботу бота і встановлюємо для pause_bot значення False
        if  bot_paused == True:
            print(f'{txcolors.WARNING}Відновлюю покупку через зміну ринкових умов, загальний
час сну: {time_elapsed}{txcolors.DEFAULT}')
            bot_paused = False

    return


def convert_volume():
    """Перетворює обсяг, наведений у QUANTITY, із USDT в обсяг кожної монети"""

    volatile_coins, number_of_coins, last_price = wait_for_price()
    lot_size = {}
    volume = {}

    for coin in volatile_coins:

        # Знаходимо правильний розмір кроку для кожної монети (у бітка 6 знаків після коми,
у інших монет - менше)
```

```
        try:
            info = client.get_symbol_info(coin)
            step_size = info['filters'][2]['stepSize']
            lot_size[coin] = step_size.index('1') - 1

            if lot_size[coin] < 0:
                lot_size[coin] = 0

        except:
            pass

        # Обчислюємо обсяг монети від QUANTITY в USDT (за замовчуванням)
        volume[coin] = float(QUANTITY / float(last_price[coin]['price']))

        # Визначаємо обсяг з правильним розміром кроку
        if coin not in lot_size:
            volume[coin] = float('{:.1f}'.format(volume[coin]))

        else:
            # якщо розмір лоту має 0 знаків після коми, зробіть обсяг цілим числом
            if lot_size[coin] == 0:
                volume[coin] = int(volume[coin])
            else:
                volume[coin] = float('{:.{}f}'.format(volume[coin], lot_size[coin]))

    return volume, last_price


def buy():
    """Розміщує ринкові ордери на покупку для кожної знайденої нестабільної монети"""
    volume, last_price = convert_volume()
    orders = {}

    for coin in volume:

        # Купуємо лише тоді, коли немає активних торгів з монетою
        if coin not in coins_bought:
            print(f"{txcolors.BUY}Готуємося до покупки {volume[coin]}
{coin}{txcolors.DEFAULT}")

            if TEST_MODE:
                orders[coin] = [{
                    'symbol': coin,
                    'orderId': 0,
                    'time': datetime.now().timestamp()
                }]

                # Лог угоди
                if LOG_TRADES:
                    write_log(f"Покупка: {volume[coin]} {coin} -
{last_price[coin]['price']}")

                continue

            # Намагаємося створити справжнє замовлення, якщо тестове замовлення не викликало
винятку
            try:
                buy_limit = client.create_order(
                    symbol = coin,
                    side = 'BUY',
                    type = 'MARKET',
                    quantity = volume[coin]
                )

            # Обробка помилки, якщо позицію неможливо розмістити
```

```
            except Exception as e:
                print(e)

            # Запускаємо блок else, якщо позицію було розміщено, і повертаємо інформацію про
замовлення
            else:
                orders[coin] = client.get_all_orders(symbol=coin, limit=1)

                # Binance іноді повертає порожній список, чекаємо поки binance не поверне
замовлення
                while orders[coin] == []:
                    print('Binance повільно повертає замовлення, знову викликаю API...')

                    orders[coin] = client.get_all_orders(symbol=coin, limit=1)
                    time.sleep(1)

                else:
                    print('Замовлення повернуто, зберігаю у файлі')

                    # Лог угоди
                    if LOG_TRADES:
                        write_log(f"Покупка: {volume[coin]} {coin} -
{last_price[coin]['price']}")


        else:
            print(f'Сигнал виявлено, але вже є активна торгівля {coin}')

    return orders, last_price, volume


def sell_coins():
    """продає монети, які досягли порогу стоп-лосс або тейк-профіт"""

    global hsp_head, session_profit

    last_price = get_price(False)
    coins_sold = {}

    for coin in list(coins_bought):
        # Визначаємо стоп-лосс і тейк-профіт
        TP = float(coins_bought[coin]['bought_at']) +
(float(coins_bought[coin]['bought_at']) * coins_bought[coin]['take_profit']) / 100
        SL = float(coins_bought[coin]['bought_at']) +
(float(coins_bought[coin]['bought_at']) * coins_bought[coin]['stop_loss']) / 100


        LastPrice = float(last_price[coin]['price'])
        BuyPrice = float(coins_bought[coin]['bought_at'])
        PriceChange = float((LastPrice - BuyPrice) / BuyPrice * 100)

