Palestra di algoritmi

28/11/2023 - gruppo A

Soluzioni

Borse di studio

```
void ric(int pos, int tot, int att, int vet[MAX]) {
        if(tot==0) { //se non ho più soldi
                //stampo vet
        } else {
                for(int j=1; j <= att; j++) { //genero tutte le possibili combinazioni
                        if(tot-j >= 0) {
                                vet[pos] = j;
                                ricorsione(pos+1, tot-j, j, vet);
                        } else {
                                break;
                        }
                }
        }
}

Nel main: ricorsione(0, N, N, vet);
```

NB: devo dichiarare ofstream out(“output.txt”) fuori dal main, in maniera globale, altrimenti non posso usarlo nella funzione.

Domino

```
struct tessera{
        int s; //numero a sinistra
        int d; //numero a destra
}
```

Per utilizzare la struct:

```
struct tessera t;
t.d = 1;
t.s = 2;
cout << t.d << “ “ << t.s;
```

```
void trova_permutazione(int pos, tessera vet[MAX], bool usata[MAX]){
        if(pos>lung_max) { lung_max = pos; }
        for(int i=0; i<N; i++) {
                if(!usata[i] && compatibile con la precedente) {
                        vet[pos] = tessera[i];
                        usata[i] = true;
                        trova_permutazione (pos+1, vet, usata);
                }
        }
}
```

Matrici

Matrici

Abbiamo visto gli array, cioè una sequenza di elementi tutti dello stesso tipo.

Cosa succede se gli elementi dell'array sono a loro volta array?

Otteniamo una matrice!

Dimensions	Example	Terminology
1	0 1 2	Vector
2	0 1 2 3 4 5 6 7 8	Matrix

Codice

Dichiarazione:

int matrix[RIGHE][COLONNE]

Accedere a un elemento:

int x = matrix[i][j]
Dove 0<=i<RIGHE e 0<=j<COLONNE

Codice

Inizializzazione:

int matrix[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}}
NB: se inizializziamo in questo modo, si può omettere il numero di righe.
Es: int matrix[][3] = {{1,2,3}, {4,5,6}, {7,8,9}}

Inizializzazione da tastiera (o da file):

```
int matrix[RIGHE][COLONNE];
for(int i=0;i<RIGHE;i++){
	for(int j=0;j<COLONNE;j++){
		cin >> matrix[i][j];
	}
}
```

NB: similmente, con questo doppio ciclo, possiamo stampare tutta la matrice, fare operazioni su tutti gli elementi, eccetera…

Esercizio (semplice)

Data in input una matrice N*M, determinare qual è la riga con la somma maggiore.

Esempio:

4 3
13 34 21
17 22 33
14 55 56
23 24 25

In questo caso dobbiamo dare in output 2 (ricordiamo che gli indici partono da 0!)

```
#define MAX 100

int N, M;
int matrice[MAX][MAX];

cin >> N >> M;
for(int i=0; i<N; i++){
        for(int j=0; j<M; j++) {
                cin >> matrice[i][j];
        }
}

int massimo = 0;
int indice = -1;

for(int i=0; i<N; i++) {
        int somma = 0;
        for(int j=0; j<M; j++){
                somma = somma + matrice[i][j];
        }
        if(somma>massimo) {massimo = somma; indice = i;}
}
cout << indice;
```

Numeri primi

Cos’è un numero primo?

Un numero primo è un numero intero, maggiore di 1, che sia divisibile solo per 1 e per sé stesso.

Sono quindi numeri primi

2, 3, 5, 7, 11, 13, 17, 19, 23, …

Attenzione: 1 non è considerato numero primo!

Come verificare se un numero N è primo?

Controlliamo tutti i numeri da 2 a N-1 e verifichiamo non siano divisori

Esempio: N=25

25%2 != 0 -> non è divisore

25%3 !=0 -> non è divisore

25%4 !=0 -> non è divisore

25%5 ==0 -> è divisore -> 25 non è primo

Alcuni miglioramenti

Al posto di andare da 2 a N-1, possiamo andare da 2 a N/2 (un numero non avrà come divisore un numero più grande della sua metà, oltre sè stesso).

Esempio: N=24, il suo divisore più grande, oltre a 24, è 12 (24/2)

Un altro metodo

Creiamo un vettore booleano di dimensione N+1.

Inizializziamo tutto il vettore a false.

Partendo da 2, mettiamo a true tutti i suoi multipli fino a N.

Avanziamo di 1 e, se è false, facciamo la stessa cosa, fino a N/2.

