Domanda 2

Punteggio max.: 2,00

Contrassegna domanda

Sono dati due vettori di interi ordinati in modo crescente e privi di ripetizioni. Si scriva una funzione che generi un vettore (allocato dinamicamente) contenente gli interi appartenenti al primo vettore e non al secondo. La funzione deve essere chiamata come segue

c = diffVett(a,na,b,nb,&nc);

a e b sono i due vettori, na e nb il numero di dati che contengono; c è il vettore risultato, allocato dinamicamente nella funzione, nc il numero di interi nel vettore risultato.

Si richiede di realizzare (SOLO) la funzione diffVett (quindi non del programma chiamante).

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
//scrivere qui il codice...
```

Domanda 3

Punteggio max.: 4,00

Contrassegna domanda

E' dato un BST avente come valori delle stringhe, che fungono anche da chiave di ricerca. Si scriva una funzione che determini la foglia a profonditá massima MAXF (in caso di uguaglianza, si selezioni la foglia con chiave maggiore). La funzione stampi a ritroso (quindi da foglia a radice) le chiavi sul cammino che connette la foglia MAXF alla radice. Il prototipo della funzione deve essere:

void BSTprintDeepest(BST b);

Si richiede, oltre alla funzione, la definizione del tipo BST (ADT di prima classe) e del tipo usato per il nodo.

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
//scrivere qui il codice...
```

Domanda 4

Punteggio max.: 6,00

Contrassegna domanda

Una superficie piana rettangolare viene suddivisa in N righe e M colonne e rappresentata da una matrice NxM di caratteri, in cui si usa in carattere '0' per rappresentare una casella libera (utilizzabile) e il carattere '1' per una casella occupata (non utilizzabile).

Si vuole realizzare una funzione che, data la matrice e le coordinate di due caselle libere (di coordinate (r0,c0) e (r1,c1)), determini il percorso di lunghezza minima per connetterle: un percorso è dato da una sequenza di caselle libere adiacenti (aventi riga o colonna in comune) a due a due: detto in altri termini, un percorso si ottiene mediante tratti orizzontali o verticali comprendenti solo caselle libere. La lunghezza di un percorso è data dal numero di caselle che lo compongono. Del percorso ottimo va solo calcolata e ritornata come risultato la lunghezza.

La funzione abbia prototipo:

int minPath(char **area, int N, int M, int r0, int c0, int r1, int c1);

Esempio (piccolo): matrice 4x5 in cui si è rappresentato con delle il percorso ottimo (di lunghezza 5) che connette la casella (1,0) alla (2,3)

01000

xxx10

01xx0

00000

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
//scrivere qui il codice...
```

PARTE PROGRAMMAZIONE (18 PUNTI)

Informazione

Contrassegna domanda

Descrizione del problema

La Regione deve decidere la localizzazione di un servizio di Pronto Soccorso per N città. Tra queste occorre selezionare M < N città che possono essere sede di Pronto Soccorso. Obiettivo del problema è determinare le sedi di pronto soccorse ed assegnare ad ognuna di esse il servizio di altre città, dati vincoli e criteri per valutare costi e benefici.

Date

- le distanze tra tutte le coppie di città, come matrice NxN di interi (ogni città dista 0 da se stessa),

- una distanza massima tollerabile MAXD, tra sede di un pronto soccorso e una città servita,

- un numero minimo MINS di città servibili da ogni sede di pronto soccorso (siccome una sede di pronto soccorso serve ovviamente se stessa, MINS tiene conto solo delle altre città NON sede di pronto soccorso)

un insieme di M città sede di pronto soccorso è accettabile se sono rispettati i vincoli seguenti:

- per ognuna delle altre N-M città esiste almeno una sede di pronto soccorso (tra le M scelte) a distanza <= MAXD

- ognuna delle sedi di pronto soccorso è in grado di servire almeno MINS città non sedi di pronto soccorso (rispettando per ognuna il vincolo precedente).

Richieste del problema

A seguire una sintesi delle richieste del problema. Per ogni richiesta si troverà una domanda dedicata nelle sezioni a seguire con una descrizione più dettagliata per le richieste.

Strutture dati e letture

Definire opportune strutture dati per rappresentare:

- L'elenco delle città (tipo ELENCO)
- La matrice delle distanze (DISTMATR)
- Un insieme di sedi di pronto soccorso (tipo SEDI, vedere i punti successivi)
- Un'assegnazione di città alle sedi (tipo SERVIZI, la soluzione del problema di ottimizzazione)

In particolare, tutte le strutture dati precedenti devono essere ADT di prima classe. Si scriva la funzione caricaDati che ritorni un elenco di città (tipo ELENCO) e una matrice delle distanze (tipo DISTMATR), leggendo un file in cui la prima riga contiene il valore N, le N righe successive i nomi di N città, le ulteriori N righe successive la matrice quadrata delle distanze: ogni riga contiene N interi separati da spazi.

Esempio di file (per semplicitá si usano nomi di un solo carattere):

