AISD lista 2

Wiktor Kuchta

1/4

Droga $u = u_1, u_2, \ldots, u_{k-1}, u_k = v$ do v jest sensowna wtedy i tylko wtedy, gdy $u_2, \ldots, u_k$ jest sensowna i najkrótsza ścieżka z u_2 do v jest krótsza niż najkrótsza ścieżka z u_1 do v.

Długości najkrótszych ścieżek z v oblicza algorytm Dijkstry.

Jest jedna sensowna ścieżka z v do v, a liczba sensownych ścieżek z $u \neq v$ do v jest sumą liczby sensownych ścieżek do v z sąsiadów u bliższych v. Aby to efektywnie obliczyć, możemy zmodyfikować algorytm Dijkstry.

```
procedure PATHS(G, v, u)
    for u in G do
        set-dist(v, ∞)
        prev[v] = undefined
        dodaj v do kolejki wierzchołków Q
    end for
    set-dist(v, 0)
    cnt[v] = 1
    while Q not empty do
        u ← pop-min-dist(Q)
        for n in neighbors(u) do
            alt ← dist(u) + c(u, n)
            if alt < dist(n) then
                set-dist(n, alt)
                prev[n] ← u
            else if dist(n) < dist(u) then
                cnt[u] ← cnt[u] + cntn
            end if
        end for
    end while
    return cnt[u]
end procedure
```

1/5

```
L ← topological-sort(G)
farthest ← first(L)
for v in L do
    prev[v] = null
    length[v] = 0
    for (u, v) in incoming-edges(v) do
        if prev is null or length[u] > length[prev] then
            prev ← u
        end if
    end for
    if prev is not null then
        length[v] ← length[prev] + 1
        if length[farthest] < length[v] then
            farthest ← v
        end if
    end if
end for
result ← [farthest]
while p = prev[last(result)] is not null do
    push-back(result, p)
end while
return reverse(result)
```

1/6

Problem to znalezienie mocy maksymalnego matchingu na grafie liczb a_i, gdzie mamy krawędź, jeśli jedna liczba jest co najmniej dwa razy większa od drugiej.

Jeśli mamy matching mocy k, to możemy utworzyć matching mocy k taki, że połączone są jedynie pierwsze k i ostatnie k liczb ciągu: Dopóki tak nie jest, to krawędź matchingu o końcu spoza tych końcówek ciągu możemy przepiąć do wolnego wierzchołka na odpowiednim końcu.

Jeśli mamy matching takiej postaci, to dalej możemy poprzepinać go tak, aby krawędzie były postaci (a_i, a_{n-k+i}). Zatem mocy największego matchingu możemy szukać sprawdzając tylko matchingi takiej postaci.

Możemy to zrobić zaczynając z (być może nieprawidłowym) takim matchingiem mocy $k = n/2$. Dla i od 1 do k, dopóki i-ta leksykograficznie krawędź matchingu jest niepoprawna (prawy koniec nie jest dwa razy większy od lewego), dekrementujemy k o 1, przepinając krawędzie.

1/7

Ustalmy numerację wierzchołków V taką, że kończy się ona na $v_j, v_{j-1}, \ldots, v_1$.

Algorytm Warshalla-Floyda w $(n-j)$-tej iteracji obliczy długości najkrótszych ścieżek między wszystkimi parami wierzchołków o numerach $\leqslant n-j$, a więc między wszystkimi wierzchołkami pozostałymi w G po usunięciu wierzchołków $v_1, \ldots, v_j$. Aby obliczyć żądane sumy D_j, po $(n-j)$-tej iteracji sumujemy odpowiedni fragment tablicy długości najkrótszych ścieżek.

W grafie mamy ważone wierzchołki, więc algorytm należy jeszcze zmodyfikować tak, że $dist[u][v]$ to waga ścieżki od u do v, $dist[v][v] = c(v)$ i obliczając wagę połączonych ścieżek odejmujemy wagę wierzchołka pośredniego, aby go nie policzyć dwukrotnie.