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

Algoritmi e Strutture Dati – 05/06/14

Esercizio 1 – Punti ≥ 6 (Parte B)

Dati n segmenti della retta delle ascisse, dove l'i-esimo segmento inizia nella coordinata $a[i]$ e termina nella coordinata $b[i]$, scrivere un algoritmo che prenda in input i vettori a, b e la dimensione n, e restituisca il sottoinsieme di segmenti indipendenti (che non si intersecano) di copertura massimale, ovvero che copre la maggior parte della retta delle ascisse.

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale.

Esercizio 2 – Punti ≥ 6 (Parte A)

Dati due alberi binari di ricerca T ed S, scrivere un algoritmo che determini se tali alberi sono uguali o meno (sia per contenuto nelle chiavi, che strutturalmente).

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale.

Esercizio 3 – Punti ≥ 8 (Parte B)

Dati un grafo non orientato $G = (V, E)$ ed un intero k, scrivere un algoritmo che ritorni **true** se è possibile colorare il grafo usando al più k colori, in modo che ogni nodo sia colorato con un colore diverso da tutti i nodi adiacenti, **false** altrimenti.

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale.

Esercizio 4 – Punti ≥ 12 (Parte B)

Sia V un vettore contenente n interi. Chiamiamo *costo* $C(i, j)$ di un sottovettore $V[i \dots j]$ di V la somma dei suoi elementi:

$$C(i,j) = \sum_{t=i}^{j} V[t]$$

Una *k-partizione* del vettore V è una divisione del vettore in k sottovettori $V[1, j_1], V[j_1{+}1, j_2], V[j_2{+}1, j_3], \dots, V[j_{k-1}{+}1, j_k]$ con $j_k = n$ e $j_t < j_{t+1}, \forall 1 \leq t < k$, ovvero tale per cui i sottovettori coprono totalmente il vettore e non si sovrappongono.
Chiamiamo *costo della partizione* il costo massimo dei suoi sottovettori. Dato un vettore V contenente n interi e un intero k,con $2 \leq k \leq n$, il problema è trovare una partizione di V di costo minimo che divide V in k sottovettori.
Ad esempio, se $V = \{2, 3, 7, -7, 15, 2\}$, una 3-partizione possibile è $\{2, 3, 7\}, \{-7, 15\}, \{2\}$, che ha costo pari a $2 + 3 + 7 = 12$. La 3-partizione $\{2, 3\}, \{7\}\{-7, 15, 2\}$ ha costo pari a $-7 + 15 + 2 = 10$ e dovrebbe essere ottima.

1. Scrivere un algoritmo che risolva il problema nel caso particolare $k = 2$, restituendo il costo della 2-partizione ottima (suggerimento: $O(n)$).

2. Scrivere un algoritmo che risolva il problema nel caso particolare $k = 3$, restituendo il costo della 3-partizione ottima (suggerimento: $O(n^2)$).

3. Scrivere un algoritmo che risolva il problema per ogni k, restituendo il costo della k-partizione ottima (suggerimento: $O(kn^2)$).

Discutere informalmente la correttezza delle soluzioni proposte e calcolare la complessità computazionale. Ovviamente i casi con $k = 2$ e $k = 3$ possono essere risolti dall'algoritmo generale, ma per questi due valori è possibile scrivere algoritmi più efficienti e/o più semplici.