        # Переконуємося, що ціна вище тейк-профіту і повторно відповідно відкориговуємо SL і
TP, якщо використовується пробний стоп-лосс
        if LastPrice > TP and USE_TRAILING_STOP_LOSS:

            # Збільшуємо тейк-профіт на TRAILING_TAKE_PROFIT
            coins_bought[coin]['take_profit'] = PriceChange + TRAILING_TAKE_PROFIT
            coins_bought[coin]['stop_loss'] = coins_bought[coin]['take_profit'] -
TRAILING_STOP_LOSS
            if DEBUG: print(f"{coin} Тейк-профіт досягнуто, коригуємо наступне його значення
{coins_bought[coin]['take_profit']:.2f}  та стоп-лосс {coins_bought[coin]['stop_loss']:.2f}
відповідно до фіксованого прибутку")
            continue
```

```
        # Переконуємося, що ціна нижче стоп-лоссу або вище тейк-профіту (якщо трейлінг стоп-
лосс не використовується) і продаєм, якщо це так
        if LastPrice < SL or LastPrice > TP and not USE_TRAILING_STOP_LOSS:
            print(f"{txcolors.SELL_PROFIT if PriceChange >= 0. else txcolors.SELL_LOSS}Тейк-
профіт або стоп-лосс досягнено, продаю {coins_bought[coin]['volume']} {coin} - {BuyPrice} -
{LastPrice} : {PriceChange-(TRADING_FEE*2):.2f}% Est:${(QUANTITY*(PriceChange-
(TRADING_FEE*2)))/100:.2f}{txcolors.DEFAULT}")

            # Намагаємося створити справжню згоду
            try:

                if not TEST_MODE:
                    sell_coins_limit = client.create_order(
                        symbol = coin,
                        side = 'SELL',
                        type = 'MARKET',
                        quantity = coins_bought[coin]['volume']

                    )

            # Обробляємо помилку, якщо позицію неможливо розмістити
            except Exception as e:
                print(e)

            # Запускаємо блок else, якщо монета була продана і створюємо словник для кожної
проданої монети
            else:
                coins_sold[coin] = coins_bought[coin]

                # Запобігаємо покупку системою цієї монети протягом наступних
TIME_DIFFERENCE хвилин
                volatility_cooloff[coin] = datetime.now()

                # Лог згоди
                if LOG_TRADES:
                    profit = ((LastPrice - BuyPrice) * coins_sold[coin]['volume'])* (1-
(TRADING_FEE*2))
                    write_log(f"Продано: {coins_sold[coin]['volume']} {coin} - {BuyPrice} -
{LastPrice} Прибуток: {profit:.2f} {PriceChange-(TRADING_FEE*2):.2f}%")
                    session_profit=session_profit + (PriceChange-(TRADING_FEE*2))
            continue

        # Виводимо раз на TIME_DIFFERENCE
        if hsp_head == 1:
            if len(coins_bought) > 0:
                print(f'Тейк-профіт або стоп-лосс ще не досягнено, не продаю {coin} поки що
{BuyPrice} - {LastPrice} : {txcolors.SELL_PROFIT if PriceChange >= 0. else
txcolors.SELL_LOSS}{PriceChange-(TRADING_FEE*2):.2f}% Est:${(QUANTITY*(PriceChange-
(TRADING_FEE*2)))/100:.2f}{txcolors.DEFAULT}')

    if hsp_head == 1 and len(coins_bought) == 0: print(f'Немає жодних монет у наявності')

    return coins_sold


def update_portfolio(orders, last_price, volume):
    """Додає кожну куплену монету до портфоліо для подальшого відстеження або продажу"""
    if DEBUG: print(orders)
    for coin in orders:

        coins_bought[coin] = {
            'symbol': orders[coin][0]['symbol'],
            'orderid': orders[coin][0]['orderId'],
            'timestamp': orders[coin][0]['time'],
            'bought_at': last_price[coin]['price'],
```

```
            'volume': volume[coin],
            'stop_loss': -STOP_LOSS,
            'take_profit': TAKE_PROFIT,
            }

