Algorytmy i Struktury Danych, 11. ćwiczenia

2023-01-11

Zadanie 11.1

Dany jest (przez listy sąsiedztwa) graf $G = (V, E)$ z wyróżnionym wierzchołkiem s. Dodatkowo każdemu wierzchołkowi przypisano dodatnią liczbę całkowitą. Zaprojektuj wydajny algorytm, który znajdzie w G najdłuższą ścieżkę o początku w s, na której liczby przypisane wierzchołkom tworzą ściśle malejący ciąg.

Rozwiązanie: Niech $f : V \to N$ oznacza funkcję, która przypisuje dla każdego wierzchołka liczbę całkowitą. Tworzymy graf $G' = (V, E')$, gdzie

$$E' = \{(u, v) \in E : f(u) > f(v)\}$$

Graf G' to **acykliczny graf skierowany** (DAG) a w nim łatwo policzyć najdłuższą ścieżkę:

Algorithm 1: Obliczanie najdłuższych ścieżek

Input: Acykliczny graf skierowany $G' = (V, E')$
Input: Tablica d zawierająca długość najdłuższej ścieżki rozpoczynającej się w danym wierzchołku
Posortuj topologicznie wierzchołki $V = (v_1, \ldots, v_n)$
foreach $i = n, \ldots, 1$ **do**
  $d[v_i] = \max(\{0\} \cup \{d[u] + 1 : (v_i, u) \in E'\})$

Złożoność czasowa i pamięciowa $O(|V| + |E|)$.

Zadanie 11.2

Skierowany graf $G = (\{1, 2, ..., n\}, E)$ nazywamy grafem przedziałowym, jeśli dla każdego wierzchołka v zbiór (numerów) wierzchołków, do których prowadzą krawędzie z v, jest przedziałem domkniętym $[l[v], r[v]]$, dla pewnych $1 \leq l[v] \leq r[v] \leq n$. W grafie mogą być pętle. Jeśli żadna krawędź nie wychodzi z v wówczas $l[v] = r[v] = 0$. Liczbę n i ciąg par $l[v], r[v]$ nazywamy zwartą reprezentacją G, a liczbę n rozmiarem tej reprezentacji.
Dana jest zwarta reprezentacja pewnego grafu G.

a) Zaprojektuj algorytm, który sprawdzi, czy graf G po usunięciu wszystkich pętli jest drzewem z korzeniem o krawędziach zorientowanych od korzenia do liści.

b) Zaprojektuj algorytm, który sprawdza, czy G jest słabo spójny (po usunięciu orientacji na krawędziach graf jest spójny).

c) Zaprojektuj algorytm, który sprawdza, czy G jest eulerowski.

Rozwiązanie: Punkt a) Jeśli $\sum_{i=1}^{n}(r[i]+1-l[i]) > 2n$ to graf G nie jest drzewem (za dużo krawędzi). Wpp możemy w czasie liniowym utworzyć standardowa reprezentację grafu G (np. jako listy sąsiedztwa) i zweryfikować czy G jest drzewem.
Punkt b) Tworzymy nowy niezorientowany graf G', który będzie zawierał dwa rodzaje krawędzi:

- $(i, l[i])$ dla $i = 1, \ldots, n$ i $l[i] \neq 0$,
- $(j, j+1)$ jeśli istnieje i takie, że $l[i] \leq j < j+1 \leq r[i]$.

Krawędzie $(j, j+1)$ możemy wyznaczyć gdy policzymy sumaryczne pokrycie prostej odcinkami $[l[i], r[i]]$. Graf G' jest spójny wtw gdy G jest słabo spójny. Graf G' ma $O(n)$ krawędzi, więc w czasie liniowym sprawdzimy czy jest spójny.
Punkt c) Sprawdź czy graf jest słabo spójny a następnie policz czy dla każdego wierzchołka $indeg[v] = outdeg[v]$. Stopnie wyjściowe można policzyć z wzoru:

$$outdeg[i] = r[i] + 1 - l[i]$$

Zauważmy, że nie musimy usuwać krawędzi typu (i, i).

Wartości *indeg* możemy policzyć korzystając z następującego algorytmu:

Algorithm 2: Obliczanie *indeg*

Input: n i tablice $l[i]/r[i]$
Output: tablica *indeg*
$A[0, \ldots, n+1] = 0$
foreach $i = 1, \ldots, n$ **do**
  $A[l[i]] += 1$
  $A[r[i]+1] -= 1$
Policz sumy prefiksowe: $indeg[i] = \sum_{j=0}^{i} A[i]$

Zadanie 11.3

Niech n będzie liczbą całkowitą większą od 2 i niech J będzie rodziną co najwyżej n różnych, domkniętych przedziałów liczb całkowitych zawartych w przedziale $[1, n]$. Grafem $G(n, J)$ nazywamy graf ($\{1, 2, ..., n\}$, $\{i - j$: istnieje przedział w J, do którego wpadają obie liczby (oba wierzchołki) i oraz $j\}$).

a) Ile jest różnych drzew BFS o korzeniu w wierzchołku 1 w grafie $G(8, J)$ dla $J = \{[1, 4], [3, 6], [5, 8]\}$?

b) Ile jest różnych drzew DFS o korzeniu w wierzchołku 1 w grafie $G(6, J)$ dla $J = \{[1, 4], [3, 6]\}$?

c) Zaprojektuj efektywny algorytm, który dla danej liczby całkowitej $n > 2$ oraz rodziny co najwyżej n różnych przedziałów J zawartych w przedziale $[1, n]$ obliczy wysokość BFS drzewa w grafie $G(n, J)$, o korzeniu w wierzchołku 1.

d) Zaprojektuj efektywny algorytm, który dla danej liczby całkowitej $n > 2$ oraz rodziny co najwyżej n różnych przedziałów J zawartych w przedziale $[1, n]$, obliczy liczbę dwuspójnych składowych w grafie $G(n, J)$.

