# MDL 9 9.12

Dominik Szczepaniak

December 9, 2023

## 1 Zadanie 5

a) Teza: Zbiór wierzchołków centralnych to albo pojedynczy wierzchołek albo para sąsiadujących wierzchołków.

Indukcja:

n=1 albo n=2 - wierzchołkiem centralnym jest albo para albo pojedyczny wierzchołek - albo pojedynczy wierzchołek albo oba są centralne.

Krok:

Załóżmy, że teza zachodzi $\forall i \leq n$ i chcemy pokazać, że dla n+1 zachodzi oraz n+1>2.

Niech drzewo T będzie miało n+1 wierzchołków. Niech T' będzie T ale bez liści. Skoro środkowe wierzchołki na ścieżce między liśćiami zostają, to T' musi mieć przynajmniej jeden wierzchołek (bo w najgorszej opcji dla drzewa o rozmiarze 3 zostanie tylko 1 wierzchołek). r() jest większe dla liścia niż dla jego sąsiada, ponieważ może przyjąć tą samą ścieżkę co sąsiad, ale przedłużyć ją o 1. No ale skoro wierzchołek centralny grafu minimalizuje r(), to wierzchołek centralny grafu na pewno nie jest liściem. Skoro usuneliśmy liście, to usuneliśmy co najmniej jeden wierzchołek, a wtedy liczba wierzchołków jest $\leq n$, czyli z założenia indukcyjnego teza dalej zachodzi.

b)

Aby znaleźć wierzchołek centralny chcemy obliczyć najdłuższą ścieżkę d w drzewie, a później znaleźć wierzchołek (lub dwa) które mają najdłuższą ścieżkę d/2.

Obliczanie średnicy drzewa to puszczenie dfs z dowolnego wierzchołka i później ponowne puszczenie dfs z wierzchołka który był najdalej od pierwotnego wierzchołka.

Zakładam, że w pamięci mam już drzewo o n wierzchołkach $\langle<\rangle 1, 2, \ldots, n >$, podanych jako lista sąsiedztwa G[], gdzie G[1] oznacza sąsiadów 1.

```
def dfs(start, odl, visited):
    visited[start] = True
    for i in G[start]:
        if(not visited[i]):
            odl[i] = odl[start]+1
            dfs(i, odl, visited)
def wierzcholek_centralny():
    visited = [False] * (n+1)
    odl = [0] * (n+1)
    dfs(1, odl, visited)
    maxV = 0
    vert = 1
    for i in range(2, n+1):
        if(maxV < odl[i]):
            maxV = odl[i]
            vert = i
    odl = [0] * (n+1)
    visited = [False] * (n+1) #resetujemy pod nowy dfs odl i visited
    dfs(vert, odl, visited)
    print(vert, odl)
    d = 0
    for i in range(1, n+1):
        d = max(d, odl[i])
    szuk = d//2
    if(d%2==0): #jeden wierzcholek
        for i in range(1, n+1):
            if(odl[i] == szuk):
                return i
    else: #dwa wierzcholki
        odl2 = [0] * (n+1)
        visited = [False] * (n+1)
        maxV = 0
        vert = 1
        for i in range(1, n+1):
            if(maxV < odl[i]):
```

```
                maxV = odl[i]
                vert = i
    dfs(vert, odl2, visited)
    wynik = []
    print(vert, odl2)

    for i in range(1, n+1):
        if(odl[i] == szuk and (odl2[i] == szuk-1 or odl2[i] == szuk
            wynik.append(i)
    for i in range(1, n+1):
        if(odl2[i] == szuk and (odl[i] == szuk-1 or odl[i] == szuk+
            wynik.append(i)
    return sorted(wynik)
```

Dfs jest standardowy więc pominę wyjaśnianie go.

Ustawiamy wszystkie visited na False i odległości na 0. Później puszczamy dfs w 1, znajdujemy najdalej znajdujący się wierzchołek od 1, oznaczmy go v. Znajdujemy go forem w odległościach i puszczamy tam znowu DFS (zerujemy odleglosci i False na visited). Najdalej znajdujący się wierzchołek od wierzchołka v daje nam średnice drzewa. Jeśli średnica jest parzysta istnieje tylko jeden wierzchołek centralny. Znajdujemy go - odległość od v do niego musi wynosić średnica / 2.

Jeśli średnica nie jest parzysta, to musimy odpalić kolejnego DFS w najdalszym wierzchołku od v i zapisać dane do nowej tablicy odl2. Wtedy szukamy wierzchołka oddalonego o średnica/2 oraz, że odległość dla sąsiedniego wierzchołka też się zgadza (średnica/2 +/- 1).

Dlaczego działa? Jeśli znajdziemy dwa końcowe punkty najdłuższej ścieżki o długości nieparzystej to wierzchołkami wewnętrznymi grafu będą dwa wierzchołki które są oddalone o średnica/2 dla obu tych punktów. Jeśli długość ścieżki jest parzysta to odległość będzie taka sama dla obu punktów, więc wystarczy znaleźć dla jednego.

