Grafy

Konrad Łakomy

22 05 2014

Zadanie do wykonania

Implementacja struktury grafu oraz algorytmów przeszukiwania:

1. Przeszukiwanie grafu w głąb DFS (ang. Depth-first search)
2. Przeszukiwanie grafu wszerz BFS (ang. Breadth-first search)
3. Wyszukiwanie najkrótszej drogi A* (ang. A-Star)

Opis implementacji grafu

Graf (ang. graph) jest strukturą zbudowaną z punktów zwanych wierzchołkami (ang. vertex) oraz łączących te punkty krawędzi (ang. edge). Położenie wierzchołków na płaszczyźnie lub w przestrzeni jest dowolne. Krawędzie symbolizują drogi, którymi możemy się poruszać po grafie.

W celu zaimplementowania struktury grafu wykorzystano **macierz sąsiedztwa (ang. adjacency matrix)** , gdzie poszczególne wiersze oznaczają wierzchołki grafu. Numer wiersza jest numerem początkowego wierzchołka krawędzi. Teraz elementy wiersza informują nas, czy ten wierzchołek jest połączony krawędzią z wierzchołkami grafu, których numery określają kolejne kolumny. Na przykład wiersz 0 definiuje wszystkie krawędzie grafu, które rozpoczynają się w wierzchołku 0.

Opis implementacji metody DFS

Metoda DFS (ang. Depth First Search - przeszukiwanie najpierw w głąb) rozpoczyna działanie w wybranym wierzchołku grafu, który oznacza jako odwiedzony. Następnie przechodzi wzdłuż dostępnej krawędzi do sąsiada tego wierzchołka, który nie został jeszcze odwiedzony. Przechodzenie jest kontynuowane dalej (w głąb grafu), aż zostanie osiągnięty wierzchołek, który nie posiada nie odwiedzonych sąsiadów. Wtedy procedura wraca do poprzednio odwiedzonego wierzchołka i kontynuuje wzdłuż kolejnej dostępnej krawędzi.

Procedurę DFS realizujemy za pomocą stosu. Stos jest strukturą danych, w której składujemy kolejne elementy, lecz odczyt zawsze odbywa się w kierunku odwrotnym - od ostatnio umieszczonego na stosie elementu.

Opis implementacji metody BFS

Metoda BFS (ang. Breadth First Search - przeszukiwanie najpierw w szerz) wykorzystuje strukturę danych zwaną kolejką (ang. queue). Jest to struktura sekwencyjna. Dane dopisujemy na jej końcu, podobnie jak na stosie. Odczyt danych odbywa się kolejno z początku kolejki tzn. dane odczytujemy z kolejki w takiej samej kolejności, w jakiej zostały zapisane w niej. Kolejkę możemy potraktować jako tzw. bufor, który przechowuje dane. Im więcej danych w kolejce, tym dłużej dany element jest przez nią przechowywany.

Opis implementacji metody A*

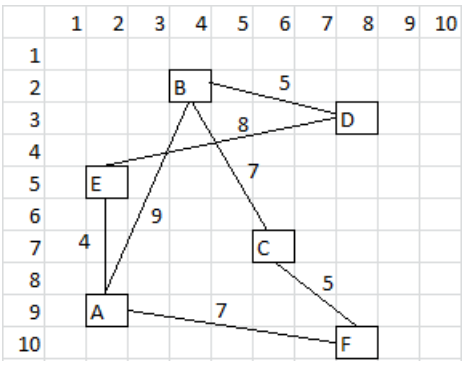
Algorytm heurystyczny znajdowania najkrótszej ścieżki w grafie ważonym z dowolnego wierzchołka do wierzchołka spełniającego określony warunek zwany testem celu. Algorytm jest zupełny i optymalny, w tym sensie, że znajduje ścieżkę, jeśli tylko taka istnieje, i przy tym jest to ścieżka najkrótsza.

Algorytm A* od wierzchołka początkowego tworzy ścieżkę, za każdym razem wybierając wierzchołek x z dostępnych w danym kroku niezbadanych wierzchołków tak, by minimalizować funkcję f(x) zdefiniowaną:

f(x) = g(x) + h(x)
gdzie:
g(x) - droga pomiędzy wierzchołkiem początkowym a x. Dokładniej: suma wag krawędzi, które należą już do ścieżki plus waga krawędzi łączącej aktualny węzeł z x.
h(x) - przewidywana przez heurystykę droga od x do wierzchołka docelowego.

