A

03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
Appello del 27/01/2022 - Prova di teoria (12 punti)

1. (2.5 punti)
Si risolva la seguente equazione alle ricorrenze mediante il metodo dello sviluppo (unfolding):

$$T(n) = 2T(n/3) + 5n \qquad n > 1$$
$$T(1) = 1 \qquad n = 1$$

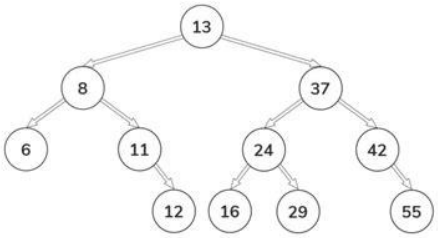
2. (2 punti)
Sia data la sequenza di interi, supposta memorizzata in un vettore:

221, 16, 89, 56, 144, 27, 33, 91, 132, 37, 49, 53

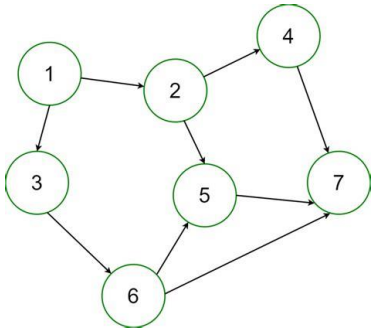
si eseguano i primi 2 passi dell'algoritmo di quicksort per ottenere un ordinamento ascendente, indicando ogni volta il pivot scelto. NB: i passi sono da intendersi, impropriamente, come in ampiezza sull'albero della ricorsione, non in profondità. Si chiede, pertanto, che siano ritornate le 2 partizioni del vettore originale e le due partizioni delle partizioni trovate al punto precedente.

3. (2 punti)
Sia data la sequenza di chiavi intere 221, 16, 89, 56, 144, 27, 33, 259. Si riporti il contenuto di una tabella di hash di dimensione 17, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si usi l'open addressing con linear probing.

4. (2 punti) inserimento in radice BST
Si inseriscano in **radice** nel BST di figura in sequenza le chiavi 7 e 10 poi si cancelli la chiave 13. Si disegni l'albero ai passi significativi.



5. (2 + 1.5 punti)
Sia dato il seguente grafo orientato:



se ne effettui una visita in profondità, considerando **1** come vertice di partenza etichettando i vertici con tempo di scoperta/tempo di fine elaborazione (**2 punti**) e gli archi con T, B, F, C (**1.5 punti**). Qualora necessario, si trattino i vertici secondo l'ordine numerico.

B

03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
Appello del 27/01/2022 - Prova di teoria (12 punti)

1. (2.5 punti)
Si risolva la seguente equazione alle ricorrenze mediante il metodo dello sviluppo (unfolding):

$$T(n) = 2T(n/3) + 6n \qquad n > 1$$
$$T(1) = 1 \qquad n = 1$$

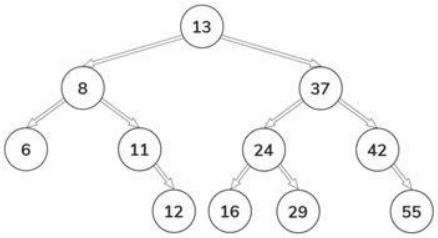
2. (2 punti)
Sia data la sequenza di interi, supposta memorizzata in un vettore:

220, 15, 88, 55, 143, 26, 32, 90, 131, 36, 48, 52

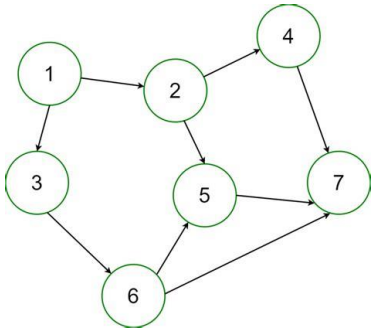
si eseguano i primi 2 passi dell'algoritmo di quicksort per ottenere un ordinamento ascendente, indicando ogni volta il pivot scelto. NB: i passi sono da intendersi, impropriamente, come in ampiezza sull'albero della ricorsione, non in profondità. Si chiede, pertanto, che siano ritornate le 2 partizioni del vettore originale e le due partizioni delle partizioni trovate al punto precedente.

3. (2 punti)
Sia data la sequenza di chiavi intere 221, 16, 89, 56, 144, 27, 33, 259. Si riporti il contenuto di una tabella di hash di dimensione 17, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si usi l'open addressing con linear probing.

4. (2 punti) inserimento in radice BST
Si inseriscano in **radice** nel BST di figura in sequenza le chiavi 7 e 10 poi si cancelli la chiave 13. Si disegni l'albero ai passi significativi.



