Esame di	**Fondamenti di informatica II - Algoritmi e strutture dati**	**12 CFU**
	Algoritmi e strutture dati V.O.	**5 CFU**
	Algoritmi e strutture dati (Nettuno)	**6 CFU**

Appello del 13-7-2017 – a.a. 2016-17 – Tempo a disposizione: *4 ore* – somma punti: 35

Istruzioni

Lanciare la macchina virtuale Oracle VirtualBox e lavorare all'interno della cartella `ESAME`, avendo cura di creare all'interno della cartella stessa:

- un file `studente.txt` contenente, una stringa per riga, cognome, nome, matricola, email; in tutto quattro righe, memorizzando il file nella cartella `ESAME`;
- una cartella `java.<matricola>`, o `c.<matricola>`, ove al posto di `<matricola>` occorrerà scrivere il proprio numero di matricola, **contenente i file prodotti per risolvere il Problema 2** (in tale cartella si copi il contenuto dell'archivio `c-aux.zip` o `java-aux.zip`); tale cartella va posizionata nella cartella `ESAME`;
- tre altri file `probl1.<matricola>.txt`, `probl3.<matricola>.txt` e `probl4.<matricola>.txt`, contenenti, rispettivamente, gli svolgimenti dei problemi 1, 3 e 4; i tre file vanno posti nella cartella `ESAME`.

È possibile consegnare materiale cartaceo integrativo, che verrà esaminato solo a condizione che risulti ben leggibile.

Per l'esercizio di programmazione (Problema 2) è possibile usare qualsiasi ambiente di sviluppo disponibile sulla macchina virtuale. Si raccomanda però di controllare che i file vengano salvati nella cartella `java.<matricola>`, o `c.<matricola>`. Si consiglia inoltre per chi sviluppa in `c` di compilare da shell eseguendo il comando `make` e poi eseguire `driver` per verificare la correttezza dell'implementazione. Analogamente si raccomanda per chi sviluppa in `java` di compilare da shell eseguendo il comando `javac *.java` e poi eseguire `java Driver` per verificare la correttezza dell'algoritmo.

N.B. Le implementazioni debbono essere compilabili. In caso contrario, l'esame non è superato.

Problema 1 Analisi algoritmo [(a) 4/30; (b) 2/30]

Si considerino i metodi Java di seguito illustrati.

```
static int problema(int a[], int i, int j) {
    if(i == j) return a[i];
    if(i + 1 == j) return a[i] + a[j];
    int n = (j + 1 - i) / 3;
    return problema(a, i, i+n-1) + problema(a, i+n, i+2*n-1) + problema(a, i+2*n, j);
}

// array a contiene valori non negativi
static double problema(int a[]) {
    if(a.length == 0) return -1;
    return problema(a, 0, a.length-1) / (double)a.length;
}
```

Sviluppare, *argomentando adeguatamente* (il 50% del punteggio dell'esercizio sarà sulle argomentazioni addotte), quanto segue:

(a) Determinare il costo asintotico dell'algoritmo descritto da `problema(int[])` in funzione della dimensione dell'input.

(b) È possibile riformulare l'algoritmo `problema` in modo non ricorsivo senza introdurre una pila? Discutere (e mostrare la riformulazione).

Problema 2 Progetto algoritmo C/Java [soglia minima: 5/30]

Con riferimento agli alberi binari di ricerca (BST), impiegati per realizzare mappe con chiavi bidimensionali (x, y) di tipo `int` non negative e memorizzate all'interno della classe/struttura `BinNode` nel vettore `coordinates` (coordinate cartesiane di un punto), risolvere al computer quanto segue, in Java o in C. Si impieghi la rappresentazione basata su classe/struttura `BST` e classe/struttura `BinNode`.

- Realizzare una funzione/metodo `BST_insert` per inserire in un BST una nuova chiave, usando la seguente relazione d'ordine fra le chiavi: $(x_1, y_1) < (x_2, y_2)$ se e solo se $(x_1 < x_2) \vee (x_1 = x_2 \ \wedge \ y_1 < y_2)$. La funzione/metodo deve restituire un riferimento/puntatore al nodo appena inserito; se un nodo con la coppia di chiavi in ingresso esiste già, allora si restituisce `null`/`NULL`. [3/30]

- Realizzare una funzione/metodo `aligned` che, dato in input un valore di ascissa, determini il numero di nodi nell'albero che hanno la stessa ascissa. [3/30]

- Realizzare una funzione/metodo `rangeQ` che, dati in input due vertici opposti di un rettangolo isotetico[1] (x_1, y_1) e (x_2, y_2), restituisca il numero di nodi inclusi nel rettangolo, che va considerato come un insieme chiuso. [4/30]

È necessario implementare i metodi in `bst.c` o `bst.java` identificati dal commento `\*DA IMPLEMENTARE*\`. In tali file è permesso sviluppare nuovi metodi se si ritiene necessario. *Non è assolutamente consentito modificare metodi e strutture già implementati.* È invece possibile modificare il file `driver` per poter effettuare ulteriori test.

Problema 3 Alberi di Fibonacci [(a) 2/30; (b) 2/30; (c) 3/30]

(a) Definire con la maggiore precisione possibile cos'è un albero di Fibonacci.

(b) Spiegare la relazione che intercorre fra gli alberi di Fibonacci e la nota sequenza numerica detta "di Fibonacci."

(c) Spiegare perché gli alberi di Fibonacci sono interessanti, spiegando in particolare cosa essi consentono di dimostrare (e come).

