Algorytmy i Struktury Danych

Lista zadań 8 - kody Huffmana, programowanie dynamiczne, grafy

1. (a) Dla podanego ciągu znaków liter z ilościami wystąpień w tekście, zasymuluj działanie algorytmu generującego kody Huffmana.
   (b) Oblicz o ile bitów otrzymana reprezentacja tekstu będzie krótsza od reprezentacji otrzymanej za pomocą kodów o stałej długości.
   (c) Mając dane drzewo kodów i zakodowany tekst wykonaj dekodowanie.
   (d) Czy kody Huffmana są wyznaczone jednoznacznie dla każdego tekstu?

   Do zadania użyj "tekstu" napisanego przy pomocy około 10 różnych znaków/liter o dość zróżnicowanej liczbie wystąpień.

2. (4 pkt) Napisz program który wykorzystując programowanie dynamiczne rozwiązuje bitoniczny problem komiwojażera (por. Cormen) w czasie wielomianowym. Oszacuj złożoność użytego przez Ciebie algorytmu. Program powinien czytać dane z pliku`miasta.txt`. W pierwszej linii pliku wejściowego jest ilość miast `n`, a w następnych `n` liniach pary współrzędnych `x[i] y[i]` każdego z miast. Wynikiem jest w plik `trasa.txt` złożonym z dwóch linii: w pierwszej jest liczba `n` i długość znalezionej trasy, a w drugiej `n` numerów miast w kolejności odwiedzania. Do długości trasy zalicza się powrót do wierzchołka wyjściowego. Numerację miast zaczynamy od `0`.

3. (2 pkt) Napisz program, który czyta z pliku graf w następującym formacie:
   (a) Pierwsza linia zawiera liczbę wierzchołków `n` oraz liczbę krawędzi `e`.
   (b) w następnych `e` liniach są po trzy liczby zadające krawędź: numer wierzchołka startowego, numer wierzchołka docelowego, oraz długość krawędzi. Wierzchołki są numerowane liczbami od 1 do `n`.

   Na podstawie pliku tworzony jest graf w reprezentacji list sąsiedztwa.

   Po wczytaniu grafu, program drukuje:
   (a) macierz sąsiedztwa,
   (b) minimalne drzewo rozpinające (algorytm Kruskala lub Prima)
   (c) drzewo najkrótszych ścieżek z wierzchołka o numerze 1 (algorytm Dijkstry): dla każdego wierzchołka (z wyjątkiem 1) drukowany jest drugi koniec krawędzi, jej długość oraz odległość całkowita od wierzchołka 1.

   Format wydruku drzew w punktach (b) i (c) jest taki sam jak dla grafu wejściowego.

   Za punkt wyjścia możesz użyć program `graph.cc` zamieszczony w serwisie panoramix.

4. (3 pkt) Dla podanego w formacie macierzy sąsiedztwa grafu zasymuluj na kartce działanie każdego z algorytmów: (a) DSF z wierzchołka A, (b) BSF z wierzchołka A, (c) MST Kruskal z wierzchołka A, (d) MST Prim, (e) Dijsktra z wierzchołka A, (f) Dijsktra z wierzchołka D.

$$\begin{pmatrix} & A & B & C & D & E & F \\ A & & 4 & & & 1 & 2 \\ B & 4 & & 2 & & 2 & \\ C & & 2 & & 8 & & \\ D & & & 8 & & 3 & 6 \\ E & 1 & 2 & & 3 & & 7 \\ F & 2 & & & 6 & 7 & \end{pmatrix}$$

   Na kartce powinien znajdować się rysunek grafu, z zaznaczonym wynikiem działania algorytmu oraz wypunktowane kolejne kroki algorytmu np. sortuję krawędzie; sprawdzam krawędź AB ale A i B są już w jednym zbiorze; sprawdzam sąsiada B ale jest już czarny; sprawdzam krawędź AB, wykonuję `union(A,B)` i dodaję AB do drzewa; `Q.getmin()` daje wierzchołek C; sprawdzam że $key(C) > key(B)+|BC|$) i robię `decrease_key(C,7)`;

5. (2 pkt) Napisz program, który (np. metodą prób i błędów, przeszukując "szachownicę" w głąb) znajduje drogę konika szachowego po szachownicy o podanych wymiarach, taką że każde pole jest odwiedzone dokładnie raz. Wypróbuj program dla kwadratowych szachownic od boku od 5 do 20.