5. (2 + 1.5 punti)
Sia dato il seguente grafo orientato:



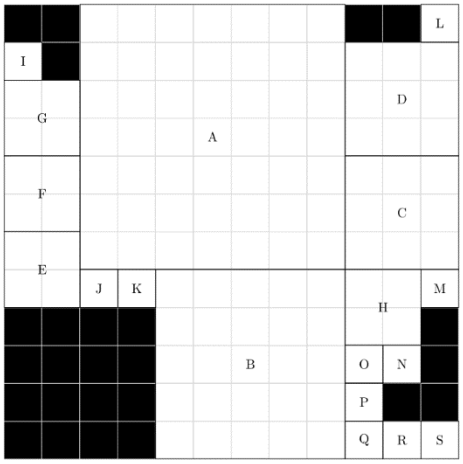
se ne effettui una visita in profondità, considerando **2** come vertice di partenza etichettando i vertici con tempo di scoperta/tempo di fine elaborazione (**2 punti**) e gli archi con T, B, F, C (**1.5 punti**). Qualora necessario, si trattino i vertici secondo l'ordine numerico.

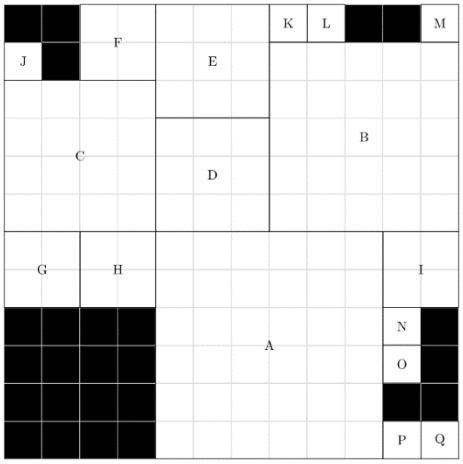
03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
Appello del 27/01/2022 - Prova di programmazione (18 punti)

1. (**18 punti**)
Una matrice rettangolare `R x C` rappresenta una griglia di gioco.
La griglia contiene celle bianche e celle nere. Le celle bianche definiscono una regione contigua nella griglia. L'obiettivo del gioco è suddividere la regione bianca nel numero minimo di sottoregioni quadrate prive di sovrapposizioni.
Si scriva un programma in C che:

- legga un primo file di testo `griglia.txt` organizzato come segue:
  - la prima riga contiene una coppia di interi `NR NC`, ossia le dimensioni della griglia
  - seguono `NR` righe di `NC` valori interi separati da spazi, ciascuno a rappresentare una cella bianca (0) o nera (1)
- legga un secondo file di testo `proposta.txt`, il cui formato è a discrezione del candidato, a rappresentare una possibile suddivisione della griglia in sottoregioni bianche e determini se sia una copertura valida (condizioni di validità: *tutte le sottoregioni bianche sono quadrate && tutta la regione bianca è coperta*) e in caso affermativo conteggi il numero di regioni usate
- si identifichi una copertura della regione bianca che fa uso del minor numero possibile di sottoregioni bianche quadrate.

Esempio.
Data la seguente griglia di gioco `12x12`, sono presentate due possibili soluzioni. La prima, a sinistra, suddivide completamente la regione bianca in 19 sottoregioni. Le regioni sono etichettate con le lettere dell'alfabeto a partire da quelle di dimensioni maggiori (regione A, 7x7) e via via decrescenti, a mero scopo illustrativo. La soluzione non è ottima. Esiste una soluzione, rappresentata nell'immagine di destra, che fa uso di 17 regioni.





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

- ***indicare nell'elaborato e nella relazione nome, cognome e numero di matricola.***
- ***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 il 30/01/2022, alle ore 12:00, 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 la presenza del candidato, <u>sulla base dell'elaborato svolto in aula</u>. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
Appello del 27/01/2022 - Prova di programmazione (12 punti)

1. (**2 punti**)
Sia data una matrice `M` di dimensione `r x c` contenente elementi interi positivi o nulli.
Si scriva una funzione `f` che generi una matrice `M'` di dimensione `r' x c'` derivata da `M` mantenendo solo le righe/colonne non contenenti valori nulli.
Si completi opportunamente il prototipo seguente in modo che la nuova matrice e le relative dimensioni siano disponibili al chiamante:

```
void f(int **M, int r, int c, ...);
```

Esempio. Siano `r` = 3, `c` = 3

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 0 & 9 \\ 7 & 8 & 9 \end{pmatrix} \rightarrow M' = \begin{pmatrix} 1 & 3 \\ 7 & 9 \end{pmatrix}$$

2. (**4 punti**)
Si scriva una funzione (wrapper) `int f(BT t)` che ricevuto in input un albero binario di interi `t` di tipo `BT` verifichi se questo rappresenti o meno un BST.
Fornire, inoltre, la definizione del tipo `BT` e del tipo nodo al suo intero, come ADT di prima classe e come quasi ADT rispettivamente.
Non è ammesso l'utilizzo di funzioni di libreria.

3. **(6 punti**)
Una matrice `M` quadrata di dimensione `N x N` rappresenta le relazioni di amicizia tra `N` persone. Ogni cella contenente il valore 1 indica che la coppia *(i,j)* è una coppia di amici. In caso contrario il valore memorizzato è 0. La relazione di amicizia è simmetrica (se *i* considera *j* suo amico, vale anche l'opposto).

- Si scriva una funzione ricorsiva in grado di suddividere le persone nel minor numero di gruppi possibili facendo in modo che tutte le persone in un gruppo siano mutualmente amiche. È ammesso che ci siano gruppi composti da una singola persona
- Si indichi esplicitamente il modello combinatorio utilizzato giustificandone la scelta
- Si descrivano e si giustifichino i criteri di pruning utilizzati o la loro eventuale assenza.