```
6
A
B
C
D
E
F
0 2 5 4 2 6
2 0 6 6 2 5
5 6 0 7 4 3
4 6 7 0 6 3
2 2 4 6 0 6
6 5 3 3 6 0
```

Problema di verifica

Si assuma di voler enumerare i possibili insiemi di M città sedi di pronto soccorso, mediante un algoritmo basato su combinazioni semplici. Si scriva la funzione checkSedi, in grado di verificare, nel caso terminale, l'accettabilità di una soluzione: NON si chiede l'algoritmo per generare le combinazioni ma solo la checkSedi. Questa deve ricevere, tra gli altri parametri (da decidere) la matrice delle distanze, la distanza massima MAXD, il numero minimo MINS, l'insieme ottenuto nel caso terminale (la soluzione, di tipo SEDI, da verificare).

Problema di ricerca e ottimizzazione

Dato un insieme valido di M sedi (tipo SEDI) di pronto soccorso, già verificato mediante checkSedi, si vuole determinare un partizionamento ottimo delle città servite, in modo tale che si rispettino i criteri seguenti:

- ogni città non sede di pronto soccorso viene assegnata a una (sola) sede di pronto soccorso, a distanza <= MAXD

- la distanza media tra le città servite e le sedi di pronto soccorso è minima

- a ognuna delle sedi di pronto soccorso si assegnano almeno MINS città.

Si scriva la funzione bestPart, che, utilizzando un algoritmo di partizionamento, dato l'elenco delle città, la matrice delle distanze, un insieme di M sedi di pronto soccorso e i valori MAXD e MINS, trovi la soluzione ottima e la ritorni come struttura dati (tipo SERVIZI). Non è necessario realizzare la funzione wrapper: è sufficiente rappresentare la funzione bestPart, che al suo interno deve usare due funzioni (da scrivere anche queste):

- checkPart: funzione di verifica da chiamare in un caso terminale

- prunePart: chiamata per fare pruning (si dica esplicitamente di quali vincoli o criteri si può fare pruning e di quali no).

Nell'esempio proposto in precedenza, supponendo che si siano selezionate come sedi di pronto soccorso {A,C}, che MAXD valga 4 e MINS valga 2, la soluzione sarebbe quella di assegnare {B,D} alla sede A e {E,F} alla sede C. Si riporta la matrice delle distanze, avendo evidenziato in rosso (sulle righe) le sedi e in grassetto le distanze compatibili con MAXD. Per la soluzione proposta, la distanza media tra le citta servite e le sedi è (2+4+4+3)/4 = 13/4.

	A	B	C	D	E	F
A	0	**2**	5	**4**	**2**	6
B	2	0	6	6	2	5
C	5	6	0	7	**4**	**3**
D	4	6	7	0	6	3
E	2	2	4	6	0	6
F	6	5	3	3	6	0

Domanda 5

Punteggio max.: 4,00

Contrassegna domanda

Strutture dati e acquisizione/lettura

Definire opportune strutture dati per rappresentare:

- L'elenco delle città (tipo ELENCO)
- La matrice delle distanze (DISTMATR)
- Un insieme di sedi di pronto soccorso (tipo SEDI, vedere i punti successivi)
- Un'assegnazione di città alle sedi (tipo SERVIZI, la soluzione del problema di ottimizzazione)

In particolare, tutte le strutture dati precedenti devono essere ADT di prima classe. Si scriva la funzione caricaDati che ritorni un elenco di città (tipo ELENCO) e una matrice delle distanze (tipo DISTMATR), leggendo un file in cui la prima riga contiene il valore N, le N righe successive i nomi di N città, le ulteriori N righe successive la matrice quadrata delle distanze: ogni riga contiene N interi separati da spazi.

Esempio di file (per semplicitá si usano nomi di un solo carattere):

```
6
A
B
C
D
E
F
0 2 5 4 2 6
2 0 6 6 2 5
5 6 0 7 4 3
4 6 7 0 6 3
2 2 4 6 0 6
6 5 3 3 6 0
```

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
//scrivere qui il codice...
```

Domanda 6

Punteggio max.: 6,00

Contrassegna domanda

Problema di verifica

Si assuma di voler enumerare i possibili insiemi di M città sedi di pronto soccorso, mediante un algoritmo basato su combinazioni semplici. Si scriva la funzione checkSedi, in grado di verificare, nel caso terminale, l'accettabilità di una soluzione: NON si chiede l'algoritmo per generare le combinazioni ma solo la checkSedi. Questa deve ricevere, tra gli altri parametri (da decidere) la matrice delle distanze, la distanza massima MAXD, il numero minimo MINS, l'insieme ottenuto nel caso terminale (la soluzione, di tipo SEDI, da verificare).

