Egzamin z ASD – zadania
27.01.2008

Zadanie 1 [12 punktów]

Dane jest n-węzłowe drzewo binarne z n różnymi kluczami w węzłach. Rozważamy następujący algorytm rekurencyjny przywracania porządku kopcowego w takim drzewie:

> Jeśli drzewo składa się z co najwyżej jednego węzła, to porządek kopcowy jest przywrócony.
> W przeciwnym przypadku znajdujemy w drzewie węzeł z najmniejszym kluczem i zamieniamy klucze z korzenia i ze znalezionego węzła; rekurencyjnie przywracamy porządek kopcowy w lewym i prawym poddrzewie korzenia.

a) **[4 punkty]** Przyjmijmy, że węzeł z najmniejszym kluczem znajdujemy jedną z metod obchodzenia drzewa (preorder, inorder lub postorder).
   a1) **[1 punkt]** Jaka jest (asymptotycznie) pesymistyczna złożoność przywracania porządku kopcowego wyrażona jako funkcja n?
   a2) **[3 punkty]** Jaka jest złożoność powyższego algorytmu dla drzewa o wysokości $\lfloor \log n \rfloor$?

b) **[8 punktów]** Zaproponuj strukturę danych, które umożliwi efektywne wykonywanie następujących operacji na n-węzłowym drzewie binarnym:

   Min(v):: znajdź w poddrzewie o korzeniu v węzeł z najmniejszym kluczem;
   Exch(u,v):: zamień klucze z węzłów u i v.

Zadanie 2 [18 punków]

Zaprojektuj strukturę danych umożliwiającą wykonywanie następujących operacji na n-wierzchołkowym lesie G (grafie nieskierowanym bez cykli):

Init(G):: utwórz strukturę danych dla grafu G zadanego przez listy sąsiedztwa (możesz przyjąć $V(G) = \{1,2, \ldots, n\}$;
Remove(u,v):: usuń krawędź $\{u, v\}$ z grafu G;
Path(u,v):: sprawdź, czy w grafie G istnieje ścieżka pomiędzy wierzchołkami u i v.

a) **[10 punktów]** Zaprojektuj takie rozwiązanie, w którym całkowity koszt wykonania operacji Init i k operacji Remove/Path nie przekracza $O(k + n\log n)$.
b) **[8 punktów]** Załóżmy, że cały ciąg k operacji jest dany „off line” (tzn. jest znany w całości z góry), a celem jest obliczenie wyników wszystkich operacji Path. Zaproponuj algorytm, który zrobi to w czasie $o(n\log n)$, gdy $k = O(n)$.

Zadanie 3 [10 punktów]

Zaproponuj efektywną strukturę danych, z pomocą której można wykonywać następujące operacje na dynamicznym zbiorze S odcinków domkniętych na prostej rzeczywistej:

Init(S):: utwórz pusty zbiór odcinków (wykonywana tylko raz na początku algorytmu);
Add(S,I):: dodaj do zbioru S nowy odcinek I, ale tylko wtedy, gdy dla każdego odcinka J z S, odcinek I ma puste przecięcie z J lub I jest zawarty w J lub J jest zawarty w I;
Delete(S,I):: usuń I z S;
Incl(S,I):: podaj ile odcinków z S jest zawartych w odcinku I.

Uwaga: W każdym zadaniu uzasadnij poprawności swoich rozwiązań i zanalizuj złożoności obliczeniowe zaproponowanych algorytmów.