        # Зберігаємо монети у json файл у тому ж каталозі
        with open(coins_bought_file_path, 'w') as file:
            json.dump(coins_bought, file, indent=4)

        print(f'Угода з ID {orders[coin][0]["orderId"]} розміщена та збережена у файлі')


def remove_from_portfolio(coins_sold):
    '''Вилучає з портфеля монети, продані через стоп-лосс або тейк-профіт'''
    for coin in coins_sold:
        coins_bought.pop(coin)

    with open(coins_bought_file_path, 'w') as file:
        json.dump(coins_bought, file, indent=4)


def write_log(logline):
    timestamp = datetime.now().strftime("%d/%m %H:%M:%S")
    with open(LOG_FILE,'a+') as f:
        f.write(timestamp + ' ' + logline + '\n')

if __name__ == '__main__':

    # Завантажуємо аргументи, а потім пропарсити налаштування
    args = parse_args()
    mymodule = {}

    # Встановлюємо значення false на початку
    global bot_paused
    bot_paused = False

    DEFAULT_CONFIG_FILE = 'Конфіг.yml'
    DEFAULT_CREDS_FILE = 'Креди.yml'

    config_file = args.config if args.config else DEFAULT_CONFIG_FILE
    creds_file = args.creds if args.creds else DEFAULT_CREDS_FILE
    parsed_config = load_config(config_file)
    parsed_creds = load_config(creds_file)

    # За замовчуванням налагодження відключено
    DEBUG = False

    # Завантажуємо системні змінні
    TEST_MODE = parsed_config['script_options']['TEST_MODE']
    LOG_TRADES = parsed_config['script_options'].get('LOG_TRADES')
    LOG_FILE = parsed_config['script_options'].get('LOG_FILE')
    DEBUG_SETTING = parsed_config['script_options'].get('DEBUG')
    TLD_USER = parsed_config['script_options'].get('TLD_USER')

    # Завантажуємо змінні для торгівлі
    PAIR_WITH = parsed_config['trading_options']['PAIR_WITH']
    QUANTITY = parsed_config['trading_options']['QUANTITY']
    MAX_COINS = parsed_config['trading_options']['MAX_COINS']
    FIATS = parsed_config['trading_options']['FIATS']
    TIME_DIFFERENCE = parsed_config['trading_options']['TIME_DIFFERENCE']
    RECHECK_INTERVAL = parsed_config['trading_options']['RECHECK_INTERVAL']
    CHANGE_IN_PRICE = parsed_config['trading_options']['CHANGE_IN_PRICE']
    STOP_LOSS = parsed_config['trading_options']['STOP_LOSS']
    TAKE_PROFIT = parsed_config['trading_options']['TAKE_PROFIT']
    CUSTOM_LIST = parsed_config['trading_options']['CUSTOM_LIST']
```

```
    TICKERS_LIST = parsed_config['trading_options']['TICKERS_LIST']
    USE_TRAILING_STOP_LOSS = parsed_config['trading_options']['USE_TRAILING_STOP_LOSS']
    TRAILING_STOP_LOSS = parsed_config['trading_options']['TRAILING_STOP_LOSS']
    TRAILING_TAKE_PROFIT = parsed_config['trading_options']['TRAILING_TAKE_PROFIT']
    TRADING_FEE = parsed_config['trading_options']['TRADING_FEE']
    SIGNALLING_MODULES = parsed_config['trading_options']['SIGNALLING_MODULES']
    if DEBUG_SETTING or args.debug:
        DEBUG = True

    # Завантажуємо креди для правильного середовища
    access_key, secret_key = load_correct_creds(parsed_creds)

    if DEBUG:
        print(f'Завантажена конфігурація нижче\n{json.dumps(parsed_config, indent=4)}')
        print(f'Облікові дані завантажено з {creds_file}')


    # Проходимо аутентифікацію з клієнтом та переконуємося, що ключ API справний
    if TLD_USER:
        client = Client(access_key, secret_key, tld='us')
    else:
        client = Client(access_key, secret_key)

    # Якщо ключ API несправний, то зупиняємо запуск сценарію
    api_ready, msg = test_api_key(client, BinanceAPIException)
    if api_ready is not True:
       exit(f'{txcolors.SELL_LOSS}{msg}{txcolors.DEFAULT}')

    # Використовуємо символи CUSTOM_LIST, якщо для CUSTOM_LIST встановлено значення True
    if CUSTOM_LIST: tickers=[line.strip() for line in open(TICKERS_LIST)]

    # Спробуємо завантажити всі куплені ботом монети, якщо файл існує і не порожній
    coins_bought = {}

    # Шлях до збереженого файлу coins_bought
    coins_bought_file_path = 'Куплені_монети.json'

    # Перевіряємо вікно цін, циклічна черга
    historical_prices = [None] * (TIME_DIFFERENCE * RECHECK_INTERVAL)
    hsp_head = -1

