Patrycja Karbownik

Koncepcja rozwiązania projektu z przedmiotu Analiza Algorytmów.

8. „Przygotowanie pod maraton"

Treść zadania:

W lasku Kampinoskim jest wiele ścieżek biegowo-rowerowych. Przygotowując się pod maraton zawodnik chce przebiec wszystkimi ścieżkami. Każdą ze ścieżek można pobiec w obu kierunkach i każda z nich ma określoną długość.

Należy wyznaczyć taką trasę biegaczowi która pokryje wszystkie ścieżki gdzie sumaryczny dystans będzie najmniejszy.

Dane wejściowe:
dla 4 skrzyżowań i 5 ścieżek:
1 2 3
2 3 4
3 4 5
1 4 10
1 3 12
Odp: 41

Zrozumienie problemu:

Problem postawiony w treści zadania jest problemem chińskiego listonosza. Sprowadza się on do znalezienia cyklu przechodzącego przez wszystkie krawędzie grafu co najmniej raz, w którym suma wag krawędzi jest najmniejszą możliwą sumą wag krawędzi spośród wszystkich takiego rodzaju cykli w grafie.

Założenia implementacji:

Informacje o grafie będą przechowywane w listach sąsiedztwa. Dla każdego wierzchołka w liście będą znajdować się jego sąsiedzi z wagą krawędzi między nimi.

Informacje o cyklu będą przechowywane w wektorze – zapis wierzchołków cyklu, oraz długość cyklu w zmiennej typu int.

Użyty język: C++

Opis algorytmu:

Problem chińskiego listonosza można podzielić na trzy podproblemy:

1. Gdy każdy wierzchołek jest parzystego stopnia (dochodzi do niego parzysta ilość krawędzi), istnieje w grafie cykl Eulera (cykl, który przechodzi przez każdą krawędź dokładnie raz) – aby otrzymać wynik, wyszukujemy cykl Eulera przy pomocy algorytmu Hierholzera i sumujemy wagi wszystkich krawędzi.
2. Dwa wierzchołki są nieparzystego stopnia – należy znaleźć najkrótszą ścieżkę między wierzchołkami nieparzystego stopnia (do tego posłuży nam algorytm Dijkstry), zdublować krawędzie, którymi prowadzi ścieżka i znaleźć cykl Eulera, a następnie zsumować wagi wszystkich krawędzi multigrafu.
3. Więcej niż dwa wierzchołki są nieparzystego stopnia:
    a. Wyszukujemy wszystkie wierzchołki nieparzystego stopnia.
    b. Za pomocą algorytmu Dijkstry znajdujemy najkrótsze ścieżki między nieparzystymi wierzchołkami.
    c. Wyszukujemy skojarzenie tych wierzchołków w pary o najmniejszej sumie wag krawędzi.
    d. Krawędzie wchodzące w skład wyznaczonych ścieżek skojarzenia dublujemy w grafie początkowym.
    e. Znajdujemy cykl Eulera i sumujemy wagi wszystkich krawędzi multigrafu.

Testowanie:

Testowanie będzie miało trzy wersje:

1. Poprawność – w kilku plikach txt będą zapisane informacje na temat grafów, a odpowiedź na postawiony problem będzie znana. Dzięki temu będzie można sprawnie sprawdzić czy algorytm działa poprawnie.
2. Poprawność – testowanie wg danych generowanych automatycznie (losowo) z ewentualną parametryzacją określaną przez użytkownika.
3. Złożoność – aby testować algorytm pod kątem złożoności, powstanie generator dużych grafów spójnych.

Algorytm generujący grafy:

Grafy potrzebne do testowania algorytmów muszą być spójne, nieskierowane i ważone. W tym celu zastosuję algorytm, który je wygeneruje:

1. Sprawdzamy czy podana przez użytkownika ilość krawędzi *e* w grafie o *n* wierzchołkach nie jest większa od ilości krawędzi w grafie pełnym, czyli *n*(n-1)/2*. Jeśli jest większa, informujemy użytkownika o problemie i prosimy o poprawienie danych. W przeciwnym razie przechodzimy do kolejnego punktu.
2. Sprawdzamy czy *e <= n*(n-1)/4*:
   a. Jeśli tak:
      i. W wektorze umieszczamy wszystkie *n* wierzchołków grafu – dalej będę nazywać ten wektor wektorem wierzchołków niedołączonych.
      ii. Losujemy jeden z wierzchołków niedołączonych i dodajemy go do listy sąsiedztwa – lista wierzchołków dołączonych (na razie bez żadnego sąsiada).
      iii. Iterujemy *n-1* razy:
         1. losujemy wierzchołek z wektora wierzchołków niedołączonych.
         2. losujemy wierzchołek z listy wierzchołków dołączonych.
         3. losujemy wagę krawędzi między wylosowanymi wierzchołkami.
         4. przenosimy pierwszy wylosowany wierzchołek do listy wierzchołków dołączonych.
         5. w liście sąsiedztwa dodajemy powstałe powiązanie.
      iv. Mamy *n-1* krawędzi w grafie. Jeśli użytkownik podał większą ilość krawędzi (*e > n-1*), musimy wylosować kolejne. W tym celu iterujemy *e-(n-1)* razy:
         1. Losujemy dwa wierzchołki i sprawdzamy czy krawędź między nimi już istnieje
            a. Jeśli nie, losujemy jej wagę, dodajemy i przechodzimy do kolejnej iteracji.
            b. Jeśli tak, wracamy do pkt. 2.a.iv.1 nie zwiększając wartości parametru iterującego.
      v. Kończymy algorytm.

   b. Jeśli nie:
      i. Tworzymy graf pełny łącząc ze sobą wszystkie wierzchołki.
      ii. Iterujemy *n*(n-1)/2 – e* razy i w każdej iteracji:
         1. Losujemy dwa różne wierzchołki.
         2. Sprawdzamy czy usunięcie krawędzi między wylosowanymi wierzchołkami nie podzieli grafu na składowe:
            a. Jeśli nie, usuwamy gałąź i przechodzimy do kolejnej iteracji.
            b. Jeśli tak, wracamy do pkt 2.b.ii.1 nie zwiększając wartości parametru iterującego.
      iii. Kończymy algorytm