Algorytmy i Struktury Danych, 5. ćwiczenia

2009-11-10

1 Plan zajęć

- izomorfizm drzew,
- d-kopce,

2 Izomorfizm drzew

Algorytm:

TreeIsomorphism(T1,T2,depth)

if $T1.height > depth$ **then**
  return $(T1.height = T2.height)$;
end if
if not TreeIsomorphism(T1,T2,depth+1) **then**
  return false;
end if
for $v \in T1.nodes[depth+1] \cup T2.nodes[depth+1]$ **do**
  {w porządku rosnących etykiet}
  dodaj $value(v)$ do listy wierzchołka $parent(v)$
end for
posortuj leksykograficznie listy $value(v)$ dla $v \in T1.nodes[depth]$
posortuj leksykograficznie listy $value(v)$ dla $v \in T2.nodes[depth]$
porównaj czy listy są identyczne, jeśli nie to **return** false
zamień etykiety $value(v)$ na liczby z zakresu $1, \ldots, n$
return true

3 Izomorfizm drzew — algorytm dla drzew nieskierowanych

Znajdź w drzewach centroidy (każde drzewo zawiera co najwyżej 2 centroidy), dla każdej kombinacji ukorzeń drzewa w centroidach i uruchom poprzedni algorytm.

Niech $w(x) = max\{|subtree(t_i)| : t_i \in adj(x)\}$. *Centroid* — wierzchołek o minimalnej wadze $w(x)$.

Find(v)

niech $c1, \ldots, c_k$ synowie wierzchołka v,

jeśli $subtree(c_i) \leq n/2$ dla $1 \leq i \leq k$, to **return** v,
wpp. niech c_j wierzchołek, taki, że $subtree(c_j) > n/2$ (jest tylko jeden o tej własności),
return FIND(c_j)

FINDCENTROID(V)

ukorzeń drzew w dowolnym wierzchołku r,
oblicz wartości $subtree(v)$ dla wszystkich wierzchołków,
return FIND(r)

4 d-kopce

d–kopiec do drzewo zupełne o stopniu d z porządkiem kopcowym (min w korzeniu). Należy pokazać, że poszczególne operacje wykonuje się w czasie:

- Min — $O(1)$
- DeleteMin — $O(d \cdot \log_d(n))$
- DecreaseKey — $O(\log_d(n))$

Koszt implementacji algorytmu Dijkstry, przy użyciu d–kopców: $O(nd \cdot \log_d(n) + m \cdot \log_d(n))$.

Zanalizować jak należy dobrać d w zależności od m i n (jeśli za d weźmiemy $max(2, \lceil m/n \rceil)$ to dostajemy $O(\frac{m \log n}{\log m/n})$).

5 Rozgłaszanie komunikatów

Dane drzewo T, należy obliczyć czas potrzebny na przesłanie komunikatów do wszystkich węzłów drzewa. Przesłanie komunikatu po jednej krawędzi zajmuje 1 jednostkę czasu.
Algorytm $O(n \log n)$:

- jeśli wierzchołek jest liściem to $czas = 0$,
- wpp. rekurencyjnie oblicz czas potrzebny na rozgłoszenie w poddrzewach,
- posortuj malejąco otrzymane czasy: $t_1, \ldots, t_k$
- $czas = max\{i + t_i : 1 \leq i \leq k\}$

Aby otrzymać algorytm $O(n)$ trzeba sprytnie obliczać wartości atrybutu $czas$.

- $Q = \{$ liście T $\}$,
- while $root \notin Q$ do
  - $x = Q.extractMin()$
  - dodaj $x.czas$ do kolejki $parent(x)$,
  - jeśli $parent(x)$ ma już pełną listę poddrzew, to policz $parent(x).czas$ i dodaj $parent(x)$ do kolejki.

Kolejkę Q można zaimplementować w tablicy (i–ty element tablicy zawiera listę wierzchołków o wartości $x.czas = i$). Sumarycznie operacje $extractMin$ zajmą czas $O(n)$. Dodawanie do kolejki zajmuje czas $O(1)$.