    # Забороняємо включення монети в volatile_coins, якщо вона вже з’явилася там менше ніж
TIME_DIFFERENCE хвилин тому
    volatility_cooloff = {}

    # Використовуємо окремі файли для тестування та реальної торгівлі
    if TEST_MODE:
        coins_bought_file_path = 'test_' + coins_bought_file_path

    # Завантажуємо json файл coins_bought якщо він існує і не порожній
    if os.path.isfile(coins_bought_file_path) and os.stat(coins_bought_file_path).st_size!=
0:
        with open(coins_bought_file_path) as file:
                coins_bought = json.load(file)

    print('Натисніть Ctrl-C, щоб зупинити сценарій')

    if not TEST_MODE:
        if not args.notimeout:
            print('ПОПЕРЕДЖЕННЯ: Використовується мережа Mainnet та реальний рахунок.
Зачекайте 30 секунд як захід безпеки')
            time.sleep(30)

    signals = glob.glob("signals/*.exs")
    for filename in signals:
```

```
        for line in open(filename):
            try:
                os.remove(filename)
            except:
                if DEBUG: print(f'{txcolors.WARNING}Не вдалося видалити зовнішній сигнальний
файл {filename}{txcolors.DEFAULT}')

    if os.path.isfile("signals/paused.exc"):
        try:
            os.remove("signals/paused.exc")
        except:
            if DEBUG: print(f'{txcolors.WARNING}Не вдалося видалити зовнішній сигнальний
файл {filename}{txcolors.DEFAULT}')

    # завантаження модулів сигналів
    try:
        if len(SIGNALLING_MODULES) > 0:
            for module in SIGNALLING_MODULES:
                print(f'Запуск {module}')
                mymodule[module] = importlib.import_module(module)
                t = threading.Thread(target=mymodule[module].do_work, args=())
                t.daemon = True
                t.start()
                time.sleep(2)
        else:
            print(f'Немає модулів для завантаження {SIGNALLING_MODULES}')
    except Exception as e:
        print(e)

    # Знаходимо початкові ціни
    get_price()
    READ_TIMEOUT_COUNT=0
    CONNECTION_ERROR_COUNT = 0
    while True:
        try:
            orders, last_price, volume = buy()
            update_portfolio(orders, last_price, volume)
            coins_sold = sell_coins()
            remove_from_portfolio(coins_sold)
        except ReadTimeout as rt:
            READ_TIMEOUT_COUNT += 1
            print(f'{txcolors.WARNING}Отримано помилку тайм-ауту від Binance. Переходимо до
повторного циклу. Поточна кількість: {READ_TIMEOUT_COUNT}\n{rt}{txcolors.DEFAULT}')
        except ConnectionError as ce:
            CONNECTION_ERROR_COUNT +=1
            print(f'{txcolors.WARNING}Отримано помилку тайм-ауту від Binance. Переходимо до
повторного циклу. Поточна кількість: {CONNECTION_ERROR_COUNT}\n{ce}{txcolors.DEFAULT}')

from tradingview_ta import TA_Handler, Interval, Exchange
import os
import time
import threading

OSC_INDICATORS = ['MACD', 'Stoch.RSI', 'Mom'] # Індикатори для використання в аналізі
осцилятора
OSC_THRESHOLD = 2 # Має бути менше або дорівнювати кількості елементів у OSC_INDICATORS
MA_INDICATORS = ['EMA10', 'EMA20'] # Індикатори для використання в аналізі ковзних середніх
MA_THRESHOLD = 2 # Має бути менше або дорівнювати кількості елементів у MA_INDICATORS
INTERVAL = Interval.INTERVAL_5_MINUTES #Терміни для аналізу

EXCHANGE = 'BINANCE'
SCREENER = 'CRYPTO'
PAIR_WITH = 'USDT'
TICKERS = 'Сигнальний_зразок.txt'
TIME_TO_WAIT = 4 # Хвилини очікування між аналізами
```

