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 9-1-2017 – a.a. 2016-17 – Tempo a disposizione: *120 minuti* – somma punti: 33.6

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

Si considerino i metodi Java di seguito illustrati.

```
static void f1(int[] a, int n, int p) {
  if((n > 0) && (a[n-1] > p)) {
    a[n] = a[n-1];
    f1(a, n-1, p);
  } else a[n] = p;
}

static void f2(int[] a, int i) {
  if(i < a.length) {
    f1(a, i, a[i]);
    f2(a, ++i);
  }
}

static void f3(int[] a) {
  if(a.length > 1) f2(a, 1);
}
```

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

(a) Determinare il costo asintotico dell'algoritmo descritto da `f3(int[])` in funzione della dimensione dell'input.

(b) Discutere se `f3` 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'etichetta di tipo stringa, in cui sono ammessi tutti i caratteri eccetto il carattere speciale `'/'`, detto *separatore*, e possono avere un numero arbitrario di figli. Dato un nodo v denotiamo con $e(v)$ l'etichetta associata a v.

Progettare un algoritmo (Java o C) che, data la radice r di un tale albero, e data una stringa s non contenente il separatore, stampi su standard output i percorsi completi[1] di tutti i nodi che hanno etichetta s. L'algoritmo deve operare in tempo lineare (si consideri costante il tempo necessario a confrontare due etichette).

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

Problema 3 Miscellanea

1. Con riferimento al tipo astratto *coda di priorità*, 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)

[1] È il tradizionale concetto di *percorso assoluto* di un file o cartella. Per un nodo v raggiungibile da r attraverso il percorso $u_1, u_2, \ldots, u_h$, essendo $u_1 = r$, $h \geq 1$ e $u_h = v$, il percorso completo è la sequenza (stringa): $/e(u_1)/e(u_2)/\cdots/e(u_h)$, ove ogni notazione $e(u_i)$ va rimpiazzata dal valore effettivo dell'etichetta.

	insert	removeMin	min
array disordinato			
array ordinato			
heap			
AVL (puro)			

2. (2 punti) Disegnare un grafo diretto di 8 nodi che sia:

   - debolmente connesso;
   - il risultato di una chiusura transitiva.

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

Un grande progetto richiede la realizzazione di n moduli software che, benché autonomi in termini di sviluppo, presentano alcune interdipendenze da cui derivano vincoli di precedenza nello sviluppo: per iniziare lo sviluppo di qualche modulo, è necessario aver completato lo sviluppo e il test di qualche altro modulo. Gli analisti hanno perciò prodotto la descrizione di massima degli n moduli, denominati $M_1, M_2, \ldots, M_n$, oltre a un lungo elenco di coppie del tipo (i, j), con $(i, j) \in \{1, \ldots, n\}^2$, a rappresentare l'esistenza del vincolo: "lo sviluppo di M_j non può iniziare prima del termine dello sviluppo e test di M_i."

Gli analisti debbono ora individuare una sequenza temporale di sviluppo dei moduli che permetta di rispettare tutti i vincoli di precedenza espressi o, in altre parole, debbono determinare una permutazione $\rho = (i_1, \ldots, i_n)$ di $(1, \ldots, n)$ tale che, sviluppando i moduli secondo l'ordine stabilito da ρ, vengano rispettati tutti i vincoli di precedenza.

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.

(b) Definire un algoritmo (pseudo-codice) per la soluzione del problema: dati in input n moduli $M_1, \ldots, M_n$ ed m coppie (vincoli) del tipo $(h, k) \in \{1, \ldots, n\}^2$, determinare una permutazione $\rho = (i_1, \ldots, i_n)$ di $(1, \ldots, n)$, essendo $i_j \in \{1, \ldots, n\}$ per $j = 1, \ldots, n$, tale che per ciascuna coppia (h, k), risulti $h = i_p$ e $k = i_q$, con $1 \leq p < q \leq n$. L'algoritmo deve restituire ρ, oppure $\emptyset$ nel caso in cui una siffatta permutazione non esiste. Seguono esempi.

Input: $\{M_1, M_2, M_3, M_4\}$ e $\{(1, 3), (2, 3)\}$; possibile output: $(2, 4, 1, 3)$; altro possibile output (egualmente ammissibile): $(1, 2, 3, 4)$; esistono altri output ammissibili.

Input: $\{M_1, M_2, M_3, M_4\}$ e $\{(1, 3), (3, 2), (2, 1), (1, 4)\}$; output: $\emptyset$.

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.