MDL 13 17.01.2024

Dominik Szczepaniak

January 18, 2024

1 Zadanie 2

Let the n vertices of the given graph G be v1, v2, . . . , vn. The Mycielski graph $y(G)$ contains G itself as a subgraph, together with n+1 additional vertices: a vertex u_i corresponding to each vertex v_i of G, and an extra vertex w. Each vertex u_i is connected by an edge to w, so that these vertices form a subgraph in the form of a star $K_{1,n}$. In addition, for each edge v_iv_j of G, the Mycielski graph includes two edges, u_iv_j and v_iu_j.

Trójkąt:

Załóżmy, że graf M_k nie zawiera trójkątów. Pokazujemy, że M_{k+1} też nie. Aby stworzyć M_{k+1} do M_k dokładamy n wierzchołków $\{u_1, u_2, ..., u_n\}$ oraz wierzchołek w. Ponieważ każdy wierzchołek u_i jest połączony z w ale nie jest połączony z żadnym $\{u_1, u_2, ..., u_n\}$ to w oczywisty sposób nie stworzymy trójkątów z wierzchołków które dodajemy. Wierzchołek w nie ma żadnego sąsiada v_i, więc pozostaje potencjalny trójkąt między $\{v_i, v_k, u_j\}$, ale u_j nie łączy dwóch sąsiednich wierzchołków v, więc jedyny trójkąt który mógłby występować jest między wierzchołkami v, ale z założenia indukcyjnego to nie zachodzi, więc nie ma trójkąta w tym grafie.

Kolorowanie:

Indukcja:

M2 to dwu wierzchołkowy graf który jest połączony za pomocą jednej krawędzi, stąd musi mieć 2 kolory, czyli podstawa indukcji X(M2) = 2 zachodzi.

Krok:

Załóżmy, $X(M_k) = k$. M_{k+1} stworzony z M_k kolorujemy:

- $c(v_i) = c(u_i)$ (bo te wierzchołki nie są połączone)

- $c(w) = k + 1$ (dla każdego w_i użyliśmy i-ty kolor więc dla w zostaje k+1).

Zatem użyliśmy k+1 kolorów.

2 Zadanie 3

Indukcja:

Dla n=1 mamy dwa regiony i malujemy je na czarno i biało.

Krok:

Załóżmy, że mamy n linii i pokolorowane regiony na czarno i biało.

Teraz jeśli przepuścimy nową linię, to przechodzi ona raz przez każdy region, przecinając każdy z nich na dwa nowe w którym każdy region sąsiadujący jest tego samego koloru. A więc potrzebujemy odwrócić kolory na jednej ze stron i mamy nasze kolorowanie. Zauważmy, że jeśli zachodziło dla n, to możemy odwrócić kolory jak chcemy i dalej będzie zachodzić, a dla nowej prostej będziemy mieć tylko nowe regiony do rozpatrzenia jako te które przecieliśmy.

Zauważmy, że ważne jest to, że jeśli odwrócimy kolory to dalej mamy kolorowanie spełniające to co chcemy.

3 Zadanie 5

W tym dowodzie korzystamy z d(v, u) = t(v, u), ale jesli mamy cykl składający się z krawędzi ujemnych wag, to nie istnieje pojęcie d(v, u), ponieważ jeśli ustalę jakieś d(v,u) to mogę przejść ten cykl jeszcze raz i dostać mniejszą wartość d(v, u).

4 Zadanie 6

Graf G:

A: (B, -4), (E, -4)
B: (A, -4), (C, 6)
C: (B, 6), (D, 3)
D: (E, 2), (C, 3)
E: (D, 2), (A, -4)
Niech A = s i C = t

Wtedy mozemy przejsc A->B->C kosztem -4 + 6 = 2
Graf G':
A->B->C daje koszt 0 + 10 = 10
A->E->D->C daje koszt 0+6+7 = 13

5 Zadanie 8

```
def czymazrodlo():
    kandytat = 0
    for i in range(1, len(macierz)):
        if(macierz[kandydat][i] and i != kandydat): #jesli otrzymamy 0 t
            continue
        else: #jesli otrzymamy 0 dla kandydata to sprawdzmy tego co ma l
            kandydat = i
    sum = 0
    for i in range(len(macierz)): #sprawdzamy czy ma krawedz do kazdego
        sum += macierz[kandydat][i]
    if(sum == len(macierz) - 1):
        sum = 0
        for i in range(len(macierz)): #sprawdzamy czy kandydat nie ma k
            sum += macierz[i][kandydat]
        return sum == 0 #jak wszystko git to mamy, a jak nie to jest fa
    return False
```

