Zadanie Park Bitowy - szkic rozwiązania

5 grudnia 2014

Proponowany szkic rozwiązania:

- zaimplementuj struktrę Drzewo, umożliwuającą na wczytanie drzewa binarnego z wejścia, wynik zapisz w tablicach `left`, `right`, `parent`,
- dodaj obliczanie tablicy `depth` zawierającą głębokość każdego wierzchołka w drzewie, $depth[1] = 0$, dla pozostałych wierzchołków $depth[v] > 0$,
- policz tablicę $links[i][v]$ dla $0 \leq i \leq \lfloor logn \rfloor$, $1 \leq v \leq n$:
  dla $i = 0$, $links[i][v] = parent[v]$,
  dla $i > 0$, $links[i][v] = links[i-1][links[i-1][v]]$ jeśli $links[i-1][v] != -1$, lub -1 wpp.
- zaimplementuj procedurę $ancestor(v, h)$, która zwraca wierzchołek oddalony od v o odległość h (idąc w kierunku korzenia), czyli np. $ancestor(v, 0) = v$, $ancestor(v, 1) = parent[v]$, $ancestor(v, depth[v]) = root = 1$:

```
res := v; i := floor(log n);
while h > 0 do
   if 2^i > h
      i := i-1
   else
      res := links[i][res]
      h := h - 2^i
return res
```

- zaimplementuj procedurę $lca(u, v)$, która zwraca najniższego wspólnego przodka u i v:

```
du := depth[u]; dv := depth[v];
if du < dv then
   v := ancestor(v, dv-du)
   dv := depth[v]
else if du > dv then
   u := ancestor(u, du-dv)
   du := depth[u]
assert du==dv
if u==v then
   return u
```

```
i := floor(log n);
while i >= 0 do
   if links[i][u] != links[i][v]
      u := links[i][u]
      v := links[j][v]
   i := i-1;
return parent[u]
```

- zaimplemenuj procedurę, która wyznaczy tablicę $far[u] = (d, v)$ oznaczającą, że najdalszy wierzchołek od u to v w odległości d

```
1. policz tablice $fardown[u]=(d,v)$ -- najdalszy wierzchołek
   w poddrzewie $u$
     for u in post-order do
        c_1 = (0, u)
        c_2 = (d+1, v) if left[u] exists and (d,v)=fardown[left[u]] or None
        c_3 = (d+1, v) if right[u] exists and (d,v)=fardown[right[u]] or None
        fardown[u] := best of c_1, c_2, c_3
2. policz tablice $farup[u]=(d,v)$ -- najdalszy wierzchołek w
   drzewie pozostałym po usunięciu synów $u$
     for u in pre-order do
        c_1 = (0, u)
        c_2 = (d+1, v) if parent[u] exists and (d,v)=farup[parent[u]] or None
        c_3 = (d+2, v) if sibling[u] exists and (d,v)=fardown[sibling[u]] or None
        farup[u] := best of c_1, c_2, c_3
3. far[u] = best of fardown[u] and farup[u]
```

- teraz do odpowiedzi na pytanie $dist(u, d)$ wystarczy:

```
(d_max, v) := far[u]
if d > d_max then return -1
l := lca(u, v)
d1 := depth[u] - depth[l]
d2 := depth[v] - depth[l]
if (d <= d1)
   return ancestor(u, d)
else
   return ancestor(v, d_max-d)
```