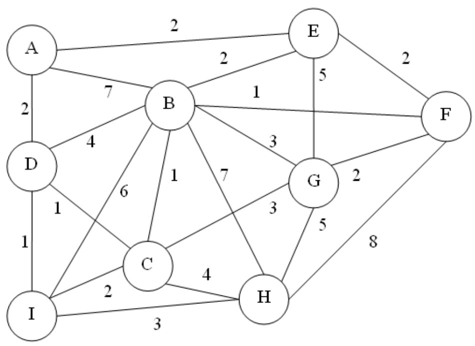
ASD - ćwiczenia XII

Algorytmy na grafach

Zadania

1. Dany jest graf $G = (V, E)$ zgodny z poniższym rysunkiem.



   (a) Zapisz ten graf w postaci tablicy list incydencji.

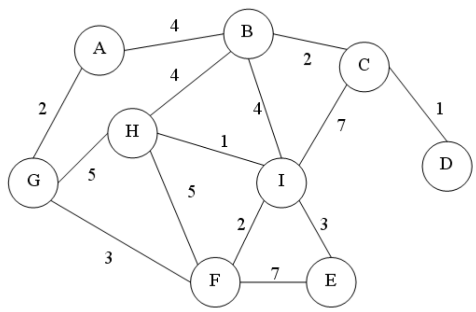
   (b) Zastosuj algorytm Dijkstry do znalezienia w tym grafie najkrótszej ścieżki z wierzchołka A do wierzchołka G. Wypisz tę ścieżkę i oblicz jej koszt. Zaznacz na grafie drzewo najkrótszych ścieżek będące rezultatem działania algorytmu Dijkstry.

   (c) Wypisz wierzchołki rozważanego grafu w kolejności, w jakiej byłyby odwiedzane z punktu startowego będącego wierzchołkiem E przez:

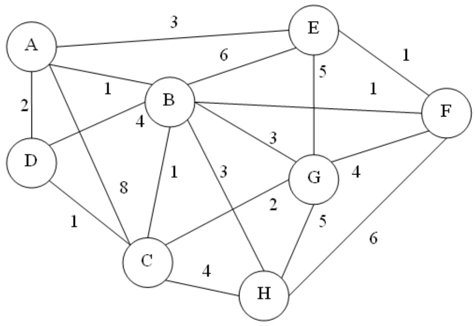
   - algorytm przechodzenia „w głąb“,
   - algorytm przechodzenia „w szerz“.

   Zakładamy, że w przypadku możliwości wyboru kilku wierzchołków jednocześnie, stosujemy porządek alfabetyczny.

2. Dla podanego poniżej grafu nieskierowanego $G = (V, E)$ wyznacz minimalne drzewo rozpinające stosując algorytm Kruskala. Wypisz krawędzie tego drzewa w kolejności zgodnej z kolejnością ich akceptowania (przyłączania). Podaj łączną wagę krawędzi minimalnego drzewa rozpinającego. W przypadku możliwości jednoczesnego wyboru kilku rozwiązań, kieruj się porządkiem alfabetycznym etykiet wierzchołków rozważanego grafu $G = (V, E)$.



3. Dla podanego poniżej grafu nieskierowanego $G = (V, E)$ wyznacz minimalne drzewo rozpinające stosując algorytm Prima. Wypisz krawędzie tego drzewa w kolejności zgodnej z kolejnością ich akceptowania (przyłączania). Podaj łączną wagę krawędzi minimalnego drzewa rozpinającego. Ustal wierzchołek C punktem startowym algorytmu. W przypadku możliwości jednoczesnego wyboru kilku rozwiązań, kieruj się porządkiem alfabetycznym etykiet wierzchołków rozważanego grafu $G = (V, E)$.



4. Dany jest kwadrat o boku n, zaczepiony lewym dolnym rogiem w punkcie $(0, 0)$ układu współrzędnych. Na powierzchni kwadratu rozmieszczono m min wybuchowych w punktach o współrzędnych całkowitych $(x[1], y[1]), (x[2], y[2]), \ldots, (x[m], y[m])$, gdzie $\forall i \in \{1, 2, \ldots, m\} \; : \; (x[i], y[i]) \neq (0, 0) \; \wedge \; (x[i], y[i]) \neq (n, n)$. Naszym zadaniem jest bezpieczne przejście (tj. bez wstąpienia na minę) z punktu $(0, 0)$ do punktu (n, n) zgodnie z poniższymi zasadami:

- poruszamy się po punktach, których współrzędne są liczbami całkowitymi,
- z dowolnego punktu o współrzędnych (i, j) możemy przejść jedynie do punktu $(i, j + 1)$ albo do punktu $(i + 1, j)$,
- dotarcie do punktu „uzbrojonego“ w minę wybuchową przedwcześnie kończy naszą podróż.

(a) Zaprojektuj algorytm, który wyznaczy długość minimalnej ścieżki, łączącej punkty $(0, 0)$ i (n, n) w bezpieczne przejście.

