НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

"На правах рукопису"
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
"___"_____2022р.

Магістерська дисертація

зі спеціальності - 122 Комп'ютерні науки
за освітньо-професійною програмою магістерської підготовки - Комп'ютерний моніторинг та геометричне моделювання процесів і систем

на тему Пошук найкоротшого шляху на графі за допомогою клітинних автоматів

Виконав: студент _2_курсу, групи_ТР-12мп__
____Куйбіда Павло Костянтинович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент, к.т.н. Залевська Ольга Валеріївна _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

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

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2022

Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”

Навчально-науковий інститут атомної та теплової енергетики
Кафедри цифрових технологій в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою "Комп’ютерний моніторинг та геометричне моделювання процесів і систем"

Спеціальності 122 Комп’ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)

«____» _____ 2022р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Куйбіди Павла Костянтиновича
(прізвище, ім’я, по батькові)

1. Тема дисертації Пошук найкоротшого шляху на графі за допомогою клітинних автоматів

Науковий керівник доцент, к.т.н Залевська Ольга Валеріївна
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “__”листопада 2022 року №_____

2. Строк подання студентом дисертації 7 грудня 2022року

3. Вихідні дані до роботи програмний продукт для моделювання та дослідження пошуку найкоротшого шляху за допомогою клітинних автоматів

4. Перелік питань, які потрібно розробити проаналізувати існуючі програмні застосунки, що досліджують алгоритми пошуку шляху за допомогою клітинних автоматів; встановити набір правил для генерації клітинного автомату, що дозволяє моделювати пошук найкоротшого шляху на графі; реалізувати програмний продукт для дослідження використання клітинних автоматів для пошуку шляху на графі

5. Орієнтований перелік ілюстративного матеріалу приклади роботи програми; діаграма класів; діаграма прецедентів

6. Орієнтований перелік публікацій Ванін В.В., Залевська О.В., Захаркін М. С.,Куйбіда П.К., Zhu Shiwei Використання алгоритмів клітинних автоматів в задачах пошуку мінімального шляху . Сучасні проблеми моделювання, (24), 45-53. вилучено із http://magazine.mdpu.org.ua/index.php/spm/article/view/3041
Ванін В.В., Залевська О.В., Захаркін М. С.,Куйбіда П.К., Zhu Shiwei Пошук найкоротшого шляху на графі за допомогою клітинного автомату. Тези XXIV

Міжнародної науково-практичної конференції «Сучасні проблеми геометричного моделювання» : Тези міжнар. наук. доп., Мелітополь, 8 верес. 2022 р. Мелітополь, 2022

Довідка про впровадження результатів наукових досліджень практику діяльності ТОВ "ПРИЗМА-13"ЛТД з іноземними інвестиціями магістерської роботи студента Куйбіди Павла Костянтиновича, навчально-наукового інституту атомної та теплової енергетики кафедри цифрових технологій в енергетиці Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», що виконана на тему: «Пошук найкоротшого шляху на графі за допомогою клітинних автоматів».

7. Дата видачі завдання «____» жовтня 2021р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримати завдання	до 1.09.2022р	
2	Практика	01.09.22р.- 26.10.22р	
3	Збір інформації	До 26.10.22р	
4	Аналіз вимог завдання, розробка методів і засобів розв’язання поставленої задачі	До 26.10.22р	
5	Розробка та тестування програмного продукту	До 26.10.22р	
6	Виконання розділів дисертації (практична частина, загальні висновки, список джерел)	До 1.11.22р	
7	Написання основних розділів автореферату	До 25.10.22р	
8	Подання в електронному вигляді роботи та анотації до неї на перевірку нормоконтролера та плагіат (UNICHECK)	21.11.22р-30.11.22р	
9	Передзахист	23.11.22р	
10	Подання пакету документів по магістерської дисертаціїта супровідних до неї документів до захисту в ЕК	7.12.22р	
11	Захист магістерської дисертації	19.12.22р	

Студент _____ Куйібіда П.К.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Залевська О.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дисертація за темою “Пошук найкоротшого шляху на графі за допомогою клітинних автоматів” виконана студентом кафедри цифрових технологій в енергетиці НН ІАТЕ Куйбідою Павлом Костянтиновичем за спеціальності 122 “Комп’ютерні науки” за освітньо-професійною програмою “Цифрові технології в енергетиці” та складається зі вступу; 5 розділів: аналізу існуючих алгоритмів пошуку найкоротшого шляху за допомогою клітинних автоматів; правил генерації та алгоритм побудови найкоротшого шляху за допомогою клітинного автомату; розробки програмного забезпечення; розробки стартап проекту, загальних висновків; списку використаних джерел, який налічує 20 джерел; 3 додатки. Загальний Обсяг роботи 78 сторінок.

Актуальність теми. Визначення оптимального маршруту між об’єктами може здійснюватися в статичному, динамічному режимах і режимах реального часу. Це питання розглядається в багатьох важливих програмах у сфері GPS, відеоігор, робототехніки, логістики та симуляції натовпу - часові середовища. Проблема пошуку шляху може мати багато різних форм, включаючи ті, що стосуються одного агенту, групи агентів, конкурентного пошуку, динамічних змін навколишнього середовища, неоднорідної місцевості, мобільних пристроїв і неповна інформація. Алгоритм пошуку шляху та створення графіка для його реалізації є двома основними компонентами пошуку рушення проблеми.

Метою роботи є розробка програмного забезпечення, яке може вирішити проблему пошуку найкоротшого шляху на графі за допомогою клітинних автоматів.

Завдання дослідження:

- Проаналізувати існуючі програмні застосунки, що досліджують алгоритми пошуку шляху за допомогою клітинних автоматів
- Встановити набір правил для генерації клітинного автомату, що дозволяє моделювати пошук найкоротшого шляху на графі

- Реалізувати програмний продукт для дослідження використання клітинних автоматів для пошуку шляху на графі на основі встановлених правил

Об’єктом дослідження є динамічні системи.

Предмет дослідження є клітинні автомати.

Методи дослідження. При дослідженнях використовується методики порівняння, абстрагування та аналізу, динамічних систем, прикладної геометрії, комп’ютерної графіки.

Ключові слова: клітинний автомат, алгоритм пошуку найкоротшого шляху, прикладна геометрія, комп’ютерна графіка, мова програмування C++.

ABSTRACT

The master's thesis on the topic "Finding the shortest path on a graph using cellular automata" was written by the student of the Department of Digital Technologies in Energy, NN IATE Kuibida Pavlo, majoring in 122 "Computer Science" under the educational and professional program "Digital Technologies in Energy" and consists of admission; 5 sections: analysis of existing algorithms for finding the shortest path using cellular automata; generation rules and the algorithm for constructing the shortest path using a cellular automaton; software development; development of a startup project, general conclusions; the list of used sources, which includes 20 sources; 3 applications. The total volume of work is 78 pages.

Actuality of theme. Determination of the optimal route between objects can be carried out in static, dynamic and real-time modes. This issue is addressed in many important applications in the fields of GPS, video games, robotics, logistics, and crowd simulation - temporal environments. The pathfinding problem can take many different forms, including those involving a single agent, a group of agents, competitive search, dynamic environmental changes, heterogeneous terrain, mobile devices, and incomplete information. A path finding algorithm and creating a graph for its implementation are the two main components of finding a solution to a problem.

The goal of the work is to develop software that can solve the problem of finding the shortest path on a graph using cellular automata.

Objectives of the study:

- To analyze the existing software applications that investigate the algorithms of finding a path with the help of cellular automata
- Establish a set of rules for the generation of a cellular automaton, which allows simulating the search for the shortest path on a graph
- Implement a software product to research the use of cellular automata for graph pathfinding based on established rules

The object of research is dynamic systems.

The subject of research is cellular automata.

Research methods. The research uses methods of comparison, abstraction and analysis, dynamic systems, applied geometry, and computer graphics.

Key words: cellular automaton, pathfinding algorithms, applied geometry, computer graphics, C++ programming language.

ЗМІСТ

ВСТУП

Визначення маршруту може здійснюватися в статичному, динамічному режимах і режимах реального часу, і воно є основною частиною багатьох важливих програм у сферах GPS, відеоігор, робототехніки, логістики та симуляції натовпу - часові середовища. За останні 20 років численні інновації підвищили точність і ефективність методів пошуку шляху, але це питання продовжує залишатися в центрі інтенсивних досліджень. Надання зручних, високопродуктивних маршрутів наразі є найважливішою темою. Проблема пошуку шляху може мати багато різних форм, включаючи ті, що стосуються одного агенту, групи агентів, конкурентного пошуку, динамічних змін навколишнього середовища, неоднорідної місцевості, мобільних пристроїв і неповна інформація. Кожне з цих питань має унікальне застосування в багатьох сферах.

Метод описаний в цій роботі визначає клітину автомату як рухомий об’єкт, який відповідає набору правил, які залежать від заповнення сусідніх клітин. Якщо в цьому місці є перешкода для руху, присвоюється значення 1 або 0 характеристичній матриці для відповідного виміру залежно від об’єму, який займає фігура. Наявність перешкоди і те, наскільки вона заповнює простір, визначають правила пересування.

За допомогою запропонованого методу можна розв’язувати задачі, використовуючи апарат клітинних автоматів для знаходження найкращого рішення, а не лише мінімального значення.

Існує багато способів інтерпретації та використання проблеми визначення найкоротшого шляху на графіку. Ось кілька прикладів того, як проблема була застосована в різних ситуаціях. Дослідження операцій — це дисципліна, яка вивчає кілька застосувань. У системах картографування, таких як Google Maps або OpenStreetMap, шляхи між реальними фізичними об’єктами визначаються за допомогою алгоритмів пошуку найкоротшого шляху на графіку.

У випадку, якщо розглядається недетермінована абстрактна машина як граф, де кожна вершина представляє стан, а кожне ребро представляє потенційний перехід, можна використовувати методи пошуку найкоротшого шляху для визначення найкращого порядку рішень для досягнення головної мети.

Цю техніку можна використовувати для пошуку рішення з найменшою кількістю ходів, наприклад, якщо вершини є станами кубика Рубіка, а ребро представляє одну дію на кубі.

Щоб визначити найкоротшу відстань у мережі доріг, звичайною практикою є визначення найкоротшого шляху на графіку. Дорожню систему можна показати у вигляді графу зі зростаючими вагами. Перехрестя доріг служать вершинами, а з'єднувальні дороги – ребрами. Довжина певного сегмента, час, необхідний для його перетину, або витрати на переміщення вздовж нього – усе це може бути пов'язано з вагою ребр. Вулиці з одностороннім рухом можна представити за допомогою орієнтованих країв.

Подібні алгоритми зазвичай мають дві фази:

1. Перша та остання вершини графа ігноруються під час попередньої обробки (при роботі з реальними даними це може зайняти до кількох днів). Потім використовуються дані, отримані від звичайного одноразового виконання;

2. З відомими початковою та кінцевою вершинами виконується запит і пошук найкоротшого шляху.

Проблема пошуку шляху може мати багато різних форм, включаючи ті, що стосуються одного агенту, групи агентів, конкурентного пошуку, динамічних змін навколишнього середовища, неоднорідної місцевості, мобільних пристроїв і неповна інформація. Кожне з цих питань має унікальне застосування в багатьох сферах. Алгоритм пошуку оптимального шляху та створення графічної інтерпретації є двома основними компонентами пошуку шляху.

1 АНАЛІЗ ІСНУЮЧИХ АЛГОРИТМІВ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ ЗА ДОПОМОГОЮ КЛІТИННИХ АВТОМАТІВ

1.1 Алгоритм Адамацького

Перший алгоритм клітинних автоматів, який вирішує проблему найкоротшого шляху, запропонував Адамацький [1], хоча можна стверджувати, що відомий алгоритм Лі можна вважати першим підходом, схожим на використання клітинних автоматів (КА). Однак алгоритм КА, запропонований Адамацький, в основному вивчає зважений граф, який також є орієнтованим. Запропонований алгоритм КА зіткнувся з трьома найпоширенішими варіантами проблеми, а саме: найкоротший шлях з одного джерела (S3P), найкоротший шлях із усіма парами (APSP) і найкоротший шлях з одним джерелом з одним пунктом призначення (S3DSP). Основною метою цієї роботи було паралельно розглянути варіанти S3P і APSP шляхом реалізації КА з регульованим радіусом околиці. Розв’язання проблеми найкоротшого шляху або визначення найпрямішого шляху в мережі між двома вершинами (тобто x і y) було апроксимовано за допомогою КА шляхом побудови досліджуваного графа на прямокутну сітку, де клітинки x і y представляють відповідні вершини. Запропоноване рішення досягається, коли хвиля збудження з коміркою x початкової точки розповсюджується в усіх напрямках і досягає комірки y.

КА моделюють живі процеси, нейронні мережі, клітинні та тваринні популяції, молекулярні рідини, мембрани та збудливі реакційно-дифузійні середовища, оскільки вони є найбільш матеріальними, сприйнятливими та практичними моделями [1]. Основна особливість збудження в середовищі полягає в тому, що сигнали можуть поширюватися без затухання на велику відстань і швидкість поширення хвилі може бути змінною. У роботі Адамацького швидкість хвилі пропорційна вазі крайового з’єднання між вузлом x і вузлом y. На початку

заданий граф відображається на клітинному масиві КА. Вихідна вершина x і кінцева вершина y графа відповідають клітинкам x і y відповідно. Осередок x збуджується, і хвиля поширюється в усіх напрямках навколо решітки, змінюючи стани комірок. Вважається, що обчислення завершено, коли хвиля досягне місця призначення y.

Визначення КА - це d-вимірна решітка L з n комірок, станів комірки Q, функція сусідства u і функція переходу f. Для кожної клітинки x функція сусідства призначає групу найближчих клітинок u(x). Функція локального переходу f відображає набір сусідніх станів на набір Q станів комірок. Використовуючи всі наведені вище характеристики, наступний стан комірки x буде визначено як стан її околиці на попередньому часовому кроці та за правилом функції переходу таким чином: $x_{t+1} = f(u(x_t))$. Еволюція КА з початковою конфігурацією c є серією переходів, таких як: $c_0 \rightarrow c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_t \rightarrow c_{t+1} \rightarrow \ldots$. У цій роботі покажчик px і вектор wxy є використовувані, які відповідають напрямку, з якого поширювалася хвиля на попередньому кроці часу, і вазі ребра між вузлами x і y відповідно. Множина Q може приймати один із таких елементів: Q = {+, #, •, 0, 1, 2, ..., ν}. Покажчик px приймає значення зі скінченної непорожньої множини Y = {1, 2, . . . , k, λ } де λ можна вважати початковим значенням px. Кожен елемент wxy вектора wx, y ∈ u(x), може перебувати в одному зі станів множини W = {∞, 0, ..., ν}. Коли вершини (вузли) x і y не з'єднані між собою ребром, орієнтованим від x до y, тоді wxy = ∞ [17].

Нехай G = ⟨V, E⟩ — орієнтований граф із n вершин, розташованих на d-вимірній дискретній ґратці, кожна вершина υ ∈ V якої з'єднана не більше ніж з k сусідніми вершинами υ1, υ2, ...υk входом ребра υ1υ, υ2υ, ..., υkυ ∈ E ваг w(υ1υ), w(υ2υ), ..., w(υkυ) ∈ {0, 1, ...ν, ∞}. Якщо якась пара υ′υ″ ∈ V , w(υ′υ″) = ∞ записується, коли в множині E немає ребра υ′υ″ (тобто немає ребра між цими вершинами). Існує принципова особливість, яку повинен мати граф G, щоб успішно відобразити його на клітинну решітку: k < n. Нехай p = (υ0, ..., υm) — найкоротший шлях від вершини υ0 до вершини υm графа G, а l(p) — довжина p. Тоді маємо, що l(p) = min{l(p′) : p′ = {υ0, ..., υm}}.

По-перше, розглядається найкоротший шлях з єдиним джерелом і єдиним пунктом призначення (S3DSP). Це можна використовувати в S3P і APSP подібним чином. У обчисленні S3DSP на початку x0 s = +, де t = 0 передбачається, а xs і xd є вузлами джерела та призначення, тоді як + є хвилею збудження. Обчислення припиняється, коли клітинка x переходить у стан # або кожна клітинка L перебуває в стані # або • (• є станом спокою).

Друге обмеження використовується для зупинки обчислень, коли немає шляху від xs до xd. Віртуальна хвиля рухається в клітинній решітці. Хвиля станів + біжить у всіх напрямках навколо решітки від комірки xs до xd або проходить усі комірки L. Покажчик px має початковий стан λ, як згадувалося раніше. Коли клітинка x приймає стан +, тоді px змінює своє початкове значення та приймає одне з набору {1, 2, . . . , k}, який є індексом сусіда, звідки надійшов стан +. Таким чином, кінцевий шлях можна легко витягти з остаточної конфігурації КА шляхом зворотного відстеження за покажчиками від клітинки xd до клітинки xs. Функція переходу f працює з функцією сусідства u(x) і вагами wxy наступним чином. Припускаючи, що комірка xt = • і деякі її сусіди з u(x) знаходяться в стані + в момент часу t. Комірка xt знаходить сусіда yt = +, який має мінімальне значення ваги, і x приймає це значення. Починаючи зі стану wxy, клітинка x переходить у стан 0, зменшуючи свій поточний стан на одиничний крок на кожному часовому кроці $x_{t+1} = x_t - 1$. Комірка xt перейде в стан +, коли xt = 0 або існує сусід yt = + і вага ребра між xt і yt є мінімальною і $w^t_{xy} < x_t$. Перехід від стану 0 до стану + і від стану + до стану # відбувається безумовно.

Коли починається симуляція, покажчики всіх клітинок перебувають у стані λ. Однак, якщо клітинка x змінює свій стан на wxy, то для деякого сусіда y j покажчик px зберігає індекс j цього сусіда. Під час зменшення wxy до 0 вказівник px можна модифікувати, якщо виконується умова $\forall y \in u(x) : y_t = + \wedge w_{xy} = \min\{w_{xy'} : y' \in u(x) \wedge y'_t = +\}$ і $w^t_{xy} < x_t$. Стан вказівника стає постійним після того, як клітинка x виходить із стану 0. Закінчуючи всі ці правила, x_{t+1} може приймати наступні стани в кожному випадку:

- #, коли xt знаходиться в стані # або +.

- +, коли xt = 0.
- •, коли xt = • і немає жодного сусіда yt = +.
- wxy, коли xt = • або xt > 0 і є принаймні один сусід із yt = +, а вага ребра між xt і yt є мінімумом інших ребер у околиці.
- xt − 1, коли xt > 0 і немає іншого сусіда yt = +, який має вагу на краю між xt і yt, щоб wxy < xt.

