# Lista 12

Łukasz Magnuszewski

## Zadanie 12

### $MST(G) \leq TSP(G)$

Zauważmy że krawędzie wchodzące w skład $TSP(G)$ uspójniają graf. A najmniejszy koszt uspójnienia grafu to $MST(G)$ czyli ta nierównośc zachodzi.

### $2MST(G) \geq TSP(G)$

Zauważmy że jak wybierzemy dowolny wierzchołek z $MST(G)$. I puścimy go w tym drzewie. To odwiedzi on każdy wierzchołek, oraz odwiedzi każda krawędź dokładnie dwa razy (raz w jedną, raz w drugą stronę). Czyli mamy kandydata na $TSP(G)$, którego koszt wynosi $2MST(G)$. Czyli $TSP(G)$ nie może być większe.

## Zadanie 1

Puśmy następującego dfsa w dowolnym punkcie. I w zmiennej ans otrzymamy punkt rozpinający jeśli on istnieje.

```
int dfs(int x, int d, vector<vector<int>>& g,
    int& ans, vector<bool>& odw)
{
    odw[x] = true;
    int low = d;

    int cnt = 0;
    for(int v : g[x]) {
        if(!odw[v]) {
            low = max(low, dfs(v, d+1, g, ans, odw));
            cnt++;
        }
    }
    if(d == 0 && cnt > 1) ans = x;
    if(low <= d && d != 0) ans = x;
    return low;
}
```

Mój algorytm to pojedyńczego wywołania dfsa, więc jego złożoność to $O(n+m)$.

A poprawnośc można uzasadnić w następujący Popatrzmy na drzewo wywołań dfs (Czyli wierzchołki orginalnego grafu oraz krawędzie którymi przeszedł dfs). Kluczowa obserwacja jest taka że wierzchołek może być połączony krawędzia nie drzewiową, tylko z wierzchołkiem który leży na ścieżce między nim a korzeniem. Bo gdyby istniała taka krawędź niedrzewiowa która nie idzie do przodka, to by dfs nią przeszedł, czyli byłaby drzewiowa sprzeczność.

Jeśli korzeń naszego drzewa ma stopień większy niż 1 to jest on punktem artykulacji. Udowodnijmy to: na pewno ten wierzchołek rozspójnia nasze drzewo, ale być może krawędzie które nie weszły w skład naszego drzewa, zapobiegają temu. Ale one mogą łączyć tylko wierzchołek i jego przodka, czyli nie mogą łączyć rozłącznych podrzew.

Teraz rozpatrzmy pozostałe wierzchołki, policzmy dla nich funkcje low. Czyli jak wysoko można zajść w górę z danego wierzchołka, mogąc iść dowolną liczbę razy krawędziami drzewowymi w dół drzewa, i maksymalnie jeden w góre krawędzią niedrzewiową. Jeśli low danego wierzchołka jest mniejsze równe od jego głębokości to jest punktem artykulacji. Bo wszystkie krawędzie niedrzewiowe jego potomków, nie mogą wyjść z jego podrzewa.

# Zadanie 2

Będę rozważał tutaj grafy spójne. Ale problem dla grafu niespójnego sprowadza się odpaleniu następującego algorytmu dla każdej spójnej i wymaga by każda spójna była dwudzielna.

Graf dwudzielny można pokolorować na 2 kolory, tak by sąsiednie wierzchołki miały różne kolory. Zauważmy że z dokładnością do izomorfizmu istnieje maksymalnie jedno dwukolorowanie (To na który z dwóch kolorów pokolorujemy pierwszy wierzchołek determinuje kolorowanie całego grafu).

```
bool dfs(int x, int col, vector<vector<int>>& g, vector<int>& color) {
    color[x] = col;
    for(int v : g[x]) {
        if(color[v] == c) return false;
        if(color[v] == 0) dfs(v, -col, g, color);
    }
    return true;
}
```

Algorytm sprowadza się do dfa, więc jego złożoność wynosi $O(n + m)$.

# Zadanie 3

```
vector<int> topo(vector<vector<int>> g, vector<int> in) {
    vector<int> s, ans;
    for(int i = 0; i < in.size(); i++) {
        if(in[i] == 0) s.push_back(i);
```

```
    }
    while(s.size()) {
        int x = s.back(); s.pop_back();
        for(int v : g[x]) {
            in[v]--;
            if(in[v] == 0) {
                s.push_back(i);
                ans.push_back(i);
            }
        }
    }
    return ans;
}
```

Obserwacja jest taka że, gdy wierzchołek ma stopień wejściowy równy zero to może on być pierwszy w sortowaniu topologicznym. Dodatkowo możemy wtedy resztę problemu sprowadzić do grafu z usuniętym pierwszym wierzchołkiem (co zmniejsza stopień wejściowy pozostałych wierzchołków).

Złożonośc tego algorytmu wynosi $O(n+m)$. Po pierwsze trzeba wyznaczyć wierzchołki które na start mają $in(x) = 0$ kosztuje to $O(n)$ operacji. W trakcie działania programu usuniemy wszystkie krawędzie, czyli $O(m)$ operacji. I każdy wierzchołek będziemy maksymalnie raz wrzucać na listę wynikową, stąd $O(n)$ operacji. Sumarycznie daję to wymaganą złożoność.

# Zadanie 6

Załóżmy niewprost że otrzymane drzewo $T$ nie jest $MSP(G)$. weźmy wtedy minimalne $i$ takie że $e_i \in MSP(G) \wedge e_i \notin T$. Wtedy istnieje $e_j$ takie że $e_j \in T \wedge e_j \notin MSP(G)$, jest tak bo drzewa o takim samym rozmiarze mają tyle samo krawędzi. Zauważmy że

# Zadanie 7

Wystarczy puścić algorytm z zadania 6 tylko dla ujemnych wag krawędzi.