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.

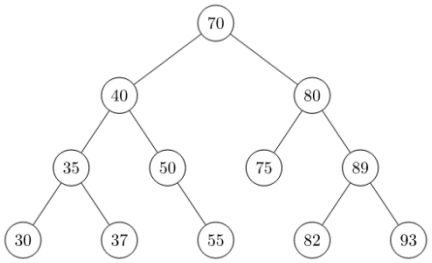
Modello di risposta
Quale tra le seguenti funzioni di hash è opportuno usare?
$h_1(k) = k \% 23$, $h_2(k) = k \% 97$
$h_1(k) = k \% 23$, $h_2(k) = 1+k \% 23$
$h(k, i) = k \% 19 + i + i^2$
$h(k, i) = k \% 23$, $h_2(k) = 1+k \% 97$
Si motivi brevemente la risposta.

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:

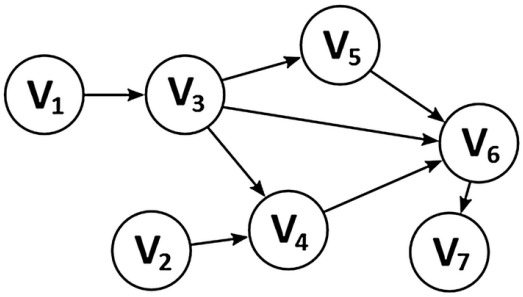


Lo si trasformi in un Order Statistic-BST aggiungendo a ciascun nodo le informazioni necessarie.

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.

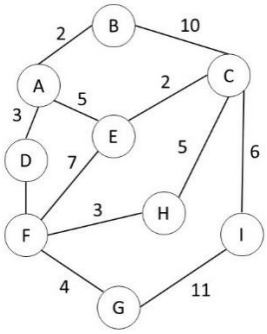
5. (2 punti)
Si ordini topologicamente il seguente DAG:



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.