Cognome e Nome _____

Matricola _____

Algoritmi e strutture dati A.A. 2018/2019

Compito n° 1

Appello del 30 gennaio 2019

Esercizio 1 (4 punti)

Sia $T(n) = \begin{cases} 9T(n/2) + cn^3 & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

con *c* e *d* costanti.

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del **teorema generale**.

Esercizio 2 (10 punti)

(2 punti) Si scriva un metodo di ricerca binaria (iterativo o ricorsivo) in pseudocodice o in Java
`static int binarySearch (int[] a, int x)`
che, preso come parametro un array **a** di numeri interi (ordinato in modo non decrescente) ed un intero **x**, restituisce la posizione di un elemento dell'array a uguale a x, se esiste, e **-1** altrimenti.

(8 punti) Due array a e b della stessa lunghezza n (ognuno dei quali non contiene elementi ripetuti) si dicono ***equiposizionali*** se per ogni posizione i=0,…,n-1, vale che a[i] è il j-esimo elemento più piccolo in a se e solo se b[i] è il j-esimo elemento più piccolo in b. Ad esempio, a={1, 8, 3} e b={10, 13, 11} sono equiposizionali.
Scrivere un metodo (in pseudocodice o in Java)
`static boolean equiposizionali (int[] a, int[] b)`
che, presi come parametro due array **a** e **b** di numeri interi, restituisce *true* se e solo se a e b sono della stessa lunghezza e sono ***equiposizionali***. Se entrambi valgono *null*, viene restituito *true*. Se solo uno dei due vale *null*, viene restituito *false*. Il metodo **non** deve modificare gli array.
Si analizzi la complessità temporale del metodo proposto: tale metodo può richiamare qualsiasi algoritmo visto a lezione e deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore, pari a un massimo 4 punti su 8*), dove n è la lunghezza degli array a e b in input.

Esercizio 3 (7 punti)

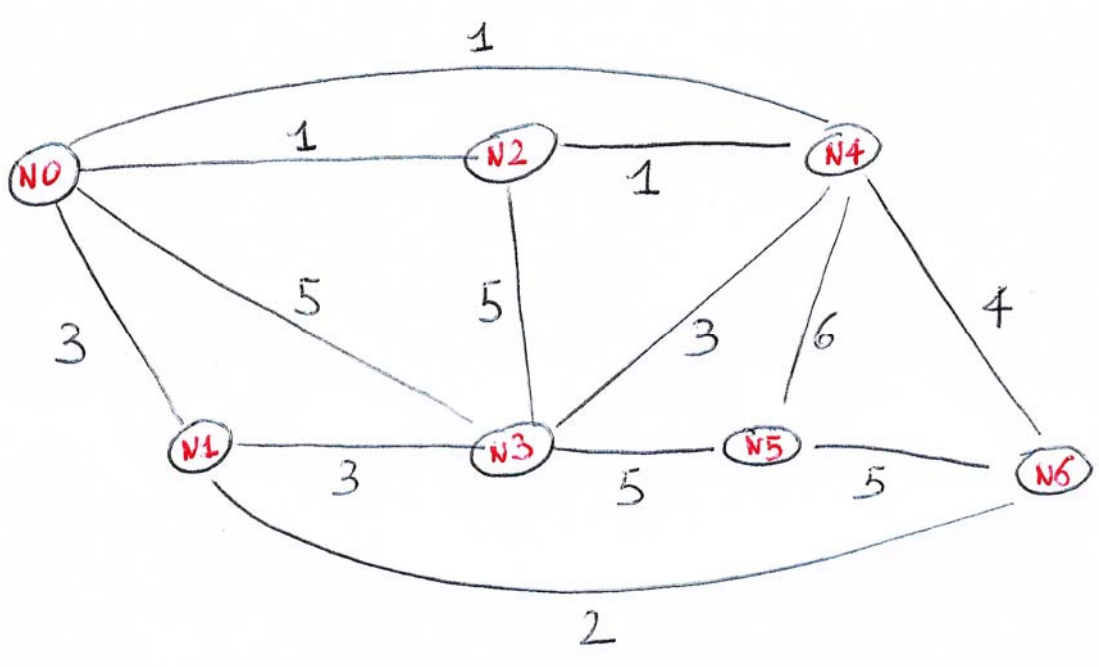
Si consideri il seguente array di numeri interi:

{5, 49, 10, 15, 11, 134, 75, 135, 6, 145}

Mostrare **passo-passo l'esecuzione dell'heap-sort** per ordinare l'array in modo non decrescente (compresa la fase iniziale di build-max-heap).

Esercizio 4 (14 punti)

Si consideri il grafo in figura.



a. Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Dijkstra** per l'albero dei cammini minimi a partire dal nodo sorgente **Nx**, dove x è ottenuto *prendendo l'ultima cifra del numero di matricola e calcolando il resto della divisione per 7.*
   Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo.
b. Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Floyd-Warshall** per i cammini minimi tra tutte le coppie, mostrando ad ogni passo la matrice delle distanze.

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).