```
FULL_LOG = False # Список результатів аналізу на консоль

def analyze(pairs):
    signal_coins = {}
    analysis = {}
    handler = {}

    if os.path.exists('signals/custsignalmod.exs'):
        os.remove('signals/custsignalmod.exs')

    for pair in pairs:
        handler[pair] = TA_Handler(
            symbol=pair,
            exchange=EXCHANGE,
            screener=SCREENER,
            interval=INTERVAL,
            timeout= 10)

    for pair in pairs:
        try:
            analysis = handler[pair].get_analysis()
        except Exception as e:
            print("Зразок сигналу:")
            print("Виняток:")
            print(e)
            print (f'Монета: {pair}')
            print (f'обробник: {handler[pair]}')

        oscCheck=0
        maCheck=0
        for indicator in OSC_INDICATORS:
            if analysis.oscillators ['COMPUTE'][indicator] == 'BUY': oscCheck +=1

        for indicator in MA_INDICATORS:
            if analysis.moving_averages ['COMPUTE'][indicator] == 'BUY': maCheck +=1


        if FULL_LOG:
            print(f'Сигнальний_бот:{pair} Осцилятори:{oscCheck}/{len(OSC_INDICATORS)} Ковзні
середні:{maCheck}/{len(MA_INDICATORS)}')

        if oscCheck >= OSC_THRESHOLD and maCheck >= MA_THRESHOLD:
                signal_coins[pair] = pair
                print(f'Сигнальний_бот: Виявлено сигнал у {pair} на
{oscCheck}/{len(OSC_INDICATORS)} осцилятори та {maCheck}/{len(MA_INDICATORS)} ковзні
середні.')
                with open('signals/custsignalmod.exs','a+') as f:
                    f.write(pair + '\n')

    return signal_coins

def do_work():
    signal_coins = {}
    pairs = {}

    pairs=[line.strip() for line in open(TICKERS)]
    for line in open(TICKERS):
        pairs=[line.strip() + PAIR_WITH for line in open(TICKERS)]

    while True:
        if not threading.main_thread().is_alive(): exit()
        print(f'Сигнальний_бот: Аналіз {len(pairs)} монет')
        signal_coins = analyze(pairs)
```

```
        print(f'Сигнальний_бот: {len(signal_coins)} монет вище
{OSC_THRESHOLD}/{len(OSC_INDICATORS)} осцилятори та {MA_THRESHOLD}/{len(MA_INDICATORS)}
ковзні середні Очікую {TIME_TO_WAIT} хвилин для наступного аналізу.')
        time.sleep((TIME_TO_WAIT*60))
from tradingview_ta import TA_Handler, Interval, Exchange
import os
import time
import threading

INTERVAL = Interval.INTERVAL_1_MINUTE #Терміни для аналізу

EXCHANGE = 'BINANCE'
SCREENER = 'CRYPTO'
SYMBOL = 'BTCUSDT'
THRESHOLD = 7 # 7 з 15 індикаторів, що вказують на продаж
TIME_TO_WAIT = 1 # Хвилини очікування між аналізами
FULL_LOG = False # Список результатів аналізу на консоль

def analyze():
    analysis = {}
    handler = {}

    handler = TA_Handler(
            symbol=SYMBOL,
            exchange=EXCHANGE,
            screener=SCREENER,
            interval=INTERVAL,
            timeout= 10)

    try:
        analysis = handler.get_analysis()
    except Exception as e:
        print("Мод_паузи:")
        print("Виняток:")
        print(e)

    ma_sell = analysis.moving_averages['SELL']
    if ma_sell >= THRESHOLD:
        paused = True
        print(f'Мод_паузи: Ринок виглядає не дуже добре, призупиняю купівлю у режимі
{ma_sell}/{THRESHOLD}. Очікую {TIME_TO_WAIT} хвилин для наступної перевірки ринку')
    else:
        print(f'Мод_паузи: Ринок виглядає добре, працюю у режимі {ma_sell}/{THRESHOLD}.
Очікую {TIME_TO_WAIT} хвилин для наступної перевірки ринку ')
        paused = False

    return paused

def do_work():

    while True:
        if not threading.main_thread().is_alive(): exit()
        # print(f'Мод паузи: Fetching market state')
        paused = analyze()
        if paused:
            with open('signals/paused.exc','a+') as f:
                f.write('yes')
        else:
            if os.path.isfile("signals/paused.exc"):
                os.remove('signals/paused.exc')

        time.sleep((TIME_TO_WAIT*60))
from tradingview_ta import TA_Handler, Interval, Exchange

