Corso: Algoritmi e strutture dati
Quiz: ASD2024 - IV Appello - Teoria

PORTALE ESAMI

	Giacomo Dandolo **s296525**
Iniziato	13 settembre 2024, 08:35
Stato	Completato
Terminato	13 settembre 2024, 09:35
Tempo impiegato	1 ora
Valutazione	Non ancora valutato
Riepilogo del tentativo	1 2 3 4 5 6

Domanda 1
Completo
Punteggio max.: 2,00

Si determini mediante un algoritmo greedy un codice di Huffman ottimo per i seguenti simboli con le frequenze specificate:

A: 5 B: 12 C: 13 D: 2 E: 15 F: 17 G: 9 H: 4 I: 7.

Il ramo sinistro abbia etichetta 0, quello destro etichetta 1.

Formato della risposta:

Quali simboli contiene l'aggregato con frequenza 23?

A D H B

Quali simboli contiene l'aggregato con frequenza 33?

I G F

Quale è la codifica di D?

10010

Quale è la codifica di E?

111

Quale è la decodifica di 10000100000110011?
AFIGH

Domanda 2
Completo
Punteggio max.: 2,50

Siano date 2 catene di $n=5$ stazioni di montaggio $S_{i,j}$ ciascuna (0 <= i <= 1, 0 <= j < 5) con i seguenti dati:

- tempi di lavorazione a nella catena i alla stazione j

0	7	9	3	4	3
1	8	5	6	4	4
	0	1	2	3	4

- tempi di trasferimento da una catena all'altra dalla stazione j-1 alla stazione j (nullo tra stazioni della stessa catena)

0	2	3	1	3
1	2	1	2	2
	0	1	2	3

- tempi di entrata nelle catene

2	4
0	1

- tempi di uscita dalle catene

2	2
0	1

· Mediante un algoritmo di programmazione dinamica si determini il costo minimo di lavorazione e la sequenza corrispondente di stazioni di montaggio.

Formato della risposta:

f[0][0] = 9
f[0][1] = 16
f[0][2] = 20
f[0][3] = 24
f[0][4] = 27
f[1][0] = 12
f[1][1] = 17
f[1][2] = 21
f[1][3] = 25
f[1][4] = 28

l[0][0] =
l[0][1] = 1
l[0][2] = 0
l[0][3] = 0
l[0][4] = 0
l[0][5] = 0
l[1][0] =
l[1][1] = 1
l[1][2] = 0
l[1][3] = 0
l[1][4] = 0
l[1][5] = 0

costo minimo: 27

catena di uscita: 0

sequenza di stazioni percorse: 0 1 0 0 0

Domanda 3

Completo

Punteggio max.: 2,00

Sia dato un albero binario con 10 nodi. Nella visita in preorder si ottiene preorder = (A, B, D, E, F, G, C, H, I, J) e in quella inorder si ottiene inorder = (B, E, D, G, F, A, C, H, J, I). Si ricostruisca l'albero binario di partenza, verificando che la visita postorder dia come risultato postorder = (E, G, F, D, B, J, I, H, C, A).

Formato della risposta:

figlio SX di A: B
figlio DX di A: C

figlio SX di B: non esiste
figlio DX di B: D

figlio SX di C: non esiste
figlio DX di C: H

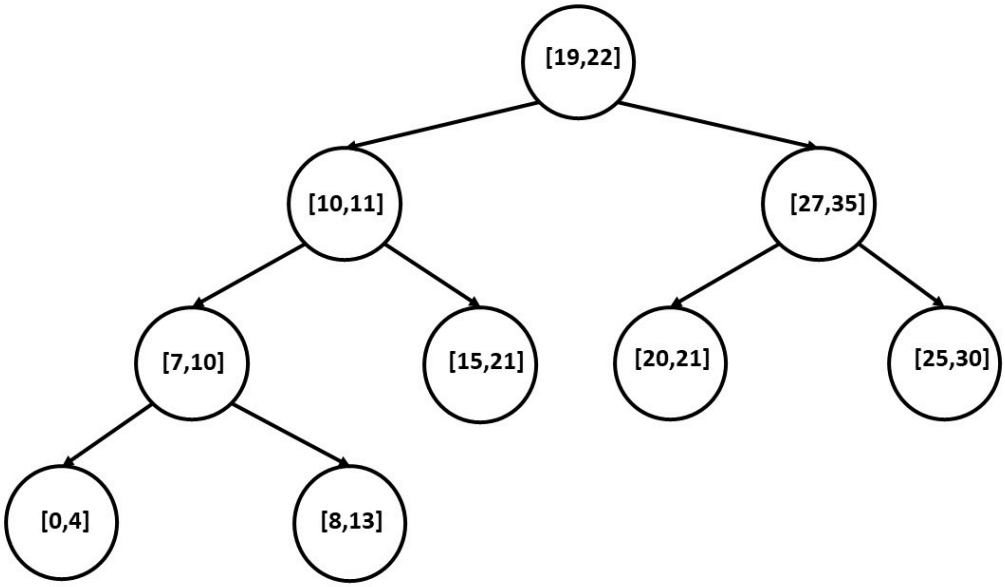
figlio SX di I: J
figlio DX di I: non esiste

Domanda 4

Completo

Punteggio max.: 2,00

Nella figura seguente è riportato un IBST incompleto. Completarlo aggiungendo in ogni nodo le informazioni necessarie.



