Algorytmy i struktury danych

Lista zadań 10

Zadanie 1

Zastosuj strukturę `UnionFind3` (znajdziesz ją pliku `UnionFind3.cc` w materiałach do wykładu 10) do sprawdzenia czy w tablicy `bool t[n][n]` istnieje ścieżka zawierająca same jedynki (true): (a) od pola `t[0][0]` do `t[n-1][n-1]` (b) od pierwszego do ostatniego wiersza (tzn. jakaś komórka z pierwszego wiersza jest połączona ścieżką z jakąś komórką z ostatniego wiersza). Za ścieżkę uważamy ciąg pól tablicy, które stykają się krawędzią (różną się o 1 numerem kolumny albo wiersza).

Zadanie 2

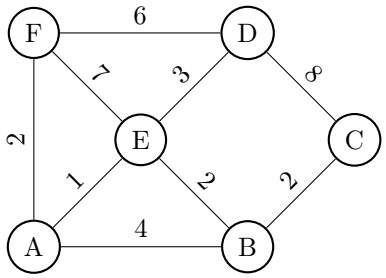
(a) Zastosuj strukturę `UnionFind` do sprawdzenia ile wysp jedynek zawiera tablica `bool t[n][n]`. Za wyspę uważamy zbiór jedynek taki, że z każdej do każdej można przejść ścieżką zawierającą same jedynki poruszając się tylko w poziomie i pionie. (b) ten sam problem rozwiąż za pomocą zmodyfikowanego algorytmu DFS.

Zadanie 3

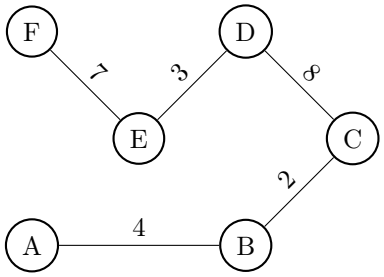
(3 pkt) Dla podanego w formacie macierzy sąsiedztwa grafu zasymuluj na kartce działanie każdego z algorytmów: (a) DSF z wierzchołka A, (b) BSF z wierzchołka A, (c) MST Kruskal, (d) MST Prim z wierzchołka A, (e) Dijsktra z wierzchołka A, (f) Dijsktra z wierzchołka D.

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

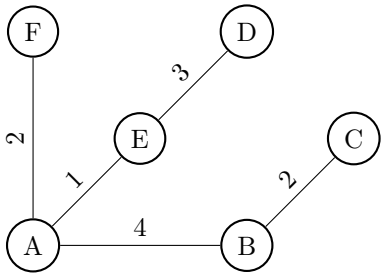
Na kartce powinien znajdować się rysunek grafu, z zaznaczonym wynikiem działania algorytmu oraz wypunktowane kolejne kroki algorytmu np. sortuję krawędzie; sprawdzam krawędź AB ale A i B są już w jednym zbiorze; sprawdzam sąsiada B ale jest już czarny; sprawdzam krawędź AB, wykonuję `union(A,B)` i dodaję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



(a) DFS z wierzchołka A

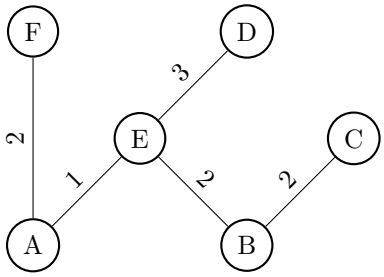


(b) BFS z wierzchołka A



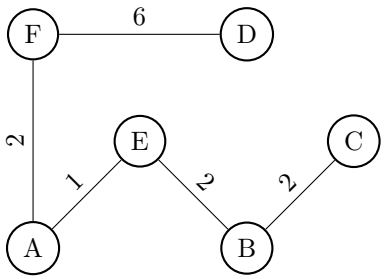
(c) MST Kruskal

Sortowanie krawędzi według wag i następnie dodawanie ich do drzewa jeśli nie tworzą cyklu.

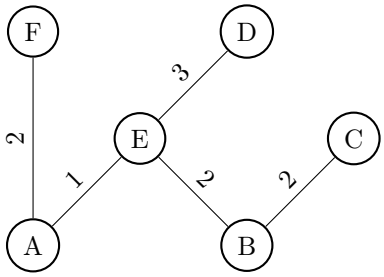


(d) MST Prim z wierzchołka A

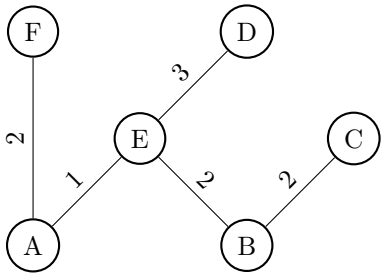
Inicjujemy tablicę odległości od grafu wartościami ∞, nastpęnie dla sąsiednich wierzchołków aktualizujemy odległości. Wybieramy wierzchołek o najmniejszej odległości, dodajemy go do grafu i powtarzamy.



(e) Dijsktra z wierzchołka A



(f) Dijsktra z wierzchołka D



Zadanie 4

(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 `graph2.cc` zamieszczony w materiałach do wykładu 11.

Zadanie 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 o boku od 5 do 20.