Cognome e Nome _____

Matricola _____

Algoritmi e strutture dati A.A. 2018/2019

Compito n° 2

Appello del 30 gennaio 2019

Esercizio 1 (4 punti)

Sia $T(n) = \begin{cases} 9T(n/4) + cn^2 & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

con *c* e *d* costanti.

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del **teorema generale**.

Esercizio 2 (10 punti)

(2 punti) Si scriva un metodo di ricerca binaria (iterativo o ricorsivo) in pseudocodice o in Java
`static int binarySearch (int[] a, int x)`
che, preso come parametro un array **a** di numeri interi (ordinato in modo non decrescente) ed un intero **x**, restituisce la posizione di un elemento dell'array a uguale a x, se esiste, e **-1** altrimenti.

(8 punti) Due array a e b della stessa lunghezza n (ognuno dei quali non contiene elementi ripetuti) si dicono ***equiposizionali*** se per ogni posizione i=0,…,n-1, vale che a[i] è il j-esimo elemento più piccolo in a se e solo se b[i] è il j-esimo elemento più piccolo in b. Ad esempio, a={1, 8, 3} e b={10, 13, 11} sono equiposizionali.
Scrivere un metodo (in pseudocodice o in Java)
`static boolean equiposizionali (int[] a, int[] b)`
che, presi come parametro due array **a** e **b** di numeri interi, restituisce *true* se e solo se a e b sono della stessa lunghezza e sono ***equiposizionali***. Se entrambi valgono *null*, viene restituito *true*. Se solo uno dei due vale *null*, viene restituito *false*. Il metodo **non** deve modificare gli array.
Si analizzi la complessità temporale del metodo proposto: tale metodo può richiamare qualsiasi algoritmo visto a lezione e deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore, pari a un massimo 4 punti su 8*), dove n è la lunghezza degli array a e b in input.

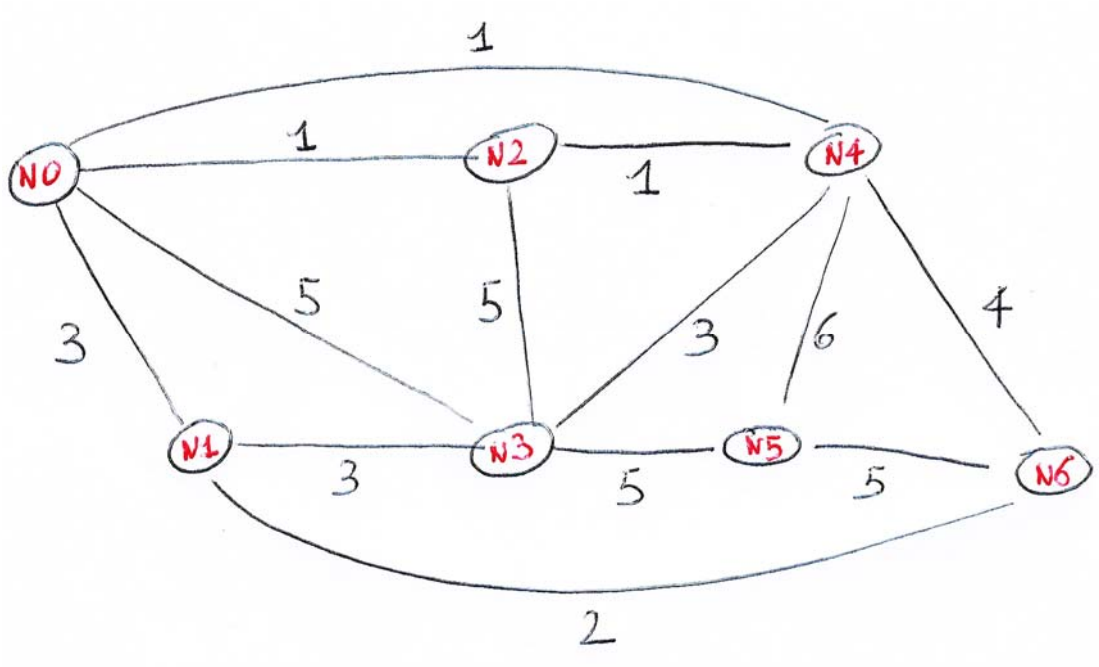
Esercizio 3 (7 punti)

Si consideri il seguente array di numeri interi:

{15, 49, 1, 15, 21, 13, 175, 35, 60, 14}

Mostrare **passo-passo l'esecuzione dell'heap-sort** per ordinare l'array in modo non decrescente (compresa la fase iniziale di build-max-heap).

Esercizio 4 (14 punti)
Si consideri il grafo in figura.



a. Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Dijkstra** per l'albero dei cammini minimi a partire dal nodo sorgente **Nx**, dove x è ottenuto *prendendo l'ultima cifra del numero di matricola e calcolando il resto della divisione per 7.*
   Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo.
b. Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Floyd-Warshall** per i cammini minimi tra tutte le coppie, mostrando ad ogni passo la matrice delle distanze.

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).