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
//scrivere qui il codice...
```

Domanda 7

Punteggio max.: 8,00

Contrassegna domanda

Problema di ricerca e ottimizzazione

Dato un insieme valido di M sedi (tipo SEDI) di pronto soccorso, già verificato mediante checkSedi, si vuole determinare un partizionamento ottimo delle città servite, in modo tale che si rispettino i criteri seguenti:

- ogni città non sede di pronto soccorso viene assegnata a una (sola) sede di pronto soccorso, a distanza <= MAXD

- la distanza media tra le città servite e le sedi di pronto soccorso è minima

- a ognuna delle sedi di pronto soccorso si assegnano almeno MINS città.

Si scriva la funzione bestPart, che, utilizzando un algoritmo di partizionamento, dato l'elenco delle città, la matrice delle distanze, un insieme di M sedi di pronto soccorso e i valori MAXD e MINS, trovi la soluzione ottima e la ritorni come struttura dati (tipo SERVIZI). Non è necessario realizzare la funzione wrapper: è sufficiente rappresentare la funzione bestPart, che al suo interno deve usare due funzioni (da scrivere anche queste):

- checkPart: funzione di verifica da chiamare in un caso terminale

- prunePart: chiamata per fare pruning (si dica esplicitamente di quali vincoli o criteri si può fare pruning e di quali no).

Nell'esempio proposto in precedenza, supponendo che si siano selezionate come sedi di pronto soccorso {A,C}, che MAXD valga 4 e MINS valga 2, la soluzione sarebbe quella di assegnare {B,D} alla sede A e {E,F} alla sede C. Si riporta la matrice delle distanze, avendo evidenziato in rosso (sulle righe) le sedi e in grassetto le distanze compatibili con MAXD. Per la soluzione proposta, la distanza media tra le citta servite e le sedi è (2+4+4+3)/4 = 13/4.

	A	B	C	D	E	F
A	0	**2**	5	**4**	**2**	6
B	2	0	6	6	2	5
C	5	6	0	7	**4**	**3**
D	4	6	7	0	6	3
E	2	2	4	6	0	6
F	6	5	3	3	6	0

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
//scrivere qui il codice...
```

PARTE TEORIA (12 PUNTI)

Domanda 1

Punteggio max.: 2,00

Contrassegna domanda

Sia data la sequenza di interi, supposta memorizzata in un vettore:

27 22 26 51 23 87 31 38 28 61 20 25 29 30

la si trasformi in un heap, ipotizzando di usare un vettore come struttura dati.
Si riportino graficamente i passi significativi della costruzione dell'heap ed il risultato finale.
Si ipotizzi che, alla fine, nella radice dell'heap sia memorizzato il valore minimo si eseguano su tale heap i primi 2 passi dell'algoritmo di heapsort.

NB: la sequenza è già memorizzata nel vettore e rappresenta una configurazione intermedia per cui la proprietà di heap non è ancora soddisfatta.

Formato della risposta:

su una sola riga elenco elementi separati da virgole o spazi:

el1, el2, el3, ... eln

Si elenchi il contenuto del vettore dell'heap al termine dell'algoritmo di trasformazione.

vettore al termine dell'algoritmo di trasformazione:

Si elenchi il contenuto del vettore dell'heap dopo l'esecuzione dei primi 2 passi di heapsort.

vettore dell'heap dopo il primo passo di heapsort:

vettore dell'heap dopo il secondo passo di heapsort:

Domanda 2

Punteggio max.: 1,50

Contrassegna domanda

Si supponga di aver memorizzato numeri tra 0 e 400 in un BST e di cercare la chiave 255.
Supponendo il BST corretto, è possibile incontrare le seguenti sequenze durante la ricerca?
Motivare brevemente la risposta.

- 2 252 309 220 330 264 255
- 2 252 209 220 240 250 255
- 2 400 202 300 270 250 255

La sequenza 2 252 309 220 330 264 255 può essere incontrata?

Sì/No

Se non può essere incontrata, indicare la prima violazione incontrata e spiegarla

La sequenza 2 252 209 220 240 250 255 può essere incontrata?