Esempio
Sia `N` = 4 e data la matrice M sottostante che rappresenta le relazioni di amicizia, il numero minimo di gruppi è 2, il primo di tre persone mutualmente amiche e il secondo di una singola persona che non ha relazioni di amicizia con nessun'altra persona.

$$M = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \rightarrow G = \{\{p_0, p_1, p_3\}, \{p_2\}\}$$

03AAX Algoritmi e Strutture Dati / 03MNO Algoritmi e Programmazione

Appello del 09/02/2022 - Prova di teoria (12 punti)

1. (2 punti)

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

9 2 4 33 3 69 13 19 7 41 0 5 6 8

- 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 massimo
- 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.

2. (2.5 punti)

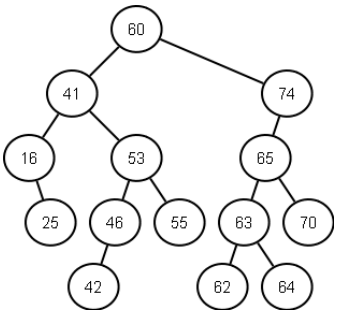
Data la catena di matrici (A_1, A_2, A_3, A_4) di dimensioni (4x6), (6x2), (2x8) e (8x3) rispettivamente, si determini mediante un algoritmo di programmazione dinamica la parentesizzazione ottima del prodotto di matrici che minimizza il numero di moltiplicazioni.

3. (1 punto)

Si converta la seguente espressione da forma prefissa a forma infissa: / * - A B / C D / E * - F G + H I

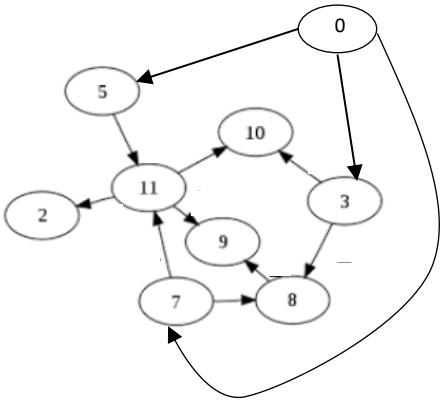
4. (2 punti)

Si partizioni il seguente BST attorno alla sesta chiave più piccola. In ogni nodo compare la chiave intera, altre eventuali informazioni non sono riportate.



5. (2.5 punti)

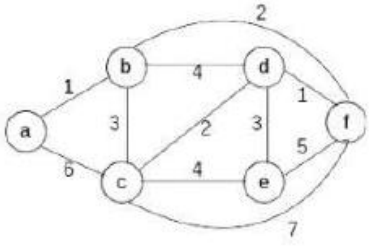
Dato il seguente grafo orientato:



si applichi l'algoritmo di Kosaraju per il calcolo delle componenti fortemente connesse, considerando **11** come vertice di partenza e, qualora necessario, trattando i vertici secondo l'ordine numerico.

6. (2 punti)

Dato il seguente grafo non orientato, connesso e pesato, se ne determini un minimum spanning tree applicando l'algoritmo di Prim a partire dal vertice **a**, disegnando l'albero e ritornando come risultato il valore del peso minimo. Si esplicitino i passi intermedi.

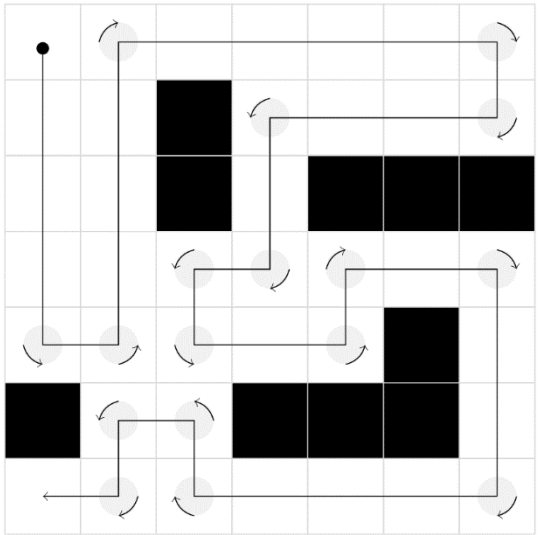


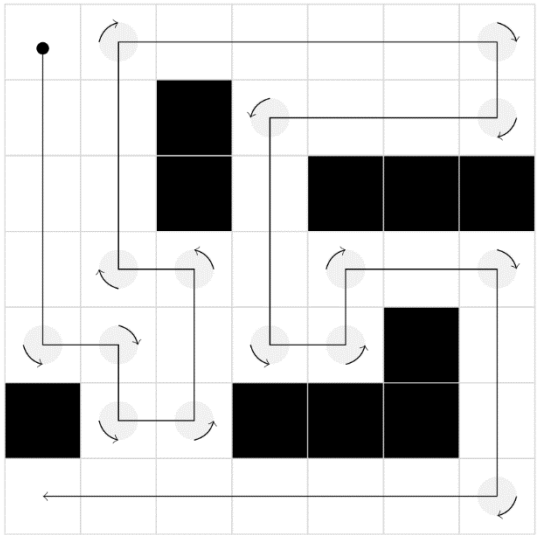
03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
Appello del 09/02/2022 - Prova di programmazione (18 punti)