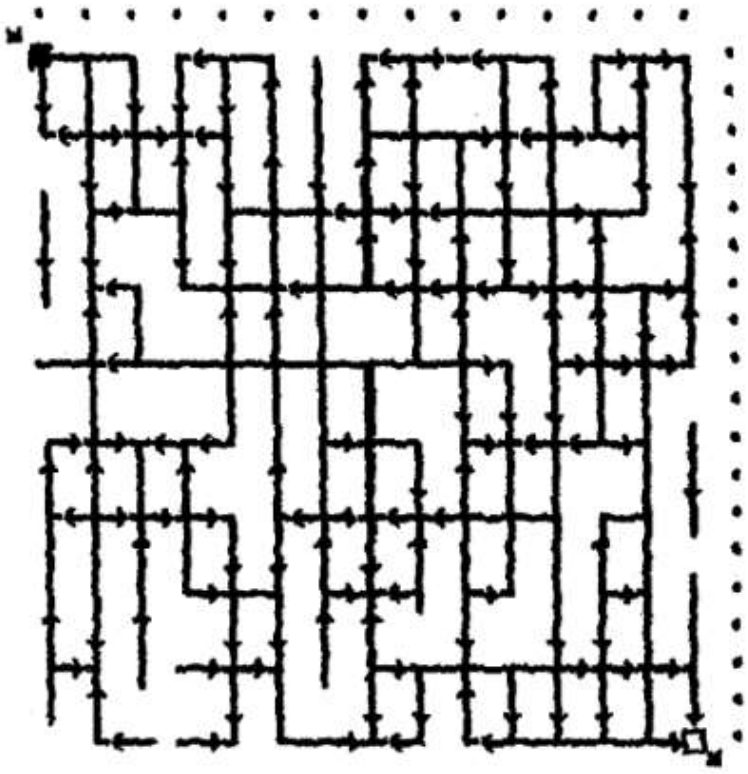


Рисунок 1.1 — Приклад орієнтованого графа G: стрілками вказано орієнтацію ребр; а перетин двох або більше прямих відповідає вершині G; чорні та порожні ящики ресурсна вершина та кінцева вершина відповідно

Приклад можна показати на рисунку 1.1. Метою є розв’язання S3DSP у двовимірній сітці, де ребра можуть приймати стани ∞ або 0, а вихідна вершина знаходиться у верхньому лівому куті, а кінцева – у нижньому правому куті. Q може приймати значення {•, +, #}, Y може приймати значення {N,W, S, E, λ }, а W може приймати значення (wxN, wxW, wxS, wxE). Символи N, W, S, E є індексами для

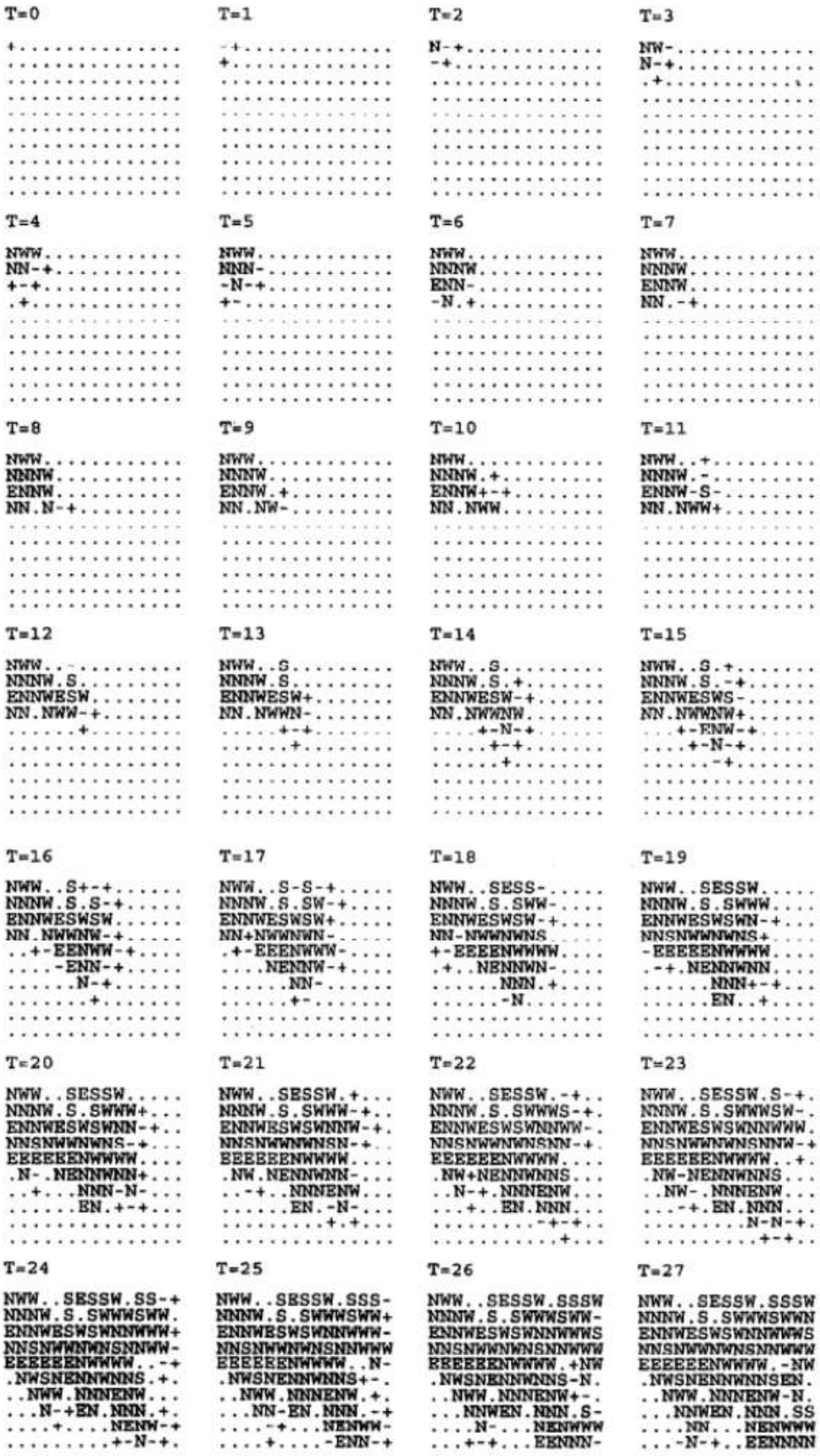
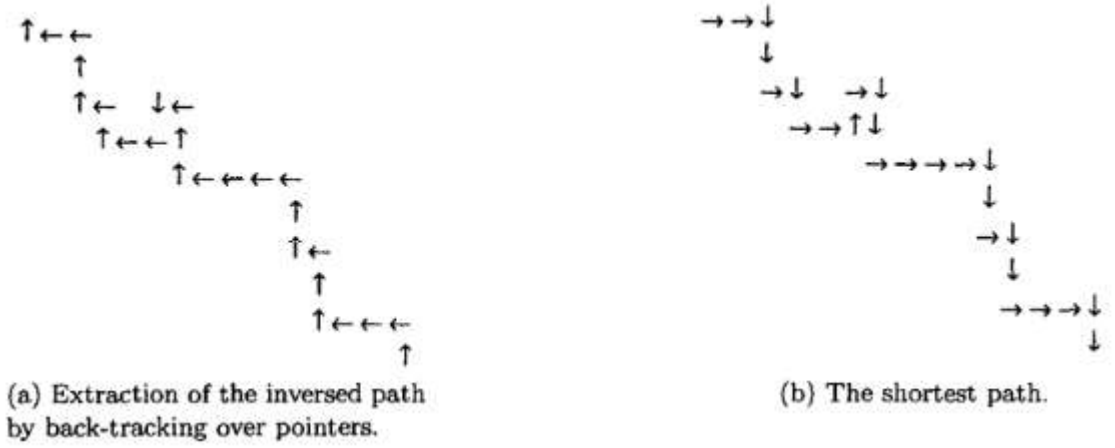


Рисунок 1.2 — Еволюція КА в обчисленні S3DSP

північних, західних, південних і східних сусідів комірки. Початкові умови: x0 s = +, ∀x ∈ L, x 6 = xs : x0 := • та px := λ . Динаміка еволюції КА показана на рисунку 1.2. Шляхи зворотного відстеження (a) і виділеного (b) можна знайти на рисунку 1.3.



(a) Extraction of the inversed path by back-tracking over pointers.

(b) The shortest path.

Рисунок 1.3 — Результати розрахунку S3DSP

Найдовший шлях у G складається з n − 1 вузлів. Таким чином, КА з n комірок, чотирьох сусідів і дев'яти станів, який моделює 2-d граф G з n вузлів, деяких граничних країв і ваг ребр {∞, 0}, може обчислити найкоротший шлях за час обчислення O(n). і APSP в O(n2) відповідно. Припускаючи, що використовується 2-d КА з n комірок, кожна з яких має чотири сусіди та 6 + 5n станів, що моделює 2-d прямокутну сітку, де ребра G-графа можуть мати вагу {0, ..., ν, ∞ }, S3DSP можна вирішити за O(νn) верхнього часу.

1.2 Алгоритм клітинних автоматів, заснований на алгоритмі Лі

Алгоритм Лі є добре відомим фундаментальним алгоритмом маршрутизації, який може знайти найкоротший шлях між двома точками в сітці. Рейнхардом Гофманом та Вільям Хохбергер було знайдено КА, подібний до алгоритму Лі, який використовує невелику кількість станів, незалежно від розміру сітки[2]. Його було

розроблено, коли досліджували виражальну силу мови клітинного опису CDL (Cellular Description Language) для різних застосувань.

1.2.1 Алгоритм Лі

Дуже добре відомим підходом до проблем маршрутизації є алгоритм Лі. Його мета - знайти оптимальний шлях між двома точками на регулярній сітці. Кожна точка сітки пов’язана з вагою. Алгоритм знаходить шлях на сітці з найменшою сумою ваг. Регулюючи вагові значення точок сітки, користувач має певний контроль над тим, що має бути оптимальним шляхом. Розглянемо приклад: користувач просто шукає найкоротший шлях між двома точками. У цьому випадку користувач вказує одиничну вагу для всіх точок сітки, і алгоритм знайде шлях із найменшою кількістю точок сітки. Це найкоротший шлях між двома вибраними точками S і E. В іншому прикладі користувач шукає шлях, який якомога менше перетинає вже існуючі шляхи. У цьому випадку користувач призначає дуже високу вагу всім точкам існуючих шляхів і дуже низьку вагу всім іншим точкам сітки. Тоді алгоритм знайде шлях із якомога меншою кількістю перетинів. Алгоритм працює в два етапи: (1) поширення хвилі від S до E і (2) побудова шляху шляхом зворотного відстеження від E до S [3].

На першому етапі обчислюються накопичені ваги (acw) для кожного вузла відносно початкової точки. Усі вільні комірки ініціалізуються до нескінченності. Накопичена вага для початкової точки S ініціалізується рівним 0. На рис. 1.4 для зразкової сітки показано розрахунок накопичених ваг. У цьому випадку вага всіх точок сітки дорівнює одній. Хвиля досягає кінцевої точки на кроці часу t = 6.

На другій фазі встановлюється фактичний шлях. Для цього алгоритм повертається від кінцевої точки E до початкової точки S. На кожному кроці в якості частини шляху вибирається сусід із найменшою накопиченою вагою.

Зауважте, що є кілька можливостей побудувати шлях від кінцевої точки до початкової точки. У кінцевій точці ви можете піднятися вгору або ліворуч (Шлях 1 і Шлях 2 на рисунку 1.4 відповідно) для цього прикладу.

Перешкоди можна моделювати нескінченними вагами в точках сітки. Оскільки тоді сума локальної ваги та накопиченої ваги сусіда завжди буде нескінченною, такі точки сітки можуть ніколи не стати частиною шляху.

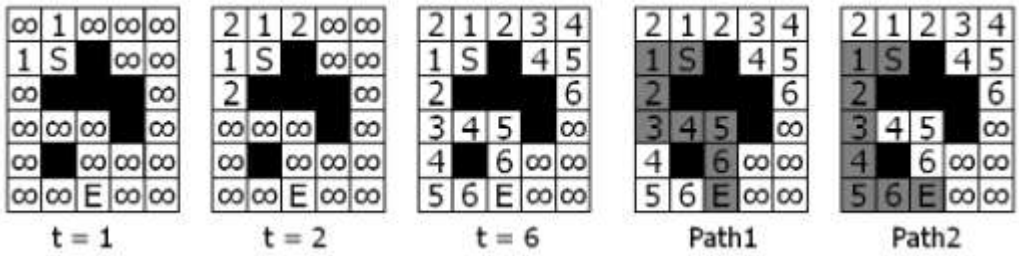


Рисунок 1.4 — Оригінальний алгоритм Лі. Хвиля поширюється від S до E

На перший погляд, цей алгоритм виглядає так, ніби він ідеально підходить для КА. На жаль, кількість станів, необхідних для виконання алгоритму, пов’язана з найдовшим шляхом або, точніше, з найбільшою накопиченою вагою, яка може статися.

1.2.2 Алгоритм Лі на основі КА

Накопичені вагові коефіцієнти в алгоритмі Лі потрібні для пошуку найкоротшого шляху. Замість того, щоб зберігати накопичені ваги, ми запам’ятовували напрямок, у якому ми повинні рухатися, щоб повернутися до вихідної точки. З цими хвильовими мітками замість накопичених ваг алгоритм потребує лише постійного набору станів. Звичайно, ми не можемо вирішити проблеми з довільними вагами в точках сітки [2].

На початку першої фази всі клітини знаходяться у вільному стані, крім початкової та кінцевої точки. На першому етапі всі комірки перевіряють, чи є в околицях комірки, які вже мають хвильову позначку. Якщо хвильову мітку знайдено, сама клітина стає хвильовою міткою до вже позначеної клітини. Функція послідовно призначає всі елементи груп сусіди і хвиля тимчасовим змінним h

(адреса комірки) і z (стан). Для кожного призначення перевіряється умова після двокрапки. Обчислення однієї функції припиняється, якщо знайдено призначення, яке задовольняє умову, тобто відповідний сусід знаходиться в стані start або в будь-якій хвилі станів. Перше призначення h=N і z=wave_up. Присвоєння z використовується лише для його побічного ефекту, щоб тимчасово зберегти стан хвилі, що відповідає досліджуваному сусіду. Якщо зараз досліджується східний сусід, цей стан комірки зміниться на wave_right, оскільки він має вказувати на цього сусіда. На рисунку 1.5 при t = 6 показана сітка зразка в кінці першої фази. Знаки хвилі позначаються маленькими стрілками. Чорні квадрати - це перешкоди. Нам довелося ввести спеціальний стан для моделювання перешкод, оскільки ми не маємо ваг у точках сітки.

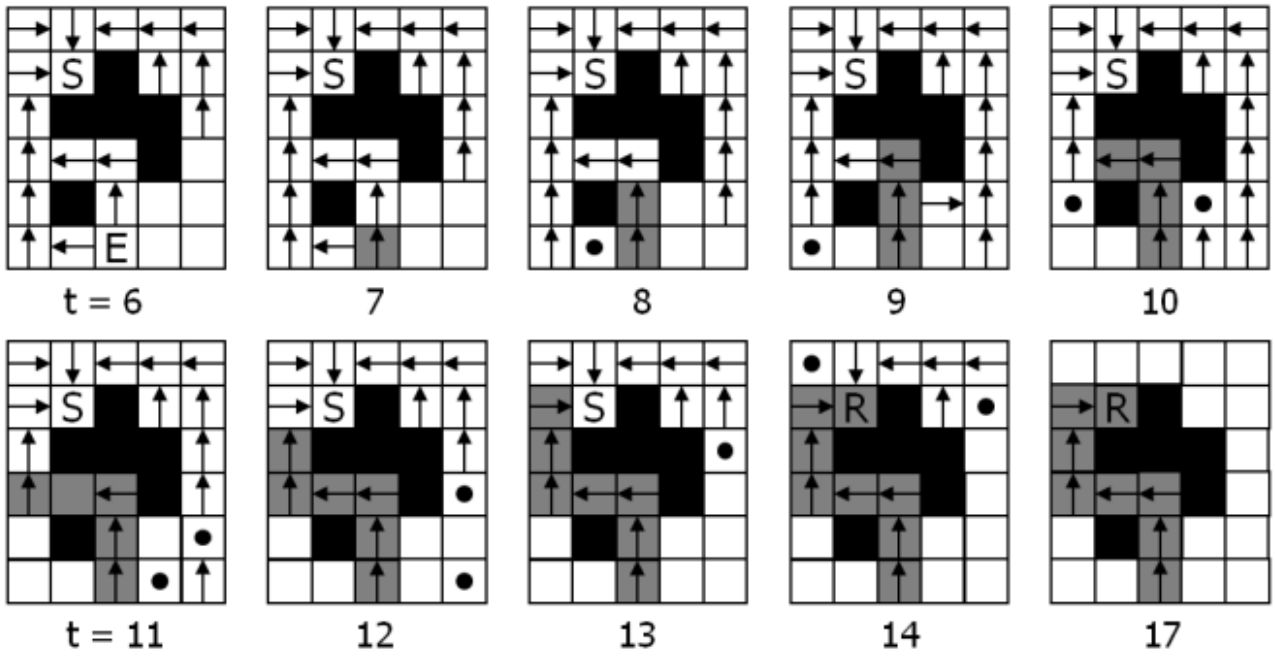


Рисунок 1.5 — Симуляція алгоритму Лі на основі клітинних автоматів

Перша фаза закінчується, коли досягається кінцева точка. Тепер шлях будується назад до початкової точки вздовж міток хвилі (рядки 29–34). Якщо клітина є одним із хвильових станів і вона бачить сусідню клітину, яка є шляхом до цієї клітини, то ця клітина стає шляхом у напрямку її попередньої позначки (рис.

1.5, t = 7 − 13). Усі непотрібні позначки хвилі мають бути очищені, щоб дозволити наступні проходи маршруту. Для цього очищаються всі комірки, які бачать сусідню комірку, шлях якої не вказує на цю комірку. Така клітина ніколи не стане частиною шляху. Також очищаються всі комірки, які бачать сусідів у відкритому стані. Оскільки побудова шляху рухається по найкоротшому шляху, неможливо, щоб клітина в чистому стані досягла клітини, яка буде на шляху, але ще не є його частиною. Комірка зі станом очищення зміниться на стан вільної на наступному часовому етапі. Часова складність для першої та другої фази становить O(p), де p — довжина шляху. Для квадратної сітки n × n максимальна довжина шляху дорівнює 2n − 1, якщо немає перешкод, і O(n2), якщо є перешкоди (шлях, подібний до спіралі) [2]. Алгоритм потребує лише 14 станів і тому може бути дуже легко реалізований апаратно.

1.3 Learning automaton

Існують проблеми з найкоротшим шляхом, у яких довжини ребер у графі можуть бути випадковими. Це ускладнює проблему. Стохастичний граф G визначається трійкою G = ⟨V, E, F⟩ , де V представляє набір вузлів, E визначає набір ребер, а матриця F є розподілом ймовірностей, що описує статистику довжин ребер. У стохастичному графі G шлях πi з довжиною ni вузлів і очікуваною довжиною Lπi від вузла джерела Vsource до вузла призначення Vdest визначається як порядок {πi,1, πi,2...πi,ni } вузлів. Vπi,1 = Vsource і Vπi,ni = Vdest є вузлами джерела та призначення, відповідно, і всі проміжні вузли є вузлами шляху πi. Припустимо, що існує r різних шляхів між Vsource і Vdest. Найкоротший шлях між вихідним вузлом і вузлом призначення визначається як шлях із мінімальною очікуваною довжиною. Стохастичні проблеми найкоротшого шляху можна згрупувати в два основні класи: перший клас спрямований на пошук апріорного рішення, яке мінімізує очікувані довжини, тоді як другий клас обчислює онлайн-рішення, яке дозволяє приймати рішення на різних етапах.