import os
```

```
import time

MY_EXCHANGE = 'BINANCE'
MY_SCREENER = 'CRYPTO'
MY_FIRST_INTERVAL = Interval.INTERVAL_1_MINUTE
MY_SECOND_INTERVAL = Interval.INTERVAL_5_MINUTES
TA_BUY_THRESHOLD = 18 # Скільки з 26 індикаторів мають вказувати на покупку
PAIR_WITH = 'USDT'
TICKERS = 'Сигнальний_зразок.txt'
TIME_TO_WAIT = 4 # Хвилини очікування між аналізами
FULL_LOG = False # Список результатів аналізу на консоль

def analyze(pairs):
    taMax = 0
    taMaxCoin = 'none'
    signal_coins = {}
    first_analysis = {}
    second_analysis = {}
    first_handler = {}
    second_handler = {}
    if os.path.exists('signals/signalsample.exs'):
        os.remove('signals/signalsample.exs')

    for pair in pairs:
        first_handler[pair] = TA_Handler(
            symbol=pair,
            exchange=MY_EXCHANGE,
            screener=MY_SCREENER,
            interval=MY_FIRST_INTERVAL,
            timeout= 10
        )
        second_handler[pair] = TA_Handler(
            symbol=pair,
            exchange=MY_EXCHANGE,
            screener=MY_SCREENER,
            interval=MY_SECOND_INTERVAL,
            timeout= 10
        )

    for pair in pairs:

        try:
            first_analysis = first_handler[pair].get_analysis()
            second_analysis = second_handler[pair].get_analysis()
        except Exception as e:
                    print("Мод_сигнальних_семплів:")
                    print("Виняток:")
                    print(e)
                    print (f'Монета: {pair}')
                    print (f'Перший обробник: {first_handler[pair]}')
                    print (f'Другий обробник: {second_handler[pair]}')
                    tacheckS = 0

        first_tacheck = first_analysis.summary['BUY']
        second_tacheck = second_analysis.summary['BUY']
        if FULL_LOG:
            print(f'Мод_сигнальних_семплів:{pair} Перший {first_tacheck} Другий
{second_tacheck}')

        if first_tacheck > taMax:
                taMax = first_tacheck
                taMaxCoin = pair
        if first_tacheck >= TA_BUY_THRESHOLD and second_tacheck >= TA_BUY_THRESHOLD:
                signal_coins[pair] = pair
```

```
                print(f'Бот_сигнальних_семплів: Виявлено сигнал у {pair}')
                with open('signals/signalsample.exs','a+') as f:
                    f.write(pair + '\n')
    print(f'Мод_сигнальних_семплів: Максимальний сигнал у {taMaxCoin} у к-ті {taMax}
(шорт)')

    return signal_coins

def do_work():
    signal_coins = {}
    pairs = {}

    pairs=[line.strip() for line in open(TICKERS)]
    for line in open(TICKERS):
        pairs=[line.strip() + PAIR_WITH for line in open(TICKERS)]

    while True:
        print(f'Мод_сигнальних_семплів: Аналіз {len(pairs)} монет')
        signal_coins = analyze(pairs)
        if len(signal_coins) == 0:
            print(f'Мод_сигнальних_семплів: Немає монет вище порогу {TA_BUY_THRESHOLD} в
обох таймфреймах. Очікую {TIME_TO_WAIT} хвилин для наступного аналізу')
        else:
            print(f'Мод_сигнальних_семплів: {len(signal_coins)} монет вище порогу
{TA_BUY_THRESHOLD} в обох таймфреймах. Очікую {TIME_TO_WAIT} хвилин для наступного аналізу')

        time.sleep((TIME_TO_WAIT*60))
from tradingview_ta import TA_Handler, Interval, Exchange
import os
import time

MY_EXCHANGE = 'BINANCE'
MY_SCREENER = 'CRYPTO'
MY_FIRST_INTERVAL = Interval.INTERVAL_1_MINUTE
MY_SECOND_INTERVAL = Interval.INTERVAL_5_MINUTES
TA_BUY_THRESHOLD = 18 # Скільки з 26 індикаторів мають вказувати на покупку
PAIR_WITH = 'USDT'
TICKERS = 'Сигнальний_зразок.txt'
TIME_TO_WAIT = 4 # Хвилини очікування між аналізами
FULL_LOG = False # Список результатів аналізу на консоль