1. **(18 punti)**
Una matrice rettangolare `R x C` rappresenta una griglia di gioco.
La griglia contiene celle bianche e celle nere. Le celle bianche sono celle su cui è possibile transitare mentre le celle nere sono inaccessibili e rappresentano degli ostacoli sulla griglia di gioco.
La cella in posizione (0,0) è sempre bianca ed è sempre il punto di partenza del cammino.
L'obiettivo del gioco è visitare tutte le celle bianche della griglia, se possibile, percorrendo un cammino semplice che esegua il minor numero possibile di cambi di direzione. La cella finale di destinazione non è definita a priori. Un cammino, purché corretto, può finire ovunque. Il movimento è possibile solo lunghe le quattro direzioni principali (nord, sud, ovest, est): non sono ammessi spostamenti in diagonale.
Si scriva un programma in C che:

- legga un primo file di testo `griglia.txt` organizzato come segue:
  - la prima riga contiene una coppia di interi `NR NC`, ossia le dimensioni della griglia
  - seguono `NR` righe di `NC` valori interi separati da spazi, ciascuno a rappresentare una cella bianca (0) o nera (1)
- legga un secondo file di testo `proposta.txt`, il cui formato è a discrezione del candidato, a rappresentare una possibile sequenza di mosse sulla griglia e determini se sia una soluzione valida secondo le regole di cui sopra. In caso affermativo conteggi il numero di cambi di direzione
- trovi una visita della griglia rispettando i vincoli di cui sopra effettuando il minor numero possibile di cambi di direzione

Esempio.
Le figure seguenti rappresentano due possibili soluzioni per la stessa griglia di gioco `7 x 7`. L'immagine a sinistra propone un cammino semplice che transita per tutte le celle bianche eseguendo 17 cambi di direzione. L'immagine a destra presenta un secondo cammino valido che usa il minimo numero di cambi di direzione: 15.





03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
Appello del 09/02/2022 - Prova di programmazione (12 punti)

1. (**2 punti**)
Sia data una matrice `M` di dimensione `r x c` contenente elementi interi positivi o nulli.
Scrivere una funzione `f` che generi una matrice `M'` di dimensione `r' x c'` derivata da `M` mantenendo solo le righe/colonne dove entrambi gli indici sono pari. Per convenzione gli indici partono da zero (pari).
Completare opportunamente il prototipo in modo che la nuova matrice e le relative dimensioni siano disponibili al chiamante.

```
void f(int **M, int r, int c, ...);
```

Esempio. Siano `r` = 3, `c` = 4

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 0 & 1 & 2 \end{pmatrix} \rightarrow M' = \begin{pmatrix} 1 & 3 \\ 9 & 1 \end{pmatrix}$$

2. (**4 punti**)
Si scriva una funzione (wrapper) `char *decode(H h, char *str)` che, ricevuto in input un albero binario `h`, rappresentante la codifica di Huffman associata a un certo set di caratteri, e una stringa di caratteri `str` che contiene una sequenza di 0/1, la decodifichi, sulla base della codifica memorizzata nell'albero `h`, ritornando come risultato una stringa decodificata.
Fornire inoltre la definizione del tipo `H` (come ADT di prima classe) e del tipo nodo al suo interno (come quasi ADT).
Non è ammesso l'utilizzo di funzioni di libreria.

3. **(6 punti**)
Una matrice `M` quadrata di dimensione `N x N` rappresenta le relazioni di amicizia tra `N` persone. Ogni cella contenente il valore 1 indica che la coppia *(i,j)* è una coppia di amici. In caso contrario il valore memorizzato è 0. La relazione di amicizia è simmetrica (se *i* considera *j* suo amico, vale anche l'opposto).

- Scrivere una funzione ricorsiva in grado di individuare il più grande gruppo di persone tale per cui ogni persona è amica con almeno altre k persone (diverse dalla persona stessa) del gruppo.
- Indicare esplicitamente il modello combinatorio utilizzato e giustificarne la scelta.
- Descrivere e giustificare i criteri di pruning utilizzati o la loro eventuale assenza.

Esempio. Sia `N` = 4, k=2

$$M = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \rightarrow g = \{p_0, p_1, p_3\}$$

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

- ***indicare nell'elaborato e nella relazione nome, cognome e numero di matricola.***
- ***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 il 12/02/2022, alle ore 12:00, 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 la presenza del candidato, <u>sulla base dell'elaborato svolto in aula</u>. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
III Appello del 27/06/2022 - Prova di teoria (12 punti)

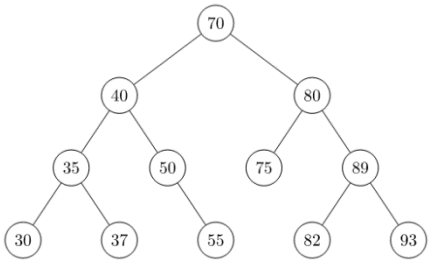
1. (2 punti)
Si determini mediante un algoritmo greedy un codice di Huffman ottimo per i seguenti simboli con le frequenze specificate: B: 5 C: 11 D: 13 F: 12 J: 13 L: 16 R: 8 X: 4 Z: 7. Il ramo sinistro abbia etichetta 0, quello destro etichetta 1. Si elenchi il contenuto della coda a priorità per frequenze crescenti di lettere/aggregati ad ogni passo dell'algoritmo.

