Patryk Mociarz 331542 Lista 12

Zad. 7.

Alg. Kruskalla dla posortowanych nierosnąco wagowo krawędzi

1. Posortuj krawędzie w takiej kolejności, by ciąg ich wag był nierosnący
$(d(e_1) \leq d(e_2) \leq d(e_3) \leq \ldots \leq d(e_n))$
2. $T \leftarrow \emptyset$
3. Dla $i = 1, 2, \ldots, m$
Jeśli dodanie do T krawędzi e_i nie tworzy cyklu, to $T \leftarrow T \cup \{e_i\}$

Zad. 9.

Aby alg. Warshalla-Floyda znajdował nie tylko długości najkrótszych dróg między parami wierzchołków, ale również te drogi, należy zastosować pomocniczą tablicę $n \times n$ przechowującą poprzedni wierzchołek na najkrótszej ścieżce między kolejną parą wierzchołków.
Na początku tablicę pomocniczą inicjalizuję tak, aby kolejny wierzchołek poprzedzał sam siebie. ($prev[i,j] = i$)

```
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            if dist[i,k] + dist[k,j] < dist[i,j]
                dist[i,j] ← dist[i,k] + dist[k,j]
                prev[i,j] ← prev[i,k]
```

Zad. 3.

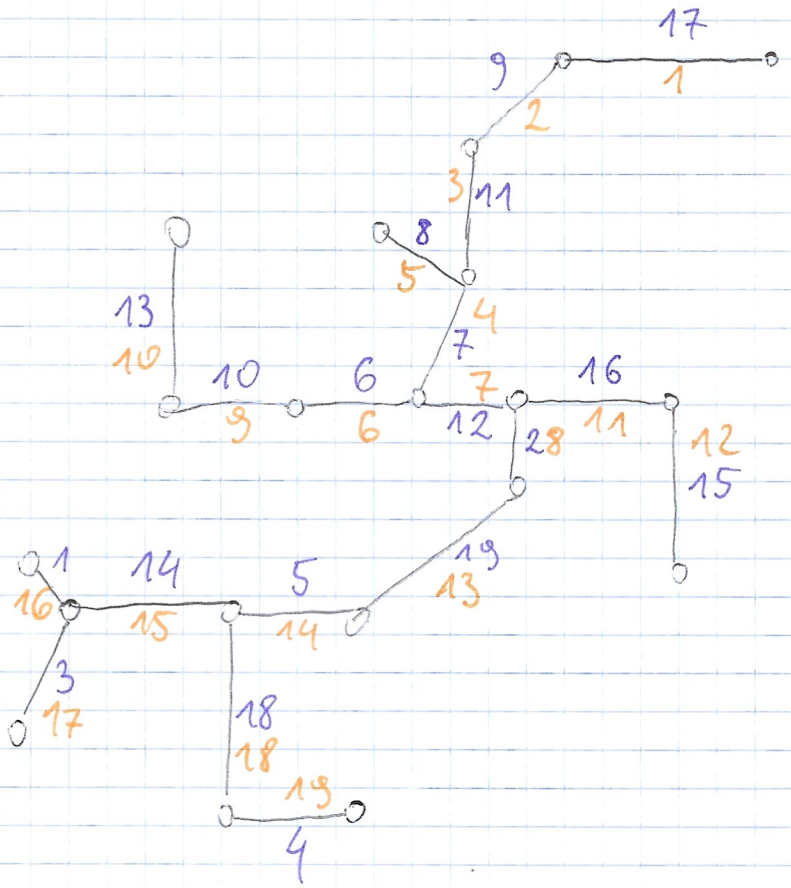
1. Tworzymy tablicę indeg przechowującą stopień wejściowy dla każdego wierzchołka

2. Używamy BFS, dodając do kolejki wierzchołki, jeśli ich indeg jest równy 0, czyli:
   - tworzymy listę
   - dodajemy do niej wierzchołki, które już mają indeg zero
   - dopóki lista nie jest pusta, zdejmujemy pierwszy element, przypisujemy mu numer (który następnie zwiększamy)
   - ~~zdejmujemy indeg~~
   - obniżamy indeg wszystkich wierzchołków na które wskazywał zdjęty wierzchołek o 1, a jeśli ich indeg będzie po odjęciu równy 0, to dodajemy je talie wierzchołki do listy.

```
for i = 1 to n
    if indeg[i] == 0: q.push(i)

while (!q.empty())
    v = q.front()
    q.pop()
    result[index++] = v
    for i = 0 to g[v].size() - 1
        w = g[v][i]
        indeg[w]--
        if (indeg[w] == 0) q.push(w)
```

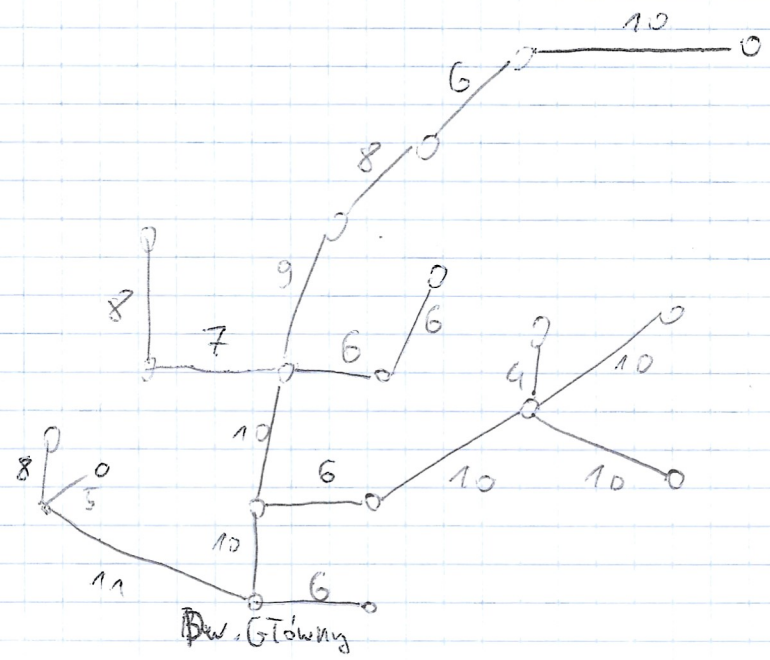
Zad. 4.



Alg. Kruskala wybiera kolejne krawędzie od najkrótszych, tak aby nie zrobić cyklu.

Alg. Prima - Dijkstry przyłącza kolejne najkrótsze do powstałego już drzewa.

Zad. 5.



(Dla pl. Grunwaldzkiego oraz Browaru drzewa tworzą się analogicznie)

Zad. 2.

Aby sprawdzić czy graf jest dwudzielny, możemy użyć przeszukiwania grafu w głąb (DFS) przypisując odwiedzanemu wierzchołkowi jeden z dwóch kolorów, a następnie sprawdzamy, czy żaden z jego sąsiadów nie posiada już takiego koloru. Jeśli tak, to graf nie jest dwudzielny.

```
bool dfs(v, k):
    color[v] = k
    for i = 0 to g[v].size()
        w = g[v][i]
        if color[w] == k  return false
        if color[w] == 0 && !dfs(w, 3-k)) return false
    return true
```

```
for i = 1 to n
    if (color[i] == 0 && !dfs(i, 1)) return false
return true
```

(kolor to samo zera na początku)