У 2006 році Бейгі та Мейболді представили мережу автоматів навчання (LA), які були названі розподіленими автоматами навчання (DLA) [4]. Більш конкретно, автомат, який діє в невідомому випадковому середовищі та покращує свою продуктивність певним чином, називається навчальним автоматом (LA). DLA — це мережа автоматів, які колективно співпрацюють для вирішення певної проблеми. DLA може бути змодельований орієнтованим графом, у якому набір вузлів графа становить набір автоматів, а набір вихідних ребер для кожного вузла утворює набір дій для відповідного автомата. Коли автомат вибирає одну зі своїх дій, інший автомат на іншому кінці краю, що відповідає вибраній дії, буде активовано [4]. Вони використовували цей інструмент, щоб запропонувати деякі ітераційні алгоритми та вирішити стохастичну проблему найкоротшого шляху. У цих алгоритмах на кожній стадії DLA визначають, які краї потрібно відібрати. Цей метод вибірки може призвести до зменшення непотрібних вибірок і, отже, до зменшення часу роботи алгоритмів. Автоматний підхід до навчання передбачає визначення оптимальної дії з набору допустимих дій. Автомат вибирає дію зі свого кінцевого набору дій, що служить вхідними даними для середовища, яке, у свою чергу, видає стохастичну відповідь у певний час. Середовище карає або винагороджує дію автомата з ймовірністю штрафу/винагороди [4]. Стан автомата оновлюється, і на наступному кроці часу вибирається нова дія.

У їхньому алгоритмі створюється мережа автоматів навчання, яка ізоморфна вхідному графу. Кожен вузол у мережі представляє LA, і дії цього вузла в LA є вихідним краєм цього вузла. Потім на етапі k вихідний автомат, який представляє вихідний вузол на графі, вибирає одну зі своїх дій на основі свого вектора ймовірності дії. Якщо дія є am, автомат Am також активується на іншому кінці ребра (s, m). Процес вибору дії та активації автомата повторюється до тих пір, поки не буде досягнуто цільовий автомат або з якихось причин пересування по краях графа неможливе або кількість відвіданих вузлів перевищує кількість вузлів у графі. Після досягнення автомата призначення довжина пройденого шляху обчислюється, а потім порівнюється з величиною, яка називається динамічним порогом.

Залежно від результату порівняння всі LA (крім автомата навчання призначення) уздовж пройденого шляху оновлюють свої ймовірності дій. Оновлення здійснюється в напрямку від джерела до призначення або навпаки. Якщо довжина пройденого шляху менша або дорівнює динамічному порогу, тоді всі LA на цьому шляху отримують винагороду, а якщо довжина пройденого шляху перевищує динамічний поріг або вузол призначення не досягнуто, то активовані автомати отримують штраф. Процес подорожі від вихідного LA до кінцевого LA повторюється доти, доки не буде досягнуто умови зупинки, коли в цей момент останнім пройденим шляхом є шлях, який має мінімальну очікувану довжину серед усіх шляхів від джерела до пункту призначення.

1.4 Алгоритми пошуку шляху на графі без використання клітинних автоматів

Розглянемо деякі найрозповсюдженіші алгоритми пошуку найкоротшого шляху на графі, що не використовують клітинні автомати.

1.4.1 Алгоритм Дейкстри

Алгоритм Дейкстри — це метод визначення найкоротшого маршруту між вершинами графа. З початкової точки об'єкта починається робота. кожного разу перевіряє найближчу недосліджену вершину, додаючи найближчі вершини до списку вершин, які слід враховувати. Якщо на графі немає ребер із від’ємними значеннями, цей алгоритм завжди визначає найкоротший маршрут між початковою точкою та бажаним пунктом призначення[6].

Порівняльна простота реалізації алгоритму Дейкстри, низька поліноміальна обчислювальна складність і потенціал для машинної реалізації є його перевагами.

До недоліків алгоритму можна віднести наступне:

- працює тільки зі структурами графів, які мають невід'ємні довжини дуг; - шукає найкоротші шляхи та їх значення лише з однієї вершини;
- щоб обчислити ТК між усіма парами вершин, необхідно повторити процес для кожної пари вершин у вихідному графі N разів;
- не шукає рефлексивних замикань;
- має досить складне програмне представлення.

1.4.2 Алгоритм Беллмана-Форда

Алгоритм Беллмана-Форда емулює найкоротші шляхи від однієї вихідної вершини до всіх інших вершин у зваженому орграфі [18]. Він повільніший, ніж алгоритм Дейкстри для тієї ж проблеми, але більш універсальний, оскільки він може обробляти графіки з деякими вагами ребер, які є від’ємними числами [6].

Даний має часову складність O (V cdot E), де V — кількість вершин у графі, а E — кількість ребер [6]. Виконується V кроків, що є причиною цієї складності. Далі проходимо всі ребра графа на кожній фазі.

Ключовою перевагою алгоритму Беллмана-Форда є його здатність працювати з негативними вагами. Алгоритм Беллмана-Форда набагато складніше зрозуміти, ніж метод Дейкстри. Через те, що графіки з від’ємними вагами зустрічаються відносно рідко, підхід Дейкстри має більше застосувань.

Алгоритм Беллмана-Форда може обробляти орієнтовані та неорієнтовані графи з невід’ємними вагами, як було зазначено раніше. Однак, якщо негативних циклів немає, він може обробляти лише орієнтовані графи з негативними вагами[6].

1.4.3 Алгоритм A*

Евристичні алгоритми пошуку включають алгоритм пошуку A*. У графі з додатними вагами ребер він використовується для визначення найкоротшого маршруту між будь-якими двома вершинами. Пітер Харт, Нільс Нільссон і Бертрам Рафаель у 1968 році.

Евристика, допоміжна функція, яка використовується алгоритмом, використовується для фокусування та прискорення пошуку. Алгоритм повний у тому сенсі, що якщо оптимальне рішення існує, воно завжди знайдено.

Для кожної ітерації враховується така вершина n, де оцінка f (n) = g (n) + h (n)[7] є найменшою. Поведінка алгоритму залежить від обраної евристичної функції h (n). Коли h (n) = 0, впливає лише g (n). A * одночасно перетворюється на алгоритм Дейкстри, який завжди знаходить найкоротший маршрут. A * завжди знайде найкоротший шлях, якщо h (n) дорівнює або менше вартості маршруту від n до мети. Процес стає повільнішим, оскільки різниця збільшується, тому що потрібно перевірити більше вершин. Вартість шляху від точки n до цілі точно дорівнює h (n): A * вибере лише вершини з найкоротшого шляху, не враховуючи інші варіанти [7].

Через те, що алгоритм A * використовує «оптимістичну» оцінку шляху через вершину, він одночасно є допустимим і пропускає мінімальну кількість вершин. Оптимізм у тому, що, якщо він проходить через цей пік, алгоритм «має шанс», що фактичне значення результату буде рівним, але не нижчим за цю оцінку. Однак, оскільки A* є інтелектуальним алгоритмом, така еквівалентність може бути можлива. A *, за визначенням, виявив шлях, коли його пошук завершено, якщо його справжня вартість нижча, ніж очікувана вартість будь-якого шляху через будь-який відкритий вузол. Однак через те, наскільки оптимістичними є ці оцінки, пов’язані вузли можна сміливо ігнорувати [8]. Іншими словами, A * є прийнятним, оскільки він ніколи не пропускає можливість зменшити довжину шляху.

Його недоліком є те, що він шукає по всіх вершинах графа. Цей алгоритм не працюватиме, якщо граф містить багато вершин.

1.5 Висновки до розділу

У цьому розділі зображено нерозривний зв’язок між СА та проблемою найкоротшого шляху. СА є дуже потужним інструментом моделювання, який може охопити основні характеристики цієї проблеми та отримати ефективні результати.

2 ПРАВИЛА ГЕНЕРАЦІЇ ТА АЛГОРИТМ ПОБУДОВИ НАЙКОРОТШОГО ШЛЯХУ ЗА ДОПОМОГОЮ КЛІТИННОГО АВТОМАТУ

2.1 Опис модулю сітки клітинного автомату

Логіка відображення базується на примітивах бібліотеки SFML таких як sf::RectangleShape, sf::Font та sf::Text, знаходиться в функції main.

Логіка керування програмою використовує систему подій бібліотеки SFML та також знаходяться в функції main.

Таблиця 2.1 Опис методів інтерфейсу:

№	Сигнатура	Вхідні парам.	Вихідні парам.	Призн.
1	Board	size	-	Конструктор классу
2	~Board	-	-	Деструктор классу
3	operator=	board	Board&	Оператор присвоєння
4	isPassable	position	bool	Перевірка можливості проходу через клітинку

Продовження таблиці 2.1

5	operator[]	position	Cell&	Отримання данних про клітинку
6	getSize()	-	sf::Vector2u	Отримання розміру сітки
7	getPassableNeighbors	sf::Vector2u	std::list<sf::Vector2u>	Отримання сусідніх кліток на котрі можна перейти
8	allocate	-	-	Створення сітки

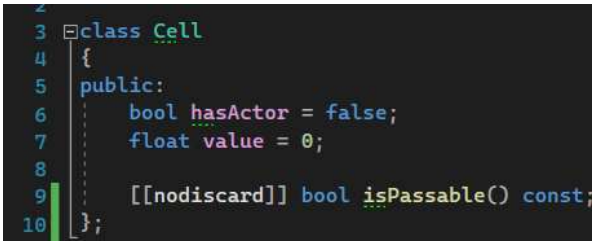


Рисунок 2.1 — Приклад коду клітинки

Вспоміжний клас Random — створений для зручності отримання випадкових чисел, є абстракцією з використанням сучасного c++ 20 templates with concepts над класами стандартної бібліотеки в тому числі mt19937 (Mersenne Twister pseudo-random generator of 32-bit numbers with a state size of 19937 bits.)

2.2 Математична модель

Клітинний автомат — це детермінована динамічна система, яка розвивається в дискретному часі і дискретному просторі, останнє зазвичай являє собою сітку. Вона складається з сітки клітин, які локально, але синхронно оновлюються по сітці відповідно до глобальної шкали часу і глобального рекурсивного правила, що регулюють еволюцію стану кожної клітини як функцію стану сусідніх клітин [15]. Хоча модель клітинних автоматів має ту ж обчислювальну потужність, що й інші універсальні моделі Тюрінга, і, отже, принципово еквівалентна, оскільки вони можуть емулювати один одного, однією з найбільш характерних особливостей клітинних автоматів є якісна різноманітність їх просторово-часових еволюцій, коли досліджуючи різні правила та різні початкові умови. Їх характерні патерни з'являються швидше, ніж в інших обчислювальних моделях, і візуально відображаються в компактній формі в результаті їх синхронного характеру, що робить їх придатними для кількісного і якісного вивчення, а також для порівняння з фізичними і природними явищами.

Двовимірні клітинні автомати вивчалися на початку 1950-х років такими людьми, як Станіслав Улам, Джон фон Нейман та Нільс Алл Баррічеллі у контексті гідродинаміки та біологічних систем. Улам та фон Нейман створили метод розрахунку руху рідини наприкінці 1950-х років. Основна концепція методу у тому, щоб розглядати рідину як групу дискретних одиниць і обчислювати рух кожної їх основі поведінки її сусідів. Улам запропонував використовувати дискретну систему для створення редукціоністської моделі самовідтворення. Джон фон Нейман розглянув ці моделі клітинних автоматів у своєму прагненні знайти або відкрити «універсальний конструктор», обчислювальну модель, яка могла б описувати себе та відтворюватись[16]. Набагато пізніше, в 1986 році, Крістофер Ленгтон виявив клітинний автомат, що самовідтворюється, на ім'я мураха Ленгтона, знайдений в невеликому просторі правил (менша станів/коротше правило), ніж створене фон Нейманом. Баррічеллі виконав багато з найраніших чисельних експериментів з клітинними автоматами як основу для штучного життя

як попередника еволюційних алгоритмів. Тип околиці, відмінний від того, що розглядав фон Нейман у двовимірних клітинних автоматах, - це околиця, названа на честь Едварда Ф. Мура, піонера теорії клітинних автоматів. Район Мура складається з дев'яти осередків: центрального осередку і восьми осередків, що оточують її [16].

2.3 Встановлення правил побудови КА

Зараз ми ближче розглядаємо КА, зосереджуючись на моделях і результатах, що представляють філософський інтерес. Хоча різноманітність систем, які можна знайти в літературі з КА, величезна, можна створити практично всі КА, налаштувавши чотири параметри, які визначають їх структуру:

- Дискретна n-вимірна решітка комірок: Ми можемо мати одновимірну, двовимірну, … , n-вимірну КА. Атомні компоненти решітки можуть мати різну форму: наприклад, двовимірна решітка може складатися з трикутників, квадратів або шестикутників. Зазвичай передбачається однорідність: усі клітини якісно ідентичні.
- Дискретні стани: на кожному дискретному часовому кроці кожна клітинка перебуває в одному і лише одному стані, $\sigma \in \Sigma$,Σ будучи набором скінченної потужності $|\Sigma|=k$.
- Локальні взаємодії: поведінка кожної клітини залежить лише від того, що відбувається в її локальному сусідстві з клітинами (що може включати або не включати саму клітину). Решітки з однаковою базовою топологією можуть мати різні визначення сусідства, як ми побачимо нижче. Важливо, однак, зауважити, що «дії на відстані» неприпустимі.
- Дискретна динаміка: на кожному кроці часу кожна комірка оновлює свій поточний стан відповідно до детермінованої функції переходу $\phi:\Sigma n \rightarrow \Sigma$ відображення конфігурацій сусідства (n-кортежів станів Σ) на Σ. Також зазвичай, хоча і не обов’язково, передбачається, що (i) оновлення є

синхронним, і (ii) ϕ приймає як вхідні дані на кроці часу t стани сусідства на безпосередньо попередньому кроці часу t-1.

Як приклад, розглянемо знамениту «Гру в життя» (або, коротше, «Життя») Джона Конвея (див. Berkelamp, Conway, & Guy 1982). «Життя» добре вписується в нашу звичну схему:

- 2-вимірна решітка квадратних комірок в ортогональній сітці.
- Σ={1,0}, тому |Σ|=2 (з причин, які ми збираємось побачити, ми можемо представити 1 як стан живого для даної клітини, 0 як стан мертвої).
- Околиця кожної комірки складається з усіх восьми сусідніх комірок (околиця Мура).
- Правило життєвого переходу виглядає наступним чином. На кожному кроці часу t з клітиною може статися одна з трьох речей:
    - Народження: якщо стан клітини при t−1 було 0 (мертва), стан комірки стає 1 (жива), якщо рівно три сусіди були 1 (живі) у t−1
    - Виживання: якщо стан клітини при t−1 було 1 (живий), стан комірки все ще дорівнює 1, якщо два або три сусіди були 1 (живі) у t−1
    - Смерть: якщо стан клітини при t−1 було 1 (жива), стан клітини стає 0 (мертва), якщо або менше двох, або більше трьох сусідів були 1 (живими) у t−1 (клітини можуть померти від «самотності» або «перенаселення»).

Відповідно до таксономії Вольфрама, життя безперечно вважатиметься КА класу 4. У цій простій обстановці виникають періодичні структури, стабільні блоки та складні рухомі моделі, навіть починаючи з дуже простої початкової конфігурації.

2.4 Алгоритм пошуку найкоротшого шляху за допомогою КА

Для того, щоб змоделювати пошук найкоротшого шляху за допомогою клітинних автоматів, нами були сформульовані наступні правила побудови:

1. Генерація перешкод
2. Встановлення напрямку руху
3. Визначаємо клітину та правила її життя, що залежать від заповнення сусідніх клітин.
4. Визначаємо можливість "проходу" через клітину: Якщо на даній позиції є перешкода руху і вона займає більше 50% об'єму клітини, то прохід закрито, в інших випадках-прохід можливий.
5. Сума одиниць є кількістю клітин, які необхідно пройти в дане положення, від початкового значення
6. Якщо мінімальний шлях не знайдено (потрапили в закуток) то алгоритм продовжує свою дію як A* алгоритм.

Правила руху а в запропонованому алгоритмі будуть визначатись наявністю перешкод та її об’ємом, тобто на скільки сильно ця перешкода заповнює простір (чи є можливість рухатись далі по клітинці, чи необхідно здійснювати обхід перешкод).

2.5 Класи клітинних автоматів

Клас 1 включає правила, що швидко створюють уніфіковані конфігурації. Правила класу 2 створюють уніфікований кінцевий шаблон чи чергування остаточних шаблонів залежно від початкових конфігурацій. Конфігурації, створені членами класу 3, значною мірою виглядають випадковими, хоча можуть бути деякі регулярні шаблони і структури [15].

Клас 4 заслуговує на особливу увагу. Якщо ми спостерігаємо всесвіт, створений Правилом 110, ми бачимо регулярні патерни (хоча і не такі регулярні, як у Правилу 108), а також деяка хаотична поведінка (хоча і не така галаслива, як у Правилу 90). Тепер основна характеристика, необхідна КА для виконання обчислень, — це здатність його правила переходу створювати «поширювані шаблони, подібні до частинок», тобто локалізовані, стабільні, але неперіодичні конфігурації груп осередків. Іноді звані в літературі солітонами, які можуть

зберігати свою форму. Ці конфігурації можна розглядати як кодування пакетів інформації, їх збереження у часі та переміщення з одного місця в інше: інформація може поширюватися у часі та просторі, не наражаючись на істотний розпад.

