Sprawozdanie – Projekt dodatkowy

Analiza algorytmów Dijkstra oraz A\*

Mikołaj Lepsy

Nr indeksu: 279941

# Wstęp

W tej pracy chciałbym przeanalizować działanie algorytmów szukających najkrótszej ścieżki: Dijkstra oraz A\*. Do analizy owych algorytmów użyjemy wizualizacji za pomocą pakietu Pygame oraz przygotowanych map. W Sprawozdaniu głęboko rozpatrzymy działanie owych algorytmów oraz pochylimy się nad kwestią wpływu „kosztów” drogi na działanie algorytmów.

# Algorytm A\*

Na zajęciach dogłębnie został przeanalizowany algorytm Dijkstry, który składa się z następujących kroków:

1. Dopóki zbiór Q nie jest pusty:
   1. Pobierz ze zbioru Q wierzchołek o najmniejszym koszcie dotarcia. Oznacz go jako v i usuń ze zbioru Q.
   2. Dla każdej krawędzi wychodzącej z wierzchołka v (oznaczmy ją jako k) wykonaj następujące czynności:
      1. Oznacz wierzchołek znajdujący się na drugim końcu krawędzi k jako u.
      2. Jeśli koszt dotarcia do wierzchołka u z wierzchołka v poprzez krawędź k jest mniejszy od aktualnego kosztu dotarcia do wierzchołka u, to:
         1. Przypisz kosztowi dotarcia do wierzchołka u koszt dotarcia do wierzchołka v powiększony o wagę krawędzi k.
         2. Ustaw wierzchołek v jako poprzednik wierzchołka u

Algorytm A\* ( A star ) jest rozszerzeniem powyższego algorytmu. Działanie A\* zależy od równania: które opisuje „predykcję” drogi do celu. Funkcja ta składa się z funkcji g(n), która opisuje aktualny „koszt” przebytej drogi oraz z funkcji h(n), która zwana jest funkcją heurestyczną.

Funkcja heurestyczna może być opisywana na wiele sposobów, jednak najczęściej wyróżnia się dwa: odległość Manhatanowska oraz odległośc Euklidesowa.

Dzięki funkcji heurestycznej, algorytm może stwierdzić, czy znajduje się bliżej celu, czy dalej oraz w którą strone powinien się kierować.

Obraz zawierający zrzut ekranu, diagram, tekst, Wielobarwność

Opis wygenerowany automatycznie

*Obraz 0: Przykładowe działanie algorytmu A\**

# Wyniki

W pracy chciałbym przedstawić wyniki przejścia obu algorytmów na 4 mapach:

W wizualizacjach mapy mają swoje zasady:

1. Bariera jest niedostępna i nie można przez nią przejść ( czarny kwadrat )
2. Istnieją 4 rodzaje ścieżek, każda o różnych wagach ( odcienie żółtego )
3. Początek jest pomarańczowy, cel – turkusowy
4. Przejścia horyzontalne NIE SĄ DOZWOLONE ( podejście Manhatanowskie )

Obraz zawierający wzór, zrzut ekranu, kwadrat, piksel

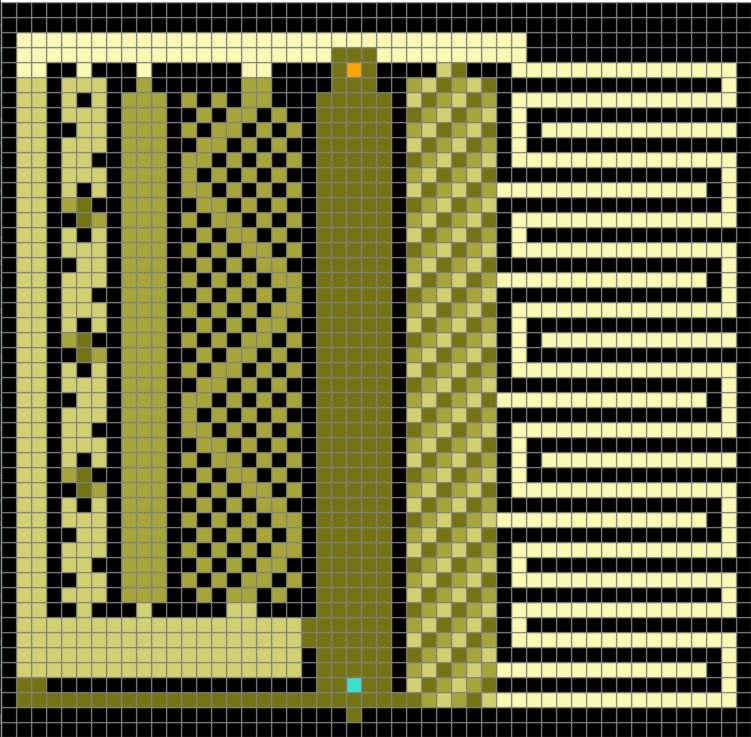
Opis wygenerowany automatycznie

*Obraz 1: Mapa 1 – Dolina. Jasne – najłatwiejsze przejście. Ciemne – najtrudniejsze przejście*