6 Zadanie 9

1. Posortujmy wierzchołki względem lewego końca odcinka któremu odpowiadają.

Wtedy dla każdego wierzchołka jego lewi sąsiedzi tworzą klikę (wszyscy mają w nim wspólny punkt, więc mają krawędzie między sobą)

2. Idąc od lewej wybieramy kolor dla wierzchołka, najmniejszy spośród niezajętych przez lewych sąsiadów wierzchołka.

Ponieważ każdy wierzchołek ma lewych sąsiadów będących kliką to ich liczność nie może przekraczać liczności największej kliki w G. Stąd liczba użytych kolorów podczas działania algorytmu bez starty ogólności jest mniejsza lub równa od liczności największej kliki w G. Czyli k <= w(G).

Wiemy, że X(G) >= w(G), bo każdy wierzchołek z kliki musi miec inny kolor.

Dodatkowo z definicji X(G) wiemy, że k>=X(G). Czyli X(G) <= k <= w(G) <= X(G), czyli k = X(G)

7 Zadanie 10

Lemat:

S jest niezależnym zbiorem G <=> S' jest pokryciem wierzchołkowym G

Dowód:

=> W S nie ma krawędzi wewnętrznych, więc dla każdej krawędzi jest albo krawędzią zewnętrzną S', albo jest pomiędzy wierzchołkami z S i S'. W obu przypadkach S' jest pokryciem wierzchołkowym.

<= Jeżeli S' jest pokryciem wierzchołkowym to każda krawędź G jest przez nie pokryta, czyli S nie ma krawędzi wewnętrznych, czyli jest zbiorem niezależnym.

Weźmy najmniejsze pokrycie wierzchołkowe, wtedy z lematu wiemy, że jego dopełnienie to niezależny zbiór wierzchołków. Ponieważ lemat to równoważność to zbiór ten jest największy (bo każde jego dopełnienie jest większe)

Stąd wiemy, że te zbiory są rozłączne i razem pokrywają cały graf G. S v S' = V

Więc $\alpha(G) + \beta(G) = n$

$\alpha(G) = n - \beta(G) = n -$ | największe skojarzenie |

G jest dwudzielny. Twierdzenie Koniga:

Największe skojarzenie możemy znaleźć algorytmem Hopcraft Karp

M <= puste

dopóki istnieje ścieżka powiększająca:

- usuń skojarzone krawędzie z M
- dodaj nieskojarzone krawędzie do M

zwróc M

8 Zadanie 11

Dobra wiemy tak - graf jest turniejem. Jest krawędź albo z u do v albo z v do u.

Rozpatrzmy dwa przypadki:

a) n jest parzyste.

Pierwszych n/2 wygrywa x+1 razy

Ostatnich n/2 wygrywa x razy

Gdzie x+1 = n/2

Każdy wygrywa z l następnymi gdzie następni są cykliczni (tj. po 6 jest 1).

Np dla n = 6

1 wygrywa z 2, 3, 4

2 wygrywa z 3, 4, 5

3 wygrywa z 4, 5, 6

4 wygrywa z 5, 6

5 wygrywa z 6, 1

6 wygrywa z 1, 2

b) n jest nieparzyste

Dla nieparzystych mamy n*(n-1)/2 meczy. No ale ponieważ n jest nieparzyste to 2 dzieli (n-1), wiec mamy podzielnosc przez n, wiec kazdy wygra (n-1)/2 meczy.

Także każdy wygrywa z (n-1)/2 następnymi gdzie następni są cykliczni np.

n = 5

1 wygrywa z 2 3

2 wygrywa z 3 4

3 wygrywa z 4 5

4 wygrywa z 5 1

5 wygrywa z 1 2

Co do podzadania 2:

Niech n = 6

$\binom{6}{2} = 15$

1 1 1 1 1 10

Suma jest 15, a to co chcemy mieć się za bardzo nie zgadza.

9 Zadanie 12

Dodajemy dwa wierzchołki s i t, gdzie s ma krawędź wychodzącą do każdego wierzchołka w grafie A o capacity b(a). Krawędź z b do t ma capacity b(b).

Krawędzie pomiędzy A i B mają capacity 1.

Niech wierzchołek a oznacza dowolny wierzchołek w A, a b oznacza dowolny wierzchołek w B.