def analyze(pairs):
    taMax = 0
    taMaxCoin = 'none'
    signal_coins = {}
    first_analysis = {}
    second_analysis = {}
    first_handler = {}
    second_handler = {}
    if os.path.exists('signals/signalsample.exs'):
        os.remove('signals/signalsample.exs')

    for pair in pairs:
        first_handler[pair] = TA_Handler(
            symbol=pair,
            exchange=MY_EXCHANGE,
            screener=MY_SCREENER,
            interval=MY_FIRST_INTERVAL,
            timeout= 10
        )
        second_handler[pair] = TA_Handler(
            symbol=pair,
            exchange=MY_EXCHANGE,
            screener=MY_SCREENER,
            interval=MY_SECOND_INTERVAL,
```

```
            timeout= 10
        )

    for pair in pairs:

        try:
            first_analysis = first_handler[pair].get_analysis()
            second_analysis = second_handler[pair].get_analysis()
        except Exception as e:
                    print("Виняток:")
                    print(e)
                    print (f'Монета: {pair}')
                    print (f'Перший обробник: {first_handler[pair]}')
                    print (f'Другий обробник: {second_handler[pair]}')
                    tacheckS = 0

        first_tacheck = first_analysis.summary['BUY']
        second_tacheck = second_analysis.summary['BUY']
        if FULL_LOG:
            print(f'{pair} Перша {first_tacheck} Друга {second_tacheck}')
        else:
            print(".", end = '')

        if first_tacheck > taMax:
                taMax = first_tacheck
                taMaxCoin = pair
        if first_tacheck >= TA_BUY_THRESHOLD and second_tacheck >= TA_BUY_THRESHOLD:
                signal_coins[pair] = pair
                print("")
                print(f'Виявлено сигнал у {pair}')
                with open('signals/signalsample.exs','a+') as f:
                    f.write(pair + '\n')
    print("")
    print(f'Максимальний сигнал у {taMaxCoin} на {taMax} (шорт)')

    return signal_coins

if __name__ == '__main__':
    signal_coins = {}
    pairs = {}

    pairs=[line.strip() for line in open(TICKERS)]
    for line in open(TICKERS):
        pairs=[line.strip() + PAIR_WITH for line in open(TICKERS)]

    while True:
        print(f'Аналізую {len(pairs)} монети')
        signal_coins = analyze(pairs)
        if len(signal_coins) == 0:
            print(f'Немає монет вище порогу {TA_BUY_THRESHOLD}')
        else:
            print(f'{len(signal_coins)} монет вище порогу {TA_BUY_THRESHOLD} у обох
таймфреймах')
        print(f'Очікую {TIME_TO_WAIT} хвилин для наступного аналізу')
        time.sleep((TIME_TO_WAIT*60))
import sys
sys.path.append('..')

import json
import os
from binance.client import Client
from datetime import datetime

# Завантажуємо допоміжні модулі
from helpers.parameters import (
```

```
    parse_args, load_config
)

# Завантажуємо модулі кредів
from helpers.handle_creds import (
    load_correct_creds
)

from colorama import init
init()

class txcolors:
    BUY = '\033[92m'
    WARNING = '\033[93m'
    SELL_LOSS = '\033[91m'
    SELL_PROFIT = '\033[32m'
    DIM = '\033[2m\033[35m'
    DEFAULT = '\033[39m'


args = parse_args()

DEFAULT_CONFIG_FILE = '../Конфіг.yml'
DEFAULT_CREDS_FILE = '../Креди.yml'

config_file = args.config if args.config else DEFAULT_CONFIG_FILE
creds_file = args.creds if args.creds else DEFAULT_CREDS_FILE
parsed_creds = load_config(creds_file)
parsed_config = load_config(config_file)

LOG_TRADES = parsed_config['script_options'].get('LOG_TRADES')
LOG_FILE = parsed_config['script_options'].get('LOG_FILE')
LOG_FILE_PATH = '../' + LOG_FILE

access_key, secret_key = load_correct_creds(parsed_creds)

client = Client(access_key, secret_key)

def write_log(logline):
    timestamp = datetime.now().strftime("%d/%m %H:%M:%S")
    with open(LOG_FILE_PATH,'a+') as f:
        f.write(timestamp + ' ' + logline + '\n')

with open('../Куплені_монети.json', 'r') as f:
    coins = json.load(f)
    total_profit = 0
    total_price_change = 0

    for coin in list(coins):
        sell_coin = client.create_order(
            symbol = coin,
            side = 'SELL',
            type = 'MARKET',
            quantity = coins[coin]['volume']
        )