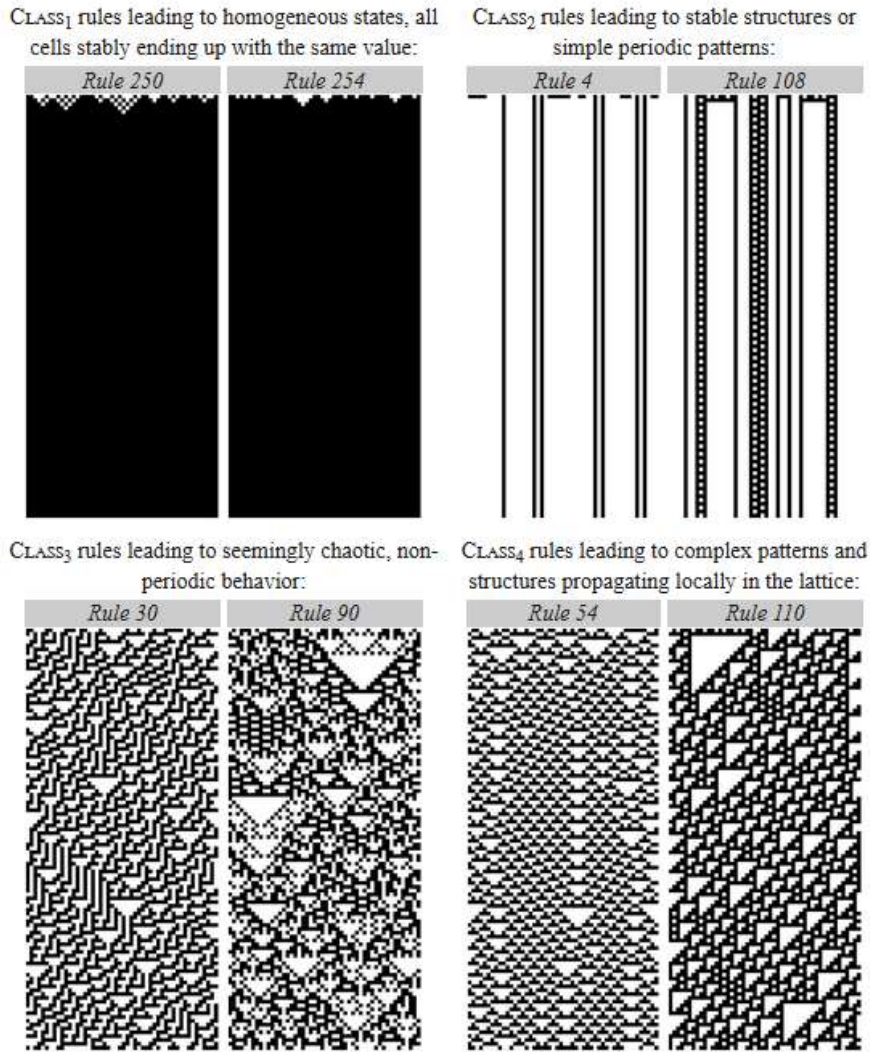


Рисунок 2.2 — Класи клітиних автоматів

Ступінь непередбачуваності у поведінці правил класу 4 також натякає на цікаві з обчислювальної точки зору особливості: за теоремою зупинки це ключова особливість універсальних обчислень, згідно з якою в принципі неможливо передбачити, чи дане обчислення буде виконано. зупинятися за певного введення. Ці ідеї привели Вольфрама до припущення, що КА класу 4 були (єдиними) здатними до універсальних обчислень. Інтуїтивно, якщо ми інтерпретуємо

початкову конфігурацію КА класу 4 як його вхідні дані, універсальний КА класу 4 може оцінити будь-яку функцію, що ефективно обчислюється, і емулювати універсальну машину Тюрінга [15]. Як ми згадували вище, правило 110 справді виявилося обчислювально універсальним.

2.6 Висновки до розділу

В даному розділі була розглянута математична модель, що базується на клітинних автоматах. Тому були розглянуті класи клітинних автоматів, правила їх побудови та програмний модуль, що реалізує сітку клітинного автомату. Був сформульований алгоритм пошуку найкоротшого шляху за допомогою клітинних автоматів.

Межі складності дуже хороші скільки ми використовували дуже обмежену форму прямокутної решітки, щоб зробити структуру задачі найбільш схожою на архітектуру обчислювального пристрою, який вирішує цю проблему. Алгоритми КА походять майже безпосередньо для біології та природи тому вони придатні до застосування на практиці. З цієї причини реалізація цих алгоритмів у масивних паралельних процесорах або нейрокомп’ютерах є подією, яка вже сталася.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Використані технології

При розробці програмного забезпечення було використано такі технології як: мова програмування - C++, бібліотека - SFML. Програму було розроблено та скомпільовано в Visual Studio. Нижче розглянемо всі технології детальніше.

3.1.1 Мова програмування C++

Об’єктно-орієнтована, узагальнена, процедурна та інші парадигми програмування підтримуються мовою програмування загального призначення C++. У 1979 році Б'ярн Страуструп почав працювати над C++ у AT&T Bell Laboratories у Мюррей-Хілл, Нью-Джерсі. Колись ця мова була відома як «сі з класами». Пізніше, у 1984 році, Страуструп змінив назву мови на C++[10]. Його джерелом є мова програмування C. Найбільш застосовним стандартом є ISO/IEC 14882:2020 (C++20), який замінює міжнародний стандарт ISO/IEC 14882:1998 (C++98).

Однією з найпопулярніших мов програмування загального призначення 1990-х була C++. Мова використовується для створення складних серверних і клієнтських програм, створення драйверів, системного програмування, створення прикладного програмного забезпечення та створення розважального програмного забезпечення, наприклад відеоігор. C# і Java — це дві сучасні мови програмування, які зазнали значного впливу C++.

Під час розробки вони доклали зусиль, щоб C++ був сумісним із C. Компілятор C++ сумісний з більшістю програм на C. Синтаксис C++ базується на синтаксисі C [10].

Рік Маскітті придумав назву «C++», яка вперше була використана в грудні 1983 року. Раніше, коли вона ще перебувала в розробці, нова мова була відома як

«С з класами». Остаточна назва походить від оператора С "++", що означає збільшення значення змінної на одиницю. Зазвичай комп’ютерні програми перейменовують, додаючи знак «+» для позначення прогресу. Цей псевдонім «вказує на еволюційний характер змін Сі», за словами Страуструпа. Рання мова програмування, не пов'язана з C++, називалася "C+".

Нововведеннями C++ порівняно з С є:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;
- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю.

Ядро мови та стандартна бібліотека є двома основними компонентами стандарту C++ для 1998 року. Бібліотека шаблонів STL, яка була створена одночасно зі стандартом, була прийнята до стандартної бібліотеки C++[9]. Однак серед програмістів C++ термін STL використовується для позначення розділу стандартної бібліотеки, який містить визначення шаблонів контейнерів, ітераторів, алгоритмів і функторів.

Стандарт С 1990 року є нормативним посиланням у стандарті C++, який не визначає функції стандартної бібліотеки, взяті безпосередньо зі стандартної бібліотеки С[11].

Переваги мови C++

- Швидкодія. Незважаючи на доступ до додаткових можливостей і інструментів, програми C++ навряд чи є повільнішими за програми С з точки зору продуктивності.

- Масштабованість. Мова програмування C++ використовується для створення програм для широкого діапазону платформ і систем.
- Робота з пам'яттю, адресами та портами на низькому рівні (що при неправильному використанні легко перетворюється на недолік).
- Потенціал для спеціалізації та виконання обчислень на етапі компіляції з використанням загальних алгоритмів для різних типів даних.
- Підтримуються різні мови програмування, фреймворки та технології, такі як метапрограмування, ООП, узагальнене програмування та класичне директивне програмування (шаблони, макроси).

Недоліки C++

- Оскільки існує так багато можливостей для порушення безпеки типу, програмам C++ досить легко включити помилку, яку важко виявити. Розробники змушені дотримуватися дуже складних вимог кодування, а не контролювати компілятор. Ці вказівки по суті обмежують C++ до більш безпечного налаштування. Більшість проблем безпеки типів C++ перенесені з C, хоча опозиція творця мови щодо автономного керування пам’яттю відіграє значну роль у цій проблемі (наприклад, збирання сміття). Як наслідок, уразливості типу «переповнення буфера» почали ідентифікувати C++.
- Погана підтримка модульності. При підключенні великої кількості модулів метод вставки файлу заголовка препроцесора (#include) для підключення інтерфейсу зовнішнього модуля значно сповільнює компіляцію.
- Багато компіляторів надають функцію попередньої компіляції файлів заголовків для усунення цієї вади (попередньо скомпільовані заголовки).
- Препроцесор C++, який він успадкував від C, досить простий [10]. Це призводить, з одного боку, до того факту, що деякі дії метапрограмування не можуть (або не можуть бути легко виконані) з його допомогою, а з іншого боку, через свою рудиментарну природу, це часто призводить до помилок і вимагає кількох кроків, щоб обійти потенційні проблеми. Деякі мови

програмування мають значно більш надійні та безпечні системи метапрограмування, такі як Scheme та Nemerle.

- Спільнота C++ побачила значне зростання використання так званого метапрограмування на основі шаблонів з кінця 1990-х років [11]. В основному він реалізує базовий функціональний інтерпретатор мови програмування, який працює під час компіляції з використанням характеристик шаблонів C++. Навіть якщо ця ідея досить приваблива сама по собі, дуже складно прийняти або не любити такий код через вищезгадані фактори. У порівнянні з іншими мовами програмування, Lisp/Scheme, Nemerle та деякі інші мають більш надійний та інтуїтивно зрозумілий механізм метапрограмування. Подібна підсистема метапрограмування шаблонів також реалізована в мові D, але вона значно простіше у використанні.
- Хоча C++ рекламується як багатопарадигмальна мова, функціональне програмування не підтримується цією мовою. Деякі бібліотеки (Loki, Boost), які використовують методи метапрограмування для додавання функціональних можливостей до мови (наприклад, підтримка лямбда-виразів/анонімних методів), частково усувають цей недолік, але їхня ефективність блідне в порівнянні з рішеннями, знайденими у функціональних мовах. Функціональні функції мови, такі як зіставлення із зразком, зазвичай дуже складно відтворити за допомогою метапрограмування.

3.1.2 Бібліотека Simple and Fast Multimedia Library

Бібліотека для розробки крос-платформного програмного забезпечення під назвою Simple and Fast Multimedia Library (SFML) була створена, щоб запропонувати простий інтерфейс прикладного програмування (API) для різних мультимедійних компонентів комп’ютерів. З прив’язками для Ada, C, Crystal, D, Euphoria, Go, Java, Julia,.NET, Nim, OCaml, Python, Ruby та Rust, він розроблений

на C++. З появою SFML 2.2 стали доступними експериментальні мобільні порти для iOS і Android [12].

Конструкція та введення вікон, а також керування контекстами OpenGL обробляються SFML.Крім того, він пропонує аудіомодуль, який використовує OpenAL, мережевий модуль для зв'язку за основним протоколом керування передачею (TCP) і протоколом дейтаграм користувача (UDP), а також графічний модуль для прямого апаратного прискорення 2D-комп'ютерної графіки, який включає рендеринг тексту за допомогою FreeType.

Програмне забезпечення, відоме як SFML, є безкоштовним із відкритим кодом і розповсюджується відповідно до ліцензії zlib/png. Він сумісний з Windows, Mac OS X, Linux і FreeBSD. Перше видання, версія 1.0, вийшло 9 серпня 2007 року, а остання версія, версія 2.5.1, вийшла 15 жовтня 2018 року [12].

На даний момент доступні такі модулі:

- Системний — усі модулі залежать від системного керування часом і потоками, отже, це важливо.
- Вікно — керування вікнами та взаємодією з користувачем.
- Завдяки графіці легко відображати графічні примітиви та зображення.
- Аудіо — пропонує інтерфейс керування аудіо.
- Мережа — для програм у мережі.

3.1.3 Інтегроване середовище розробки Visual Studio

Microsoft Visual Studio — це інтегроване середовище розробки (IDE). Окрім веб-сайтів, веб-додатків, веб-сервісів і мобільних додатків, він використовується для створення комп'ютерних програм [13]. Платформи розробки програмного забезпечення Microsoft, зокрема Windows Store, Windows Presentation Foundation, Windows API та Windows Forms, використовуються Visual Studio. Він може створювати як рідний, так і керований код.

Рефакторинг коду та IntelliSense (функція завершення коду) підтримуються редактором коду в Visual Studio. Налагоджувач на рівні джерела та на рівні машини

функціонують із вбудованим налагоджувачем. Профайлер коду, конструктор для розробки додатків графічного інтерфейсу користувача, веб-дизайнер, конструктор класів і конструктор схем бази даних є додатковими вбудованими інструментами. Він приймає плагіни, які додають функціональні можливості майже на всіх рівнях, як-от додавання підтримки для програм керування джерелами (наприклад, Subversion і Git) і нових наборів інструментів, таких як редактори та візуальні дизайнери для мов програмування зі спеціалізованими доменами, або набори інструментів для різних процесів розробки програмного забезпечення (наприклад, Azure, DevOps Client: Team Explorer).

Поки існує спеціальна служба для цієї мови, Visual Studio дозволяє редактору коду та налагоджувачу підтримувати (в різному ступені) практично кожну мову програмування на додаток до 36 інших мов програмування. C, C++, C++/CLI, Visual Basic.NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML і CSS – лише деякі з вбудованих мов. Серед інших мов через плагіни пропонується підтримка Python, Ruby, Node.js і M [13]. У минулому Java (і J#) підтримувалися.

Версія Visual Studio для спільноти є найбільш фундаментальною версією та є безкоштовною. «Безкоштовна повнофункціональна IDE для студентів, розробників із відкритим кодом і індивідуальних розробників», — йдеться у слогані Visual Studio Community Edition [14].

3.2 Опис модулів SFML

SFML надає простий інтерфейс до різних компонентів вашого комп'ютера, для полегшення розробки ігор і мультимедійних застосунків. Складається з п'яти модулів: системного, віконного, графічного, аудіо та мережевого.

3.2.1 Модуль SFML Graphics

Основним завданням графічного модуля SFML є малювання тексту, зображень, ліній, форм та інших об’єктів, які можна малювати, на екрані. Інші його

обов’язки включають створення спеціалізованих об’єктів, керування геометрією екрана та створення зовнішніх файлів (включаючи зображення та шрифти) у пам’яті.

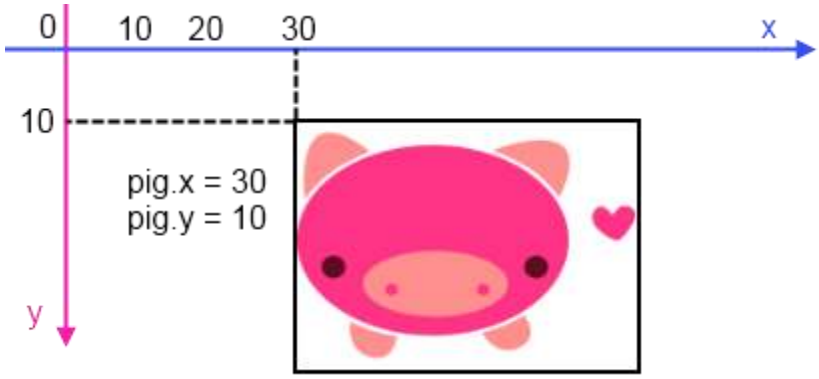


Рисунок 3.1 — Система координат

Верхній лівий кут екрана, де корінь системи координат, знаходиться на (0, 0). Більші значення розташовані далі праворуч на горизонтальній осі абсцис. Більші значення розташовані нижче по вертикальній осі у. Важливо мати на увазі, що розташування (0, 0) також вирівнюється за верхнім лівим кутом пікселя, а це означає, що під час створення ліній шириною 1 піксель ви можете зіткнутися з деякими проблемами відтворення для деяких функцій.

3.2.2 Система подій SFML

Для спрощення обробки дій користувача в SFML реалізовано систему подій. Під подією мається на увазі певна дія користувача, наприклад, натискання клавіші на клавіатурі (sf::Event::KeyPressed), клік мишки (sf::Event::MouseButtonPressed) або закриття вікна програми (sf::Event::Closed).

Програмно, подія є прикладом використання патерну проектування "Команда". Команда — це поведінковий патерн, який перетворює запит в об'єкт, що містить всю інформацію про запит. Цей патерн дозволяє передавати запити як

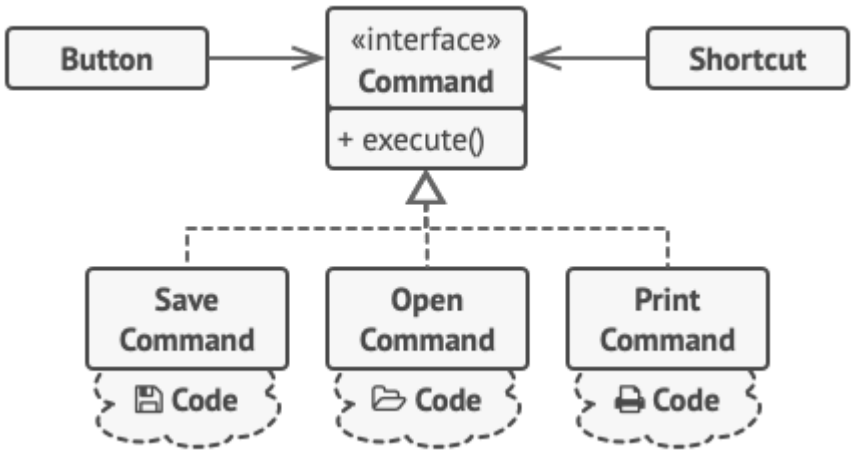


Рисунок 3.2 — Приклад патерну Команда

параметри методів або функцій, контролювати час виконання запиту (наприклад, відкласти його або створити стек запитів, що будуть виконуватись послідовно) і обробляти запити, що не можуть бути виконані[20].

3.3 Діаграма прецедентів

Діаграма варіантів використання — це графічне зображення можливої взаємодії користувача з системою [19].

Рисунок 3.3 — Діаграма прецедентів

Користувач має можливість згенерувати поле, перейти до наступного кроку і знайти найкоротший шлях.

3.4 Діаграма класів

Діаграма класів — це тип діаграми статичної структури, яка описує структуру системи, показуючи класи системи, їхні атрибути, операції (або методи) і зв’язки між об’єктами [19].

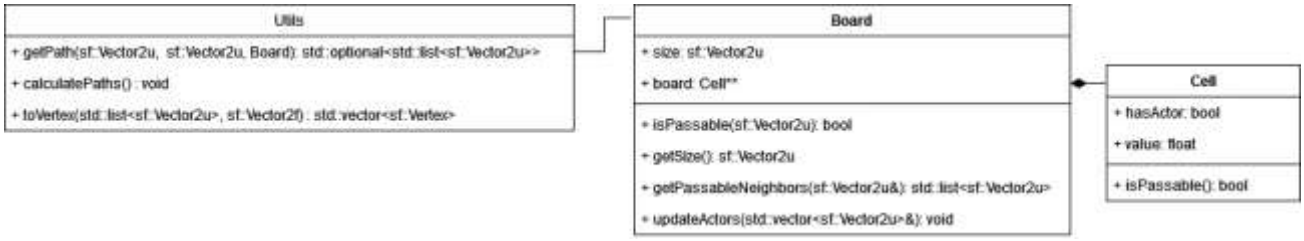


Рисунок 3.4 — Діаграма класів

клас Board — відповідає за сітку клітинного автомату
клас Cell — відповідно за клітинку
глобальна функція getPath (на діаграмі - в блоці Utils) — реалізація алгоритму A* для сітки клітин, використовує вкладену структуру node, що використовується в алгоритмі для відвіданих та не відвіданих комірок, для оцінки евристики використовується Евклідова відстань.

3.5 Модульна структура програми

Модульне програмування — парадигма програмування, орієнтована на зменшення складності програмних продуктів та можливості перенесення окремих рішень з одних програмних проектів у інші.

