Egzamin z ASD – zadania
08.02.2006

Zadanie 1 [8 pkt]
Zaprojektuj drzewo decyzyjne znajdujące medianę z pięciu różnych elementów w co najwyżej sześciu porównaniach.

Zadanie 2 [12 pkt]
Trójkątowym kaktusem nazywamy spójny graf w którym każda dwuspójna składowa jest cyklem długości trzy (trójkątem). Niech G=(V,E) będzie n-wierzchołkowym trójkątowym kaktusem. Przyjmujemy, że G jest reprezentowany za pomocą list sąsiedztwa.
a) [2 pkt] Podaj liczbę krawędzi G jako funkcję n?
b) [5 pkt] Pokaż w jaki sposób zastosować przeszukiwanie w głąb do prawidłowego pokolorowania wierzchołków grafu 3 kolorami. Zapisz swój algorytm w pseudokodzie.
c) [5 pkt] Zaproponuj efektywny algorytm znajdowania najliczniejszego skojarzenia w G.

Uzasadnij poprawność swoich rozwiązań i dokonaj analizy złożonośco obliczeniowej zaproponowanych algorytmów.

Zadanie 3 [8 pkt]
Tablica *p[1..n] of 1..n* opisuje zbiór drzew z korzeniami (las), rozpiętych na zbiorze wierzchołków {1,2,...,n} – *p[j]* jest ojcem wierzchołka *j* w lesie, jeżeli tylko *p[j] <> j*; jeżeli *p[j] = j*, to *j* jest korzeniem drzewa.

Na lesie wykonujemy następujące operacje:

parent(j):: podaj wartość *p[j]*;

union(j,k):: połącz dwa drzewa o korzeniach *j, k* w jedno, czyniąc korzeń jednego z nich ojcem drugiego korzenia;

contract(j):: jeśli *j* nie jest korzeniem, to
  niech *D* będzie zbiorem wszystkich dzieci *j* lub *p[j]*, z wyłączeniem *j*;
  wybierz wierzchołek *j* lub *p[j]*;
  niech *x* będzie wybranym wierzchołkiem, a *y* tym drugim;
  jeśli *p[j]* nie jest korzeniem, to uczyń ojcem *x* wierzchołek *p[p[j]]*, w przeciwnym razie niech *p[x] := x*;
  uczyń *x* ojcem wierzchołka *y*, jak i wszystkich wierzchołków z *D.*

Zaprojektuj sposób efektywnego wykonywania ciągu operacji *parent, union* lub *contract.* Dokonaj analizy złożoności obliczeniowej zaproponowanych algorytmów.

Zadanie 4 [12 pkt]
Zaprojektuj strukturę danych do przechowywania przedziałów domkniętych, która będzie umożliwiać wykonywanie następujących operacji, każdą w czasie O((log n)^2):

Insert(x,y) - dodaj przedział *[x,y]*
Remove(x,y) - usuń przedział *[x,y]*
Count(x,y) - oblicz liczbę przedziałów w strukturze zawartych w przedziale *[x,y]*

O liczbach *x,y* wiadomo, że są liczbami całkowitymi z przedziału *[1,n]*, dla pewnego, ustalonego całkowitego *n* > 1. Uzasadnij poprawność swojego rozwiązania.