Algorytmy i Struktury Danych, 14. ćwiczenia

2020–01–15

Spis treści

1 Find-Union implementacja

Algorithm 1: Init(n)

foreach $i \in \{1..n\}$ **do**
    $p[i] = -1$
    $size[i] = 1$

Algorithm 2: Find(i)

if $p[i] = -1$ **then**
    return i
else
    $p[i] := Find(p[i])$
    return $p[i]$

Algorithm 3: Union(i, j)

```
i = Find(i)
j = Find(j)
if i ≠ j then
    if size[j] > size[i] then
        (i, j) = (j, i)
    p[j] = i
    size[i] = size[i] + size[j]
```

2 Problem 21–1, Minimum "off–line"

Dany ciąg operacji INSERT(x) ($x \in 1, \ldots, n$, każda wartość jest dodawana co najwyżej 1 raz). oraz EXTRACT-MIN. Należy obliczyć rezultaty poszczególnych operacji EXTRACT-MIN (należy pamiętać, że cały ciąg operacji jest z góry dany).
Przykład:

$$4, 8, E, 3, E, 9, 2, 6, E, E, E, 1, 7, E, 5$$

Rozwiązanie: Rozbijamy ciąg wywołań na podciągi jednorodne:

$$I_1, E, I_2, \ldots, I_m, E, I_{m+1}$$

Gdzie każdy zbiór I_j to jakiś podzbiór kluczy (być może pusty!).

Algorithm 4: Off-Line-Minimum

```
for i ∈ 1, ..., n do
    wyznacz j takie, że i ∈ I_j
    if j ≠ m + 1 then
        extracted[j]=i
        niech l będzie najmniejszą wartością wiekszą niż j, dla której
          zbiór I_l istnieje
        I_l = I_j ∪ I_l (zbiór I_j zostaje zniszczony)
```

3 Problem wyznaczania głębokości

(w nowym wydaniu Cormena, problem na numer 21–2)
Dany jest las $\mathcal{F} = \{T_i\}$ ukorzenionych drzew z trzema operacjami:

- Make-Tree(v) tworzy drzewo składające się z węzła v,
- Find-Depth(v) zwraca głębokość węzła v w jego drzewie
- Graft(r,v) ustawia jako ojca węzła r węzeł v (zakładamy, że r jest korzeniem swojego drzewa T, oraz $v \notin T$)

Algorithm 5: Make-Tree(v)

Make-Set(v) (czyli $link[v] = nil$, $size[v] = 1$)
$parent[v]$=nil (ojciec wierzchołka v w lesie $\mathcal{F}$)
$d[v]$=0 (pseudo-głębokość v)

Algorithm 6: Find-Depth(v)

(symulujemy $Find(v)$ i sumujemy wartości $d[v]$ na ścieżce wyznaczonej przez wskaźniki $link$)
if $link[v] = nil$ **then**
    return $d[v]$
else
    niech $u = link[v]$
    $d_1 =$ Find-Depth(u)
    if $link[u] \neq nil$ **then**
        $d[v] += d[u]$
        $link[v] = link[u]$

Algorithm 7: Graft(r, v)

$parent[r] = v$
$h =$ Find-Depth(v)
$r' = Find(r)$
$v' = Find(v)$
$d[r'] += h + 1$
if $size[r'] \leq size[v']$ **then**
    $link[r'] = v'$
    $size[v'] += size[r']$
else
    (w Find-Union podłączamy węzły odwrotnie niż w lesie)
    $link[v'] = r'$
    $size[r'] += size[v']$
    $d[v'] = d[v'] - d[r']$

4 System różnych reprezentantów

Dana jest rodzina I, n niepustych podzbiorów zbioru $\{1, 2, \ldots, n\}$, z których każdy to całkowitoliczbowy przedział postaci $[i, j]$, $i \leq j$. Zaprojektuj efektywny algorytm sprawdzania, czy zadana rodzina posiada system różnych reprezentantów, a jeśli tak, to podaje jeden z nich.

