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

Algoritmi e Strutture Dati – 7/1/16

Esercizio 1 – Punti ≥ 4 (Parte A)

Un vettore V contenente n interi si dice 5-crescente se risulta vero che:

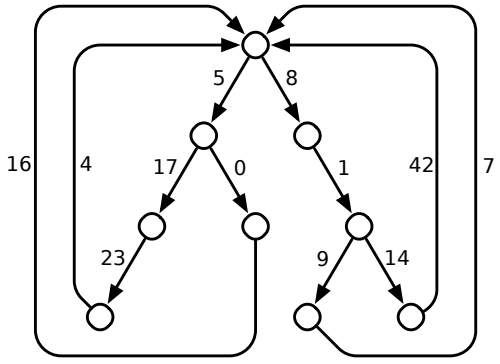
$$\forall i, j : (1 \leq i, j \leq n \wedge j \geq i + 5) \Rightarrow V[j] > V[i]$$

Si analizzi la complessità degli algoritmi di ordinamento MergeSort, QuickSort (versione *non probabilistica*), InsertionSort quando l'input è una sequenza 5-crescente. Giustificare la risposta, discutendo dei casi pessimo, medio e ottimo (se è necessario distinguere).

Esercizio 2 – Punti ≥ 8 (Parte A)

Un grafo è un "albero binario alloopato" (si pronuncia "allupato") se deriva da un albero binario nel modo seguente: è presente un nodo del grafo per ogni nodo dell'albero; è presente un arco per ogni coppia (padre,figlio), orientato dal padre al figlio; è presente un arco da ogni foglia alla radice dell'albero binario, chiudendo così un ciclo (loop). Gli archi sono pesati da una funzione $w : E \to \mathbb{Z}$; gli archi possono avere peso negativo, ma non esistono cicli la cui somma dei pesi sia negativa.
Dati in input un grafo alloopato $G = (V, E)$ con $n \geq 2$ nodi, il nodo r radice, la funzione di pesi w e due nodi u, v, scrivere un algoritmo che restituisca il peso del cammino minimo da u a v.

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale. La valutazione terrà conto del costo computazionale della soluzione, tenuto conto che è possibile risolvere il problema in $O(n)$.



Esercizio 3 – Punti ≥ 10 (Parte B)

Sia dato un insieme $\mathcal{X}$ di n intervalli della retta reale, descritti tramite due vettori $a[1 \ldots n]$ e $b[1 \ldots n]$, dove l'i-esimo intervallo è descritto da $[\ a[i], b[i]\ ]$ (estremi inclusi).
La copertura $C(\mathcal{X})$ di $\mathcal{X}$ è l'unione di tutti i punti della retta reale che sono inclusi negli intervalli in $\mathcal{X}$: $C(\mathcal{X}) = \bigcup_{I \in \mathcal{X}} I$
Scrivere un algoritmo che restituisca il più piccolo sottoinsieme $\mathcal{Y} \subseteq \mathcal{X}$, fra tutti quelli la cui copertura è uguale alla copertura di $\mathcal{X}$ ($C(\mathcal{Y}) = C(\mathcal{X})s$).

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

Ad esempio, dati gli intervalli $\mathcal{X} = \{[2, 4], [4, 7], [8, 10][3, 6]\}$, il sottoinsieme $\mathcal{Y} = \{[2, 4], [4, 7], [8, 10]\}$ ha la stessa copertura di $\mathcal{X}$.

Esercizio 4 – Punti ≥ 10 (Parte B)

In un certo linguaggio di programmazione che siete obbligati ad utilizzare, l'unica operazione disponibile sulle stringhe è il "taglio": data una stringa S di L caratteri ed una posizione k, con $1 \leq k < L$, l'operazione di divisione taglia la stringa in due stringhe di k e $L - k$ caratteri. Ad esempio, dividere `alberto` in posizione 2 restituisce `"al"` e `"berto"`. Il costo di questa operazione è pari a L, in quanto richiede di copiare la stringa originale in due diverse zone di memoria.
Supponete di avere in input una stringa di L caratteri e un vettore V di n posizioni in cui tagliare la stringa. Ad esempio, dati `"checompitodifficile"` e $V = \{3, 10\}$, il risultato è `"che"`, `"compito"`, `"difficile"`. L'ordine in cui vengono effettuati i tagli è importante, perchè a seconda dell'ordine si ottengono costi diversi. Ad esempio, supponete di avere una stringa di 19 caratteri `"checa..odiesercizio"` e di doverla spezzare nelle posizioni 3, 8, 10, ottenendo così `"che"`, `"ca..o"`, `"di"`, `"esercizio"`:

- Ordine: $3, 8, 10$: costo $19 + 16 + 11 = 46$
- Ordine: $10, 8, 3$: costo $19 + 10 + 8 = 37$
- Ordine: $10, 3, 8$: costo $19 + 10 + 7 = 36$

Scrivere un algoritmo che, dati $V[1 \ldots n]$ e L, restituisca il costo minimo necessario per spezzare la stringa originale secondo quanto descritto in V.

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