02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II
Appello del 09/09/2016 - Prova di teoria (12 punti)

1. (1 punto)
Si ordini in maniera discendente il seguente vettore di interi mediante selection sort:

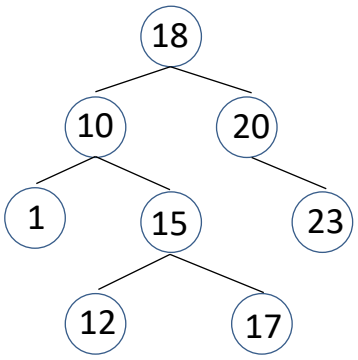
5 4 10 7 6 4 0 1 6 5 0 2 7 5 0 3 0 4 9

Si indichino i passaggi principali.

2. (1 punto)
Si esprima in notazione prefissa e postfissa la seguente espressione aritmetica mediante visita dell'albero binario corrispondente: ((A - B) / (C+(D*E))) * ((F-G) / (H*I))

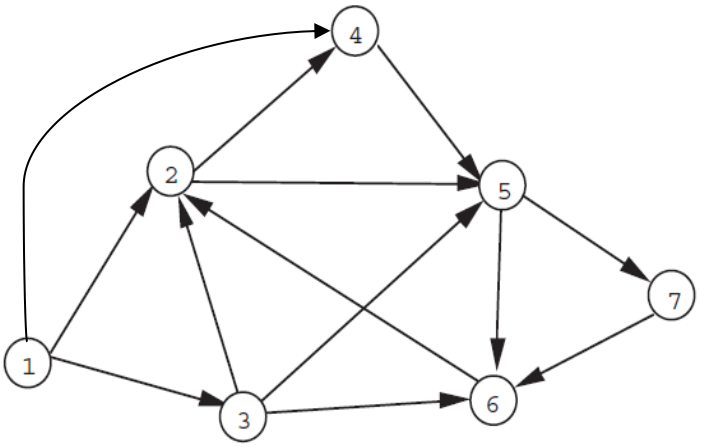
3. (2 punti)
Sia data la sequenza di chiavi RAWFZIEV dove ciascun carattere è individuato dal suo ordine progressivo nell'alfabeto (A=1, ..., Z=26). Si riporti la struttura di una tabella di hash di dimensione 17, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si supponga di utilizzare l'open addressing con double hashing. Si definiscano opportune funzioni di hash h_1 e h_2.

4 (1 punto)
Si inseriscano in foglia nel BST di figura in sequenza le chiavi: 11, 4, poi si cancelli la chiave 15, riportando a ogni passo l'albero risultante.



5 (0.5+0.5+0.5 punti)
Si visiti in pre-order, in-order e post-order l'albero binario dell'esercizio 4.

6 (1.5 + 1.5 + 2.5 punti)
Sia dato il seguente grafo orientato:



- lo si rappresenti come lista delle adiacenze e matrice delle adiacenze (**1.5 punti**)
- se ne effettui una visita in ampiezza, considerando **1** come vertice di partenza (**1.5 punti)**
- se ne determinino le componenti fortemente connesse mediante l'algoritmo di Kosaraju (**2.5 punti).**

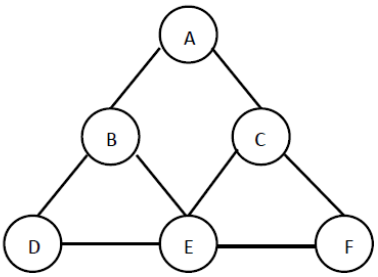
Qualora necessario, si trattino i vertici secondo l'ordine numerico.

02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 09/09/2016 - Prova di programmazione (18 punti)

Nella Teoria dei Grafi si definisce **Insieme Indipendente** o **Insieme Stabile** un insieme IS di vertici di un grafo G = (V, E) tale per cui, qualunque coppia di vertici $v_1 \in$ IS e $v_2 \in$ IS si consideri, non esiste un arco che li renda adiacenti $(v_1, v_2) \notin$ E.
Dato un grafo G, un **Insieme Indipendente Massimale** è un insieme indipendente che non è sottoinsieme di alcun altro Insieme Indipendente.

Esempio: dato il seguente grafo non orientato {A,D,E,F} non è un insieme indipendente, {D,F} è un insieme indipendente ma non massimale, {A,D,F} è un insieme indipendente massimale.



Un grafo non orientato G = (V, E) è memorizzato in un file mediante l'elenco dei suoi archi con il seguente formato:

$$idV_1 \ idV_2$$

che indica che (idV1, idV2) $\in$ E, dove idV1$\in$ V e idV2$\in$ V. Poiché il numero di vertici del grafo non è noto a priori, si suggerisce di calcolarlo tramite lettura preliminare del file e/o caricamento in una tabella di simboli. Ogni vertice è individuato mediante un identificatore alfanumerico di lunghezza massima uguale a 20 caratteri. Si può assumere che non esistano archi duplicati. Non è lecito assumere nessuna forma di ordinamento degli archi.