2. (2 punti)
Sia data la sequenza di chiavi intere 13 181 267 302 98 110 45 207. Si riporti il contenuto di una tabella di hash di dimensione 13, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si usi il linear chaining.

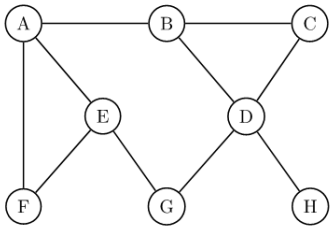
3. (1.5 punti)
Si supponga di aver memorizzato tutti i numeri tra 0 e 1000 in un BST e di cercare la chiave 255. Quali tra le seguenti sequenze non possono essere incontrate durante la ricerca, essendo il BST corretto?

924 220 911 244 898 250 362 255 — 925 202 911 240 912 245 255
2 300 387 219 266 382 381 278 255 — 2 252 401 398 330 344 397 255

4. (1.5 punti)
Si cancelli la chiave 70 dal seguente BST:

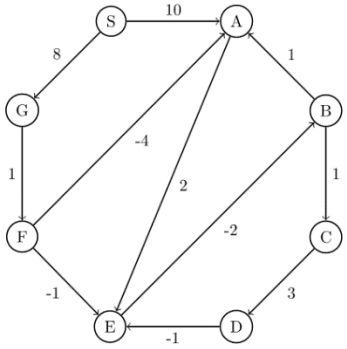


5. (0,5 + 0,5 + 2 punti)
Dato il seguente grafo non orientato



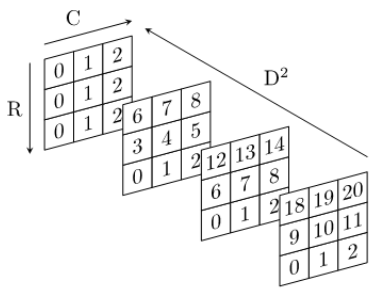
lo si rappresenti come lista e matrice delle adiacenze (**0,5 + 0,5 punti**) e se ne effettui una visita in ampiezza a partire dal vertice **A** (**2 punti**).

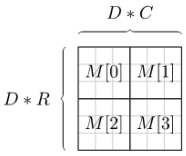
6. (2 punti)
Si determinino per il seguente grafo orientato pesato mediante l'algoritmo di Bellman-Ford i valori di tutti i cammini minimi che collegano il vertice **S** con ogni altro vertice.



03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
III Appello del 27/06/2022 - Prova di programmazione (12 punti)

1. (**2 punti**)
Una funzione riceve una matrice tridimensionale `M` di dimensioni `(D²)xRxC`. La funzione `flatten` alloca una nuova matrice bidimensionale di dimensioni `(DxR)x(DxC)`, ricopia i contenuti della matrice ricevuta come argomento con lo schema illustrato a seguire e ritorna la nuova matrice. In sostanza, nella matrice risultante ogni M[i]-esimo livello viene riposizionato "per righe". Nell'esempio seguente `D` vale 2.





18	19	20	12	13	14
9	10	11	6	7	8
0	1	2	0	1	2
6	7	8	0	1	2
3	4	5	0	1	2
0	1	2	0	1	2

2. (4 punti)
Sia dato un albero di grado `N`, i cui nodi sono definiti dalla seguente struttura C:

```
struct node {
  char *key;
  struct node *children[N];
};
```

Definire la struttura wrapper dell'albero `nTREE` come ADT di prima categoria evidenziando la suddivisione tra file dei vari contenuti. Si realizzi una funzione wrapper `int countIf(nTREE t);` e la corrispondente funzione di visita ricorsiva, che visiti l'albero e ritorni il conteggio del numero di nodi aventi grado (numero di figli) maggiore del grado del rispettivo nodo padre. Il nodo radice, che non ha un padre, conta 1 per default.

3. (6 punti)
Si vogliono generare tutte le sequenze alfabetiche di `k` (parametro del programma) caratteri che rispettino i seguenti vincoli:

- si possono usare solo lettere minuscole (da 'a' a 'z') e maiuscole (da 'A' a 'Z')
- al massimo la metà dei caratteri possono essere lettere minuscole
- lo stesso carattere, senza distinguere tra maiuscolo o minuscolo, non può apparire più di `p` (parametro del programma) volte consecutivamente.

- Si scriva una funzione ricorsiva in C in grado di generare tutte le sequenze secondo le regole di cui sopra, stampando solo quelle accettabili.
- Si giustifichi la scelta del modello combinatorio adottato.
- Si descrivano i criteri di pruning adottato o il motivo della loro assenza.