Uwaga: na potrzeby tego zadania dwa drzewa przeszukiwania różnią się wtedy, gdy istnieje wierzchołek, który w obu drzewach ma różnych ojców.

Rozwiązanie: Punkt c) Zauważmy, że jeśli $u \leq v$ to $\mathsf{depth}[u] \leq \mathsf{depth}[v]$. Stąd jeśli obliczymy dla każdego wierzchołka $\mathsf{left}[v] = \min\{u : (u, v') \in J \text{ oraz } v' > v\}$ to możemy policzyć głębokość każdego wierzchołka (w drzewie BFS) korzystając z następującego wzoru:

$$\mathsf{depth}[u] = \begin{cases} 0 & \text{jeśli } u = 1 \\ \mathsf{depth}[\mathsf{left}[u]] + 1 & \text{wpp} \end{cases}$$

Zadanie 11.4

Zaprojektuj wydajny algorytm, który sprawdzi, czy w danym silnie spójnym grafie istnieje zamknięta (zorientowana) marszruta o nieparzystej długości. Jeżeli odpowiedzią jest TAK, znajdź jedną z takich marszrut.

Rozwiązanie: Policz drzewo BFS z dowolnego wierzchołka s i dla każdego wierzchołka wyznacz jego głębokość w drzewie BFS $d[v]$. Niech (u, v) to krawędź taka, że $d[u] \not\equiv d[v]$ (mod 2):

- jeśli taka krawędź istnieje to odpowiedzią jest TAK, ponieważ możemy skonstruować następującą marszrutę:
  - niech P_u (P_v) to ścieżka $s \to u$ ($s \to v$) korzystająca z krawędzi drzewa BFS,
  - wyznacz (np. za pomocą DFS) dowolną ścieżkę $P = v \to s$,
  - niech $M_1 = P_u + uv + P$, $M_2 = P_v + P$
  - dokładnie jedna z marszrut M_1/M_2 ma nieparzystą długość
- jeśli krawędź nie istnieje to graf jest dwudzielny więc odpowiedzią jest NIE.

Zadanie 11.5

Dany jest n-kąt wypukły W, którego wierzchołki są ponumerowane $1, 2, \ldots, n$ w kolejności ich występowania na obwodzie, $n > 2$. Ponadto danych jest k przekątnych w wielokącie W. Zaprojektuj wydajny algorytm, które sprawdza, czy istnieje para przecinających się przekątnych we wnętrzu wielokąta.

Rozwiązanie:

Zadanie sprowadza się do sprawdzenia czy istnieją dwa (niedomknięte) przedziały $I_1 = (i, j)$ i $I_2 = (k, l)$, takie, że $I_1 \not\subseteq I_2$, $I_2 \not\subseteq I_3$, $I_1 \cap I_2 \neq \emptyset$.

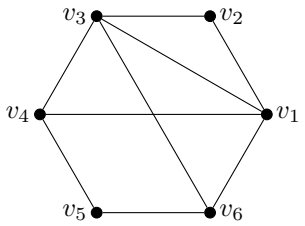

Figure 1: Przykład wielokąta z przekątnymi $(1,3)$, $(1,4)$ i $(3,6)$

Sprawdzenie czy istnieją takie przedziały możem wykonać przeglądając przedziały w odpowiednim porządku. Tworzymy kolejkę zdarzeń:

$$E = \{(l_i, +, (l_i, r_i)) : 1 \leq i \leq n\} \cup \{(r_i, -, (l_i, r_i)) : 1 \leq i \leq n\}$$

Kolejkę porządkujemy wg następującego porządku, $(t_i, s_i, (l_i, r_i)) < (t_j, s_j, (l_j, r_j))$, wtw:

- $t_i < t_j$,
- lub $t_i = t_j$ i $s_i =$ - i $s_j = +$,
- lub $t_i = t_j$ i $s_i = s_j = +$ i $r_i > r_j$,
- lub $t_i = t_j$ i $s_i = s_j =$ - i $l_i > l_j$,

Algorithm 3: Sprawdzenie czy istnieją dwa przecinające się przedziały

```
Utwórz kolejkę zdarzeń E i uporządkuj ją wg zdefiniowanego porządku
S = ∅ (pusty stos przedziałów)
foreach (t_i, s_i, (l_i, r_i)) ∈ E do
    /* Niezmiennik:  l ≤ t_i ≤ r dla (l, r) ∈ S                    */
    if s_i = + then
        Niech (l, r) = Top(S)
        if r < r_i then
            /* l < l_i < r i (l_i, r_i) ⊈ Top(S) i (l_i, r_i) ∩ Top(S) ≠ ∅  */
            return TAK
        else
            Push(S, (l_i, r_i))
    else
        /* s_i = − i Top(S) = (l_i, r_i)                           */
        Pop(S)
return NIE
```