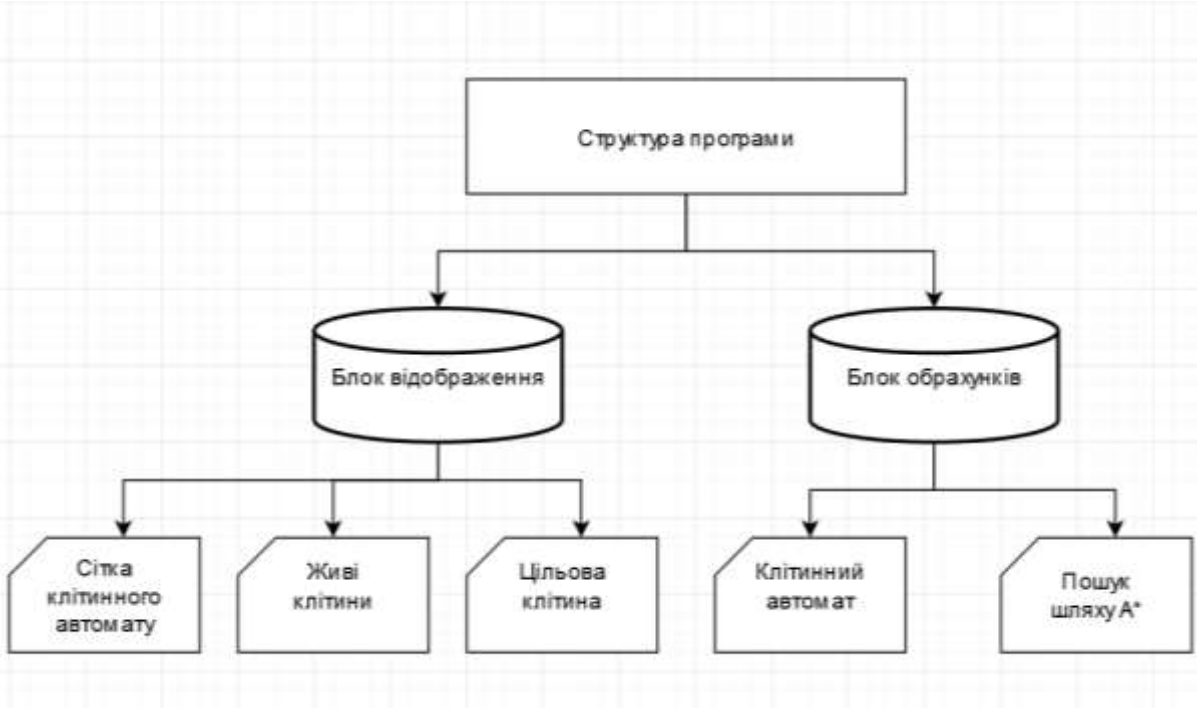


Рисунок 3.5 — Модульна структура програми

На рисунку 3.5 зображена модульна структура програми, що була створена при розробці програми.

3.6 Висновки до розділу

В даному розділі було розглянуто основні моменти з розробки програмного забезпечення, а саме опис обраної мови програмування та середовища розробки. Також було продемонстровано основу структури проекту у вигляді діаграми прецедентів та діаграми класів.

4 АНАЛІЗ РОБОТИ АЛГОРИТМУ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ ЗА ДОПОМОГОЮ КЛІТИННИХ АВТОМАТІВ

4.1 Опис роботи алгоритму

Надалі буде продемонстровано приклади роботи програми, а саме використання оптимального алгоритму пошуку у клітинних автоматах та опис різних ситуацій, в яких знаходиться початкова координата. На рисунках 4.1- 4.5 показано ітераційні дані, які були зроблені для першого випадку, а у рисунках 4.6-4.12 – другі.

Рисунок 4.1 — Початок роботи програми (випадок 1)

На рисунку 4.2 зображено знаходження шляху на другій ітерації алгоритму, де автомат наближається до своєї кінцевої цілі.

Рисунок 4.2 — Робота програми на першій ітерації (випадок 1)

На рисунку 4.3 продемонстровано вже третю ітерацію, де поступово агенти наближаються до кінцевого результату свого пересування.

Рисунок 4.3 — Робота програми на другій ітерації (випадок 1)

Далі за допомогою рисунку 4.4 можна побачити третю ітерацію пошуку найкоротшого шляху на ландшафті клітинних автоматів, де автомати далі просувається до своєї кінцевої цілі. На рисунку 4.5 два агенти досягли кінцевої точки.

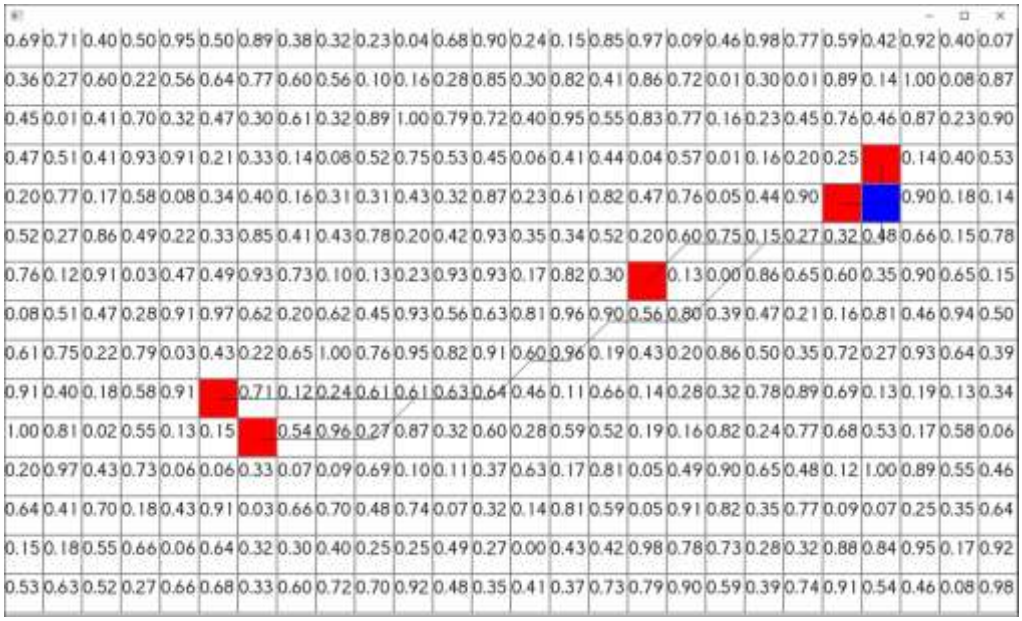


Рисунок 4.4 — Робота програми на третій ітерації (випадок 1)

Рисунок 4.5 — Робота програми при досягненні мети одночасно двома клітинами (випадок 1)

У другому випадку немає агентів, що досягнуть мети одночасно.

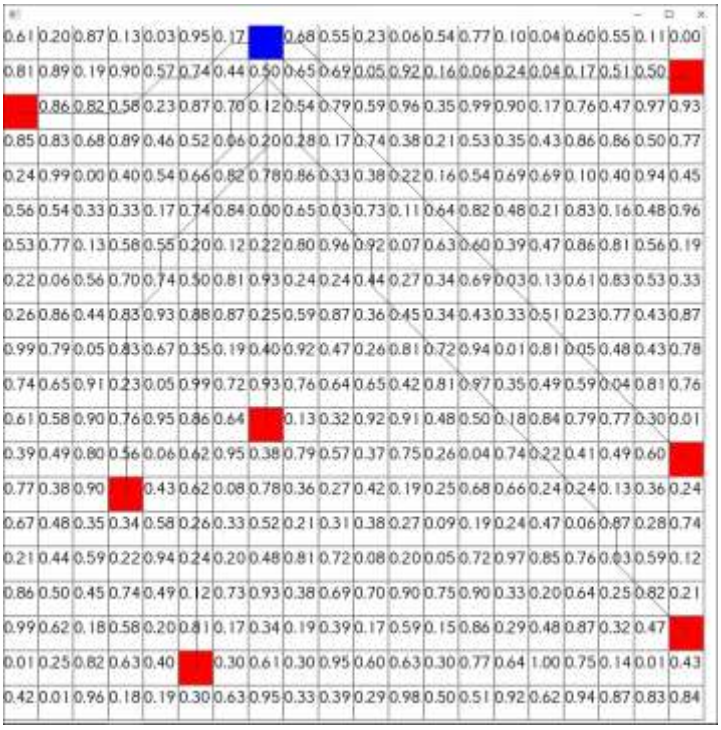


Рисунок 4.6 — Початок роботи програми (випадок 2)

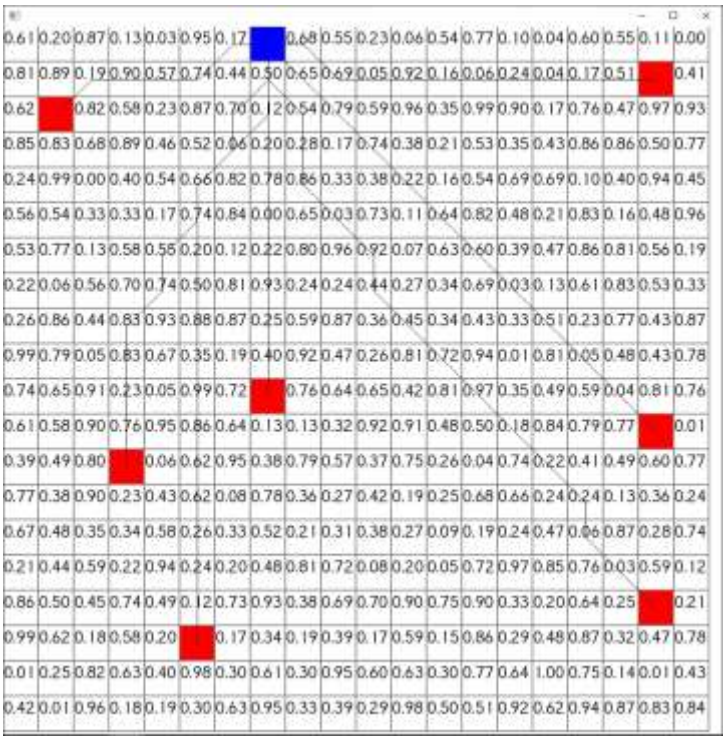


Рисунок 4.7 — Робота програми (випадок 2)

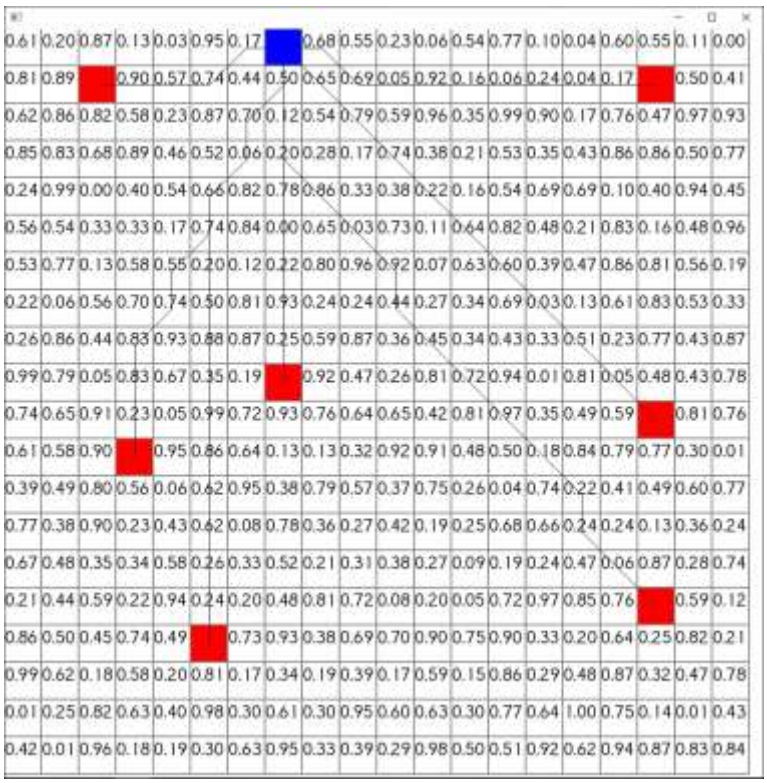


Рисунок 4.8 — Робота програми (випадок 2)

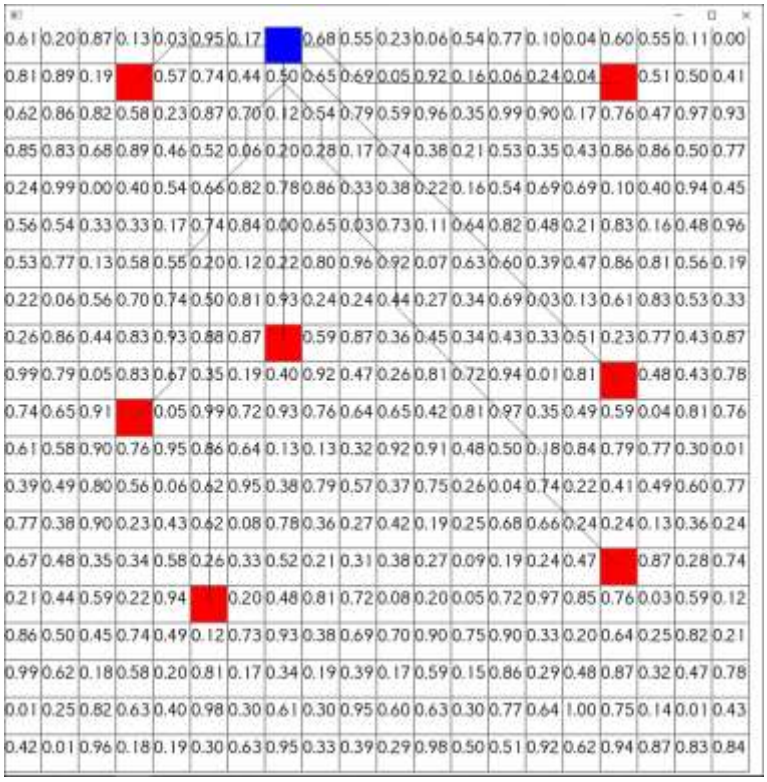


Рисунок 4.9 — Робота програми (випадок 2)

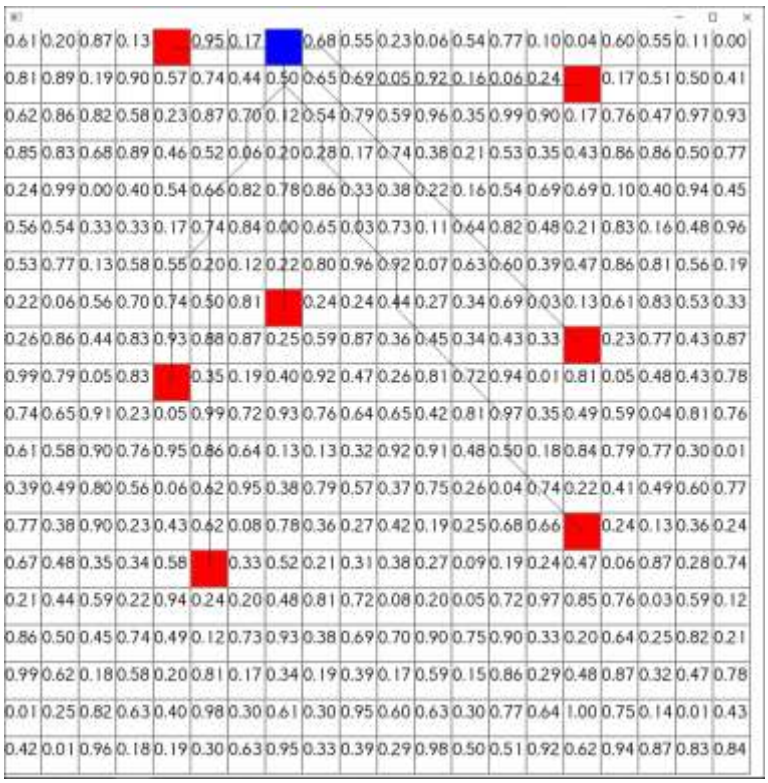


Рисунок 4.10 — Робота програми (випадок 2)

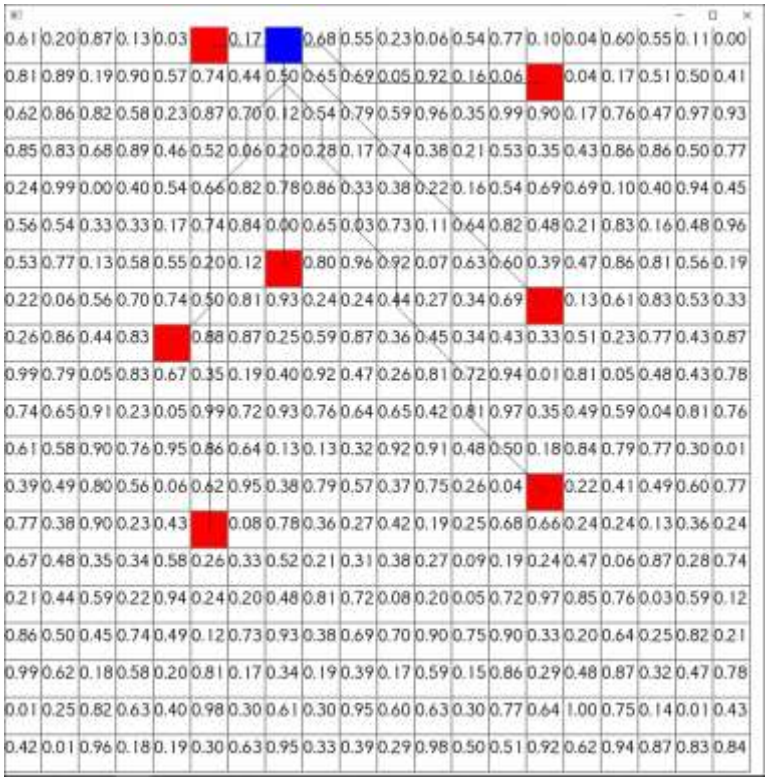


Рисунок 4.11 — Робота програми (випадок 2)

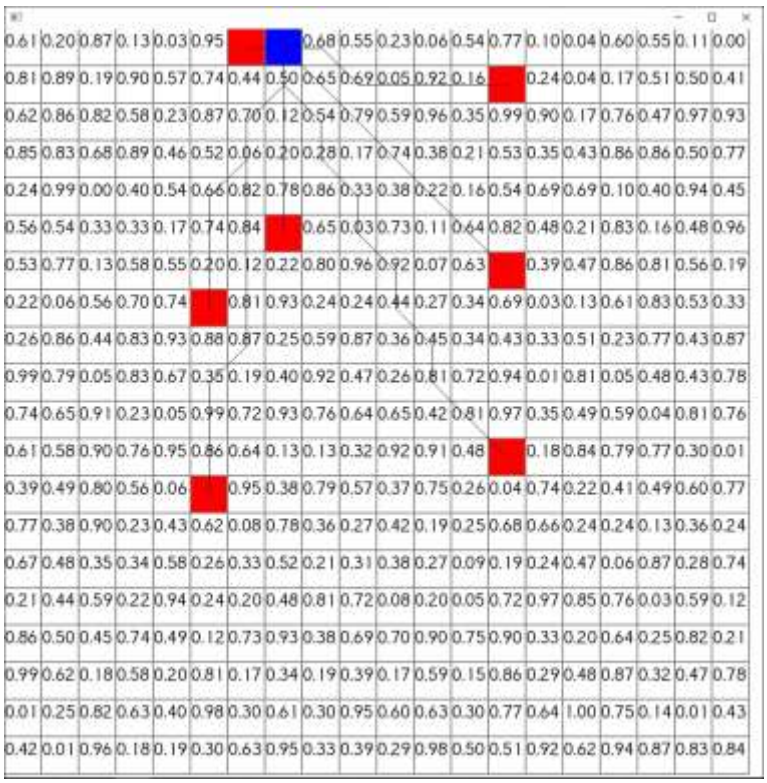


Рисунок 4.12 — Робота програми (випадок 2)

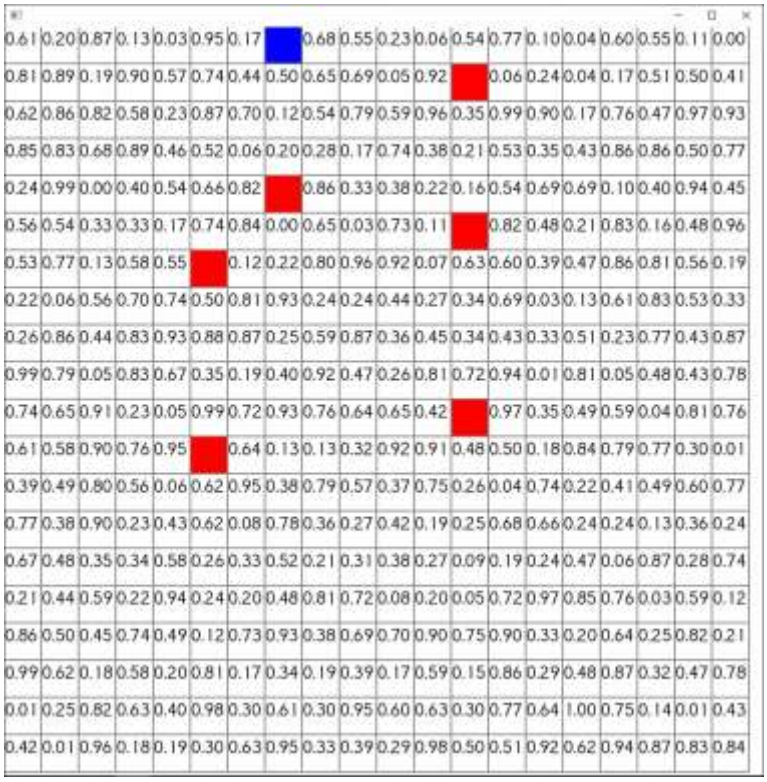


Рисунок 4.13 — Робота програми при знаходженні кінця шляху врізнобій

Після певної кількості ітерацій алгоритм знаходить найкоротший шлях на графі серед об’єктів, що рухаються, та різних заповнень вершин за допомогою клітинних автоматів

4.2 Опис обраного алгоритму та його переваги в користуванні з клітинними автоматами

Не зважаючи на існуючі рішення та підходи до знаходження найкоротшого шляху на графі, задача залишається актуальною, оскільки не існує єдиного підходу, що задовільнив би вимоги до розв’язку в різних областях застосування рішення. Розвиток клітинних апаратів дозволив застосувати його методи для вирішення поставленої задачі.

