

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 $R \times C$ rappresenta una griglia di gioco.

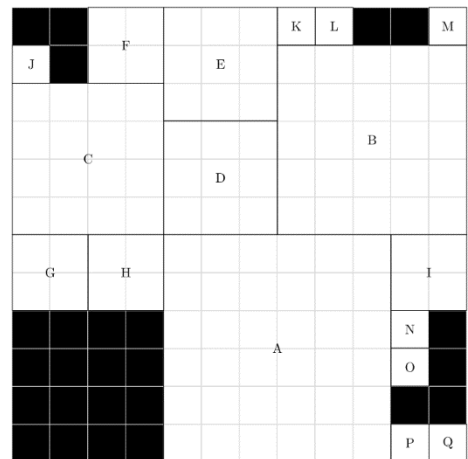
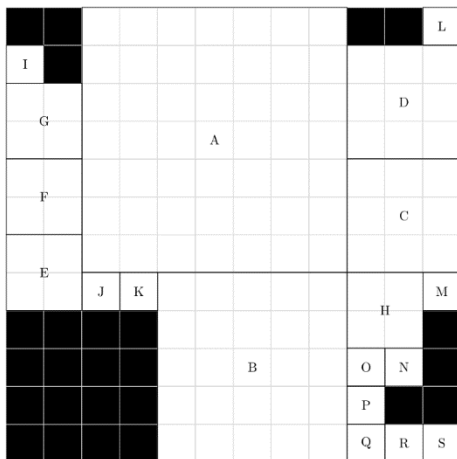
La griglia contiene celle bianche e celle nere. Le celle bianche definiscono una regione contigua nella griglia. L'obiettivo del gioco è suddividere la regione bianca nel numero minimo di sottoregioni quadrate prive di sovrapposizioni.

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 $NR \quad NC$, 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 12×12 , 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, 7×7) 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 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 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, 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
Appello del 27/01/2022 - Prova di programmazione (12 punti)

1. (2 punti)

Sia data una matrice M di dimensione $r \times c$ contenente elementi interi positivi o nulli.

Si scriva una funzione f che generi una matrice M' di dimensione $r' \times c'$ 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 $N \times N$ 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 programmazione (18 punti)

1. (18 punti)

Una matrice rettangolare $R \times C$ 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.

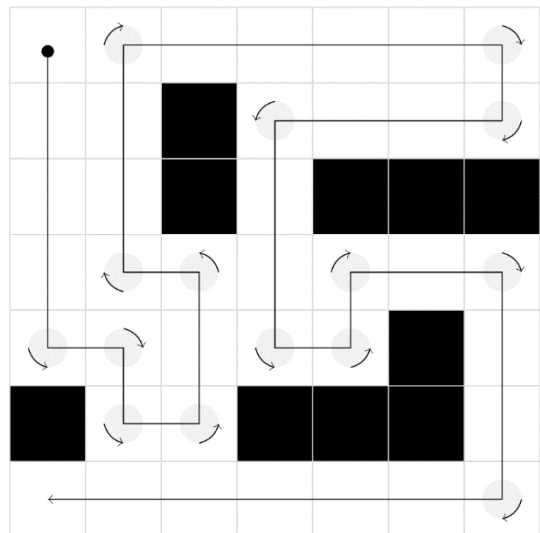
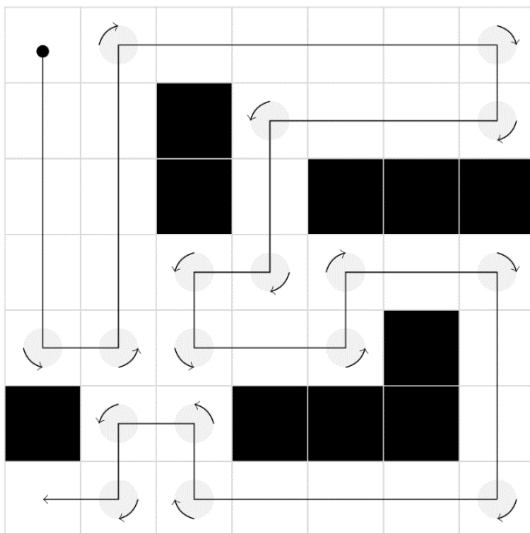
La cella in posizione (0,0) è sempre bianca ed è sempre il punto di partenza del cammino.

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 lungo 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 $NR \quad NC$, 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 7×7 . 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 $r \times c$ contenente elementi interi positivi o nulli.

Scrivere una funzione f che generi una matrice M' di dimensione $r' \times c'$ 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 $N \times N$ 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 giustificare 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 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 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, sulla base dell'elaborato svolto in aula. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.

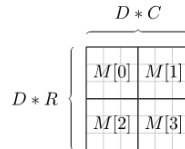
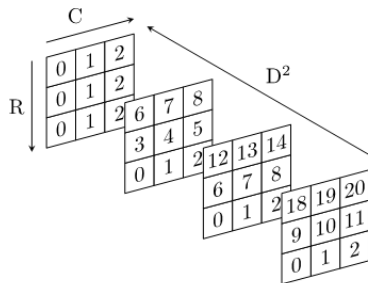
03AAX Algoritmi e Strutture Dati

03MNO Algoritmi e Programmazione

III Appello del 27/06/2022 - Prova di programmazione (12 punti)

1. (2 punti)

Una funzione riceve una matrice tridimensionale M di dimensioni $(D^2) \times R \times C$. La funzione `flatten` alloca una nuova matrice bidimensionale di dimensioni $(D \times R) \times (D \times C)$, 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.

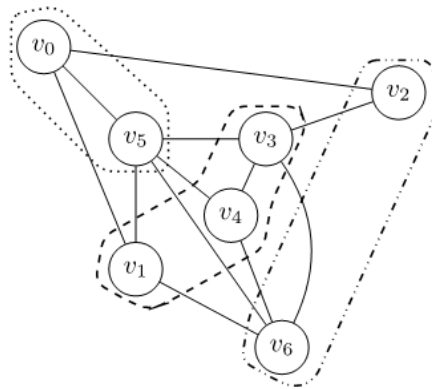
03AAX Algoritmi e Strutture Dati
03MNO Algoritmi e Programmazione
III Appello del 27/06/2022 - Prova di programmazione (18 punti)

Dato un grafo $G = (V, E)$ non orientato non pesato, viene definita **domatic partition** una partizione $\{V_1, V_2, \dots, V_k\}$ 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 $0 \leq v < V, 0 \leq w < V$

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 sottoinsieme 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 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 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 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, 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

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 $R \times C$ 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 $\langle \text{parola} \rangle \langle \text{valore} \rangle$, dove $\langle \text{parola} \rangle$ è una stringa senza spazi di massimo 15 caratteri e $\langle \text{valore} \rangle$ è un intero positivo a rappresentare il valore della parola.

Le regole del gioco sono 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
ix										T	R	E		O	O	G		D
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 predilige quella composta dal maggior numero di parole.