Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 21 settembre 2017

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1, 2, 3 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 4 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su tutto ciò che consegnate.*

1. Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `20 24 33 18 16 25 28 23`

(a) Disegnate l'albero AVL.

(b) Disegnate l'albero 2-3.

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ampiezza* dell'albero AVL.

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine anticipato* dell'albero AVL.

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato* dell'albero AVL.

(f) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine anticipato "al rovescio"* dell'albero AVL, cioè visitando prima la radice, poi i sottoalberi destro e sinistro (i sottoalberi, se non vuoti, sono visitati ricorsivamente nello stesso modo).

(g) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato "al rovescio"* dell'albero AVL, cioè visitando i sottoalberi destro e sinistro, e infine la radice (i sottoalberi, se non vuoti, sono visitati ricorsivamente nello stesso modo).

2. Considerate la seguente sequenza di numeri memorizzata in un array che deve essere ordinata in modo crescente:
`20 24 33 18 16 25 28 23`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo le prime due iterazioni del ciclo principale dell'algoritmo.

3. Nel riquadro che segue ciascuna affermazione, scrivete V se l'affermazione è vera, F se è falsa:

(a) Gli algoritmi basati sulla *programmazione dinamica* utilizzano la ricorsione per risolvere efficientemente problemi per cui altri tipi di algoritmi risultano del tutto inefficienti. ☐

(b) Quando il fattore di carico di una tabella hash risulta elevato, è utile riordinare gli elementi nella tabella per evitare che le operazioni di ricerca impieghino troppo tempo. ☐

(c) Esiste un algoritmo per moltiplicare due matrici quadrate $n \times n$ contenenti numeri interi, che effettua un numero di operazioni che, in funzione di n, cresce meno di n^3. ☐

(d) L'elemento massimo di un array non ordinato di n elementi può essere trovato effettuando $O(\log n)$ confronti. ☐

(e) Per trovare il massimo e il minimo della diagonale di una matrice $n \times n$, è necessario, prima di tutto, costruire un array ordinato contenente gli elementi della diagonale. Pertanto, se si utilizza un algoritmo basato su confronti, il numero di confronti necessari è almeno dell'ordine di $n \log n$. ☐

(f) Il massimo e il minimo di una matrice quadrata $n \times n$ possono essere trovati effettuando $O(n \log n)$ confronti. ☐

(g) Ogni grafo non orientato con n vertici e almeno $2n$ archi possiede un albero ricoprente. ☐

(h) Applicando l'algoritmo di Prim a un grafo non orientato *non connesso* si ottiene una foresta di alberi che ricopre il grafo. ☐

(i) Applicando l'algoritmo di Kruskal a un grafo non orientato *non connesso* si ottiene una foresta di alberi che ricopre il grafo. ☐

(j) Se P≠NP allora per trovare il cammino minimo tra due nodi di un grafo orientato e pesato è necessario tempo esponenziale. ☐

4. Un istituto di credito vuole realizzare un nuovo software per i propri terminali bancomat. Al momento del prelievo, il terminale dovrà erogare la somma richiesta dal cliente utilizzando un numero di banconote minimo, in base alle banconote disponibili al momento. Poiché la risposta deve essere fornita in tempi rapidi, si decide di risolvere il problema mediante un algoritmo greedy.

Input: La quantità $q \geq 0$ di banconote disponibili per ciascun taglio; la somma richiesta da cliente.
Output: Il numero di banconote erogate per ciascun taglio o, *in alternativa*, un messaggio che comunichi che la somma richiesta non è erogabile.
Semplificazione: I tagli delle possibili banconote sono fissati: € 10, 20, 50, 100, 200.

Esempio: Date le disponibilità indicate nella tabella a destra, se la somma richiesta è € 260 una soluzione ottima è data da una banconota da € 200, una da € 50 e una da € 10. La soluzione che consiste di una banconota da € 200 e tre banconote da € 20 non è ottima.

€	10	20	50	100	200
q	74	80	80	15	10

Cosa si richiede:

(a) Descrivete *sinteticamente a parole* e poi ad alto livello in pseudocodice, un algoritmo che utilizzando la *tecnica greedy* cerchi di risolvere il problema. Nel caso l'algoritmo non riesca a determinare una soluzione corrispondente alla cifra richiesta dal cliente, dovrà comunicare che la somma non è erogabile.

(b) Indicate se l'algoritmo trova sempre una soluzione ottima oppure no, motivando la risposta. In particolare, in caso di risposta negativa, presentate un esempio in cui l'algoritmo determina una soluzione che *non è* ottima (indicate l'input, l'output prodotto dal vostro algoritmo e una soluzione ottima).

(c) È possibile che il vostro algoritmo indichi erroneamente che la somma richiesta non è erogabile? Motivate la risposta. In particolare, in caso di risposta affermativa presentate un esempio in cui l'algoritmo indica erroneamente che la somma richiesta non è erogabile (indicate l'input, le scelte effettuate dall'algoritmo, e una possibile soluzione).

Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 21 settembre 2017

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1, 2, 3 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 4 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su tutto ciò che consegnate.*

1. Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `22 26 35 20 18 27 30 25`

(a) Disegnate l'albero AVL.

(b) Disegnate l'albero 2-3.

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ampiezza* dell'albero AVL.

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine anticipato* dell'albero AVL.

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato* dell'albero AVL.

(f) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine anticipato "al rovescio"* dell'albero AVL, cioè visitando prima la radice, poi i sottoalberi destro e sinistro (i sottoalberi, se non vuoti, sono visitati ricorsivamente nello stesso modo).

(g) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato "al rovescio"* dell'albero AVL, cioè visitando i sottoalberi destro e sinistro, e infine la radice (i sottoalberi, se non vuoti, sono visitati ricorsivamente nello stesso modo).

2. Considerate la seguente sequenza di numeri memorizzata in un array che deve essere ordinata in modo crescente:
`22 26 35 20 18 27 30 25`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo le prime due iterazioni del ciclo principale dell'algoritmo.

3. Nel riquadro che segue ciascuna affermazione, scrivete V se l'affermazione è vera, F se è falsa:

(a) Il massimo e il minimo di una matrice quadrata $n \times n$ possono essere trovati effettuando $O(n \log n)$ confronti. ☐

(b) L'elemento massimo di un array non ordinato di n elementi può essere trovato effettuando $O(\log n)$ confronti. ☐

(c) Per trovare il massimo e il minimo della diagonale di una matrice $n \times n$, è necessario, prima di tutto, costruire un array ordinato contenente gli elementi della diagonale. Pertanto, se si utilizza un algoritmo basato su confronti, il numero di confronti necessari è almeno dell'ordine di $n \log n$. ☐

(d) Esiste un algoritmo per moltiplicare due matrici quadrate $n \times n$ contenenti numeri interi, che effettua un numero di operazioni che, in funzione di n, cresce meno di n^3. ☐

(e) Quando il fattore di carico di una tabella hash risulta elevato, è utile riordinare gli elementi nella tabella per evitare che le operazioni di ricerca impieghino troppo tempo. ☐

(f) Se P≠NP allora per trovare il cammino minimo tra due nodi di un grafo orientato e pesato è necessario tempo esponenziale. ☐

(g) Applicando l'algoritmo di Kruskal a un grafo non orientato *non connesso* si ottiene una foresta di alberi che ricopre il grafo. ☐

(h) Applicando l'algoritmo di Prim a un grafo non orientato *non connesso* si ottiene una foresta di alberi che ricopre il grafo. ☐

(i) Ogni grafo non orientato con n vertici e almeno $2n$ archi possiede un albero ricoprente. ☐

(j) Gli algoritmi basati sulla *programmazione dinamica* utilizzano la ricorsione per risolvere efficientemente problemi per cui altri tipi di algoritmi risultano del tutto inefficienti. ☐

4. Un istituto di credito vuole realizzare un nuovo software per i propri terminali bancomat. Al momento del prelievo, il terminale dovrà erogare la somma richiesta dal cliente utilizzando un numero di banconote minimo, in base alle banconote disponibili al momento. Poiché la risposta deve essere fornita in tempi rapidi, si decide di risolvere il problema mediante un algoritmo greedy.

Input: La quantità $q \geq 0$ di banconote disponibili per ciascun taglio; la somma richiesta da cliente.
Output: Il numero di banconote erogate per ciascun taglio o, *in alternativa*, un messaggio che comunichi che la somma richiesta non è erogabile.
Semplificazione: I tagli delle possibili banconote sono fissati: € 10, 20, 50, 100, 200.

Esempio: Date le disponibilità indicate nella tabella a destra, se la somma richiesta è € 260 una soluzione ottima è data da una banconota da € 200, una da € 50 e una da € 10. La soluzione che consiste di una banconota da € 200 e tre banconote da € 20 non è ottima.

€	10	20	50	100	200
q	74	80	80	15	10

Cosa si richiede:

(a) Descrivete *sinteticamente a parole* e poi ad alto livello in pseudocodice, un algoritmo che utilizzando la *tecnica greedy* cerchi di risolvere il problema. Nel caso l'algoritmo non riesca a determinare una soluzione corrispondente alla cifra richiesta dal cliente, dovrà comunicare che la somma non è erogabile.

(b) Indicate se l'algoritmo trova sempre una soluzione ottima oppure no, motivando la risposta. In particolare, in caso di risposta negativa, presentate un esempio in cui l'algoritmo determina una soluzione che *non è* ottima (indicate l'input, l'output prodotto dal vostro algoritmo e una soluzione ottima).

(c) È possibile che il vostro algoritmo indichi erroneamente che la somma richiesta non è erogabile? Motivate la risposta. In particolare, in caso di risposta affermativa presentate un esempio in cui l'algoritmo indica erroneamente che la somma richiesta non è erogabile (indicate l'input, le scelte effettuate dall'algoritmo, e una possibile soluzione).