Formato della risposta:

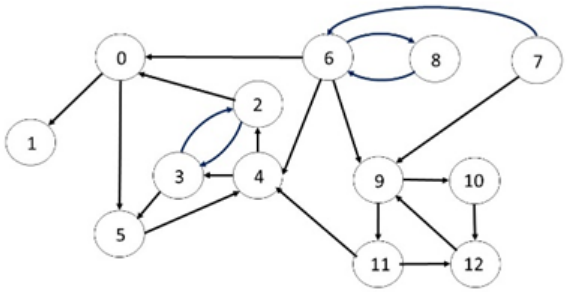
informazione extra nodo: tempo di fine dell'intervallo massimo nel sottoalbero definito da un nodo, con nodo compreso

[19,22]: 35
[10,11]: 21
[27,35]: 35
[7,10]: 13
[15,21]: 21
[20,21]: 21
[25,30]: 35
[0,4]: 4
[8,13]: 13

Domanda 5
Completo

Sia dato il seguente grafo orientato:

Punteggio max.: 2,00



se ne effettui una visita in profondità, considerando **0** come vertice di partenza etichettando i vertici con tempo di scoperta/tempo di fine elaborazione. Qualora necessario, si trattino i vertici secondo l'ordine numerico.

Formato della risposta:

per ogni vertice tempo di scoperta / tempo di fine elaborazione

Vertice 0: 0/11

Vertice 1: 1/2

Vertice 2: 5/8

Vertice 3: 6/7

Vertice 4: 4/9

Vertice 5: 3/10

Vertice 6: 12/23

Vertice 7: 24/25

Vertice 8: 13/14

Vertice 9: 15/22

Vertice 10: 16/19

Vertice 11: 20/21

Vertice 12: 17/18

Domanda 6

Completo

Punteggio max.: 1,50

In base alla visita in profondità dell'esercizio precedente, si etichettino gli archi del grafo con le etichette T, B, F e C.

Formato della risposta:

per ogni arco etichetta T, B, F o C

0-1: T
0-5: T
2-0: B
2-3: T
3-2: B
3-5: B
4-2: T
4-3: F
5-4: T
6-0: C
6-4: C
6-8: T
6-9: T
7-6: C
7-9: C
8-6: B
9-10: T
9-11: T
10-12: T
11-4: C
11-12: C
12-9: B

Corso: Algoritmi e strutture dati
Quiz: ASD2024 - IV Appello - Programmazione

PORTALE ESAMI

	Giacomo Dandolo **s296525**
Iniziato	13 settembre 2024, 10:01
Stato	Completato
Terminato	13 settembre 2024, 11:41
Tempo impiegato	1 ora 40 min.
Valutazione	Non ancora valutato
Riepilogo del tentativo	i i 1 2 3 4 i i 5 6 7

Informazione

Attenzione!

Indicare quale tipologia di esame si intenda svolgere rispondendo alla domanda a scelta multipla seguente (domanda 1).
Si noti che è possibile leggere tutte o parte delle domande prima di effettuare la scelta e rispondere alla domanda 1.
Le domande dalla 2 alla 6 sono dedicate all'esame semplificato (traccia da 12pt).

In particolare:

- la domanda 2 fa parte dell'esercizio da 2 punti.
- la domanda 3 fa parte dell'esercizio da 4 punti.
- le domande 4,5,6 fanno parte dell'esercizio da 6 punti.

Le domande dalla 7 in poi sono dedicate all'esame completo (traccia da 18pt).

Un apposito separatore fa da intervallo tra le domande del compito da 12pt e le domande del compito da 18pt.

Informazione

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

- se non indicato diversamente, è consentito utilizzare chiamate a funzioni standard, 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 riferiti dal codice dovranno essere inclusi 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 LUNEDI 13/09/2024, alle ore 23:59, mediante caricamento su Portale
- assicurarsi di caricare l'elaborato nella **Sezione Elaborati relativa all'a.a. *2023/24***.
- 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 la presenza del candidato, sulla base dell'elaborato svolto durante l'appello. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.

Domanda 1
Completo
Non valutata

Indicare quale tipologia di esame si intende svolgere.

Scegli un'alternativa:

a. 12 Punti - Semplificato

b. 18 Punti - Completo

Risposta corretta.

Ignorare il feedback di questa domanda.

La risposta corretta è: 18 Punti - Completo

Domanda 2

Risposta non data

Punteggio max.: 2,00

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).

Domanda 3

Risposta non data

Punteggio max.: 4,00

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.

