# Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

Prova di laboratorio - Appello del 23 gennaio 2019

## Note importanti

- Leggete attentamente il testo degli esercizi e le indicazioni su come svolgerli. Se avete dubbi sul significato delle richieste, chiedete chiarimenti!
- I file utili allo svolgimento degli esercizi, e indicati nel testo, sono contenuti nell'archivio zip, nella cartella `allegati`.
- Gli esempi di input/output proposti nel testo sono anch'essi contenuti nell'archivio zip, in file di text separati, nella cartella `esempi`.
- Leggete attentamente anche le indicazioni su come preparare le risposte. Per ogni esercizio è richiesto di preparare un file: in alcuni casi si tratta di un file di testo, in altri casi di un programma in C. Per ogni esercizio viene indicato il nome con cui salvare il file; è importante rispettare questa indicazione.
- Nella prima riga di tutti i file che consegnerete, **scrivete nome, cognome e matricola**.
- Dopo esservi autenticati, caricate sul sito `upload.di.unimi.it` tutti i file contenenti le vostre risposte. I nomi dei file devono essere i seguenti: `es1-struttura.txt`, `es2-readBoard.c`, `es3-grafo.txt`, `es4-min.c`. `es5-comandi.c`.

## Esercizio 1: Struttura dati misteriosa

Il programma in allegato `es1-struttura.c` contiene l'implementazione di una struttura dati classica, qui indicata genericamente col termine `struttura`. Tale implementazione si basa a sua volta sull'uso di liste concatenate, e di funzioni per la loro gestione (inserimento, cancellazione, stampa). Il programma contiene anche una funzione `main` che può essere usata per fare dei test.

Analizzate il programma e rispondete alle domande seguenti.

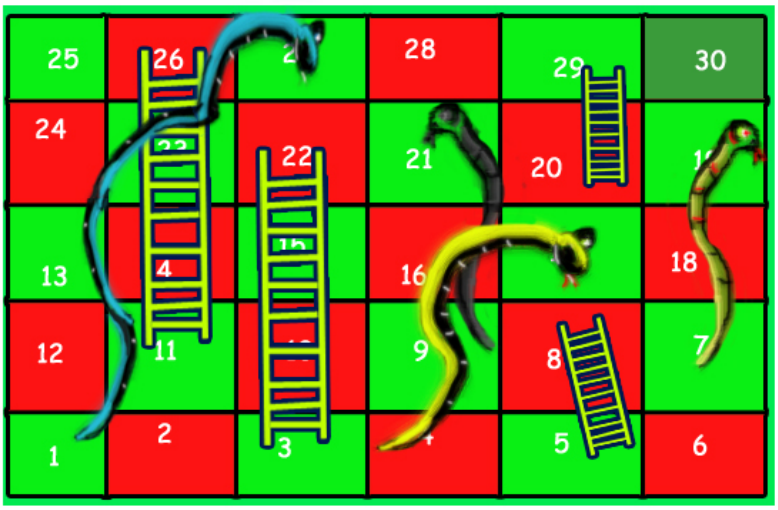
1. Individuate di che struttura dati si tratta e descrivete come è implementata.
2. Cosa rappresentano i membri `a` e `b`?
3. Cosa fanno le funzioni `f1` e `f2`? Scegliete dei nomi più significativi per queste funzioni.

**Note per la consegna.** Scrivete le risposte in un file di testo con nome `es1-struttura.txt`.

**Gli esercizi rimanenti riguardano tutti il gioco "Scale e serpenti" presentato qui di seguito.**

## Scale e serpenti - regole del gioco

Il gioco "Scale e serpenti" si gioca su una griglia rettangolare. che contiene numeri, scale e serpenti, come nella seguente figura:




Siano $r$ e $c$ il numero di righe e di colonne nella griglia, e $n = r \times c$ il numero totale di caselle. La griglia contiene i numeri da 1 a $n$, uno per casella. Si parte dalla casella 1, si lancia un dado e ci si sposta in avanti in base al numero ottenuto. Le scale e i serpenti determinano dei *salti*: quando si raggiunge una casella che contiene la base di una scala, si deve salire la scala fino a raggiungere la casella in cui si trova la cima della scala; quando si raggiunge una casella con la bocca di un serpente, si deve scendere fino alla casella in cui si trova la coda del serpente. L'obiettivo è raggiungere la casella numerata con $n$. Se con un lancio si supera il numero $n$, la partita è persa.

**Esempio.** Nella griglia di gioco rappresentata in figura, se si lancia il dato tre volte ottenendo i numeri 2, 6, 2, si procede come segue:

- dalla casella 1, avendo lanciato 2, si raggiunge la casella 3 che contiene la base di una scala, quindi si sale fino alla casella 22;
- dalla casella 22, avendo lanciato 6, si raggiunge la casella 28;
- infine, dalla casella 28, avendo lanciato 2, si raggiunge 30.

Si vince la partita anche con le sequenze di lanci (2, 2, 6), oppure (2, 5, 2, 6, 2), oppure (5, 6, 6, 6, 6 ).