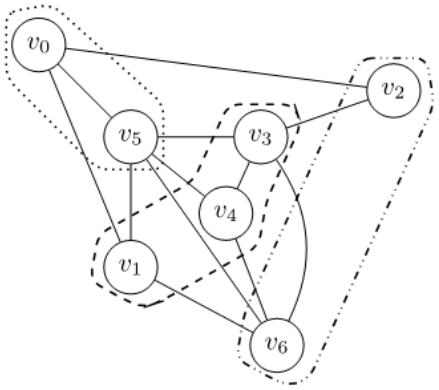
PER ENTRAMBE LE PROVE DI PROGRAMMAZIONE (18 o 12 punti):

- ***indicare nell'elaborato e nella relazione nome, cognome e numero di matricola.***
- ***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 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 giovedì 30/06/2022, alle ore 23:59, 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 la presenza del candidato, sulla base dell'elaborato svolto in aula. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
III Appello del 27/06/2022 - Prova di programmazione (12 punti)

Dato un grafo G = (V, E) non orientato non pesato, viene definita **domatic partition** una partizione $\{V_1, V_2, \ldots, V_k\}$ dei vertici del grafo tale per cui ogni sottoinsieme V_i è un **dominating set** per il grafo stesso.
Si definisce **dominating set** un sottoinsieme D dei vertici di un grafo tale per cui ogni vertice non in D è adiacente ad almeno un vertice di D.

Esempio:
Sia dato il seguente grafo,



la partizione di V $\{V_1 = \{v_0, v_5\}, V_2 = \{v_1, v_3, v_4\}, V_3 = \{v_2, v_6\}\}$ è una **domatic partition** in quanto soddisfa i criteri per essere una partizione ed ognuno dei sottoinsiemi che vi appartengono è un **dominating set.** La sua cardinalità è 3 ed è anche la cardinalità massima.

Si legga in una opportuna struttura dati un grafo G dal file `g.txt` che ha il seguente formato:
- sulla prima riga compare il numero di vertici V
- segue un numero non definito di coppie (v,w) a rappresentare gli archi del grafo, con 0 <= v < V, 0 <= w < V

Si assuma che il grafo non sia un multigrafo e che sia semplice.

Si legga una proposta di partizionamento da file, in un formato a discrezione del candidato, e si valuti se questa sia una **domatic partition** valida, che quindi:
- rappresenti un partizionamento valido dei vertici
- sia tale che ogni sottinsieme sia un dominating set

Si individui la **domatic partition** a cardinalità massima, ossia la partizione composta dal maggior numero possibile di dominating set per il grafo in input.

03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
IV Appello del 02/09/2022 - Prova di teoria (12 punti)

1. (2 punti)
Sia data una coda a priorità di dati inizialmente vuota implementata mediante uno heap. Sia data la sequenza di interi e carattere *:

20 32 19 51 * * 28 * 74 9 81 * * 17 * 41

dove ad ogni intero corrisponde un inserimento nella coda a priorità e al carattere * un'estrazione con cancellazione del **massimo**. Si eseguano **in sequenza** le operazioni di cui sopra.

Modello di risposta

- effettuata l'inserzione di 28 nella coda a priorità che risulta dalle operazioni precedenti, elencare il contenuto del vettore che implementa lo heap:

- effettuata l'inserzione di 81 nella coda a priorità che risulta dalle operazioni precedenti, elencare il contenuto del vettore che implementa lo heap:

- effettuata l'inserzione di 41 nella coda a priorità che risulta dalle operazioni precedenti, elencare il contenuto del vettore che implementa lo heap:

2. (2 punti)
Sia dato un albero binario con 12 nodi. Nella visita si ottengono le 3 seguenti sequenze di chiavi:

preorder	60	30	3	5	1	20	9	13	47	16	18	21
inorder	5	3	1	30	20	9	60	47	13	18	16	21
postorder	5	1	3	9	20	30	47	18	21	16	13	60

Modello di risposta
Avendo ricostruito l'albero binario a partire da tali visite, chi sono:

- il figlio sinistro e destro nel nodo che contiene la chiave 20
- il figlio sinistro e destro nel nodo che contiene la chiave 13
- il figlio sinistro e destro nel nodo che contiene la chiave 3.

3. (2 punti)
Sia data la sequenza di chiavi intere 33 263 109 144 100 113 86 159 58. Si riporti il contenuto di una tabella di hash di dimensione 23, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si usi l'open addressing con double hashing.

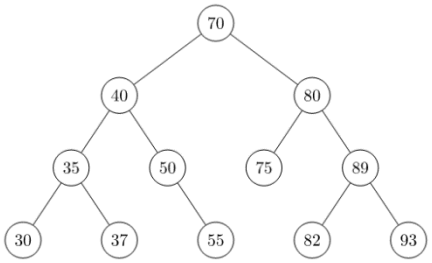
Modello di risposta
Quale tra le seguenti funzioni di hash è opportuno usare?
$h_1(k) = k \% 23$, $h_2(k) = k \% 97$
$h_1(k) = k \% 23$, $h_2(k) = 1+k \% 23$
$h(k, i) = k \% 19 + i + i^2$
$h(k, i) = k \% 23$, $h_2(k) = 1+k \% 97$
Si motivi brevemente la risposta.

Si elenchino le collisioni per la chiave 113.