Sì/No

Se non può essere incontrata, indicare la prima violazione incontrata e spiegarla

La sequenza 2 400 202 300 270 250 255 può essere incontrata?

Sì/No

Se non può essere incontrata, indicare la prima violazione incontrata e spiegarla

Domanda 3

Punteggio max.: 2,00

Contrassegna domanda

Sia data la sequenza di chiavi intere 211, 26, 79, 46, 154, 17, 43, 230, 16. Si riporti il contenuto di una tabella di hash, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si usi l'open addressing con linear probing. Si determini la dimensione minima M della tabella di per avere un fattore di carico alpha < 0.4.

Formato della risposta:

dimensione minima della tabella

M =

Quale tra le seguenti funzioni di hash è opportuno usare?

$h_1(k) = k \% 17, h_2(k) = 1+k \% 17$

$h(k) = k \% 23$

$h(k) = k \% 19$

$h_1(k) = k \% 23, h_2(k) = 1+k \% 97$

Si motivi brevemente la risposta.

Collisioni chiave 16:

cella1 dove vi è collisione:

...

cellaN dove vi è collisione:

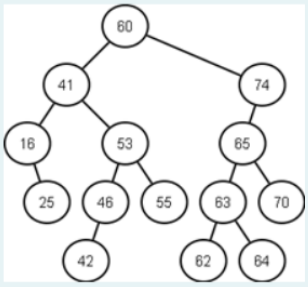
Al termine dell'inserzione, in quale cella si trova la chiave 230?

Domanda 4

Punteggio max.: 2,00

Contrassegna domanda

Si inserisca in **radice** nel BST di figura la chiave 45 e poi si cancelli la chiave 60.



Dopo l'inserimento di 45, chi sono i figli sinistro e destro della radice?

figlio SX=

figlio DX =

Dopo la cancellazione di 60, chi sono i figli sinistro e destro di 65?

figlio SX=

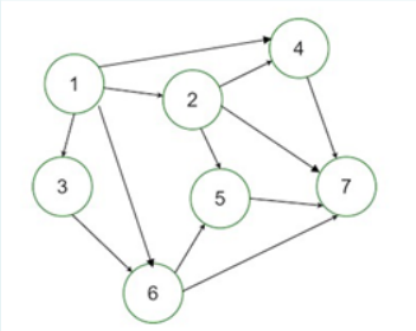
figlio DX =

Domanda 5

Punteggio max.: 2,00

Contrassegna domanda

Si ordini topologicamente il seguente DAG. Si consideri **1** come vertice di partenza. Qualora necessario, si trattino i vertici secondo l'ordine numerico e si assuma che la lista delle adiacenze sia anch'essa ordinata numericamente.



Formato della risposta:

per ogni vertice tempo di scoperta / tempo di fine elaborazione

Vertice 1:

Vertice 2:

Vertice 3:

Vertice 4:

Vertice 5:

Vertice 6:

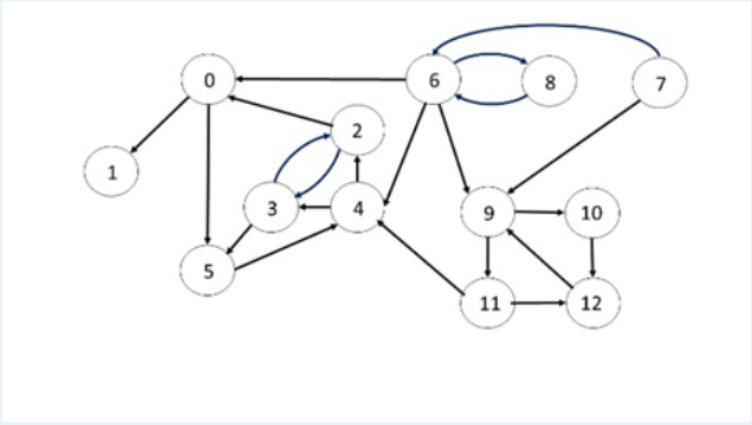
Vertice 7:

Domanda 6

Punteggio max.: 2,50

Contrassegna domanda

Dato il seguente grafo orientato:



Si applichi l'algoritmo di Kosaraju per il calcolo delle componenti fortemente connesse, considerando **0** come vertice di partenza. Qualora necessario, si trattino i vertici secondo l'ordine numerico e si assuma che la lista delle adiacenze sia anch'essa ordinata numericamente.

Formato della risposta:

Elencazione delle SCC come insiemi di vertici:

SCC1 =

SCC2 =

....

SCCn =