        BuyPrice = float(coins[coin]['bought_at'])
        LastPrice = float(sell_coin['fills'][0]['price'])
        profit = (LastPrice - BuyPrice) * coins[coin]['volume']
        PriceChange = float((LastPrice - BuyPrice) / BuyPrice * 100)

        total_profit += profit
        total_price_change += PriceChange

        text_color = txcolors.SELL_PROFIT if PriceChange >= 0. else txcolors.SELL_LOSS
```

```
        console_log_text = f"{text_color}Продаю: {coins[coin]['volume']} {coin} - {BuyPrice}
- {LastPrice} Прибуток: {profit:.2f} {PriceChange:.2f}%{txcolors.DEFAULT}"
        print(console_log_text)

        if LOG_TRADES:
            timestamp = datetime.now().strftime("%d/%m %H:%M:%S")
            write_log(f"Продаю: {coins[coin]['volume']} {coin} - {BuyPrice} - {LastPrice}
Прибуток: {profit:.2f} {PriceChange:.2f}%")

    text_color = txcolors.SELL_PROFIT if total_price_change >= 0. else txcolors.SELL_LOSS
    print(f"Загальний прибуток: {text_color}{total_profit:.2f}{txcolors.DEFAULT}. Загальна
зміна ціни: {text_color}{total_price_change:.2f}%{txcolors.DEFAULT}")

os.remove('../Куплені_монети.json')
import yaml
import argparse



def load_config(file):
    try:

        with open(file, encoding="utf-8") as file:
            return yaml.load(file, Loader=yaml.FullLoader)
    except FileNotFoundError as fe:
        exit(f'Не можливо знайти {file}')

    except Exception as e:
        exit(f'Стався виняток...\n {e}')



def parse_args():
    x = argparse.ArgumentParser()
    x.add_argument('--debug', '-d', help="додатковий логінг", action='store_true')
    x.add_argument('--config', '-c', help="Шлях до config.yml")
    x.add_argument('--creds', '-u', help="Шлях до файлу з кредами")
    x.add_argument('--notimeout',    help="Не    використовуйте    тайм-аут    у    prod",
action="store_true")
    return x.parse_args()
from sys import exit



def load_correct_creds(creds):
    try:

        return creds['prod']['access_key'], creds['prod']['secret_key']

    except TypeError as te:
        message = 'Креди відформатовані некоректно\n'
```

```
        message += f'TypeError:Exception:\n\t{str(te)}'
        exit(message)
    except Exception as e:
        message = 'Сталась Fallback Exception :(\n'
        message += f'Exception:\n\t{str(e)}'
        exit(message)


def test_api_key(client, BinanceAPIException):
    #Перевіряє чи повертають надані ключі API помилки

    try:
        client.get_account()
        return True, "Ключ API успішно перевірено"

    except BinanceAPIException as e:



        if e.code in  [-2015,-2014]:
            bad_key = "ключ API відформатовано неправильно..."
            america = "якщо ви зараз в TLD країні (VPN), то треба оновити конфігурацію та
встановити TLD: True"
            ip_b = "якщо встановлене блокування IP на своїх ключах, переконайтеся що цей IP
дозволена (ipinfo.io/ip)"

            msg  =  f"Ваш  ключ  API  неправильний,  IP  заблокований,  або  неправильні
TLD/дозволи...\n Скоріш за все: {bad_key}\n  {america}\n  {ip_b}"

        elif e.code == -2021:
            issue = "Проблема з позначкою часу"
            desc = "Переконайтеся, що ОС синхронізована за часом із сервером"
            msg  =  f"Позначка  часу  для  цього  запиту  на  1000  мс  випередила  час  сервера\n
{issue}\n  {desc}"
        elif e.code == -1021:
            desc  =  "Час  ОС  не  синхронізовано  належним  чином...Синхронізуйте  час  ntp  з
'pool.ntp.org'"
            msg = f"{desc}\nМожливо спробуйте це:\n\tsudo ntpdate pool.ntp.org"
        else:
            msg = "Зіткнулись з кодом помилки API, який не було чітко виявлено...\n"
            msg += str(e)

        return False, msg
```

```
except Exception as e:
    return False, f"Стався виняток:\n{e}"
```