Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 2 - Grafy

UWAGA1:
Nie należy stosować gotowych kontenerów STL do implementacji zadań.

UWAGA2:
Wszystkie struktury danych oraz algorytmy należy implementować zgodnie z opisem na wykładzie.

UWAGA3:
Zadania poświęcone badaniu efektywności algorytmów grafowych zakładają badanie efektywności w zależności od metody reprezentacji grafów. Należy uwzględnić reprezentacje grafu w postaci macierzy sąsiedztwa oraz listy sąsiedztwa. Badania należy wykonać dla 5 różnych liczb wierzchołków w grafie V (np. 10, 50, 100, 500 i 1000) oraz następujących gęstości grafu: 25%, 50%, 75% oraz dla grafu pełnego. Dla każdego zestawu parametrów: algorytm, reprezentacja grafu, liczba wierzchołków i gęstość grafu należy wygenerować po 100 losowych instancji, natomiast w sprawozdaniu umieścić wyniki uśrednione.

Zadania

1. Zadania na ocenę dst (3.0):

1. Należy zaimplementować tablicę haszującą przechowującą N integerów z wykorzystaniem funkcji haszującej w postaci x%N, gdzie x reprezentuje wprowadzaną wartość, a N jest wielkością tablicy haszującej.
   (a) Użytkownik powinien mieć możliwość zdefiniowania ilości wprowadzanych elementów (należy zweryfikować, że jest to liczba pierwsza),
   (b) wartości zapisywane do tablicy powinny być generowane losowo bez powtórzeń,
   (c) należy zaimplementować trzy sposoby rozwiązywania kolizji:
      i. linkowanie
      ii. próbkowanie liniowe
      iii. podwójne haszowanie z drugą funkcją haszującą w postaci q - x%q, gdzie q <N jest liczbą pierwszą.
   (d) należy zaprezentować działanie operacji dodawania, wyszukiwania oraz usuwania elementów z tablicy, za każdym razem wyświetlając ilość próbek potrzebnych do dodania i wyszukania elementów.

2. Należy zaimplementować graf przechowujący elementy określonego typu. Należy napisać funkcje wykonujące podstawowe operacje na grafie zgodnie z informacjami podawanymi na wykładzie.
   (a) Należy zaimplementować graf za pomocą listy sąsiedztwa zgodnie z wytycznymi na wykładzie
   (b) Należy zaimplementować graf za pomocą macierzy sąsiedztwa zgodnie z wytycznymi na wykładzie.

2. Zadania na ocenę db (4.0):

1. Należy zaimplementować graf przechowujący elementy określonego typu. Należy napisać funkcje wykonujące podstawowe operacje na grafie zgodnie z informacjami podawanymi na wykładzie.
   (a) Należy zaimplementować graf za pomocą listy sąsiedztwa zgodnie z wytycznymi na wykładzie
   (b) Należy zaimplementować graf za pomocą macierzy sąsiedztwa zgodnie z wytycznymi na wykładzie.
2. Należy zaimplementować algorytmy Kruskala i Prima oraz przeprowadzić analizę efektywności tych algorytmów zgodnie z informacjami w Uwadze 3.

Sprawozdanie na ocenę db(4.0)

Sprawozdanie powinno zawierać:

- krótkie wprowadzenie,
- opis badanych algorytmów z omówieniem ich złożoności obliczeniowej
- omówienie przebiegu eksperymentów i przedstawienie uzyskanych wyników (w postaci tabel i wykresów)
- podsumowanie i wnioski (w przypadku niezgodności uzyskanych wyników z przewidywanymi spróbować wyjaśnić przyczyny),
- bibliografia (materiały wykorzystane do wykonania ćwiczenia, w tym strony internetowe).

3. Zadania na ocenę bdb (5.0):

Opis problemu

Problem najkrótszej drogi (ścieżki) w grafie między dwoma wierzchołkami polega na znalezieniu w grafie ważonym najkrótszego połączenia pomiędzy tymi wierzchołkami. Szczególnymi przypadkami tego problemu są:

- znalezienie najkrótszej ścieżki od wybranego wierzchołka do wszystkich pozostałych wierzchołków
- znalezienie najkrótszej ścieżki pomiędzy dwoma wybranymi wierzchołkami

Do rozwiązywania tego problemu służą (między innymi) dwa następujące algorytmy:

a) Dijkstry - przy założeniu, że w grafie nie ma wag ujemnych. Pesymistyczna złożoność obliczeniowa algorytmu Dijkstry wynosi O(E+VlogV)

b) Bellmana-Forda – dopuszczalne są wagi ujemne, ale niedopuszczalne jest istnienie cyklu o koszcie ujemnym. Pesymistyczna złożoność obliczeniowa algorytmu wynosi O(VE)

Opis Zadania

Zbadać efektywność jednego z powyższych algorytmów w zależności od sposobu reprezentacji grafu (w postaci macierzy i listy) oraz gęstości grafu. Badania należy wykonać dla 5 różnych liczb wierzchołków V oraz następujących gęstości grafu: 25%, 50%, 75% oraz dla grafu pełnego. Dla każdego zestawu: reprezentacja grafu, liczba wierzchołków i gęstość należy wygenerować po 100 losowych instancji, zaś w sprawozdaniu umieścić wyniki uśrednione. Aby otrzymać ocenę bardzo dobrą program należy napisać obiektowo. W przypadku algorytmu Dijkstry zaleca się implementację kolejki w formie kopca.

Program oprócz opcji badania efektywności (wygenerowania danych, pomiarów czasu, etc.) musi mieć możliwość wczytania grafu z pliku tekstowego oraz możliwość zapisu wyniku działania algorytmu (najkrótsza droga z pierwszego wierzchołka do wszystkich pozostałych wierzchołków). Format danych w pliku tekstowym jest następujący:

ilość_krawędzi *ilość_wierzchołków* *wierzchołek_startowy*

W kolejnych liniach znajdują się definicje krawędzi grafu (trójka danych):

wierzchołek_początkowy *wierzchołek_końcowy* *waga*

Wierzchołki numerowane są od zera. W pliku wynikowym dla każdego wierzchołka należy podać: koszt drogi oraz ciąg wierzchołków od wierzchołka startowego.

Sprawozdanie na ocenę bdb(5.0)

Wyniki należy przedstawić w tabelach oraz w formie wykresów. Wykresy powinny ilustrować zależność czasu wykonania algorytmu (oś Y) w funkcji ilości wierzchołków (oś X). Zaleca się przygotowanie następujących wykresów:

1. Wykresy typu 1 (osobne wykresy dla każdej reprezentacji grafu) – wykresy liniowe, których parametrem jest gęstość grafu oraz typ algorytmu (czyli 3x2=6 linii na rysunek).
2. Wykresy typ2 (osobne wykresy dla każdej gęstości grafu) – w formie linii których parametrem jest typ algorytmu i typ reprezentacji (czyli 4 linie na każdy rysunek).

Bibliografia

1. Cormen T., Leiserson C.E., Rivest R.L., Stein C., Wprowadzenie do algorytmów, WNT
2. Drozdek A., C++. Algorytmy i struktury danych, Helion
3. http://www.algorytm.org/algorytmy-grafowe/algorytm-prima.html
4. http://www.algorytm.org/algorytmy-grafowe/algorytm-kruskala.html
5. http://users.v-lo.krakow.pl/~toma/algorytmy/Algorytmy%20grafowe.pdf