Teraz jeśli przepuścimy Floyd-Fulkersona przez ten graf to otrzymamy największy możliwy podzbiór krawędzi który spełnia założenia naszego zadania. Dlaczego? Bo tak działa Floyd Fulkerson, a do tego wierzchołki z A mogą przyjąć maksymalnie b(a) wartość z przepływu, więc mogą wybrać maksymalnie b(a) krawędzi wychodzących z nich, co chcemy otrzymać w zadaniu (check). Wierzchołki z b natomiast mogą przyjąć maksymalnie b(b) przepływu, ponieważ jeśli przyjmą więcej to nie będą mogły przerzucić tego do t, bo przekroczą capacity krawędzi b -> t (która wynosi b(b)), czyli też mamy to co chcemy w zadaniu (check).

10 Zadanie 13

1. Przechodźmy po kolei po krawędziach w grafie G:
   - usuńmy krawędź z grafu
   - oznaczmy wierzchołki które łączyła ta krawędź przez u i v.
   - znajdźmy dijkstrą najkrótszą scieżkę między nimi.
   - dodajmy wagę krawędzi która usunęliśmy
   - $min_weight_cycle = min(min_weight_cycle, tocowyszo)$

   2. zwrócmy min_weight_cycle

   Dijstra to (E log V)
   Krawędzi jest E
   więc mamy O(E * (E log V))

11 Zadanie 14

Usuńmy krawędzie wchodzące do s i wychodzące z t - jeśli wejdziemy do s to możemy zacząć nową ścieżke i nie zabierać krawędzi, więc nie wracajmy do s. Tak samo jeśli wyjdziemy z t to będziemy zabierać krawędzie, które mogłyby być inną ścieżką.

Pokażmy, że maksymalny flow jest >= liczbie rozłącznych ścieżek dla grafu gdzie capacity na krawędziach jest równe 1.

Jeśli capacity na krawędziach jest równe 1, to jeśli przejdziemy jakąś ścieżka z s do t to zużyjemy wszystkie krawędzie którymi szliśmy w tej ścieżce.

Czyli jeśli wykorzystamy wszystkie rozłączne ścieżki to dojdziemy do t przynajmniej k razy, gdzie k to ilość rozłącznych ścieżek. Czyli maksymalny flow >= k - liczbie rozłącznych ścieżek.

Załóżmy że maksymalny flow jest większy od liczby rozłącznych ścieżek.

Rozważmy dwa przypadki:

1) możemy dotrzeć tylko za pomocą rozłącznych ścieżek do t

No ale jeśli możemy dotrzeć tylko za pomocą rozłącznych ścieżek do t, to maksymalny flow jest równy liczbie rozłącznych ścieżek od s do t.

2) użyjemy jakiś ścieżek które dzielą krawędzie (nie są rozłączne)

No ale to jest niemożliwe, ponieważ capacity = 1, więc jeśli ścieżki mają wspólną krawędź to nie możemy wykorzystać tej krawędzi więcej niż raz.

Czyli mamy, że maksymalny flow jest >= liczbie rozłącznych ścieżek oraz, że maksymalny flow nie może być większy niż liczba rozłącznych ścieżek, jesli capacity = 1, więc maksymalny flow = liczbie rozłącznych ścieżek. W takim razie użyjmy floyda pałkersona do znalezienia maksymalnego flowu w tym grafie.

12 Zadanie 15

Przekształćmy capacity na wierzchołki.

Jak to robimy?

Dzielimy każdy wierzchołek v na nowe dwa wierzchołki v_{in} oraz v_{out} i dodajemy krawędź (v_{in}, v_{out}) z capacity równa capacity wierzchołka. Później zmieniamy wszystkie krawędzie które były postaci (u, v) na (u, v_{in}) i wszystkie krawędzie postaci (v, w) na krawędzie postaci (v_{out}, w)

Usuńmy wszystkie krawędzie wchodzące do s - i tak nie będą potrzebne do wyniku bo nie spełniają warunków zadania. Tak samo usuńmy wszystkie krawędzie wychodzące z t, na tej samej podstawie.

Mamy teraz możliwość użycia floyda pałkersona. Po prostu wszędzie jest capacity = 1 i szukamy największego podzbioru krawędzi. Jeśli wszędzie capacity będzie równe 1 to do każdego wierzchołka będziemy mogli wejść tylko raz i z każdego wyjść tylko raz. W takim razie znajdziemy maksymalną liczbę ścieżek z s do t, ponieważ floyd pałkerson maksymalizuje flow, czyli dostaniemy największą możliwą ilość ścieżek, ponieważ flow będzie największy.