Zastosowana heurystyka jest typu Manhattan (odległość dwóch węzłów to suma ich odległości w pionie i w poziomie – „poruszamy się po ulicach") - nie ma ona własności niedoszacowania, ale kosztem pewności wyniku przyspiesza znacznie obliczenia.
Mając dane węzły o współrzednych W1=[x1,y1] i W2=[x2,y2] odległość typu manhattan to D=|(x2-x1)+(y2-y1)|.

Przykłady struktur grafu - Przykład

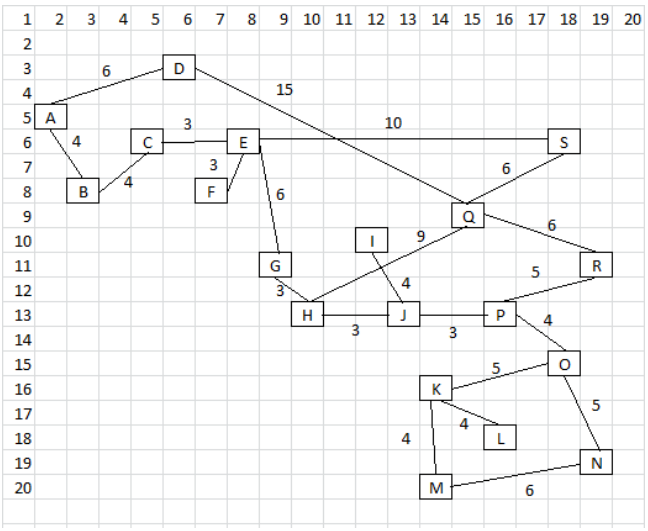


Rysunek 1: Przykład nr. 1

Benchmark:

1. BFS:
   (a) REZULTAT: C B F A D E
   (b) ILOŚĆ POWTÓRZEŃ ITERACJI: 36
   (c) CZAS WYKONYWANIA: 0.011131 ms
2. DFS:
   (a) REZULTAT: C F A E D B
   (b) ILOŚĆ POWTÓRZEŃ ITERACJI: 36
   (c) CZAS WYKONYWANIA: 0.014372 ms
3. A*:
   (a) REZULTAT: C F A E
   (b) ILOŚĆ POWTÓRZEŃ ITERACJI: 24
   (c) CZAS WYKONYWANIA: 0.082143 ms

Przykłady struktur grafu - Przykład 2



Rysunek 2: Przykład nr. 2

Benchmark:

1. BFS:
   (a) REZULTAT: M K N L O P J R H I Q G D S E F A C B
   (b) ILOŚĆ POWTÓRZEŃ ITERACJI: 361
   (c) CZAS WYKONYWANIA: 0.146883 ms
2. DFS:
   (a) REZULTAT: M N O P R Q S E G H J I F C B A D K L
   (b) ILOŚĆ POWTÓRZEŃ ITERACJI: 361
   (c) CZAS WYKONYWANIA: 0.127952 ms
3. A*:
   (a) REZULTAT: M K O P J H G E
   (b) ILOŚĆ POWTÓRZEŃ ITERACJI: 152
   (c) CZAS WYKONYWANIA: 0.105696 ms

Wnioski

Na podstawie przeprowadzonych pomiarów można zauważyć że algorytm A* jest najbardziej efektywnym algorytmem przeszukiwania grafu. Dzieje się tak ponieważ w algorytmie A* dodaje się heurystycznie wypracowaną inteligencje, aby kierować poszukiwaniami zamiast poruszać się na ślebo.

Przeszukiwanie wszerz odnajduje rozwiązanie optymalne jeśli takie istnieje lecz może w tym celu badać olbrzymią liczbe węzłów, ponieważ nie próbuje inteligetnie wybierać kolejności badania ruchów. Z kolei w przeszukiwaniu w głąb próbuje się szybko znaleźć rozwiązanie postępując do przodu z analizą ruchów ile się tylko da. Musi być ono jednak ograniczone gdyż w przeciwnym razie mogłoby prowadzić do bezowocnego przeszukiwania drzewa. W tym celu stosuje się dwa zbiory stanów do zapisywania wierzchołków. Zbiór otwarty w którym znajdują się wierzchołki które trzeba jeszcze odwiedzić oraz zbiór zamknięty w którym znajdują się wierzchołki już odwiedzone.