Algorithm 8: SystemRóżnychReprezentantów(I)

```
for i ∈ 1, ..., n + 1 do
    Make-Set(i)
    Last[i] = i
posortuj przedziały I wg. drugiej i pierwszej współrzędnej
for [l, r] ∈ I do
    i = Last[Find-Set(l)]
    if i ≤ r then
        przypisz i jako reprezentanta [l, r]
        i' = Last[Find-Set(i + 1)]
        Union(i, i')
        Last[Find-Set(i')] = i'
    else
        BRAK ROZWIĄZANIA
```

Algorithm 9: KruskalMST

```
foreach v ∈ V do
    MakeSet(v)
T = ∅
foreach (u, v) ∈ E (w porządku niemalejących wag w(e)) do
    if Find(u) ≠ Find(v) then
        T = T ∪ (u, v)
        Union(u, v)
```

5 Algorytm Floyda-Warshalla

Algorytm oblicza najkrótsze ścieżki pomiędzy każdą parą wierzchołków. Algorytm działa poprawnie, jeśli w grafie nie istnieje cykl u ujemnej wadze.

```
1: for i, j ∈ V do
2:     dist[i, j] = d[i, j]
3: end for
4: for i ∈ V do
5:     for v1 ∈ V do
6:         for v2 ∈ V do
7:             dist[v1, v2] = min(d[v1, v2], d[v1, i] + d[i, v2])
8:         end for
9:     end for
10: end for
```

6 Cykl Eulera

Cykl Eulera w grafie skierowanym G istnieje wtedy i tylko wtedy gdy:

- dla każdego wierzchołka $v \in V(G)$ mamy $indeg(v) = outdeg(v)$.
- nieskierowana wersja grafu G (tzn. taka w której ignorujemy zwrot krawędzi), jest spójna,

- $C = \emptyset$
- tak długo jak G nie jest pusty, oblicz dowolny cykl i dołącz go do C,

7 Silne spójne składowe

Wierzchołki x, y należą do tej samej silnej spójnej składowej, jeśli istnieją ścieżki z x do y oraz z y do x.

SCC(G)

- uruchom algorytm DFS, oblicz czasy $f[v]$ dla każdego wierzchołka,
- wyznacz graf G^T (transpozycja grafu G)
- przeglądaj alg. DFS, wierzchołki G^T w kolejności malejących czasów $f[v]$,
- zwróć każde drzewo DFS jako osobną silną spójną składową.

Lemat 1. *Silne spójne składowe są identyczne w grafach G i G^T.*

Lemat 2. *Niech C i C' będą różnymi silnie spójnymi składowymi grafu skierowanego $G = (V, E)$ i niech $u, v \in C$, natomiast $u', v' \in C'$. Załóżmy, że istnieje ścieżka $u \to u'$ w G. Wówczas w G nie może istnieć ścieżka z $v' \to v$.*

Def. $d(U) = min_{u \in U} d[u]$, $f(U) = max_{u \in U} f[u]$.

Lemat 3. *Niech C i C' będą różnymi silnie spójnymi składowymi w skierowanym grafie $G = (V, E)$. Załóżmy, że istnieje krawędź $(u, v) \in E$, taka, że $u \in C$ i $v \in C'$. Wówczas $f(C) > f(C')$.*

Lemat 4. *Niech C i C' będą różnymi silnie spójnymi składowymi w skierowanym grafie $G = (V, E)$. Załóżmy, że istnieje krawedź $(u, v) \in E^T$, taka, że $u \in C$ i $v \in C'$. Wówczas $f(C) < f(C')$.*

Twierdzenie 1. *Algorytm SCC poprawieni oblicza silnie spójne składowe w skierowanym grafie G.*

8 Zadanie Graf Inwersji

Dana permutacja π_n, definiujemy graf

$$G = (V = \{1, \ldots, n\}, E = \{(i, j) : i < j \text{ i } \pi_i > \pi_j\})$$

Należy wyznaczyć spójne składowe w grafie G.
Rozwiązanie (trickowe):

Lemat 5. *Niech π_k to permutacja liczb $\{1, \ldots, k\}$, taka, że:*

- *dla* $1 \leq i < k$: $\max\{\pi[1, \ldots, i]\} \neq i$,
- $\max\{\pi[1..k]\} = k$.

To graf inwersji permutacji π_k jest spójny.

Dowód. Załóżmy, że dla udowodniliśmy, że $\pi[i+1,\ldots,k]$ tworzą jedną spójną składową.

Dla dowolnego wierzchołka $i < k$, niech $\pi[x] = \max(\pi[1,\ldots,i]) > i$, $\pi[y] = \min(\pi[i+1,\ldots,k]) \leq i$. Mamy dwa przypadki:

- $x = i$, stąd istnieje inwersja (i, y) i wierzchołek i leży w tej samej spójnej składowej co $i+1,\ldots,k$.
- $x < i$, stąd istnieją inwersje (x, i) i (x, y) więc również wierzchołek i leży w tej samej spójnej składowej co $i+1,\ldots,k$.

□

Algorithm 10: ZadanieGrafInwersji

```
i = 1; CurrMax = 0
for j = 1,...,n do
    CurrMax = max(CurrMax, π_j)
    if CurrMax = j then
        wierzchołki π_i,...,π_j tworzą spójną składową
        i = j + 1; CurrMax = 0;
```