MDL 8 30.11

Dominik Szczepaniak

December 9, 2023

1 Zadanie 1

z definicji

2 Zadanie 2

Weźmy najdłuższą ścieżkę w grafie G, niech ta ścieżka będzie zadana przez wierzchołki $P = set v_1, v_2, \ldots, v_k$.

Wszyscy sąsiedzi v_1 muszą być w P inaczej P mogłoby być przedłużone. Ponadto z degree v_1 ma k sąsiadów w P. Niech j będzie maksymalnym indexem sąsiada v_1. Z degree mamy więc, że $j \geq k$. Mamy więc cykl $set v_1, v_2, \ldots, v_j, v_1$ który na pewno ma długość co najmniej k+1.

3 Zadanie 3

Wiemy, że ilość krawędzi to (z lematu o uściskach dłoni) $\frac{1}{2} * \sum_{i=1}^{n} deg(v_i) = \frac{1}{2} * \sum_{i=1}^{n} (i * t_i)$

Oraz sumując stopnie wierzchołków i odejmując 1:

$n - 1 = |E| = \sum_{i=1}^{n} (t_i) - 1$

Czyli:

$\sum_{i=1}^{n} (t_i) - 1 = \frac{1}{2} * \sum_{i=1}^{n} (i * t_i)$

$\frac{1}{2} * \sum_{i=1}^{n} (i * t_i) - \sum_{i=1}^{n} (t_i) + 1 - = 0$

$\sum_{i=1}^{n} (i * t_i) - \sum_{i=1}^{n} (t_i) + 2 = 0$

$\sum_{i=1}^{n} (t_i(i - 2)) + 2 = 0$

$\sum_{i=3}^{n} (t_i(i - 2)) + 2 + t_1(-1) + t_1(0) = 0$

$\sum_{i=3}^{n} t_i(i-2) + 2 - t_1 = 0$
$t_1 = \sum_{i=3}^{n} t_i(i-2) + 2$

Liczba liści nie jest zależna od ilości wierzchołków o stopniu 2 w drzewie. Spowodowane jest to tym, że dodanie wierzchołka o stopniu 2 przedłuży tylko część drzewa i nie zostaną dodane żadne nowe liście.

4 Zadanie 4

Drzewo z definicji jest acyklicznym grafem spojnym. Jesli sa dwie sciezki z u do v, to mozemy przejsc ta sciezka, pozniej isc do wierzcholka, pozniej wrocic do v i znowu isc do u -> mamy cykl, wiec sprzecznosc.

5 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 liściami 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.

6 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:

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

Co chcieliśmy pokazać.

Obie implikacje zachodzą więc twierdzenie jest prawdziwe.

7 Zadanie 7

W grafie Q_k dwa wierzchołki są sąsiednie, gdy różnią się dokładnie jedną współrzedna.

W takim razie sasiednie weirzcholki maja rozna parzystosc wystapien 1.

W takim razie mozemy podzielic dwa podzbiory.

A - zbior wierzcholkow w ktorym 1 wystepuje parzysta ilosc razy

B- zbior wierzcholkow w ktorych l wystepuje nieparzysta ilosc razy

Skoro dwa sasiednie wierzcholki maja rozna parzystosc wystapien 1, to nie moga nalezec do tego samego podzbioru, zatem graf jest dwudzielny.

8 Zadanie 8

problem kolorownia grafu

```
visited = [False] * (n+1)
color = [False]*(n+1)
def sprawdz(G, start, visited, color):
    for u in G[start]:
        if(not visited[u]):
            visited[u]=True
            color[u] = not color[start]
            if(not sprawdz(G, u, visited, color)):
                return False
        elif(color[start] == color[u]):
            return False

    return True
```

Czemu działa? - Chcemy pokolorować graf na dwa różne kolory A i B. Zaczniemy więc od dowolnego wierzchołka w tym grafie i chcemy pokolorować wszystkich jego NIEODWIEDZONYCH sąsiadów na inny kolor niż nasz wierzchołek. Jeśli sąsiad był odwiedzony i ma ten sam kolor to możemy skończyć, no bo dwa sąsiednie wierzchołki mają już ten sam kolor. Jeśli sąsiad nie był odwiedzony i ma inny kolor to wykonujemy rekurencyjnie DFS na tym wierzchołku kolorując wszystkich jego sąsiadów w ten sam sposób. Jeśli dla jakiegokolwiek wywołania rekurencyjnego wyjdzie nam fałsz, to oczywiście cofając się rekurencyjnie przekazujemy również fałsz (if not sprawdz).

Całkowita złożoność to liczba wierzchołków + liczba krawędzi (wchodzimy do każdego wierzchołka a w każdym wierzchołku ilość krawędzi (for dla sąsiada) będzie się sumować do ogólnej liczby krawędzi).

9 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

...

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

Oznaczmy wielomian

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

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

$a_i \leq k \forall i \in \{1, 2, \dots, k+1\}$, 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 $w(k) = x^k + x^{k-1} + x^{k-2} + \dots + x^1 + x^0$
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 $a_i \leq k \forall i \in \{1, 2, \dots, k+1\}$ 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.

10 Zadanie 10

Niech P będzie najdłuższą ścieżką w grafie G, gdzie v to ostatni wierzchołek w tej ścieżce.

W takim razie wszyscy 3 sąsiedzi v są w P (bo jeśli nie to można przedłużyć o tego sąsiada)

Niech Ci sąsiedzi to będą odpowiednio a, b i c.

Rozpatrzmy 3 ścieżki:
1. Krawędź ua
2. Krawędź ub która poźniej jest przedłużona ścieżka ba.
3. Krawędź uc która później jest przedłużona ścieżką ca.
Te ścieżki dzielą punkt końcowy, więc ich połączenie jest cyklem.
Z zasady szufladkowej 2 z tych ścieżek mają taką samą parzystość.
Jesli więc weźmiemy dwie z tą samą parzystością dostaniemy cykl parzysty.