Problema 4 Problema su grafi [(a) 3/30; (b) 5/30; (c) 4/30]

Un centro rurale sito in un luogo particolarmente impervio è stato finalmente raggiunto dal sistema di distribuzione di energia elettrica ed è stata predisposta, nei pressi del centro rurale, una mini-centrale di distribuzione, sita in una posizione nota (x_0, y_0). Il centro rurale è composto da n edifici di coordinate note $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, che debbono essere tutti raggiunti

[1] Isotetico: con lati paralleli agli assi coordinati.

da un cavo elettrico: è essenziale che ciascun edificio sia raggiunto dal servizio di fornitura di energia elettrica, ma non è indispensabile che un cavo colleghi direttamente la centrale a ogni edificio. Infatti, è parimenti accettabile che un edificio sia raggiunto dalla fornitura di energia elettrica indirettamente, cioè attraverso altri edifici, come illustrato in Fig. 1, ove la centrale C è direttamente collegata agli edifici 1, 2, 3, 4 e 5, ma l'edificio 7 è raggiunto da cavi dopo che questi hanno già raggiunto l'edificio 3, così come l'edificio 6 è collegato a C attraverso 3 e 7. Ad ogni modo, nel centro rurale è possibile collegare con cavi qualunque coppia di punti, siano essi edifici o centro di distribuzione. La Fig. 2 mostra un'altra soluzione ammissibile, in cui tutti gli edifici, tranne il num. 1, sono collegati a C in modo indiretto. Per ovvie ragioni di risparmio si evitano le situazioni ridondanti (collegamenti multipli).

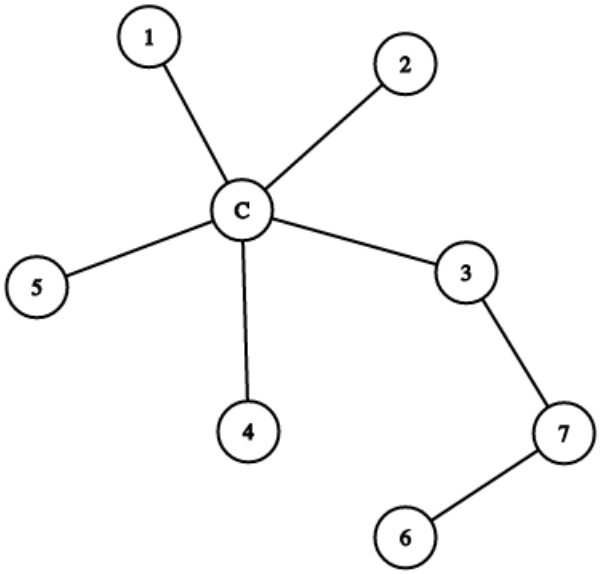

Fig. 1

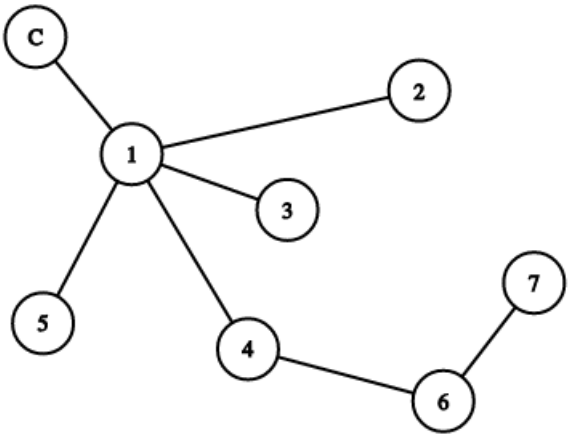

Fig. 2

Ciò premesso, occorre risolvere il seguente problema: date le coordinate degli n edifici e quelle del centro di distribuzione C, collegare, direttamente o indirettamente, tutti gli edifici al centro C minimizzando la lunghezza totale del cavo impiegato (la somma cioè delle lunghezze delle tratte). Ai fini dell'esame sviluppare quanto segue:

(a) Formulare il problema posto come un problema su grafi. A tal fine occorre dapprima definire il grafo a cui si farà riferimento, definendone vertici, archi e le caratteristiche generali (semplice, diretto, pesato ecc.), e chiarendo come esso sia collegato al problema assegnato; quindi bisogna formulare quest'ultimo come problema su grafi, utilizzando in particolare una terminologia appropriata, pertinente al mondo dei grafi.
N.B. Qui specificare *quale problema su grafi* deve essere risolto, non *come*.

(b) Definire un algoritmo (pseudo-codice) per la soluzione del problema: date in input le coordinate[2] di n edifici $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ e quelle della centrale (x_0, y_0), determinare, restituendolo in uscita, un insieme di tratte che minimizzi la quantità totale di cavo impiegato. Ciascuna tratta è espressa come una coppia (h, k), essendo h e k due elementi di $\{0, 1, 2, \ldots, n\}$, avendo stabilito di denotare con il numero 0 la centrale C e con l'intero i l'edificio di coordinate (x_i, y_i).

L'algoritmo dovrà dapprima convertire l'input in un opportuno grafo, risolvere su tale grafo il problema posto, così come descritto al punto (a), e quindi fornire in output le grandezze richieste.
N.B. Lo pseudo-codice privo di indentazione riceve penalizzazione del 20%.

[2]Le coordinate sono numeri reali.

(c) Scegliere una idonea rappresentazione del grafo e determinare i costi computazionali (tempo e spazio) dell'algoritmo sviluppato al punto (b). Il 50% del punteggio sarà attribuito alle giustificazioni addotte.
N.B. Lo svolgimento del punto (c) non sarà valutato se il punto (b) non presenta un algoritmo sostanzialmente corretto.