Si elenchino le collisioni per la chiave 58.

Al termine dell'inserzione, quale chiave contiene la cella all'indice 12?

4. (2 punti)
Sia dato il seguente BST:

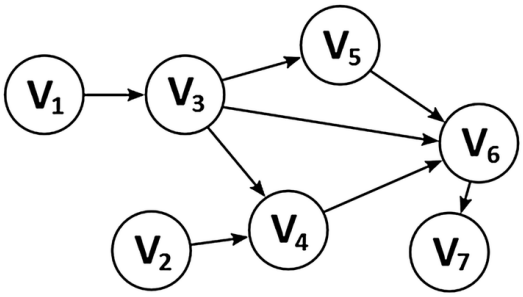


Lo si trasformi in un Order Statistic-BST aggiungendo a ciascun nodo le informazioni necessarie.

Modello di risposta

- Quali informazioni devono essere aggiunte ad ogni nodo? Breve spiegazione.
- Al termine, quali informazioni conterrà la radice?
- E il nodo con chiave 50?
- Se si sta cercando la chiave di rango 4, quali saranno gli archi percorsi? Elencarli nella forma coppie di vertici separate da virgole.

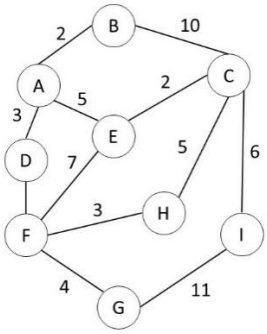
5. (2 punti)
Si ordini topologicamente il seguente DAG:



Si consideri $\mathbf{V_1}$ come vertice di partenza e, qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.

Modello di risposta
per ogni vertice tempo di scoperta / tempo di fine elaborazione
Vertice V_1:
Vertice V_2:
Vertice V_3:
Vertice V_4:
Vertice V_5:
Vertice V_6:
Vertice V_7:

6. (2 punti)
Si determini mediante l'algoritmo di Kruskal l'albero ricoprente minimo per il grafo non orientato, pesato e connesso in figura.



Modello di risposta

per i passi dell'algoritmo richiesti si elenchino su righe separate:

- quale/quali archi sono considerati
- quale/quali tra questi viene/vengono eventualmente scelto/scelti:

Passo 1:

- archi considerati:
- archi scelti:

Passo 3:

- archi considerati:
- archi scelti:

Passo 5:

- archi considerati:
- archi scelti:

03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
IV Appello del 02/09/2022 - Prova di programmazione (12 punti)

1. (**2 punti**)
Una funzione riceve due vettori di interi ordinati, di dimensione nota. La funzione alloca un nuovo vettore `v3` di dimensione opportuna per memorizzare i soli elementi di `v1` che non appaiano anche in `v2`, conservando l'ordinamento. Eventuali elementi duplicati di `v1` vanno inseriti in `v3` una sola volta.
Completare il prototipo a seguire, in modo che `v3` e la sua dimensione effettiva siano disponibili a chi invoca la funzione, e implementare la funzione richiesta.

```
... f(int *v1, int *v2, int dim1, int dim2, ...)
```

Es.:

```
v1 = {1,2,2,3,4,5}, dim1 = 6
v2 = {1,3}, dim2 = 2
v3 = {2,4,5}, dim3 = 3
```

2. (**4 punti**)
Si supponga di dover implementare un albero di grado 3, i cui nodi contengono valori interi.
Definire la struttura wrapper dell'albero (come ADT di prima categoria) e dei nodi (a libera scelta del candidato), evidenziando la suddivisione tra file dei vari contenuti. Si realizzi inoltre una funzione wrapper C `void countIf(nTREE t, ...);` e la corrispondente funzione di visita ricorsiva, che visiti l'albero e ritorni il conteggio del numero di nodi aventi esattamente 1, esattamente 2 o esattamente 3 figli. La funzione deve quindi calcolare e rendere disponibili a chi la invoca tre valori distinti, per rispondere alla richiesta posta.

3. (**6 punti**)
Dato un vettore di interi `I`, di dimensione `dimI`, e un vettore `S`, di dimensione `dimS`, contenente una sequenza di somme obiettivo, scrivere una funzione ricorsiva che valuti se sia possibile ottenere le somme riportate in `S` utilizzando gli elementi del vettore `I`. Non è necessario usare tutti gli elementi di `I`, ma ogni elemento può essere usato al massimo una volta. Una volta trovata una soluzione valida, interrompere la ricerca: non è necessario individuare tutte le soluzioni possibili.

- Si scriva la funzione ricorsiva richiesta.
- Si giustifichi la scelta del modello combinatorio adottato.
- Si descrivano i criteri di pruning adottato o il motivo della loro assenza.

Es. 1:

```
I = {1,2,3,4,5,6}, dimI = 6
S = {1,7,7}, dimS = 3
```

È possibile ottenere il risultato voluto, come: (1), (2+5), (3+4)

Es. 2:

```
I = {1,3,4,5,5}, dimI = 5
S = {7,7}, dimS = 2
```

Non è possibile ottenere il risultato voluto.