(b) Oszacuj złożoność swojego algorytmu względem rozmiaru boku kwadratu n oraz liczby zaminowanych pól m.

5. Niech Q będzie zbiorem n planowanych stacji kolejowych $Q[1,]\, Q[2], \ldots, Q[n]$, których współrzędne geograficzne są z góry ustalone. Elementy zbioru Q reprezentowane są przy pomocy następującej struktury danych:

```
struct Station {
  real N_S, W_E;           // współrzędne geograficzne
};
```

(a) Zaproponuj algorytm (przedstaw słownie metodę rozwiązania problemu), którego rezultatem będzie szkielet sieci kolejowej $\mathcal{R}$ taki, że:

- dla dowolnych $Q[i], Q[j] \in Q$ istnieje trasa kolejowa łącząca te stacje,
- wszystkie rozwidlenia (skrzyżowania) torów kolejowych mogą znajdować się jedynie na terenie stacji kolejowych,

- łączna suma długości torów kolejowych tworzących poszukiwany szkielet $\mathcal{R}$ ma być minimalna z możliwych.

(b) Podaj pseudokod funkcji

`Graph FIND_SCHEMA(Station Q[]),`

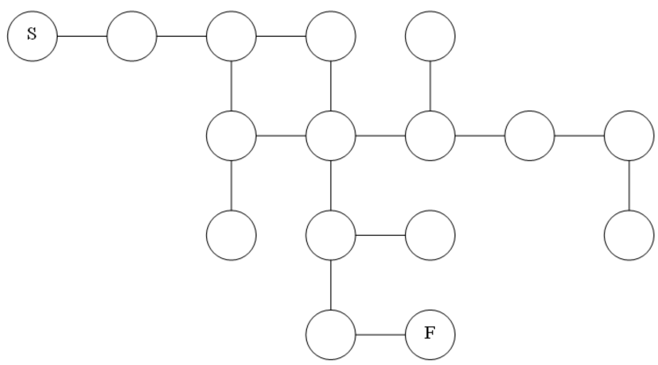
będącej implementacją przedstawionego algorytmu (w kodzie procedury można używać dowolnych algorytmów przedstawionych na wykładzie, bez potrzeby wypisywania ich implementacji, należy jednak wyjaśnić jak i do czego się ich używa).

(c) Oszacuj złożoność funkcji `FIND_SCHEMA()` względem liczby n planowanych stacji kolejowych.

6. Załóżmy, że pewien labirynt utworzony w płaszczyźnie euklidesowej składa się z n pomieszczeń i m korytarzy o następującej charakterystyce:

   - z/do każdego z pomieszczeń prowadzą co najwyżej cztery korytarze zwrócone na strony świata: północną (N), południową (S), wschodnią (E) oraz zachodnią (W),
   - wszystkie pomieszczenia mają ten sam wymiar,
   - każdy korytarz jest idealnie prosty i wszystkie korytarze mają tą samą długość,
   - każdy korytarz jest obustronnie zakończony pomieszczeniami,
   - z każdego pomieszczenia istnieje droga przez labirynt (naprzemienny ciąg pomieszczeń oraz korytarzy rozpoczynający i kończący się pomieszczeniem) prowadząca do dowolnego innego pomieszczenia,
   - istnieją w labiryncie dwa szczególne pomieszczenia: pomieszczenie startowe s oraz pomieszczenie końcowe t (wyjście z labiryntu),
   - pomieszczenie startowe B i końcowe F nie są tym samym pomieszczeniem,
   - istnieje w labiryncie droga prowadząca z pomieszczenia B do F przechodząca przez d pomieszczeń (łącznie z pomieszczeniem startowym i końcowym), gdzie $2 \leq d \leq 2 \cdot \sqrt{n}$.

   Oczywiście każdy tak zdefiniowany labirynt możemy traktować jako niezorientowany graf $G = (V, E)$, gdzie $|V| = n$ i $|E| = m$. Np. dla $n = 16$, $m = 16$ przykładem grafu jest:



   Zaprojektuj procedurę `FIND_PATH(Graph G)`, która dla dowolnego grafu $G = (V, E)$ reprezentującego pewien labirynt składający się z n pomieszczeń i m korytarzy, wyznaczy drogę (poda ciąg krawędzi tego grafu) z pomieszczenia S do F tak, że liczba korytarzy odwiedzonych w trakcie działania tego algorytmu wynosi $O\left(3^d\right)$.

7. Które z podanych poniżej stwierdzeń jest prawdziwe, odpowiedź uzasadnij:

   (a) dla dowolnego grafu prostego $G = (V, E)$ algorytm Prima i algorytm Kruskala generują identyczne drzewo rozpinające,

   (b) dla dowolnego grafu prostego $G = (V, E)$ algorytm Dijkstry generuje minimalne drzewo rozpinające.