Застосування аналізу часової складності алгоритмів охоплює широкий спектр галузей, включаючи комп'ютерну науку, штучний інтелект, обробку сигналів, оптимізацію, криптографію та інші[4]. Вивчення часової складності допомагає розробити швидкі та ефективні алгоритми для важливих задач і сприяє розвитку нових технологій.

У практиці аналізу часової складності алгоритмів використовуються різні математичні методи та інструменти, такі як О-нотація, теорія ймовірностей, статистика, математичне моделювання та інші. Для складних алгоритмів може бути складно точно визначити їх часову складність аналітично, тому часто використовуються емпіричні методи, які базуються на експериментах та спостереженнях[5].

Аналіз часової складності алгоритмів є важливою складовою комп'ютерних наук та інших суміжних галузей. Вивчення та розуміння часової складності допомагає розробникам та дослідникам покращити ефективність алгоритмів, зменшити витрати обчислювальних ресурсів і розробити нові інноваційні рішення.

Список літератури:

1. *Часова складність алгоритму [Електронний ресурс] – Режим доступу: http://surl.li/iofrq*
2. *Горошко Ю. В. Про часову складність алгоритмів. Комп'ютерно-орієнтовані системи навчання : зб. наук. праць. Київ : Вид-во НПУ імені М. П. Драгоманова, 2015. Вип. 15(22). С. 27-31.*
3. *Антонов Дзігора http://r.donnu.edu.ua/bitstream/123456789/1479/1/ForConvert_001.pdf*

УДК 004.6

Гончар А. А., студентка 1 курсу
Спеціальності 122 «Комп'ютерні науки»
Науковий керівник:
Гончар В. М., асистент кафедри
Інформаційних технологій

АЛГОРИТМ ЗНАХОДЖЕННЯ ОПТИМАЛЬНИХ ВЕРШИН ГРАФА

Донецький національний університет імені Василя Стуса, м. Вінниця

Проектування складних систем, вивчення їхніх властивостей та управління ними вимагає розробки математичних моделей. Вивчення властивостей систем за допомогою математичних моделей часто є єдиним способом дослідження складних систем і вирішення найважливіших практичних завдань. Застосування теорії графів для побудови математичних моделей обумовлено тим, що можна описати широкий спектр об'єктів і процесів. Це дає можливість автоматично

шукати оптимальні маршрути, цільову доступність і мережеве планування. Одним із напрямків таких додатків є відеоігри, які вимагають розробки інтерфейсів, візуалізації рухів користувача та персонажа, розробки алгоритму та розрахунку рухів рухомих персонажів, в процесі застосовуються ігри у жанрі Roguelike.

Цілі роботи. Удосконалення алгоритму A* з усуненням недоліків алгоритму за допомогою методу проходу всіх вершин зваженог неорієнтованого графу.

Аналіз останніх досліджень і публікацій. Виходячи з [2], усі відомі алгоритми розв'язування заданої задачі можна розділити на 3 групи:

• алгоритм Дейкстри (для знаходження оптимального шляху між двома вершинами);

• алгоритм Форда-Бельмана (для пошуку найкращого шляху між усіма парами вершин);

• алгоритмі Флойда-Уоршалла (для знаходження найкоротшого шляху між парами вершин).

При створенні відповідного графа ми вважаємо відстань між вершинами нескінченною, а самі вершини ізольованими. За вагу вершини ми беремо кількість ітерацій, які забезпечують доступ до вершини. Ми дивимося на графік від верхньої лівої вершини до нижньої правої. Для кожної вершини ми перевіряємо, чи є вона кінцевою вершиною чи перешкодою.

Висновки та перспективи. Алгоритм A* був вдосконалений, тому з'явилася можливість вивчати динаміку кінцевих вершин, враховуючи поведінку всіх персонажів і забезпечуючи взаємодію гравця з ігровою картою. Наведений алгоритм може бути використаний при розробці програмного забезпечення робототехніки.

Алгоритм A* використовує допоміжні евристичні функції для спрямування напрямку пошуку, що значно скорочує час роботи алгоритму Дейкстри. Алгоритм A* проходить по всіх вершинах графа, переконавшись у знаходженні оптимального рішення (якщо воно існує) [3]. Згідно з алгоритмом, вершини класифікуються на три категорії відповідно до знайдених шляхів: невідомі вершини, відомі вершини та повністю досліджені. Вершини, чий шлях відомий, вважаються відомими, тоді як вершини, чий найкоротший шлях відомий, вважаються досліджуваними [4].

Головна частина. На початку алгоритму всі вершини вважаються невідомими. Спочатку розгляньте вузли, суміжні з початковим вузлом; виберіть серед них вузол із найменшим значенням f(x) і відобразіть цей вузол як відомий. На кожному етапі алгоритм використовує набір шляхів від початкової вершини до всіх ще не відкритих вершин графа, розміщених у пріоритетній черзі. Пріоритет шляху визначається значенням функції $f(x)=g(x)+h(x)$, де $g(x)$ – вага вершини, а вартість шляху від початкової вершини , $h(x)$ — функція евристичної формули для оцінки вартості шляху від вершини x до кінцевої вершини. Вага вершини відноситься до кількості можливих шляхів від початкової вершини до даної вершини [5].