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

- ***indicare nell'elaborato e nella relazione nome, cognome e numero di matricola.***
- ***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 mercoledì 07/09/2022, alle ore 23:59, 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 la presenza del candidato, <u>sulla base dell'elaborato svolto in aula</u>. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
IV Appello del 02/09/2022 - Prova di programmazione (18 punti)

Un crucipuzzle è un gioco enigmistico che consiste nel cercare delle parole, generalmente attinenti ad un tema e presenti in un elenco, all'interno di una griglia di lettere.

La griglia di gioco è una scacchiera rettangolare di dimensione `RxC` contenente solo lettere maiuscole, memorizzata nel file "`griglia.txt`" con il seguente formato:

- Sulla prima riga appare la coppia di interi `R` e `C`
- Seguono `R` righe di `C` caratteri, senza spazi, a rappresentare la griglia di gioco.

Le parole da ricercare sono memorizzate in un file "`parole.txt`", di lunghezza non nota, in cui ogni riga riporta una coppia `<parola> <valore>`, dove `<parola>` è una stringa senza spazi di massimo 15 caratteri e `<valore>` è un intero positivo a rappresentare il valore della parola.

Le regole del gioco solo le seguenti:

- Le parole possono essere cercate nel solo verso di lettura in orizzontale (da sinistra a destra), verticale (dall'alto in basso) o diagonale (da sinistra a destra e dall'alto in basso). Una parola deve essere trovata nella sua interezza: non sono ammessi match parziali
- Data una coppia di parole trovate nello schema, le due parole possono incrociarsi condividendo al massimo una lettera. Non sono ammesse sovrapposizioni di sottostringhe di lunghezza superiore a un carattere
- Il valore della "soluzione" è pari alla somma dei valori delle parole trovate nella griglia
- Non è detto che sia possibile individuare tutte le parole dell'elenco nello schema di gioco

Nell'esempio proposto a seguire è presentata una generica griglia di gioco e una sua versione "risolta" in cui sono state individuate 21 parole. Prestare attenzione a come la parola `SORT` (Oriz.,II,12) non possa essere trovata in (Diag.,VII,7) poiché la sovrapposizione con `QUICKSORT` (Diag.,II,2) violerebbe uno dei vincoli di cui sopra (massimo un carattere in comune data una coppia di parole).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
I	K	K	N	Z	T	B	E	L	L	M	A	N	T	S	O	X	N	S
II	A	Q	T	H	U	P	A	T	H	O	E	S	O	R	T	C	B	D
III	L	C	U	L	O	G	N	J	S	V	P	C	V	I	P	A	A	I
IV	B	L	M	I	N	L	I	V	L	E	E	A	J	C	R	H	C	J
V	E	V	N	A	C	A	L	G	O	R	I	T	M	O	U	R	G	K
VI	R	T	K	W	L	K	P	V	H	T	D	F	L	R	N	N	N	S
VII	O	B	O	O	L	I	S	U	Y	I	F	Y	V	S	I	X	G	T
VIII	G	R	A	F	O	C	S	O	F	C	S	N	I	I	N	M	E	R
IX	A	D	E	S	F	C	W	T	R	E	U	O	T	O	G	P	D	A
X	C	B	F	S	B	D	O	P	A	T	S	D	O	N	T	F	G	B
XI	R	J	G	N	M	Q	Q	U	E	U	E	O	H	E	Z	V	E	F
XII	N	F	Y	C	A	M	M	I	N	O	M	I	N	I	M	O	D	B

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
I						B	E	L	L	M	A	N		S				
II	A	Q				P	A	T	H			S	O	R	T			D
III	L		U	L	O	G	N			V				I	P	A		I
IV	B			I						E				C	R		C	J
V	E				C	A	L	G	O	R	I	T	M	O	U			K
VI	R				L	K				T	D			R	N			S
VII	O	B	O	O	L	I	S			I	F			S	I			T
VIII	G	R	A	F	O		S	O		C	S	N		I	N		E	R
IX							T	R	E		O			O	G		D	A
X		B	F	S					A	T		D		N			G	
XI						Q	U	E	U	E	O			E			E	
XII				C	A	M	M	I	N	O	M	I	N	I	M	O		

Richieste:

1. Definire e implementare delle opportune strutture dati per rappresentare:
   - la griglia di gioco (tipo `GRID`)
   - l'elenco di parole da cercare (tipo `WORDS`)
   - una soluzione al problema di ricerca (tipo `SOL`)

   In aggiunta alle tre strutture dati esplicitamente richieste, è possibile definire tutti i tipi ausiliari ritenuti opportuni, a supporto delle tre strutture principali. Si presti attenzione a includere esplicitamente le funzioni dedicate alle letture dei file di cui sopra.
2. Acquisire da un file "`proposta.txt`", il cui formato è a descrizione del candidato (e di cui il candidato deve fornire anche una breve descrizione), una soluzione al problema di ricerca. La proposta è valida se le parole che la compongono fanno effettivamente parte della lista da cercare, si trovano nella griglia di gioco e rispettano i vincoli di orientamento e sovrapposizioni ammessi indicati in precedenza
3. Scrivere una funzione ricorsiva in grado di individuare la soluzione a valore massimo possibile. A fronte di soluzioni dal valore equivalente, si prediliga quella composta dal maggior numero di parole.