Cognome e Nome _____

Matricola _____

Algoritmi e strutture dati A.A. 2017/2018

Appello del 17 settembre 2018

Esercizio 1 (8 punti)

Si consideri il seguente algoritmo di ricerca binaria ricorsiva, da eseguire su un array ordinato in modo non decrescente:

```
function ternarySearchRic(int[] A, int p, int r, int v)
     if r-p+1 <= 2
        if A[p] == v
            return p
        else if A[r] == v
            return r
        else
            return -1
     q1=p+((r-p+1)/3)
     q2=p+2*((r-p+1)/3)
     if v>=A[p] and v<=A[q1]
        return ternarySearchRic (A, p, q1, v)
     else if v>A[q1] and v<=A[q2]
        return ternarySearchRic (A, q1+1, q2, v)
     else
        return ternarySearchRic (A, q2+1, r, v)
```

Scrivere la ricorrenza *T(n)* che indica il tempo di esecuzione dell'algoritmo nel caso peggiore. Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso di un metodo a scelta.

Esercizio 2 (8 punti)

Si scriva un algoritmo (in pseudocodice o in Java) che, presi in input

- un intero **M**,
- un array **estratti** di numeri interi (compresi tra 1 e **M**) di dimensione **k**,
- un array bidimensionale **giocati** di numeri interi (compresi tra 1 e **M**) con **n** righe e **k** colonne,

restituisca il numero di righe dell'array **giocati** che contengono gli stessi elementi dell'array estratti (non necessariamente nello stesso ordine).

Se ad esempio M=90 e k=6, tale problema trova quanti 6 sono stati totalizzati a Superenalotto.

Si analizzi la complessità computazionale dell'algoritmo proposto in funzione di M, n e k. Tale algoritmo non deve modificare gli array ricevuti in input e deve avere complessità temporale **O(M + nk)** nel caso peggiore (*algoritmi con complessità temporale peggiore danno luogo a una valutazione minore, pari a un massimo 4 punti*).

Suggerimento: utilizzare un array ausiliario di dimensione M per memorizzare la *funzione caratteristica* dell'insieme dei numeri estratti.

Esercizio 3 (9 punti)

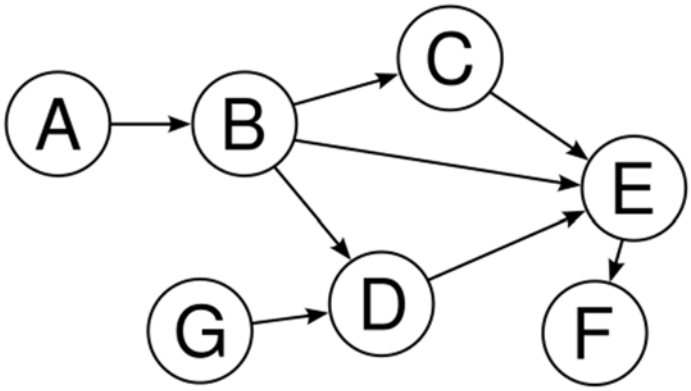
Si consideri la sequenza di chiavi intere

65, 94, 10, 140, 115, 24, 75, 35, 60, 125

a. Mostrare **passo-passo** l'heap di minimo che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato le chiavi sopra elencate
b. Mostrare la tabella hash di dimensione **13** con liste di trabocco che si ottiene a partire dalla tabella vuota inserendo nell'ordine indicato le chiavi sopra elencate
c. Mostrare la tabella hash di dimensione **13** con indirizzamento aperto (tramite hashing doppio) che si ottiene a partire dalla tabella vuota inserendo nell'ordine indicato le chiavi sopra elencate (si assuma che il secondo parametro della funzione dell'hashing doppio sia **12**).

Esercizio 4 (11 punti)

Si consideri il grafo non diretto aciclico (DAG) in figura.



- Mostrare la rappresentazione del grafo tramite liste di adiacenza
- Mostrare la rappresentazione del grafo tramite matrice di adiacenza
- Eseguire **passo-passo** la visita in profondità del grafo, indicando il tempo di inizio e fine visita per ogni nodo (si fa riferimento all'algoritmo DFS che prende in input un grafo e richiama al suo interno l'algoritmo DFS-VISIT su vari nodi sorgente).
- Sfruttando la visita del passo precedente, individuare un ordinamento topologico dei nodi del DAG.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** minuti.
- Scrivere **subito** nome, cognome, matricola e numero del compito su OGNI FOGLIO (**compreso questo**).
- Durante la prova scritta **non** è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- È possibile consultare appunti, libri, dispense o qualsiasi altro materiale.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco il proprio libretto (o altro documento di identità).