Domanda 4

Risposta non data

Punteggio max.: 6,00

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
```

Informazione

PAGINA DI INTERMEZZO

La prova da 12 punti termina prima di questa pagina di intermezzo.

La prossima domanda rappresenta l'inizio della prova da 18 punti.

Informazione

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

Completo

Punteggio max.: 5,00

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...
```

```
// ELENCO.h
```

```
typedef struct elenco_t *ELENCO;
```

```
ELENCO ELENCOinit(int maxN);
```

```
ELENCO ELENCOfile(FILE* file); // ottiene l'elenco delle città da file

int ELENCOgetN(ELENCO e); // ottieni numero N di città


// ELENCO.c
// include ELENCO.h

typedef struct elenco_t {

  char citta[30];

  int maxN;

} *ELENCO;

}

// implementazione funzioni in ELENCO.h


// DISTMATR.h

typedef struct distmatr_t DISTMATR;

DISTMATR DISTMATRinit(int N);

DISTMATR DISTMATRfile(FILE* file, DISTMATR e); // ottiene da file la matrice delle distanze

int DISTMATRgetVal(DISTMATR d, int i, int j); // ottiene da una DISTMATR il valore nella cella (i,j)


// DISTMATR.c

// include DISTMATR.h

typedef struct distmatr_t {

  int **m;

  int N;

} *DISTMATR;


// SEDI.h

// include ELENCO.h, DISTMATR.h

typedef struct sedi_t SEDI;

SEDI SEDIinit(int maxN);

int SEDIgetN(SEDI s); // ottiene il numero di città

int SEDIgetNumS(SEDI s); // ottiene il numero di sedi

DISTMATR SEDIgetD(SEDI s); // ottiene la matrice delle distanze

int SEDIisSede(SEDI s, int i); // controlla se è una sede l'elemento in indice i
```

```
// SEDI.c

// include SEDE.h

typedef struct sedi_t {

 int *s; // usata per tenere conto delle sedi: 0 = non sede, 1 = sede

  int numS; // tiene conto del numero delle sedi, inizialmente 0

  DISTMATR d; // usata per tenere conto delle distanze i-j

 int N; // tiene conto del numero di città

} *SEDI;

// implementazione funzioni in SEDE.h


// SERVIZI.h

typedef struct servizi_t SERVIZI;

SERVIZIinit(int N, float media);

SERVIZIcount(SERVIZI s, int i); // conta il numero di città a cui arri

// SERVIZI.c

typedef struct servizi_t {

  int *reach; // definisce quali città sono raggiunte da quale sede

  int maxN;

  int N; // tiene conto del numero di sedi usate

  float media;

} *SERVIZI;


// main.c

// include ELENCO.h, DISTMATR.h, SEDI.h, SERVIZI.h

void caricaDati(FILE* file, ELENCO *e, DISTMATR *d) {

 *e = ELENCOfile(file);

 *d = DISTMATRinit(ELENCOgetN(*e));

 *d = DISTMATRfile(file, *d);

}
```

Domanda 6

Completo

Punteggio max.: 5,00

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...
```

```
// main.c

int checkSedi(DISTMATR d, int MAXD, int MINS, SEDI s) {
 int i, j, flag, n;

 // check per condizione numero minimo di città disponibili affinché abbia senso MINS

 if ((SEDIgetN(s) - SEDIgetNumS(s)) < MINS)

   return 0;

 // check per raggiungimento sedi

 for (i = 0; i < SEDIgetN(s); i++)

   if (SEDIisSede(s, i) == 0) {

     flag = 0;

     for (j = 0; j < SEDIgetN(s) && flag == 0; j++)

       if (i != j && SEDIisSede(s, j) == 1 && DISTMATRgetVal(SEDIgetD(s), i, j) <= MAXD) {

         flag = 1;

         break;

       }

     if (flag == 0)

       return 0;

   }

 // check per numero minimo di città per sedi

 for (i = 0; i < SEDIgetN(s); i++)

   if (SEDIisSede(s, i) == 1) {

     n = 0;

     for (j = 0; j < SEDIgetN(s); j++)

       if (i != j && SEDIisSede(s, j) == 0 && DISTMATRgetVal(SEDIgetD(s), i, j) <= MAXD)

         n++;
```

```
    if (n < MINS)
```

```
      return 0;
```

```
   }
```

```
return 1;
```

```
}
```

Domanda 7

Completo

Punteggio max.: 8,00

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...
```

```
```

```
// SERVIZI.c

int checkPart(SERVIZI s, DISTMATR d, SEDI *M, int N, int MAXD, int MINS) {

 int i, j, k, *check, curr;

 check = (int*) calloc(SERVIZIgetN(s), sizeof(int));

 for (i = 0; i < SERVIZIgetN(s); i++)

  for (j = 0; j < DISTMATRgetN(d) && check[j] != 1; j++);

  curr = SERVIZIgetUsedVal(s, j);

  for (k = 0; k < DISTMATRgetN(d); k++) {

   if (curr == SERVIZIgetUsedVal(s, k) && DISTMATRgetVal(curr, k) <= MAXD)

    check[k] = 1;

  }

  if (SERVIZIcount(s, curr) < MINS)

   return 0;

 return 1;

}

SERVIZI bestPart(SERVIZI *s, ELENCO e, DISTMATR d, SEDI *M, int MAXD, int MINS) {

}
```