Визначаємо клітину як об’єкт який рухається по певному набору правил, що залежать від заповнення сусідніх клітин. Якщо на даній позиції є перешкода руху, то в залежності від того який об’єм займає фігура присвоюємо значення 1 або 0 в характеристичній матриці відповідної розмірності. Правила руху визначаються наявністю перешкоди, на скільки сильно ця перешкода заповнює простір (чи є можливість рухатись далі по клітинці, чи необхідно здійснювати обхід перешкод). Наприклад: нехай об’єм який заповнює фігура є менше за 0,9 умовних одиниць – тоді початковий об’єкт може рухатись в даному напрямку, якщо ж більше за 0,9 то прохід через дану клітину заборонено.

Запропонований підхід дозволяє використовувати апарат клітинних автоматів до вирішення задач не тільки пошуку мінімального значення, а й для визначення оптимального рішення задачі.

4.3 Висновки до розділу

В даному розділі було розглянуто основні моменти з використання програмного забезпечення, а саме продемонстровано ітеративний процес виконання програми для двох випадків досягання мети: одночасного та врізнобій.

5 РОЗРОБКА СТАРТАП ПРОЕКТУ

5.1 Опис ідеї проекту

Ми розглянемо зміст запропонованої ідеї, її потенційні можливості застосування, чим вона відрізняється від своїх попередників, а також ключові переваги, які може отримати користувач продукту. У таблиці 4.1 наведено результати аналізу.

Таблиця 5.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигода для користувачів
Розробити застосунок для пошуку найкоротшого шляху	Транспортна сфера	Можливість застосування для розробки маршрутів з найкоротшим шляхом
	Авіаційна сфера	Можливість застосування для розробки авіа маршрутів з найкоротшим шляхом

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Можливі конкуренти			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Запропонована архітектура	Google	Maps			
1	Визначення найкоротшого шляху	+	+	+			+

Продовження таблиці 5.2

2	Відображення кожного кроку	+	+	-			+
3	Можливість масштабування	-	+	+	+		
4	3D візуалізація даних	-	+	+	+		
5	Веб додаток	-	+	+	+		

Основними відмінностями є спосіб відображення кроків та результатів, а також швидкість і точність.

5.2 Технологічний аудит ідеї проекту

У цьому розділі перераховані технології, які можна використовувати для реалізації проекту. Результат наведено у таблиці 5.3.

Таблиця 5.3 – Технології для реалізації ідеї проекту

№	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Побудова клітинного автомату	Використовує вкладену структуру node, що використовується в алгоритмі для відвіданих та не відвіданих комірок, для оцінки евристики використовується Евклідова відстань. логіка відображення базується на примітивах бібліотеки SFML таких як sf::RectangleShape, sf::Font та sf::Text, знаходиться в	Необхідні: C++, SFML, Cmake	Доступні, безкоштовні
2	Алгоритм найкоротшого шляху			

Продовження таблиці 5.3

№	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
3	Логіка відображення та керування	функції main логіка керування програмою використовує систему подій бібліотеки SFML.		
4	Генерація випадкових чисел	Є абстракцією з використанням сучасного сахару (c++ 20 templates with concepts) над класами стандартної бібліотеки в тому числі mt19937 (Mersenne Twister pseudo-random generator of 32-bit numbers with a state size of 19937 bits.)	Необхідні: C++	Доступні, безкоштовні

Обрані технології реалізації ідеї проекту: C++, Cmake, Visual Studio, C++ 20; SFML, C++ STD. Обрані технології є фактично набором інструментів та фреймворків для реалізації мети додатку, не потребують доробки.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 5.4 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Економний аналіз великих шляхів	Малий і середній бізнес і незалежні власники бізнесу, які потребують аналізу даних, але не мають необхідних коштів для виконання цих завдань	- Наявність фінансування - Для аналізу потрібні різні обсяги даних	- Стабільність - Постійна підтримка - Впровадження оновлень
Обробка даних на пристроях з низькими технічними можливостями	Ті, хто хоче аналізувати дані, але не може дозволити собі хмарні обчислення		

Продовження таблиці 5.4

Зручне керування завантаженими наборами даних	Підприємства малого та середнього бізнесу, а також незалежні власники бізнесу	- Наявність фінансового забезпечення - Різний об’єм необхідних для аналізу даних	- Стабільність - Постійна підтримка Впровадження оновлень

Також необхідно оцінити фактори, які можуть завадити реалізації проекту. Оцінку наведено у таблиці 5.5.

Таблиця 5.5 – Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Поява конкурентів	Цілком можливо, що з’являться суперники, які зможуть виробляти продукт вищого калібру або розробляти вже популярну та визнану послугу. Можливо, почнуть з'являтися більш продуктивні продукти.	Розробка альтернативних методів аналізу даних з використанням технології екранування пам’яті та вдосконалення поточного рішення за допомогою нових технологій

Продовження таблиці 5.5

Фактор	Зміст загрози	Можлива реакція компанії
Відсутність репутації сервісу	Середні та малі підприємства можуть неохоче завантажувати свої статистичні дані в неперевірену службу, оскільки вони хвилюються, що до них можуть отримати доступ конкуренти.	Створення можливостей шифрування даних і отримання сертифіката безпеки від авторитетного стороннього аудитора

У таблиці 5.6 представлено фактори можливості системи.

Таблиця 5.6 – Фактори можливостей

Фактор	Зміст можливості	Можлива реакція компанії
Невелика кількість конкурентів	На ринку обробки великих даних зараз небагато конкурентів, і їхні пакети програмного забезпечення часто використовують горизонтальну кластеризацію для підвищення обчислювальних можливостей системи.	Розробка продукту спрямована на мінімізацію використання оперативної пам’яті та зниження витрат на обчислення

Продовження таблиці 5.6

Фактор	Зміст можливості	Можлива реакція компанії
Можливість побудови власної репутації	Не маючи попереднього досвіду, новий сервіс на ринку обробки великих даних має всі шанси створити власну репутацію.	Пошук клієнтів, поширення інформації про платформу та мінімальна націнка на калькуляційні витрати
Демонстрація вигоди використання розробленого додатка	Порівняння потужності та економічності розробленого сервісу з конкурентами	Створення веб-сторінки програми, яка містить детальну інформацію про робочі алгоритми програми, ефективність вибраних методів, візуалізацію графіка та порівняння швидкості, з якою обробляються запити, з аналогічними наборами даних у розробленому середовищі та середовищах, що використовуються конкурентами

Продовження таблиці 5.6

Демонстрація простоти користування розробленим додатком	Швидке занурення до інтерфейсу пошуку найкоротшого шляху	Створення безкоштовних обмежених за часом підписок для привернення уваги потенційних клієнтів до корисності розробленої системи
Зростання попиту на системи аналізу даних	Збільшення клієнтської бази	Привернення уваги потенційних користувачів та реклама

Майбутні виклики включають комерціалізацію комп’ютерних даних і розробку зручних для користувача інтерфейсів, що може зменшити кількість можливих споживачів. З кожним випуском швидкість, з якою обробляються запити та працює система в цілому, повинна збільшуватися, щоб протистояти цій загрозі.

Вивчення ринкової пропозиції має вирішальне значення перед приєднанням до ринку. Для цього необхідно з'ясувати загальні характеристики ринкового суперництва.

У таблиці 5.7 наведено ступеневий аналіз конкуренції на ринку.

Таблиця 5.7 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - чиста	Можливість необмеженої конкуренції на ринку	Можливість чесної конкуренції
2. За рівнем конкурентної боротьби – міжнародний	На ринку є іноземні компанії-конкуренти.	Створіть англомовний додаток, щоб залучити більше користувачів з-за кордону.
3. За галузевою ознакою – міжгалузева	Широкий спектр галузей може використовувати розроблену архітектуру.	Уніфікована підтримка сукупних запитів для кількох наборів даних
4. Конкуренція за видами товарів – товарно-видова	Види товарів схожі за функціональністю	Враховувати недоліки методів обробки даних
6. За характером конкурентних переваг – цінова	Ціна - дуже важливий фактор при виборі продукту	Приділяти більше уваги вартості розробки та впровадження
7. За інтенсивністю – немарочна	Бренд не впливає на впізнаваність продукції	

Крім того, для кращого розуміння ринку потрібне ретельне вивчення конкурентного середовища галузі. Результати проведеного аналізу наведені у таблиці 5.8.

Таблиця 5.8 – Аналіз конкуренції в галузі за М.Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Google	Apple	Значна кількість постачальників	Активно диктують умови	Збільшення кількості аналогічних товарів
Висновки:	Доволі відома компанія, тому конкуренція інтенсивна	Компанія має потенціал запуску технологій аналізу великих масивів даних	Постачальник не впливає на ринок	Мають основний вплив	Велика кількість товарів замінників

На основі результатів дослідження конкурентів було прийнято рішення щодо доцільності роботи на ринку, незважаючи на наявність конкуренції. Для подальшого розвитку проекту необхідно підвищити ефективність методів обробки даних. Дані результати були враховані при порівняльному аналізі факторів конкурентоспроможності в таблиці 5.9.

Таблиця 5.9 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Ціна	Використовуючи методи екранування пам’яті, розроблене архітектурне рішення має на меті знизити вартість обробки великих обсягів даних.
2	Швидкість виконання запитів	Використання методів віртуальних стовпців для аналізу даних і виконання запитів дуже швидко без копіювання чи дублювання даних у пам’яті серверного комп’ютера позитивно впливає на швидкість виконання запитів.
3	Швидкість завантаження наборів даних	Можливість обробки великих наборів шляхів

Використовуючи дані конкурентоспроможності проекту, необхідно проаналізувати сильні і слабкі сторони проекту (таблиця 5.10) та навести SWOT-аналіз стартап-проекту (таблиця 5.11).

Таблиця 5.10 – Порівняльний аналіз сильних та слабких сторін архітектурного рішення для аналізу великого обсягу статистичних даних на пристроях з низькими технічними можливостями

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з хмарними технологіями для даних						
			-3	-2	-1	0	+1	+2	+3
1.	Ціна	20		+					
2.	Швидкість виконання запитів	14					+		
3.	Швидкість завантаження наборів даних	16					+		
4.	Гнучкість маніпулювання набором даних	12			+				

Розроблений проект має переваги щодо мінімальних витрат на обчислення, швидкості виконання операцій та адаптивності при використанні набору даних. Наявність більш відомих конкурентів і початкова відсутність довіри до таких підприємств є слабкими сторонами.

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки для виведення стартап-проекту на ринок. Далі необхідно проаналізувати строки реалізації та ймовірність отримання ресурсів. Результати аналізу наведені у таблиці 5.12.

Таблиця 5.11 – SWOT-аналіз стартап-проекту

Сильні сторони: Поєднання методів швидкого завантаження даних, економічного використання дорогих серверних компонентів, що знижує вартість продукту, і найбільш ефективних способів обробки статистичних запитів	Слабкі сторони: Нова компанія, яка виходить на ринок і потребує сильної рекламної фірми
Можливості: Удосконалюються методи аналізу даних, створюються нові інструменти для управління наборами даних.	Загрози: Зниження вартості надання послуг за допомогою хмарних сервісів. вихід конкурентів, які використовуватимуть аналогічні методи обробки даних. Ціна жорстких дисків може бути значною.

Таблиця 5.12 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтований комплекс заходів) ринкової поведінки	Ймовірність отримання результатів	Строки реалізації
1	Створення мінімального продукту з набором найголовніших функцій та швидкий вихід на ринок	Висока	Від 2 до 5 місяців
2	Створення потужного сервісу з великою кількістю різноманітних операцій з даними	Середня	Від 8 до 12 місяців

Було розглянуто два основні підходи: створення продукту, який мав би всі заплановані можливості, або створення мінімального продукту для тестування ринку та вимірювання введення користувачів. Перший варіант було обрано, оскільки відгуки та переваги користувачів є вирішальним компонентом у визначенні найкращого курсу дій для розвитку проекту. Ми зможемо визначити, які додаткові функції потрібні майбутнім клієнтам, і зможемо задовольнити їхні потреби, не витрачаючи час на способи, які вони не використовуватимуть, оскільки ми вже створили основну функціональність системи.

5.4 Розроблення ринкової стратегії проекту

Для створення ринкової стратегії проаналізуємо цільові групи потенційних користувачів. Результати аналізу наведені у таблиці 5.13.

Таблиця 5.13 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Фізичні особи підприємці	Середня	Середній	Низька	Низька
2	Компанії малого бізнесу	Висока	Високий	Низька	Низька
3	Компанії середнього бізнесу	Середня	Низький	Висока	Середня

Які цільові групи обрано: в першу чергу необхідно звернути увагу на

компанії малого бізнесу, оскільки вони не мають можливості витрачати великі суми коштів на аналіз даних, але відчувають гостру необхідність.

На основі отриманої інформації розробимо базову стратегію розвитку. Результати аналізу наведено у таблиці 5.14.

Таблиця 5.14 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Вихід на профільний ринок з сервісом аналізу великих об’ємів даних	Масовий маркетинг серед компаній малого бізнесу	Комбінація рішення для аналізу даних при низькому використанні дорогої ОЗУ та зручні механізми маніпулювання даними, а також необмежений об’єм вхідних наборів даних	Стратегія лідерства по витратах

На наступному етапі необхідно обрати стратегії конкурентної поведінки. На основі базової стратегії розвитку розробимо стратегію конкурентної поведінки (таблиця 5.15).

Таблиця 5.15 – Визначення базової стратегії конкурентної перевірки

Чи є проект «першопрохідцем» на ринку?	Чи буде бізнес шукати нових клієнтів чи відбиратиме поточних у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Ні, проект не є першопрохідцем на ринку, але володіє унікальними методами обробки наборів даних	Компанія зробить свій продукт доступним як для нових, так і для існуючих клієнтів, які зараз користуються послугами конкурентів у цьому регіоні.	Компанія не буде копіювати конкурентів	Стратегія наслідування лідеру

За стратегію конкурентної поведінки було обрано стратегію наслідування лідеру, за допомогою якої вдасться уникнути боротьби з лідерами ринку.

За допомогою проведених досліджень визначена стратегія позиціонування з урахуванням основних вимог до проекту та стратегій його розвитку, які наведені у таблиці 5.16.

Таблиця 5.16 – Визначення стратегій позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Низька ціна, швидкість, функціональні можливості	Стратегія лідерства по витратах	Низька вартість обробки даних, висока швидкість, зручний інтерфейс роботи з датасетами	Економічність, простота, швидкість, зручність

У результаті основою ринкової стратегії буде залучення організацій малого бізнесу, які відчувають потребу аналізувати дані для майбутнього розвитку ринку, але не мають необхідної фінансової підтримки.

5.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 5.17 підсумовуються результати попереднього аналізу конкурентоспроможності проекту.

Таблиця 5.17 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Дешевий аналіз статистичних даних	Використання оперативної пам’яті сервера якомога менше зменшує витрати, пов’язані з отриманням аналізу результатів набору даних.	Зі збільшенням обсягу даних для аналізу всі конкуренти прагнуть використовувати більше оперативної пам'яті, що значно підвищує вартість обчислень.
2	Швидкі методи завантаження датасетів на сервер	Можливість завантаження архівів даних та окремих файлів на сервер, що забезпечує більш швидку та адаптовану модель маніпулювання даними.	Зазвичай вони швидко завантажують дані, але не ділять їх на менші частини, що призводить до частих збоїв під час завантаження дуже великих файлів даних. Причиною цього є непостійне підключення користувача до Інтернету.
3	Керування наборами даних	Зручна обробка та керування наборами даних досягається завдяки створенню інтуїтивно зрозумілого інтерфейсу користувача.	Подібні системи залежать від того, що користувач добре розуміє комп’ютери та їхні системи, що вимагає значного часу для ознайомлення з інфраструктурою проекту.

Надалі розробляється тривіальна маркетингова модель товару: уточняється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 5.18).

Таблиця 5.18 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td>Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td>Віддалений доступ та управління набором даних, низька вартість отримання результатів запитів, висока швидкість аналізу даних</td></tr>
<tr><td rowspan="3">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td></tr>
<tr><td>1. Додаток для пошуку найкоротшого шляху на графі</td></tr>
<tr><td>Марка: назва організації-розробника + назва товару</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td>До продажу: базова версія</td></tr>
<tr><td>Після продажу: постійна модернізація</td></tr>
<tr><td colspan="2">За рахунок чого потенційний товар буде захищено від копіювання: методи обробки та аналізу наборів даних, а також методи їх зберігання на сервері. Тобто захист ідеї товару буде відбуватися шляхом захисту інтелектуальної власності на комплексне поєднання властивостей і характеристик.</td></tr>
</table>

Наступним кроком є встановлення цінових орієнтирів, яких необхідно дотримуватися при встановленні ціни потенційного продукту (кінцева ціна визначається під час фінансово-економічного аналізу проекту). Цей крок включає аналіз вартості порівнянних або замінних продуктів, а також рівня доходів споживачів цільового ринку.

