Cognome: **Nome**: **# Matricola**: **Riga**: **Col**:

Algoritmi e Strutture Dati - Prova d'esame - Problemi
27/05/11

Esercizio 0 Scrivere correttamente nome, cognome, numero di matricola, riga e colonna.

Esercizio 1 - Punti 6

Scrivere una funzione ricorsiva che prende in input il puntatore alla radice di un albero binario i cui nodi contengono interi positivi distinti e restituisca la lunghezza del più lung cammino monotono crescente radice-discendente, dove il discendente non è necessariamente foglia; con lunghezza si intende il numero totale di *archi* attraversati; e con monotona crescente si intende che i valori contenuti nei nodi della sequenza devono essere ordinati in senso crescente da radice a discendente. Argomentare la sua correttezza e analizzare il suo costo computazionale.

Esercizio 2 - Punti 5+5+2

Si consideri il seguente problema. Data un'espressione $E = C_1O_1C_2O_2 \ldots C_{n-1}O_{n-1}C_n$, con $n \geq 2$, dove gli elementi C_i sono interi positivi e gli elementi $O_j \in \{+, \cdot\}$ sono operatori di somma o di moltiplicazione (dati). Si cerchi una parentesizzazione di valore minimo dell'espressione, utilizzando programmazione dinamica. Ad esempio, $5 + 13 \cdot 2$ può essere parentesizzata come $((5 + 13) \cdot 2) = 36$ oppure con un valore minimo $(5 + (13 \cdot 2)) = 31$.

1. Scrivere una formula ricorsiva che rappresenti la sottostruttura ottima del problema
2. Scrivere un algoritmo che restituisca il valore minimo dell'espressione
3. Scrivere un algoritmo che stampi l'espressione con valore minimo.

Esercizio 3 - Punti 6+6

In un vettore V di n interi non necessariamente ordinato, si dice double-gap un indice i, $1 \leq i < n$, tale che $V[i+1]–V[i] \geq 2$.

1. Dato un vettore V di $n \geq 2$ interi tale che $V[n]–V[1] \geq n$, provare che V ha almeno un double-gap. Suggerimento: per induzione.
2. Progettare un algoritmo che, dato un vettore V di $n \geq 2$ interi tale che $V[n]–V[1] \geq n$, restituisca la posizione (primo indice) del double-gap in $O(\log n)$ tempo.

Esercizio 4 - Punti 4+8

Si considerino n job da sottomettere ad un processore, ognuno caratterizzato da una deadline $D[i]$ e da un guadagno $G[i]$, $1 \leq i \leq n$. I vettori G e D contengono interi positivi; per semplicità assumiamo che tutti i valori siano distinti. Tutti i job hanno durata standard 1. Se il job i è eseguito entro l'istante $D[i]$ produrrà un guadagno $G[i]$, altrimenti è inutile eseguirlo perchè il guadagno sarà nullo. L'obiettivo è trovare una sequenza di esecuzione che massimizzi il guadagno.
1) Si consideri il seguente algoritmo greedy.

```
maxgain(integer[] D, integer[] G, integer n)
  { ordina i vettori D, G per guadagno decrescente, ovvero ∀i, j, 1 ≤ i < j ≤ n : G[i] < G[j] }
  integer t ← 0
  for i ← 1 to n do
      if t + 1 ≤ D[i] then                    % Decide se il job i può essere schedulato al tempo t
          print i
          t ← t + 1
```

L'algoritmo considera i job per valori decrescenti di guadagno, escludendo ovviamente quelli che hanno passato la scadenza. Si dimostri, con un controesempio, che questo algoritmo greedy non è corretto (ovvero non massimizza il guadagno).

2) Si descriva un algoritmo greedy corretto. Se ne dimostri la correttezza, utilizzando l'approccio usato a lezione: dimostrate che qualsiasi soluzione ottima può essere trasformata in una soluzione che segue la vostra scelta greedy.