MDM Lista 12

Weronika Jakimowicz

ZAD. 1.

```
odwiedzone = [0] * n
skoki = [0] * n

nozyczki = -1

def dfsik(j, studnia):
    odwiedzone[j] = 1
    skoki[j] = studnia
    zmienna = 0

    for i in sasiedzi[j]:
        if not odwiedzone[i]:
            zmienna++
            skoki[j] = max(skoki[j], dfsik(i, studnia+1))

    if studnia == 0 and zmienna > 1:
        nozyczki = j

    if skoki[j] <= studnia
        nozyczki = j

    return skoki[j]


def rozspojnia():
    dfsik(0, 0)
    if nozyczki == -1:
        print("NIE␣ISTNIEJE␣WIERZCHOLEK␣ROZSPOJNIAJACY")
    else:
        print(nozyczki)
```

ZAD. 2.

```
odwiedzone = [0] * n
kolorowanie = [0] * n

def dfsik(j, kolor):
    odwiedzone[j] = 1
    kolorowanie[j] = kolor

    for i in sasiedzi[j]:
        if not odwiedzone[i]:
            if not dfsik(i, kolor * -1): return False

        if kolorowanie[i] == kolorowanie[j]:
            return False
    return True
```

ZAD. 3.

```
odwiedzone = [0] * n
kolorowanie = [0] * n

pom = []

def dfsik(j):
    odwiedzone[j] = 1
    for i in sasiedzi[j]:
        if not visited[i]: dfs(i)

    pom.append(i)

def topo():
    for i in range(len(pom)):
        kolejnosc.append(pom[n-i])
```

ZAD. 9

```
for i in range(n):
    for j in range(n):
        for k in range(n):
            if (d[j][i] < INF && d[i][k] < INF):
                nc = d[j][i] + d[i][k]
                if d[j][k] > nc:
                    d[j][k] = nc
                    p[j][k] = p[j][i] + p[i][k]
```

ZAD. 12.

Niech G będzie grafem z nieujemnymi wagami na krawędziach. Niech MST(G) *oznacza długość najlżejszego drzewa spinającego w* G*, a* TSP(G) *oznacza długość najkrótszej drogi komiwojażera w* G *(komiwojażer może odwiedzać wierzchołki wielokrotnie). Wykaż, że*

$$\mathrm{MST(G)} \leq \mathrm{TSP(G)} \leq 2 \cdot \mathrm{MST(G)}$$

$2 \cdot \mathrm{MST(G)} \geq \mathrm{TSP(G)}$ - w najgorszym przypadku będziemy szli przez najmniejsze drzewo rozpinające do najodleglejszych wierzchołków i wracali do korzenia po tej samej drodze, czyli po każdej "gałęzi" przejdziemy dwukrotnie.

$\mathrm{TSP(G)} \geq \mathrm{MST(G)}$ - w najlepszym przypadku po prostu przejdziemy jeden raz po każdej gałęzi najmniejszego drzewa rozpinającego, bo odwiedza ono każdy wierzchołek jednokrotnie po jak najkrótszej drodze.

ZAD. 14.

Graf jest krawędziowo k-spójny gdy jest spójny i usunięcie z niego co najwyżej $(k-1)$ *krawędzi nie rozspójnia go. Używając przepływów w sieciach pokaż, że* G *jest krawędziowo* k*-spójny* $\iff$ *między dwoma wierzchołkami istnieje* k *krawędziowo rozłącznych dróg.*

$\Leftarrow$

Niech G będzie grafem takim, że między dowolnymi dwoma wierzchołkami jest k krawędziowo rozłącznych dróg. Weźmy $F \subseteq E(G)$ zbiór krawędzi taki, że $G \setminus F$ jest grafem rozłącznym. Z założenia wiemy, że dla dowolnych $v, w \in G$ istnieje k rozłącznych dróg w G między tymi dwoma wierzchołkami, więc aby $G \setminus F$ było rozłączne, to musimy co najmniej jedną krawędź z każdej z tych k dróg wyjąć. Czyli $|F| \geq k$.

$\Rightarrow$

Niech G będzie grafem krawędziowo k-spójnym. Zamieńmy ten graf na digraf rozbijając dowolną krawędź $\{v, w\} \in G$ na dwie skierowane krawędzie: (v, w) oraz (w, v). Chcemy dodać do niego źródło s oraz ujście t i każdej krawędzi przypiszemy pojemność 1. Wiemy, że maksymalny przepływ w grafie jest równy pojemności minimalnego cięcia. W przypadku grafu k-spójnego minimalne cięcie, które rozłącza wszystkie drogi między s a t jest równe k, czyli maksymalny przepływ w G jest równy k, a to znaczy, że idąc od źródła do ujścia musimy odwiedzić co najmniej k różnych krawędzi, bo każda z nich może pomieścić co najwyżej 1.