*Obraz zawierający wzór, ścieg, kwadrat, Symetria

Opis wygenerowany automatycznie*

*Obraz 2: Mapa 2 – labirynt, łatwo się zgubić*

**

*Obraz 3: Mapa 3 – ścieżki. 7 różnych ścieżek o różnym koszcie przejścia*

Poniżej w tabelach zostały przedstawione wyniki obu algorytmów po znalezieniu najkrótszej (według nich ) ścieżek:

|  |  |  |
| --- | --- | --- |
| Mapa 1 | Dijkstra | A\* |
| Przejrzane Nody | 797 | 1010 |
| Czas | 14.6644 | 16.5194 |

*Tabela 1: Wynik na mapie 1. Znalezione ścieżki są identyczne. Dijkstra wygrywa*

|  |  |  |
| --- | --- | --- |
| Mapa 2 | Dijkstra | A\* |
| Przejrzane Nody | 784 | 659 |
| Czas | 13.8672 | 12.1898 |

*Tabela 2: Wynik na mapie 2. Znalezione ścieżki są identyczne. A\* wygrywa*

|  |  |  |
| --- | --- | --- |
| Mapa 3 | Dijkstra | A\* |
| Przejrzane Nody | 1064 | 1034 |
| Czas | 18.6330 | 18.3981 |

*Tabela 3: Wynik na mapie 1. Znalezione ścieżki są identyczne. Remis*

Można zauważyć, że algorytmy działają dosyć podobnie, jednak przygotowana mapa 4 ukazuje największą zaletę A\*, ponieważ Dijkstra podczas porównywania sąsiadujących z nią Nodów, zawsze szuka tego o najmniejszej wadze.

Obraz zawierający wzór, Prostokąt, Symetria, kwadrat

Opis wygenerowany automatycznie

*Obraz 4: Mapa dodatkowa – słabość Dijkstry*

Między celem a początkiem, znajdują się Nody o niskiej wadze, reszta dostępnych jest o wadze najniższej. Dzieki funkcji heurestycznej która „podpowiada” algorytmowi w którą strone powinien się kierować, może on przejść przez ścieżkę o większej wadze, zanim sprawdzi wszystkie Nody o wadze najniższej.

|  |  |  |
| --- | --- | --- |
| Mapa 4 | Dijkstra | A\* |
| Przejrzane Nody | 722 | 144 |
| Czas | 20.5084 | 2.2348 |

*Tabela 4: Wynik na mapie 4. Znalezione ścieżki bardzo się różnią. A\* zdecydowanie wygrywa*

# Dodatkowe konkluzje

Można zauważyć, że oba algorytmy są bardzo podobne, jednak A\* zdobył swoją popularność w przemieszczaniu przestrzeni pustych, w których jest wiele możliwych ścieżek do obrania. Przykład:

Obraz zawierający Wielobarwność, zrzut ekranu, kwadrat, Prostokąt

Opis wygenerowany automatycznie

*Obraz 5.1: Podejście A\**

*Obraz zawierający Wielobarwność, wzór, zrzut ekranu, linia

Opis wygenerowany automatycznie*

*Obraz 5.2: Podejście Dijkstry*

Algorytm A\* jest zdecydowanie szybszy.

Dodatkową uwagą jeszcze może być różnica wag, wtedy działanie algorytmu zmienia się w zależności od mnożnika wag

Kod pisałem sam, dlatego proszę o wyrozumiałość w kwestii niektórych bugów, jeszcze nie jest na wszystkie warianty przetestowane, niektóre funkcjonalności jeszcze powinny być dodane, lecz nie byłem w stanie rozwiązać pewnych problemów z przekazywaniem argumentów. Kod jest po angielsku ( zawsze tak piszę ), uczyłem się i wzorowałem z następujących materiałów:

[https://pl.wikipedia.org/wiki/Algorytm\_A\*](https://pl.wikipedia.org/wiki/Algorytm_A*)

<https://www.youtube.com/watch?v=abHftC1GU6w>

<https://www.youtube.com/watch?v=dQw4w9WgXcQ>