In questo modo, se alla fine di questo processo la posizione N è ancora a false, sappiamo che è primo.

Vediamo meglio con un esempio.

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f
	f	f	v	f	v	f	v	f	v	f	v	f	v	f	v	f
	f	f	v	f	v	f	v	v	v	f	v	f	v	v	v	f
	f	f	v	f	v	f	v	v	v	f	v	f	v	v	v	f
	f	f	v	f	v	f	v	v	v	f	v	f	v	v	v	f
	f	f	v	f	v	f	v	v	v	f	v	f	v	v	v	f
Numeri primi:	2	3		5		7				11		13				17

```
int N;
bool vet[MAX];
cin >> N;


for(int i=2; i<=N; i++) {
        if(vet[i]==false){
                int j = i;
                while(j<=N){
                        j+=i;
                        vet[j] = true;
                }
        }
}
if(vet[N] == false) { cout << “Primo!” }
else { cout << “Non primo!”; }
```

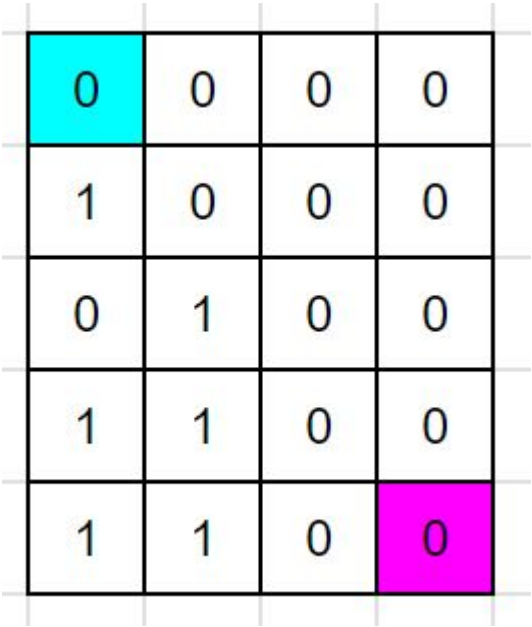
Esercizio

Matrice prima:

https://training.olinfo.it/#/task/matrice/statement

Matrici come mappe

Matrice come mappa



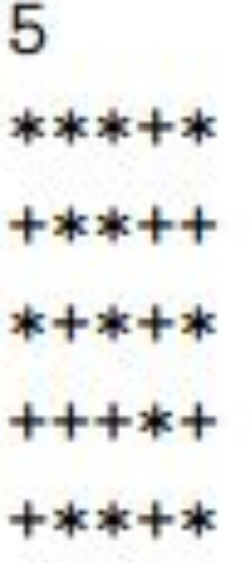
Alcuni problemi delle olimpiadi chiedono di vedere una matrice come una mappa.

In questo caso immaginiamo le celle della matrice come piastrelle e di poterci muovere solo sulle caselle con gli 0 (quelle con 1 hanno delle trappole).

Dobbiamo trovare se esiste un percorso dalla casella azzurra alla casella rosa.

Input

NB: la matrice di solito è data come stringhe di asterischi (*) e più (+), come in questo esempio. Per prendere in input usare il seguente metodo:




```
in >> N;

for(int i=0; i < N; i++) {
        in.get(); // leggo il carattere 'a capo'
        for(int j=0; j < N; j++) {
            m[i][j] = in.get();
        }
}
```

“Muoversi” sulla mappa

```
int mappa[MAX][MAX] m;
bool visitata[MAX][MAX] v;        //inizializzata a false

void walk(int r, int c) {         //riga e colonna (coordinate della cella in cui siamo)
        visitata[r][c] = true;
        //Questi 2 array rappresentano le 8 direzioni in cui posso andare
        int ar[] = {-1,-1,0,1,1,1,0,-1};      //Array righe (es. -1 significa vai nella riga sopra)
        int ac[] = {0,-1,-1,-1,0,1,1,1};      //Array colonne (es. 1 significa vai verso destra)

        for(int i=0; i < 8; i++) {            //Vai in una delle 8 direzioni
                int new_row = r + ar[i];
                int new_col = c + ac[i];

                if (isValid(new_row, new_col) && !visitata[new_row][new_col]) {   //Se è valida
                        walk(new_row, new_col);                                   //Visito la cella
                }
        }
}
```

Nel main: walk(0,0)

```
bool isValid(int r, int c) { //Ritorna true se la cella e' dentro la matrice
        if (r < 0 || c < 0 || r >= N || c >= N)
                return false;
        return true;
}
```

Esercizio

- mappa (https://training.olinfo.it/#/task/mappa/statement)