Si vuole scrivere un programma C in grado di:

- ricevere 3 nomi di file parametri sulla riga di comando:
  - il primo file contenente la descrizione del grafo
  - il secondo file contenente un elenco di vertici uno per riga
  - il terzo file su cui memorizzare il risultato
- leggere il grafo e memorizzarlo in un'opportuna struttura dati
- verificare se il grafo letto dal secondo file è davvero un insieme indipendente
- identificare un insieme indipendente massimale e memorizzarlo su un terzo file. A video si visualizzi la cardinalità dell'insieme così identificato, detta numero di indipendenza.

PER ENTRAMBE LE PROVE DI PROGRAMMAZIONE (18 o 12 punti):

- ***indicare nell'elaborato e nella relazione (oltre a nome, cognome e numero di matricola) anche il nome del corso per cui si sta sostenendo l'esame (AP, APA I+II).***
- ***Se non indicato diversamente, è consentito utilizzare chiamate a funzioni <u>standard</u>, quali ordinamento per vettori, funzioni su FIFO, LIFO, liste, BST, tabelle di hash, grafi e altre strutture dati, considerate come librerie esterne.***
- ***Gli header file devono essere allegati all'elaborato (il loro contenuto riportato nell'elaborato stesso). Le funzioni richiamate, inoltre, dovranno essere incluse nella versione del programma allegata alla relazione. I modelli delle funzioni ricorsive non sono considerati funzioni standard.***
- ***Consegna delle relazioni (per entrambe le tipologie di prova di programmazione) entro lunedì 12/09/2016 ore 24:00 mediante caricamento su Portale. Le istruzioni per il caricamento sono pubblicate sul Portale nella sezione Materiale). QUALORA IL CODICE CARICATO CON LA RELAZIONE NON COMPILI CORRETTAMENTE, VERRÀ APPLICATA UNA PENALIZZAZIONE. Si ricorda che la valutazione del compito viene fatta senza discussione o esame orale, sulla base dell'elaborato svolto in aula. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 09/09/2016 - Prova di programmazione (12 punti)

1. (2 punti)

Scrivere la funzione

```
int subMatMax (int **mat, int r, int c, int n);
```

che ricevuta una matrice di interi `mat` con `r` righe e `c` colonne, ricerchi la sotto-matrice quadrata di dimensione `n` la cui somma degli elementi è massima.

Esempio: data la matrice seguente

	0	1	2	3
0	5	2	3	1
1	3	1	6	4
2	3	0	5	2

con `r=3` e `c=4`, se `n=2` la sottomatrice con somma dei suoi elementi massima, pari a 17, è riportata nelle celle con sfondo grigio.

2. (4 punti)

Scrivere la funzione:

```
int distance (nnode_t *root, int key1, int key2);
```

che, ricevuto un BST di radice root (di chiave intera) e due valori interi `key1` e `key2`, restituisce il numero di archi che è necessario attraversare per raggiungere il nodo di chiave `key1` da quello di chiave `key2` (o viceversa).

Suggerimento: partendo dalla radice, la funzione `distance` proceda ricorsivamente lungo il BST fintanto che il percorso per le chiavi `key1` e `key2` è comune e dia origine a due visite separate quando il percorso di ricerca per le due chiavi si suddivide.

3. (6 punti)

Come regalo di compleanno u ragazzo vuole comprare alla sua ragazza *k* libri. In libreria sono disponibili *n* libri, ciascuno dei quali appartiene a 1 e 1 solo tra *m* generi letterari. Vale $k \leq m$. I *k* libri devono essere di generi diversi. Sia dato un vettore `int *vet` di `n` elementi che registra nella posizione *i* il genere del libro i-esimo ($1 \leq$ `vet[i]` $\leq m$). Si scriva la funzione:

```
int birthday (int *vet, int n, int m, int k);
```

che stampa tutte le possibili maniere diverse di scegliere *k* libri di generi diversi.

Esempio 1: se

`vet = (2 1 1 4 3) n=5` (libri numerati da 0 a 4) `m=4` (generi numerati da 1 a 4) `k=3`

le soluzioni sono:

`(0,1,3), (0,1,4), (0,2,3), (0,2,4), (0,3,4), (1,3,4), (2,3,4)`

Esempio 2: se

`vet = (1 2 3 1 2 3) n=6` (libri numerati da 0 a 5) `m=3` (generi numerati da 1 a 3) `k=2`

le soluzioni sono:

```
(0,1), (0,2), (0,4), (0,5), (1,2), (1,3),
(1,5), (2,3), (2,4), (3,4), (3,5), (4,5)
```