## 2 Zadanie 6

Implikacja z lewej w prawo:

Z lematu o uściśnięciach dłoni $\sum_{i=1}^{n} = 2|E| = 2(n-1)$ (drzewo o n wierzchołkach ma n-1 krawędzi).

Implikacja z prawo w lewo:

Udowodnijmy bez straty ogólności, że dla posortowanego nierosnąco ciągu d mamy, że dwa ostatnie wyrazy będą stopnia 1 (będą liściami).

Indukcja po n:

n=2

d1 + d2 = 2(2-1) = 2

d1 = 1, d2 = 1

Załóżmy, że dla n zachodzi. Czyli dla n+1 chcemy pokazac, ze $\sum_{i=1}^{n+1} = 2n$

Niech naszym nowym wierzchołkiem będzie $v_{n+1}$ o stopniu $d_{n+1} = 1$ (z założenia). Nasz nowy wierzchołek musi być połączony z dowolnym wierzchołkiem. Oznaczmy ten wierzchołek przez $v_j$ i niech jego stopień będzie równy $d_j$.

W naszym nowym ciągu więc $d'_j = d_j + 1$ (bo nowy wierzchołek)

Mamy więc:

$\sum_{i=1,i\neq j}^{n}(d_i + d'_j) + d_{n+1} = \sum_{i=1}^{n}(d_i) + 1 + d_{n+1} = 2(n-1) + 1 + 1 = 2n - 2 + 2 = 2n$

Co chcieliśmy pokazać.

Obie implikacje zachodzą więc twierdzenie jest prawdziwe.

# 3 Zadanie 9

Zakładam, że wartość liczbowa nodów nie ma znaczenia tj. drzewo (V, E) = ((1, 6), (1, 6)) jest tym samym drzewem co (V, E) = ((1, 2), (1, 2)).

Załózmy, ze mamy $deg(v_i) = k$. Jakie mamy mozliwosci drzewa k krawedziowego?

1. Sciezka o dlugosci k
2. Korzen ma k krawedzi
3. Korzen ma k-1 krawedzi i kolejny ma 1 krawedz
4. Korzen ma k-2 krawedzi i kolejny ma 2 krawedzie
5. Korzen ma k-2 krawedzi i kolejny ma 1 krawedz do kolejnego ktory ma jedna krawedz

$\vdots$

Zauważmy, że aby otrzymac $deg(v_i) = k$ potrzebujemy $k$+1 wierzcholkow.

Oznaczmy wielomian

$w(k) = a_1 x^k + a_2 x^{k-1} + a_3 x^{k-2} + \cdots + a_k x^1 + a_{k+1} * x^0$

Gdzie $a_1$ to ilosc krawedzi wychodzacych dla wierzcholka 1 które nie idą do wierzchołka <= 1, $a_2$ - ilosc krawedzi wychodzacych dla wierzcholka 2 które nie idą do wierzchołka <= 2, itd.

Mamy k+1 wierzcholków wiec liczb $a_{k+1}$ (numeracja od 1).

$\sum_{i=1}^{k}(a_i) = k$ (suma krawędzi wychodzących z wierzchołków musi się sumować do k, bo tyle mamy łącznie krawędzi).

oraz

$a_i \leq k \forall i \in \{1, 2, \ldots, k+1\}$, gdyż mamy co najwyżej k krawędzi

Zauważmy, że każde drzewo może mieć takie przedstawienie - nie ma w tym grafie cykli - ponieważ krawędź wychodząca może iść tylko do wierzchołka z większym ID, stąd możemy podzielić wierzchołki na poziomy.

Zauważmy teraz, że nasze możliwości drzewa k-krawędziowego możemy opisać tym wielomianem. W szczególności:

1. Scieżka długości k to wielomian $w(k) = x^k + x^{k-1} + x^{k-2} + \cdots + x^1 + x^0$

2. Korzeń k krawędziowy: $w(k) = k * x^k$

3. Korzeń k-1 krawędziowy i kolejny 1 krawędź: $w(k) = (k-1) * x^k + x^{k-1}$

5. Korzen ma k-2 krawedzi i kolejny ma 1 krawedz do kolejnego ktory ma jedna krawedz: $w(k) = (k-2) * x^k + x^{k-1} + x^{k-2}$

Nasz graf ma przedstawiać każde drzewo. Jeśli w takim razie wybierzemy dowolny wielomian w(x) z dowolnymi współczynnikami które sumują się do k oraz $a_i \leq k \forall i \in \{1, 2, \ldots, k+1\}$ to możemy go opisać naszym grafem, ponieważ każdy wierzchołek $i$ ma stopień $\leq k$ oraz suma krawędzi w grafie jest co najmniej k (np. z zadania 2 wiemy, że istnieje ścieżka o długości k), także nasz graf spełnia ten wielomian, więc spełnia opisanie każdego możliwego drzewa o k krawędziach.