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

Algoritmi e Strutture Dati - 03/11/2016

Esercizio -1 Iscriversi allo scritto entro la scadenza. In caso di inadempienza, -1 al voto finale.

Esercizio 0 Scrivere correttamente nome, cognome, numero di matricola, riga e colonna. In caso di inadempienza, -1 al voto finale.

Nota

In questo compito, sono specificate le signature delle funzioni da utilizzare quando risolvete il problema. Se avete bisogno di una funzione con più parametri, magari definita in maniera ricorsiva, usate la funzione proposta come wrapper da cui richiamate la vostra funzione.

Esercizio 1 – Punti ≥ 6 (Parte A)

Trovare un limite **superiore** alla seguente equazione di ricorrenza, utilizzando il metodo di sostituzione (detto anche per tentativi), facendo particolare attenzione ai casi base.

$$T(n) = \begin{cases} 27\,T(\lfloor n/9 \rfloor) + n\sqrt{n} & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Esercizio 2 – Punti ≥ 9 (Parte A)

Si consideri un grafo orientato aciclico $G = (V, E)$, in cui ogni nodo $v \in V$ sia associato ad un *fattore* reale positivo $v.f$. Un cammino crescente è un cammino $C = v_1, v_2, \ldots, v_k$ di k nodi e $k-1$ archi tale per cui, per ogni $i \in \{2, 3, \ldots, k\}$

- $v_{i-1}.f < v_i.f$
- $(v_{i-1}, v_i) \in E$

Scrivere un algoritmo **int** longestIncreasing(GRAPH G) che preso in input G, restituisca la lunghezza del più lungo cammino crescente presente nel grafo.

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

Esercizio 3 – Punti ≥ 7 (Parte A)

Si consideri un albero binario con radice T, in cui ogni nodo v sia associato ad un *peso* intero (non necessariamente positivo) $v.p$. Il *costo* di una foglia è dato dalla somma dei pesi dei nodi appartenenti al cammino semplice radice-foglia (ricordiamo che un cammino semplice è un cammino che attraversa i collegamenti dal padre ai figli, mai dai figli al padre). Ad esempio, considerato un cammino semplice $C = v_1, v_2, \ldots, v_k$ composto da k nodi, dove v_1 è la radice e v_k è una foglia, allora il costo di v_k è dato da $\sum_{i=1}^{k} v_i.p$ Una foglia ha costo *massimale* se il suo costo è maggiore o uguale al costo di ogni altra foglia.
Scrivere un algoritmo **int** maxCount(TREE T) che dato in input T, restituisca il numero di foglie che hanno costo massimale.

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

Esercizio 4 – Punti ≥ 8 (Parte A)

Sia V un vettore contenente n interi, con le seguenti proprietà:

- $n \geq 3$
- $V[n] > 0$ e $V[1] < 0$
- per ogni indice $i \in \{2 \ldots n\}$, $|V[i-1] - V[i]| \leq 1$

Scrivere un algoritmo **int** zero(**int**[] V, **int** n) che restituisca un indice i tale per cui $V[i] = 0$, se questo esiste, -1 altrimenti. Soluzioni con complessità $\Omega(n)$ non verranno tenute in considerazione.

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

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

Algoritmi e Strutture Dati - 03/11/16

Esercizio -1 Iscriversi allo scritto entro la scadenza. In caso di inadempienza, -1 al voto finale.

Esercizio 0 Scrivere correttamente nome, cognome, numero di matricola, riga e colonna. In caso di inadempienza, -1 al voto finale.

Nota

In questo compito, sono specificate le signature delle funzioni da utilizzare quando risolvete il problema. Se avete bisogno di una funzione con più parametri, magari definita in maniera ricorsiva, usate la funzione proposta come wrapper da cui richiamate la vostra funzione.

Esercizio 1 – Punti ≥ 6 (Parte A)

Trovare un limite **inferiore** alla seguente equazione di ricorrenza, utilizzando il metodo di sostituzione (detto anche per tentativi), facendo particolare attenzione ai casi base.

$$T(n) = \begin{cases} 64\,T(\lceil n/16 \rceil) + n\sqrt{n} & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Esercizio 2 – Punti ≥ 8 (Parte A)

Sia V un vettore contenente n interi, con le seguenti proprietà:

- $n \geq 3$
- per ogni indice $i \in \{2 \dots n\}$, $|V[i-1] - V[i]| \leq 1$
- $V[1] > 0$ e $V[n] < 0$

Scrivere un algoritmo **int** zero(**int**[ ] V, **int** n) che restituisca un indice i tale per cui $V[i] = 0$, se questo esiste, -1 altrimenti. Soluzioni con complessità $\Omega(n)$ non verranno tenute in considerazione.

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

Esercizio 3 – Punti ≥ 7 (Parte A)

Si consideri un albero binario con radice T, in cui ogni nodo v sia associato ad un *fattore* reale positivo $v.f$. Il *costo* di una foglia è dato dal prodotto dei fattori di tutti i nodi appartenenti al cammino semplice radice-foglia (ricordiamo che un cammino semplice è un cammino che attraversa i collegamenti dal padre ai figli, mai dai figli al padre). Ad esempio, considerato un cammino semplice $C = v_1, v_2, \dots, v_k$ composto da k nodi, dove v_1 è la radice e v_k è una foglia, allora il costo di v_k è dato da $\prod_{i=1}^{k} v_i.f$ Una foglia ha costo *minimale* se il suo costo è minore o uguale al costo di ogni altra foglia.
Scrivere un algoritmo **int** minCount(TREE T) che dato in input T, restituisca il numero di foglie che hanno costo minimale. Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale.

Esercizio 4 – Punti ≥ 9 (Parte A)

Si consideri un grafo orientato aciclico $G = (V, E)$, in cui ogni nodo $v \in V$ sia associato ad un *peso* intero $v.p$ (non necessariamente positivo). Un *cammino decrescente* è un cammino $C = v_1, v_2, \dots, v_k$ di k nodi e $k-1$ archi tale per cui, per ogni $i \in \{2, 3, \dots, k\}$

- $v_{i-1}.p > v_i.p$
- $(v_{i-1}, v_i) \in E$

Scrivere un algoritmo **int** longestDecreasing(GRAPH G) che preso in input G, restituisca la lunghezza del più lungo cammino decrescente presente nel grafo. Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale.