Algorytmy i Struktury Danych, 13. ćwiczenia

2017-01-20

Spis treści

1 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 1: 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)
```

2 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 2: 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 3: 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 4: 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)
  $link[v'] = r'$
  $size[r'] += size[v']$
  $d[v'] = d[v'] - d[r']$

3 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 5: 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
```