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

Appello del 2-2-2017 – a.a. 2016-17 – Tempo a disposizione: *120 minuti* – somma punti: 34

Problema 1 Analisi algoritmo [(a) 6/30; (b) 3/30]

Si considerino i metodi Java di seguito illustrati.

```
// assumere a[i] > 0 per ogni i
static void rs(int[] a) {
  if(a.length < 2) return;
  int max = a[0];
  for(int i = 1; i < a.length; i++) if(a[i] > max) max = a[i];
  rs(a, 0, a.length, 0x1 << lg2(max));
}

// assumere a[i] > 0 per ogni i
static void rs(int[] a, int start, int endp1, int mask) {
  if((endp1 - start <= 1) || (mask == 0)) return;
  int j = start, k = endp1;
  while(j < k) {
    while((j < k) && ((a[j] & mask) == 0x0)) j++;
    while((j < k) && ((a[k-1] & mask) != 0x0)) k--;
    if(j < k) swap(a, j, k-1);
  }
  rs(a, start, j, mask >>> 1);
  rs(a, j, endp1, mask >>> 1);
}

static void swap(int[] a, int j, int i) {
  int tmp = a[i]; a[i] = a[j]; a[j] = tmp;
}

// assumere x > 0
static int lg2(int x) {
  int r = -1;
  while(x > 0) { x /= 2; r++; }
  return r;
}
```

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

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

(b) Discutere se `rs` opera *in place* oppure no (risposte del tipo "sì, opera in place" o "no, non opera in place", ma prive di discussione, saranno completamente irrilevanti).

Problema 2 Progetto algoritmo C/Java [8/30]

Per rappresentare un file system viene impiegato un albero i cui nodi possiedono un campo `name` di tipo stringa, in cui sono ammessi tutti i caratteri eccetto il carattere *separatore* `'/'`, e un campo `size` di tipo intero, che esprime la dimensione del file o della cartella[1]; tali nodi possono inoltre avere un

[1]Per dimensione di casella si intende la dimensione della struttura dati che mantiene l'elenco dei contenuti della cartella e non la somma delle dimensione dei file/cartelle in essa contenuti.

numero arbitrario di figli. In un tale albero un nodo interno rappresenta una cartella, mentre una foglia può rappresentare un file o una cartella vuota. Naturalmente, per determinare l'occupazione di una cartella è necessario sommare le dimensione di tutti gli oggetti in essa contenuti.

Ciò premesso, si richiede di progettare un algoritmo (Java o C) che, data la radice r di un tale albero, e dato un intero positivo t, stampi su standard output i percorsi assoluti di tutte le cartelle i cui contenuti occupano almeno t bytes. L'algoritmo deve operare in tempo lineare (si consideri costante il tempo necessario a concatenare stringhe).

N.B. Il codice sorgente privo di indentazione riceve penalizzazione del 20%.

Problema 3 Complessità temporali

Con riferimento al tipo astratto *mappa ordinata*, e alle strutture concrete riportate in tabella, indicare i costi computazionali (temporali) di *caso peggiore* delle operazioni elencate. Usare la notazione Θ. (Costo corretto: +0.3; costo errato: −0.15; costo assente: 0)

	put	get	remove	predecessor	max
lista disordinata					
lista ordinata					
BST					
AVL					

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

B	C	D	E	F	
d_1	d_2	d_3	d_4	d_5	A
	d_6	d_7	d_8	d_9	B
		d_{10}	d_{11}	d_{12}	C
			d_{13}	d_{14}	D
				d_{15}	E

Un'ampia area geografica è servita da un sistema di trasporto ferroviario che consiste di n stazioni e di m tratte, essendo ogni tratta un collegamento (bidirezionale) fra due stazioni, di cui è nota la lunghezza. L'azienda di trasporto intende definire un piano di investimenti e ha bisogno di una tabella che esprime, per ogni coppia di stazioni, la distanza su ferrovia, transitando attraverso 0 o più stazioni intermedie. Un esempio di tabella da produrre è mostrato nella figura a sinistra, ove si assume per semplicità che ci siano solo le stazioni A, B, C, D, E ed F. Ciascuna cella deve contenere la distanza (ferroviaria) d_i fra la coppia di stazioni associata alla cella. Ciò premesso, si richiede di:

(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 posto; 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: dati in input n stazioni $T_1, \ldots, T_n$, un insieme F di m tratte del tipo $\{h, k\} \in F \subseteq \{1, \ldots, n\}^2$, ove la tratta $\{h, k\}$ denota il collegamento ferroviario (bidirezionale) fra T_h e T_k, e una funzione $w : F \mapsto \mathbb{R}^+$, determinare le $n(n-1)/2$ distanze ferroviarie fra tutte le possibili coppie di stazioni, ove ciascuna distanza è un elemento del dominio $\mathbb{R}^+ \cup \{\infty\}$, essendo la distanza ∞ una convenzione che indica la mancanza di un collegamento attraverso ferrovia fra le due stazioni. Specificare con chiarezza come l'output viene rappresentato.
N.B. Lo pseudo-codice privo di indentazione riceve penalizzazione del 20%.

(c) Determinare i costi computazionali dell'algoritmo (tempo e spazio) nel caso di rappresentazione basata su *liste di adiacenza* e su *matrice di adiacenza*. Il 50% del punteggio sarà attribuito alle giustificazioni addotte.