Таблиця 5.19 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Безкоштовний додаток	Зазвичай безкоштовний додаток	1 тис. дол – 7 тис. дол	200 дол. – 700 дол.

Наступним кроком є визначення оптимальної системи збуту (таблиця 5.20).

Таблиця 5.20 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Клієнти прагнуть отримувати товар за меншою ціною за рахунок конкуренції	Постійний доступ до сервісу	Однорівневий чи нульовий канал збуту	Власноруч та через посередників

Таблиця 5.21 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Аналіз даних у Excel	Інтернет, телефон	Моніторинг та прогнозування	Звернути увагу на ціну, зручність та швидкість порівняно з конкурентами	Дешевий та якісний аналіз даних

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування та визначену специфіку поведінки клієнтів (таблиця 5.21).

5.6 Висновки до розділу

В даному розділі було проведено аналіз створення стартапу на основі використання запропонованого алгоритму пошуку найкоротшого шляху на графі за допомогою клітинних автоматів. А саме описано ідею стартапу та проведено технічний аудит описаної ідеї. Також було проаналізовано ринкові можливості запуску такого стартап проекту, для чого було розроблено ринкову стратегію та маркетингову програму для даного стартап проекту.

ВИСНОВКИ

В даній роботі було продемонстровано та проаналізовано основні алгоритми пошуку найкоротшого шляху на графі та реалізовано у вигляді додатку що вирішує проблему пошуку найкоротшого шляху на графі у прикладі, який має вигляд двовимірного клітинного автомату. Було описано всі використані технології та кроки реалізації застосунку. Описано ідею стартапу для додатку. Всі отримані навички та знання, необхідні для побудови та використання даного продукту будуть подальші використані у нових або оновлених роботах.

У результаті аналізу, можна зробити висновок, що даний проект є досить перспективним, так як сфера аналізу даних зростає з кожним роком, а вартість обчислень лише збільшується. Існуючі альтернативні рішення є значно дорожчими у використанні і тому стають не досить привабливими для ведення бізнесу. Завдяки здешевленню аналізу даних можна залучити нові компанії, які раніше не мали можливості користування даною сферою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Identification Of Cellular Automata by Adamatzky, A. Taylor & Francis, 1994.
2. Solving routing problems with cellular automata by Reinhard W. Hoffmann & William C. Hochberger, 1996.
3. Lee, C.Y.: An algorithm for path connections and its applications. IRE Transactions on Electronic Computers EC-10(2), 346–365 (1961)
4. Beigy, H., Meybodi, M.R.: Utilizing distributed learning automata to solve stochastic shortest path problems. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 14(05), 2006.
5. "A note on two problems in connexion with graphs" by Dijkstra, E. W., 1959, S2CID 123284777.
6. Bellman-Ford Algorithm. Дата оновлення: 25.11.2021 – URL: https://www.simplilearn.com/tutorials/data-structure-tutorial/bellman-ford-algorithm (дата звернення 28.11.2022).
7. The Quest for Artificial Intelligence by Nilsson, Nils J., 2009-10-30, ISBN 9780521122931.
8. M. N. S. Swamy, K. Thulasiraman Graphs, Networks, and Algorithms — Wiley, 1981p. ISBN-13 978-0471035039.
9. "Lecture: The essence of C++. University of Edinburgh" by Stroustrup, Bjarne from 28 April 2015.
10. C++ Applications by Stroustrup, Bjarne, 17 February 2014.
11. The C++ Programming Language (4th Edition) by Stroustrup, Bjarne, 9 May 2013, ISBN 0-201-88954-4.
12. SFML Documentation Версія 2.5.1 [Електронний ресурс] – URL: https://www.sfml-dev.org/documentation/2.5.1/ (дата звернення 28.11.2022).
13. Visual Studio 2022 – URL: https://visualstudio.microsoft.com/vs/ (дата звернення 28.11.2022)

14. "Visual Studio Product Lifecycle and Servicing" Microsoft Docs. Дата оновлення 30.11.2022 – URL: https://learn.microsoft.com/en-us/visualstudio/productinfo/ vs-servicing (дата звернення 01.12.2022).
15. "Cellular Automata" Stanford Encyclopedia of Philosophy. Дата оновлення 22.08.2017 – URL: https://plato.stanford.edu/entries/cellular-automata/ (дата звернення 01.12.2022).
16. Amoroso, Serafino / Patt, Yale N. Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures. - J. Comput. Syst. Sci.
17. "Cellular Automata Applications in Shortest Path Problem" by Michail-Antisthenis Tsompanas, Nikolaos I Dourvas, Konstantinos Ioannidis, Georgios Ch. Sirakoulis, December 2017, DOI: 10.1007/978-3-319-77510-4_8
18. Ford L. Flows in Networks / Ford L., Fulkerson D. — Princeton: Princeton University Press, 1962.
19. The unified modeling language reference manual by James Rumbaugh, Ivar Jacobson, Grady Booch., Addison-Wesley. 1990. ISBN 0-201-30998-X
20. Behavioral Patterns – Command [Електронний ресурс] – URL: https://refactoring.guru/design-patterns/command (дата звернення 28.11.2022)

ДОДАТОК А

Пошук найкоротшого шляху на графі за допомогою клітинних автоматів

ТЕЗИ

Аркушів 2

Київ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УКРАЇНСЬКА АСОЦІАЦІЯ З ПРИКЛАДНОЇ ГЕОМЕТРІЇ

МЕЛІТОПОЛЬСЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО

МЕЛІТОПОЛЬСЬКА ШКОЛА ПРИКЛАДНОЇ ГЕОМЕТРІЇ

ТЕЗИ ДОПОВІДЕЙ

24 МІЖНАРОДНОЇ
НАУКОВО – ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

СУЧАСНІ ПРОБЛЕМИ ГЕОМЕТРИЧНОГО МОДЕЛЮВАННЯ

УКРАЇНА, МЕЛІТОПОЛЬ
08-09 ВЕРЕСНЯ 2022 р.

Ванін В.В., д.т.н.,
Залевська О.В., к.т.н.,
Захаркін М. С.,
Куйбіда П.К.
Національний технічний університет України Київський політехнічний інститут імені Ігоря Сікорського(Україна)
Чжан Мінцзюнь
Інститут обробки інформації (Information Research Institute), Технологічний університет Цілу - Шаньдунська академія наук (КНР)

ПОШУК НАЙКОРОТШОГО ШЛЯХУ НА ГРАФІ ЗА ДОПОМОГОЮ КЛІТИННОГО АВТОМАТУ

Не зважаючи на існуючі рішення та підходи до знаходження найкоротшого шляху на графі, задача залишається актуальною, оскільки не існує єдиного підходу, що задовільнив би вимоги до розв'язку в різних областях застосування рішення. Розвиток клітинних апаратів дозволив застосувати його методи для вирішення поставленої задачі.

При такому підході визначаємо клітину як об'єкт який рухається по певному набору правил, що залежать від заповнення сусідніх клітин. Якщо на даній позиції є перешкода руху, то в залежності від того який об'єм займає фігура присвоюємо значення 1 або 0 в характеристичній матриці відповідної розмірності. Правила руху визначаються наявністю перешкоди, на скільки сильно ця перешкода заповнює простір (чи є можливість рухатись далі по клітинці, чи необхідно здійснювати обхід перешкод). Наприклад: Нехай об'єм який заповнює фігура є менше за 0,5 умовних одиниць – тоді початковий об'єкт може рухатись в даному напрямку, якщо ж більше за 0,5 то прохід через дану клітину заборонено. Запропонований підхід дозволяє використовувати апарат клітинних автоматів до вирішення задач не мільки пошуку мінімального значення, а й для визначення оптимального рішення задачі.

ДОДАТОК Б

Пошук найкоротшого шляху на графі за допомогою клітинних автоматів

СТАТТЯ

Аркушів 14

Київ 2022

Міністерство освіти і науки України

Українська асоціація з прикладної геометрії

Мелітопольський державний педагогічний університет
імені Богдана Хмельницького

Мелітопольська школа прикладної геометрії

СУЧАСНІ ПРОБЛЕМИ МОДЕЛЮВАННЯ

ЗБІРНИК НАУКОВИХ ПРАЦЬ

Випуск 24

Наукове фахове видання
(категорія Б)

Мелітополь – 2022 р.

УДК [51+514+721+004.92]–047.58(062.552)
ББК 22.1я5
С 91

Свідоцтво про державну реєстрацію друкованого засобу масової інформації: Серія КВ № 21030-10830Р від 29.09.2014 р.
Збірник наукових праць включено до Переліку наукових фахових видань України з технічних наук (наказ Міністерства освіти і науки України № 241 від 09.03.2016)

Рекомендовано до друку та поширення через мережу Інтернет Вченою радою МДПУ імені Б. Хмельницького, протокол № 3 від 27 вересня 2022 р.

С 91 Сучасні проблеми моделювання: зб. наук. праць / МДПУ ім. Б. Хмельницького; гол. ред. кол. Ю.М. Ковальов. – Мелітополь: Видавництво МДПУ ім. Б. Хмельницького, 2022.– Вип. 24. – 203 с.

Збірник містить статті за результатами досліджень з теорії та практики моделювання, розглядаються актуальні наукові та прикладні проблеми геометричного моделювання, методика постановки та проведення наукових та дослідницьких експериментів, результати наукових досліджень, питання підготовки фахівців та науковців.

Випуск призначений для науковців, викладачів, аспірантів і студентів.

УДК [51+514+721+004.92]–047.58(062.552)
ББК 22.1я5

ISSN 2313-125X

ЗМІСТ

УДК 518.14

ВИКОРИСТАННЯ АЛГОРИТМІВ КЛІТИННИХ АВТОМАТІВ В ЗАДАЧАХ ПОШУКУ МІНІМАЛЬНОГО ШЛЯХУ

Ванін В.В., д.т.н.,
vaninvladimir30@gmail.com, ORCID: 0000-0001-7008-
7269 Залевська О.В., к.т.н.,
o.zalevska@kpi.ua, ORCID: 0000-0002-3163-
1695 Захаркін М.С.
zaharmisha70@gmail.com, ORCID: 0000-0003-4609-8130
Куйбіда П.К.,
pavel.kuybida@gmail.com ORCID:0000-0002-4767-
3937 Zhu Shiwei

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»(Київ,Україна)

У роботі розглядається використання клітинних автоматів для удосконалення п ошуку найкоротшого шляху між об'єктами, на шляху до яких є перепони. В запропонованому алгоритмі, за основу взято клітинний автомат, а саме гра
«Життя). Правила побудови структури клітинного автомату вимагають змін, в залежності від вимог, що стоять перед користувачем. В статті наведений можлививай варіант пошуку найкоротшого шляху за домопогою клітинного автомату на прикладі гри. Метою гри є дістатисьь певної наперед заданої клітинки ігрового поля за мінімальну кількість кроків.Задача має багато аналогів та використовується в різних сферах науки від звичайних ігр до закрітіх військових стратегій.
Розглянуті існуючі аналоги аглоритмів пошуку найкоротшого шляху володіють рядом недоліків. До них можна віднести роботу лише з

структурою графів, знаходження найкоротшого шляху тільки з однієї вершини та інші. Також існує недолік, що притаманний всім методам - вони завжди намагаються просуватися в напрямку мети, навіть якщо це неправильний шлях. Ці недоліки приводять до збільшення часу реалізації алгоритму, застрягання персонажів в кутках та між перепонами.

В наведенному алгоритмі поле розглядається як елемент поля клітинного автомату. На першому кроці випадковим чином генеруються перешкоди, що не знакають протячгом всієї гри. Визначаємо клітину, як об'єкт який рухається по певному набору правил, що залежать від заповнення сусідніх клітин. Якщо на даній позиції є перешкода руху, то в залежності від того який об'єм займає фігура присвоюємо значення 1 або 0 в характеристичній матриці відповідної розмірності. Правила руху визначаються наявністю перешкоди, на скільки сильно ця перешкода заповнює простір (чи є можливість рухатись далі по клітинці, чи необхідно здійснювати обхід перешкод).

При наведеному алгоритмі зникають наведені недолі алгоритму, а швидкість виконання значно зменшується.

Ключові слова. *Мінімальний шлях, клітинний автомат, правила побудови клітинних автоматів, алгоритми пошуку мінімальної відстані.*

Постановка проблеми. Визначення оптимального маршруту між об'єтками може здійснюватися в статичному, динамічному режимах і режимах реального часу. Це питання розглядається в багатьох важливих програмах у сфері GPS, відеоігор, робототехніки, логістики та симуляції натовпу - часові середовища. Проблема пошуку шляху може мати багато різних форм, включаючи ті, що стосуються одного агенту, групи агентів, конкурентного пошуку, динамічних змін навколишнього середовища, неоднорідної місцевості, мобільних пристроїв і неповна інформація. Алгоритм пошуку шляху та створення графіка для його реалізації є двома основними компонентами пошуку рушення проблеми.

Аналіз останніх досліджень та публікацій. Розглянемо існуючі алгоритми

знаходження мінімальнаго щляху між об’єктами. Алгоритм Дейкстри — це метод визначення найкоротшого маршруту між вершинами графа[11]. З початкової точки об'єкта починається робота. кожного разу перевіряє найближчу недосліджену вершину, додаючи найближчі вершини до списку вершин, які слід враховувати. Якщо на графі немає ребер із від’ємними значеннями, цей алгоритм завжди визначає найкоротший маршрут між початковою точкою та бажаним пунктом призначення [18].

Порівняльна простота реалізації алгоритму Дейкстри, низька поліноміальна обчислювальна складність і потенціал для машинної реалізації є його перевагами. До недоліків алгоритму можна віднести наступне:

- працює тільки зі структурами графів, які мають невід'ємні довжини дуг; - шукає найкоротші шляхи та їх значення лише з однієї вершини;
- щоб обчислити ТК між усіма парами вершин, необхідно повторити процес для кожної пари вершин у вихідному графі N разів;
- не шукає рефлексивних замикань;
- має досить складне програмне представлення

Алгоритм Беллмана-Форда має часову складність O (V cdot E), де V — кількість вершин у графі, а E — кількість ребер [19]. Виконується V кроків, що є причиною цієї складності. Дальше проходимо всі ребра графа на кожній фазі.

Ключовою перевагою алгоритму Беллмана-Форда є його здатність працювати з негативними вагами. Алгоритм Беллмана-Форда набагато складніше зрозуміти, ніж метод Дейкстри. Через те, що графіки з від’ємними вагами зустрічаються відносно рідко, підхід Дейкстри має більше застосувань.

Алгоритм Беллмана-Форда може обробляти орієнтовані та неорієнтовані графи з невід’ємними вагами, як було зазначено раніше. Однак, якщо негативних циклів немає, він може обробляти лише орієнтовані графи з негативними вагами [19].

Формулювання цілей статті. Проаналізувати специфіку роботи алгоритмів пошуку найкоротшого шляху, розглянути засоби реалізації програмної системи, описати та розробити основні функції програмної реалізації в якій буде застосовуватися обраний алгоритм.

Основна частина. Визначення маршруту може здійснюватися в статичному, динамічному режимах і режимах реального часу, і воно є основною частиною багатьох важливих програм у сферах GPS, відеоігор, робототехніки, логістики та симуляції натовпу - часові середовища. За останні 20 років численні інновації підвищили точність і ефективність методів пошуку шляху, але це питання продовжує залишатися в центрі інтенсивних досліджень.

Проблема пошуку шляху може мати багато різних форм, включаючи ті, що стосуються одного агенту, групи агентів, конкурентного пошуку, динамічних змін навколишнього середовища, неоднорідної місцевості, мобільних пристроїв і неповна інформація. Кожне з цих питань має унікальне застосування в багатьох сферах. Алгоритм пошуку оптималного шляху та створення графічноъ ынтерпритацыъ є двома основними компонентами пошуку шляху.

Цей метод визначає клітину автомату як рухомий об’єкт, який відповідає набору правил, які залежать від заповнення сусідніх клітин. Якщо в цьому місці є перешкода для руху, то значенню клытини присвоюэться значеняння обэму який займає перешкода. Дані заносяться до характеристичної матриці. Характеристистична матриця повністю визначає рух персонажа.

Наведемо діаграму прецендентів та класів застосунку, що реалізують наведений алгоритм (рис.1 та рис.2).

Рисунок 1 — Діаграма прецедентів

Перш за все користувач має можливість згенерувати поле, перейти до наступного

кроку і знайти найкоротший шлях.

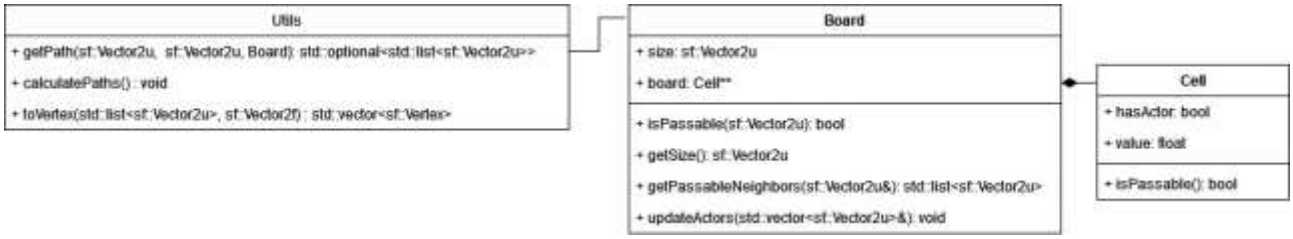


Рисунок 2 — Діаграма класів

- Отже, застосунок буде складатись з таких модулів як:
- клас Board — відповідає за сітку клітинного автомату
- клас Cell — відповідно за клітинку
- глобальна функція getPath (на діаграмі - в блоці Utils) — реалізація алгоритму A* для сітки клітин, використовує вкладену структуру node, що використовується в алгоритмі для відвіданих та не відвіданих комірок, для оцінки евристики використовується Евклідова відстань.

За допомогою запропонованого методу можна розв’язувати задачі, використовуючи апарат клітинних автоматів для знаходження найкращого рішення, а не лише мінімального значення.

Розглянемо приклад використання пошуку оптимального алгоритму у клітинних автоматах та опис різних ситуацій, в яких знаходиться початкова координата. На рисунку 3 показано ітераційні дані, які були зроблені для

першого випадку, а у рисунку 4 – другі та на рисунку 5 вказано знайдений шлях.