Усі перераховані вище алгоритми працюють для невеликої кількості вершин. Ці алгоритми не рекомендуються, оскільки час їх дії експоненціально збільшується зі збільшенням кількості вершин.

Алгоритм A* враховує як відстань між вершинами, так і оцінку шляху, що визначає його використання як основу в даному завданні. При знаходженні шляху між вершинами алгоритм A* обходить мінімальну кількість вершин, оскільки використовує «оптимістичну» оцінку шляху через вершини. Оптимістичний означає, що під час обходу вершин він гарантує, що справжнє значення результату ніколи не буде меншим за задане значення евристичної функції. Ми встановлюємо евристичну функцію як евклідову відстань між двома точками на прямій.

Для реалізації алгоритму будемо використовувати інформований пошук. Інформований пошук – це стратегія пошуку рішень в просторі визначених станів, що відносяться до конкретного завдання [5].

Інформацію про конкретну задачу будемо заносити до евристичного функціоналу, за допомогою якого, на кожному кроці будемо приймати рішення про продовження алгоритму чи його завершення.

Під час обробки вершин ми зберігаємо дані в стек і використовуємо їх для подальших досліджень. Використання стека замість стандартної черги в алгоритмі A* забезпечує швидший доступ до будь-якої вершини дослідження.

Удосконалений алгоритм має включати в себе наступні кроки:

1. Створення двох списків вершин – ті що очікують розгляду і вже досліджені. У перший список додається вершина старту, список досліджених вершин на момент старту алгоритму є порожнім.

2. Для розрахунку кожної вершини знаходиться значення функціоналу $f(x)=g(x)+h(x)$, де $g(x)$ - відстань від старту до вершини, $h(x)$ - приблизну відстань від вершини до мети. Кожна вершина зберігає посилання на точку, з якої в неї прийшли.

3.Зі списку вершин на розгляд вибирається вершина з найменшим значенням функціоналу $f(x)$. Позначимо її X.

4. Якщо X - мета, то маршрут знайдено.

5. Переносимо X зі списку тих що чекають на розгляд у список вже досліджених.

6. Для кожної з вершини X суміжні вершини позначимо як У. Покроково розглядаємо вершини У:

6.1.Якщо Y вже знаходиться в списку досліджуваних вершин - пропускаємо її.

6.2.Якщо Y ще немає в списку вершин, що очікують дослідження - додаємо до її та враховуємо зв’язок з вершиною X через змінні $Y.g(x)$ та $Y.h(x)$.

6.3.Якщо ж Y є в списку на розгляд – перевіряємо: якщо X. $g(x)$ + відстань від X до Y < $Y.g(x)$, ми прийшли в точку Y коротшим шляхом. Тоді замінюємо $Y.g(x)$ на X. $g(x)$ + відстань від X до Y, а точку, з якої прийшли в Y, на X

7. Маршруту не існує, якщо список вершин на дослідження є порожнім, а поставлена мета не досягнута.

Список літератури

1.Stuart Russell, Peter Norvig Artificial Intelligence: A Modern Approach, 2004, Prentice Hall, ISBN 3-8273-7089-2

2.P. E. Hart, N. J. Nilsson, B. Raphael A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics SSC4 (2), pp. 100—107, 1968.

3.P. E. Hart, N. J. Nilsson, B. Raphael Correction to «A Formal Basis for the Heuristic Determination of Minimum Cost Paths», SIGART Newsletter, 37, pp. 28-29, 1972.

4.Anthony Stentz Optimal and Efficient Path Planning for Partially-Known Environments, Original D paper, ICRA International Conference on Robotics and Automation, 2022.*

5. Introduction to the A Algorithm, URL: https://www.redblobgames.com/pathfinding/a-star/introduction.html.*

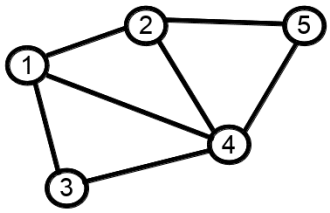
УДК 004.6

Бевзюк А.Ю., студентка 1 курсу
спеціальності 122 «Комп'ютерні науки»
Науковий керівник:
Гончар В.М., асистент кафедри
інформаційних технологій

ПОБУДОВА І ВИКОРИСТАННЯ МАТРИЦЬ СУМІЖНОСТІ І МАТРИЦЬ ВІДСТАНЕЙ

Донецький національний університет імені Василя Стуса, м. Вінниця

Дискретна математика — це розділ математики, який має справу з дискретними об'єктами, такими як цілі числа, графи та множини. Одним з підрозділів даної дисципліни, є теорія графів, у якій матриці суміжності та матриці відстаней є двома важливими інструментами для представлення та аналізу графів.



	1	**2**	**3**	**4**	**5**
1	0	1	1	1	0
2	1	0	0	1	1
3	1	0	0	1	0
4	1	1	1	0	1
5	0	1	0	1	0

Рис.1 Приклад неорієнтованого графа та його матриці суміжності

У теорії графів матриця суміжності — це квадратна матриця, яка представляє граф. Матриця має рядок і стовпець для кожної вершини в графі, а запис у позиції (i, j) дорівнює 1, якщо між вершинами i та j є ребро, і 0 в іншому випадку. Матрицю суміжності можна побудувати і для орієнтованого графа, і для неорієнтованого, і для графа з петлями [1].