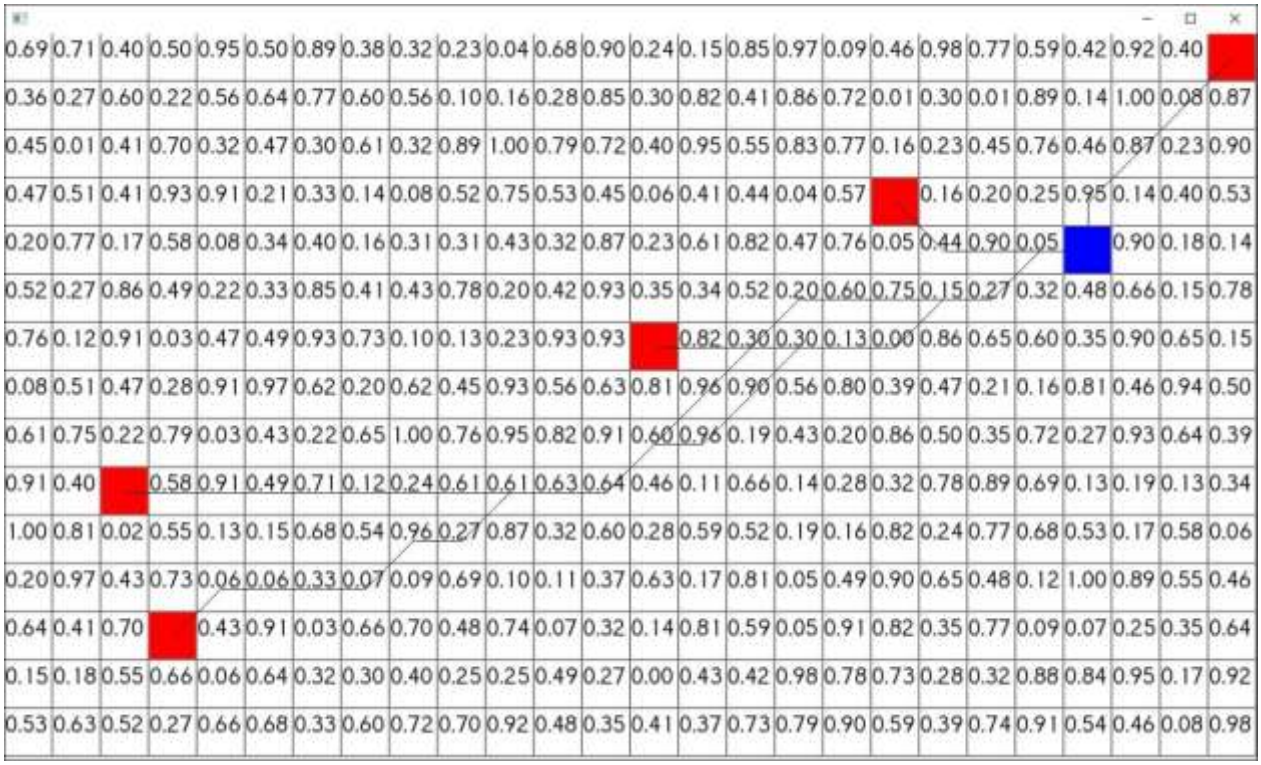


Рисунок 3. Початок роботи програми. Випадок 1.

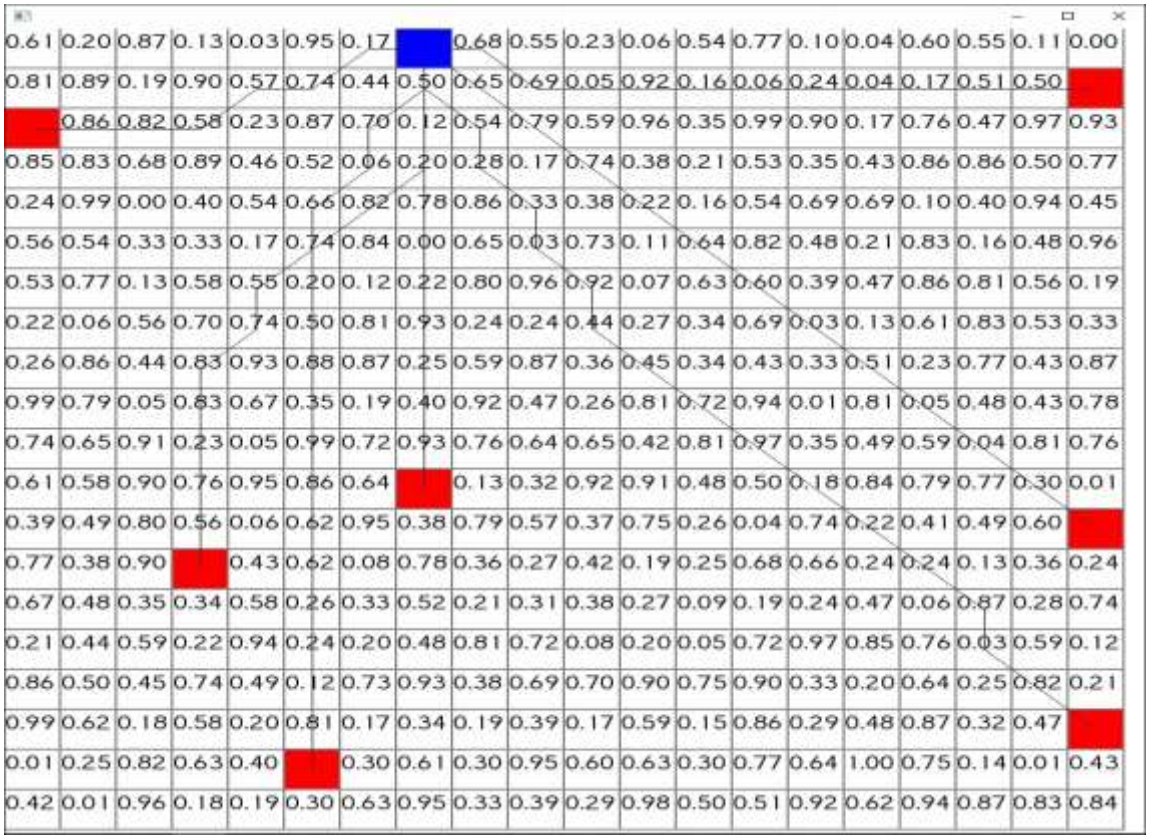


Рисунок 4. Перша ітерація реалізації другого випадку побудованого алгоритму.

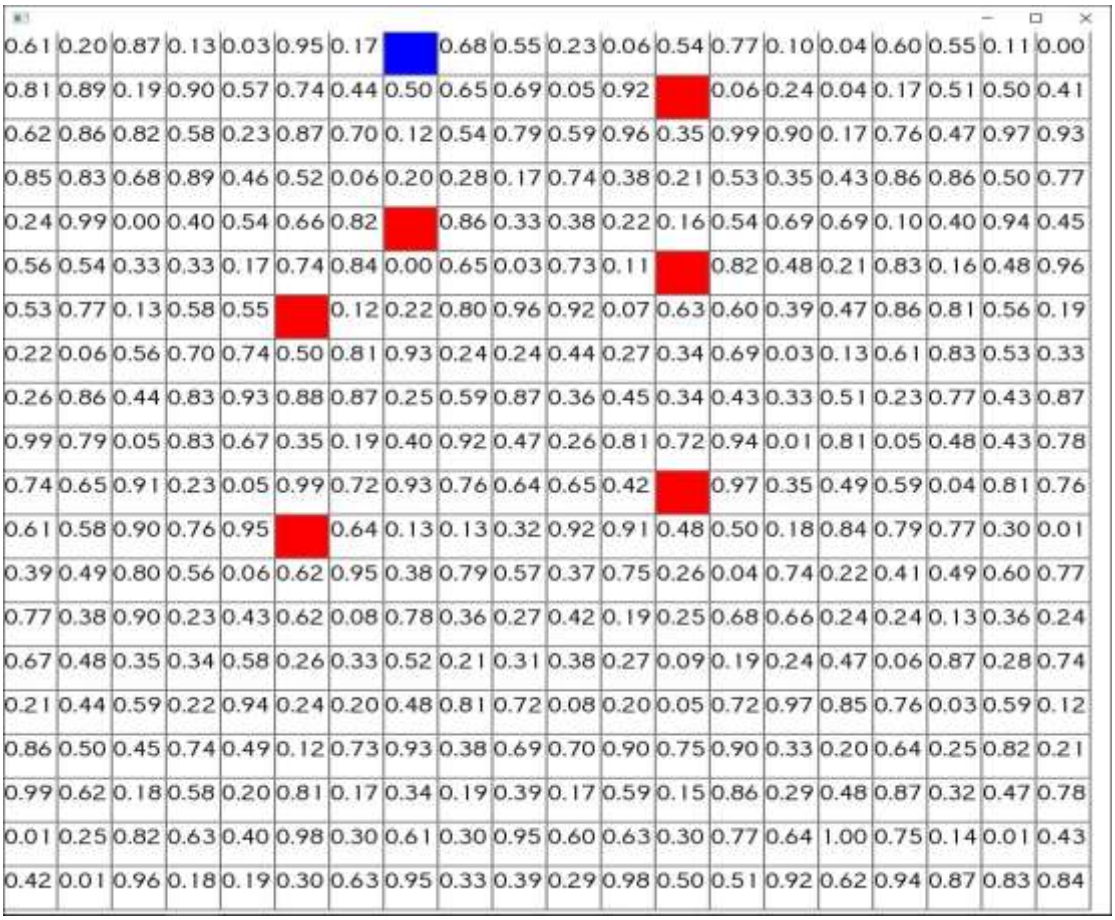


Рисунок 5. Результат роботи програми. Випадок 2.

Висновки. Запропонований спосіб знаходження оптимального мінімального шляху з перешкодами між двох об'єктів є швидший за існуючі аналоги та позбавлений таких недоліків як робота тільки зі структурами графів, які мають невід'ємні довжини дуг, пошук найкоротшого шляху та значення лише з однієї його вершини. Встановлені правила побудови клітинного автомату дають можливість не лише знаходження шляху між вершинами, а й дослідити автомат на оптимальність знайденого шляху.

Література.

1. Structure in communication nets. Proceedings of the Symposium on Information Networks (1955), Shimbel A.

2. O. Zalevska *et al.*, "Construction and Study of the Mathematical Model for the System Using Three-Dimensional Cellular Automata,"(2021) *2021 IEEE 16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, pp. 49-52, doi:

10.1109/CADSM52681.2021.9385235.

3. Ванін, В. В., Залевська, О. В., & Чередніченко, В. О. (2019). СТРУКТУРА ТРИВИМІРНОГО КЛІТИННОГО АВТОМАТУ ДЛЯ ПОБУДОВИ ЗОБРАЖЕННЯ ДИНАМІЧНИХ СИСТЕМ. *Сучасні проблеми моделювання*, (15), 43-50. https://doi.org/10.33842/2313-125X/2019/15/43/150

4. Залевська , О., Фіногенов , О., Ібнухсейн , І., & Суворова , В. (2021). ВИКОРИСТАННЯ ЗАСОБІВ ОНЛАЙН ТЕХНОЛОГІЙ ДЛЯ ПОБУДОВИ ТРИВИМІРНИХ КЛІТИННИХ АВТОМАТІВ. *Сучасні проблеми моделювання*, (22), 39-47. https://doi.org/10.33842/22195203/2021/22/39/47

5. "Experiments with the Graph Traverser program" by Doran, J. E.; Michie, D. 1966-09-20. ISSN 0080-4630. S2CID 21698093.

6. The Quest for Artificial Intelligence by Nilsson, Nils J., 2009-10-30, ISBN 9780521122931.

7. A* Search Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/a-search-algorithm/.

8. Dijkstra's Shortest Path Algorithm - A Detailed and Visual Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual- introduction/ (дата звернення 28.11.2022).

9. M. N. S. Swamy, K. Thulasiraman Graphs, Networks, and Algorithms — Wiley, 1981p. ISBN-13 978-0471035039.

10. Labeling Algorithm for Shortest Paths on Road Networks. / [Abraham I., Delling D., Goldberg A., Werneck R.]. - Philadelphia. - Symposium on Experimental Algorithms, 2011. — pp. 230-241.

11. Dijkstra E. A note on two problems in connexion with graphs. // In.: Numerische Mathematik. - 1959. - V. 1.— pp. 269-271.

12. Ford L. Flows in Networks / Ford L., Fulkerson D. — Princeton: Princeton University Press, 1962. — 253 p.

13. Bellman-Ford Algorithm Visually Explained. Дата оновлення: 08.07.2020 – URL: https://blog.devgenius.io/bellman-ford-algorithm-visually- explained-e940b6edb00a (дата звернення 28.11.2022).

14. A. Makarenko, B. Goldengorin and D. Krushinsky, "Game ‘Life’ with Anticipation Property", *Cellular Automata. ACRI 2008. Lecture Notes in Computer Science*, vol. 5191, 2008, [online] Available: https://doi.org/10.1007/978–3-540-79992-4_10

USE OF CELLULAR AUTOMATA ALGORITHMS IN MINIMUM PATH SEARCH PROBLEMS

V. Vanin, O. Zalevska, M. Zakharkin, P. Kuibida Zhu Shiwei

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (Kyiv, Ukraine)

The paper considers the use of cellular automata to improve the search for the shortest path between objects with obstacles on the way. The proposed algorithm is based on a cellular automaton, namely the game "Life). The rules for constructing the structure of the cellular automaton require changes, depending on the requirements of the user. The article presents a possible variant of finding the shortest path with the help of a cellular automaton using the example of a game. The aim of the game is to reach a certain predetermined cell of the playing field for a minimum number of steps.The problem has many analogues and is used in various fields of science from ordinary games to fixed military strategies.

The considered existing analogues of shortest path search agglomerations have a number of drawbacks. These include working only with the structure of graphs, finding the shortest path from only one vertex and others. There is also a drawback

that is inherent in all methods - they always try to move in the direction of the goal, even if it is the wrong path. These disadvantages lead to an increase in the time of implementation of the algorithm, stuck characters in the corners and between obstacles.

In the above algorithm, the field is considered as a field element of a cellular automaton. In the first step, obstacles are randomly generated, which do not appear throughout the game. We define a cell as an object that moves according to a certain set of rules that depend on the filling of neighboring cells. If there is an obstacle at a given position, then depending on the volume occupied by the figure, we assign a value of 1 or 0 in the characteristic matrix of the corresponding dimension. The rules of movement are determined by the presence of an obstacle, how much this obstacle fills the space (is it possible to move further along the cell, or is it necessary to bypass the obstacles).

With this algorithm, the above disadvantages of the algorithm disappear, and the speed of execution is significantly reduced.

Keywords. Minimum path, cellular automaton, rules for constructing cellular automata, algorithms for finding the minimum distance.

References

1. Structure in communication nets. Proceedings of the Symposium on Information Networks (1955), Shimbel A.

2. O. Zalevska et al., "Construction and Study of the Mathematical Model for the System Using Three-Dimensional Cellular Automata,"(2021) IEEE 16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), pp. 49-52, doi: 10.1109/CADSM52681.2021.9385235.

3. Vanin, V. V., Zalevska, O. V., & Cherednichenko, V. O. (2019). THE STRUCTURE OF A THREE-DIMENSIONAL CELLULAR AUTOMATON FOR IMAGE CONSTRUCTION OF DYNAMICAL SYSTEMS. Modern problems of modeling, (15), 43-50. https://doi.org/10.33842/2313-125X/2019/15/43/150

4. Zalevska, O., Finogenov, O., Ibnukhsein, I., & Suvorova, V. (2021). The use of online technologies for the construction of three-dimensional cellular automata. Modern problems of modeling, (22), 39-47. https://doi.org/10.33842/22195203/2021/22/39/47

5. "Experiments with the Graph Traverser program" by Doran, J. E.; Michie, D. 1966-09-20. ISSN 0080-4630. S2CID 21698093.

6. The Quest for Artificial Intelligence by Nilsson, Nils J., 2009-10-30, ISBN 9780521122931.

7. A* Search Algorithm [Electronic resource] - Access mode to the resource: https://www.geeksforgeeks.org/a-search-algorithm/.

8. Dijkstra's Shortest Path Algorithm - A Detailed and Visual Introduction [Electronic resource] – Access mode to the resource: https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/ (accessed 28.11.2022).

9. M. N. S. Swamy, K. Thulasiraman Graphs, Networks, and Algorithms - Wiley, 1981. ISBN-13 978-0471035039.

10. Labeling Algorithm for Shortest Paths on Road Networks (2011) / [Abraham I., Delling D., Goldberg A., Werneck R.]. - Philadelphia. - *Symposium on Experimental Algorithms,* - pp. 230-241.

11. Dijkstra E. A note on two problems in connection with graphs: Numerische Mathematik (1959).- V. 1.- pp. 269-271.

12. Ford L. Flows in Networks / Ford L., Fulkerson D. - *Princeton: Princeton University Press*, 1962. - 253 p.

13. Bellman-Ford Algorithm Visually Explained. Date of update: 08.07.2020 - URL: https://blog.devgenius.io/bellman-ford-algorithm-visually-explained-e940b6edb00a (accessed 28.11.2022).

ДОДАТОК В

Пошук найкоротшого шляху на графі за допомогою клітинних автоматів

АКТ ВПРОВАДЖЕННЯ

Аркушів 1

Київ 2022

ТОВ підприємство з іноземними інвестиціями
ПРИЗМА – 13
Україна, м. Київ, пр-т. Голосіївський , 100/2
Тел./Факс : (+38 044) 462 48 18

IBAN рахунок:
UA343006140000026002500298954
в ПАТ "ПАТ "КРЕДІ АГРІКОЛЬ БАНК",
м.Київ, МФО 300615
Свідоцтво ПДВ 37074905
ІПН 216490826504; Код ЕДРПОУ 21649087

LTD Venture with foreign investments
PRISMA – 13
pr. Golosiyevskiy, 100/2, Kyiv, Ukraine
Tel. / Fax: (+38 044) 462 48 18

IBAN:
UA343006140000026002500298954
"CREDIT AGRICOLE", Kiyv, Code 300614

Evidence 37034905
Ind. 216490826504; Code 21649087

№ ___ від "___"_____2022 г.

Проректору
КПІ ім.Ігоря Сікорського
Олексію ЖУЧЕНКО

Довідка

про впровадження результатів наукових досліджень
в практику діяльності ТОВ "ПРИЗМА-13"ЛТД з іноземними інвестиціями

У магістерській роботі студента Куйбіди Павла Костянтиновича, навчально-наукового інституту атомної та теплової енергетики кафедри цифрових технологій в енергетиці Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», що виконана на тему: «Пошук найкоротшого шляху на графі за допомогою клітинних автоматів» запропоновано ряд заходів щодо оптимізації роботи товариства та створення нового напрямку його роботи.

Практичну цінність має вивчення та дослідження динаміки викидів шкідливих виробництв від промислових об'єктів, а також прогнозування викидів на найближче майбутнє. Запропонована модель оцінки промислової стабільності буде взята за основу роботи нового відділу підприємства.

Окремо слід зазначити можливість прогнозування викиду шкідливих речовин на найближчі роки, що забезпечує можливість їх врахування при аналізі рентабельності та стабільності роботи підприємства в майбутньому.

Надані рекомендації, що базуються на проведеному аналізі, та пропозиції, щодо створення нового відділу товариства, врахування екологічної стабільності на інших підприємствах України будуть використані в подальшій практичній діяльності підприємства.

Заступник директора з IT _____ Майстерко Ігор Васильович