**Descrizione sintetica della griglia di gioco** Ai fini del gioco e del calcolo delle mosse, la griglia di gioco può essere descritta sinteticamente tramite il numero $n$ e la sequenza di interi $a_1, a_2, \ldots a_n$ che indicano gli eventuali salti. Più precisamente:

- se la casella numerata con $i$ non contiene né la base di una scala né la bocca di un serpente, allora $a_i$ è pari a 0;
- altrimenti $a_i$ è il numero della casella in cui saltare, una volta raggiunta la casella numerata con $i$.

**Esempio** La descrizione sintetica della griglia sopra disegnata è il seguente:

```
30
0 0 22 0 8 0 0 0 0 0 26 0 0 0 0 0 4 0 7 29 9 0 0 0 0 0 1 0 0 0
```

## Esercizio 2: scale e serpenti - trova gli errori

Il programma in allegato `es2-readBoardErrato.c` legge da standard input le informazioni relative alla griglia di gioco (vedi sotto le specifiche del formato di input) e produce in output la descrizione sintetica della griglia stessa, come definita in precedenza. Il programma contiene alcuni errori: individuateli e correggeteli. Per ogni correzione, annotate un commento che spiega qual era l'errore e come l'avete corretto. NOTA IMPORTANTE: effettuate le modifiche necessarie a correggere gli errori, ma *senza stravolgere il programma*.

**Note per la consegna.** Correggete il programma contenuto nel file, annotandolo con i commenti opportuni, infine salvate il file con nome `es2-readBoard.c`.

**Il programma deve:**

- leggere da standard input i dettagli relativi alla griglia e alla posizione di serpenti e scale (vedi sotto le specifiche del formato di input);
- stampare su standard output la descrizione sintetica della griglia, come definita nella sezione precedente.

**Formato di input** L'input è formato da:

- due numeri $r$ e $c$;
- un numero $s$;
- $s$ linee che rappresentano ciascuna la posizione di un serpente: per ogni serpente sono indicati i valori delle caselle che contengono rispettivamente la bocca e la coda del serpente;
- un numero $l$;
- $l$ linee che rappresentano ciascuna la posizione di una scala: per ogni scala sono indicati il valore della casella che contiene la base della scala e il numero di caselle coperte dalla scala stessa.

**Esempio di esecuzione** Nel caso della griglia rappresentata in figura, il serpente con la coda nella casella 1 e la bocca nella casella 27 è indicato dalla coppia `27 1`; la scala che ha la base nella casella 11 e la cima nella casella 26 è indicata dalla coppia `11 4`. Nel complesso, l'input è dato da:

```
5 6
4
27 1
21 9
17 4
19 7
4
11 4
3 4
20 2
5 2
```

L'output corretto è il seguente:

```
30
0 0 22 0 8 0 0 0 0 0 26 0 0 0 0 0 4 0 7 29 9 0 0 0 0 0 1 0 0 0
```

## Esercizio 3: scale e serpenti - modellazione con grafo

1. Descrivete e formalizzate il gioco in termini di grafi, specificando in particolare cosa rappresentano i nodi e cosa rappresentano gli archi. Indicate inoltre le caratteristiche fondamentali del grafo (è orientato? è connesso? ha delle regolarità particolari? ecc... )

2. Progettate due algoritmi che siano in grado rispettivamente di:
   a) Calcolare il numero minimo di lanci necessario per vincere la partita.
   b) Stampare una sequenza di lanci di lunghezza minima che consente di vincere la partita.

3. Modificate gli algoritmi suddetti in modo che si possa specificare la casella di partenza.

4. Modificate ulteriormente gli algoritmi in modo da evitare le mosse che utilizzano scale e serpenti. Ad esempio, nel primo caso l'algoritmo deve calcolare il numero minimo di mosse necessarie per poter vincere la partita partendo da una data casella, ma senza usare né scale né serpenti.

**Note per la consegna** Scrivete le risposte in un file di testo con nome `es3-grafo.txt`.

**Esempio** Se la griglia di gioco è quella rappresentata nella figura, il numero minimo di lanci per raggiungere 30 partendo dalla casella 1 è 3 (ad esempio con i lanci 2, 6, 2). Senza usare scale né serpenti, il numero minimo di lanci invece è 5 (ad esempio con i lanci 5, 6, 6, 6, 6).

## Esercizio 4: scale e serpenti - implementazione

Scrivete un programma che implementi l'algoritmo che avete progettato e descritto nell'esercizio 3, in risposta alla richiesta 2a).

Il programma deve leggere da standard input la descrizione sintetica della griglia di gioco (come specificata alla fine di pagina 2) e deve stampare su standard output il numero minimo di lanci di dado necessari a vincere la partita partendo dalla casella col numero 1.

Potete utilizzare, se lo ritenete opportuno, porzioni di codice fornite o sviluppate per gli esercizi precedenti.

**Esempio di esecuzione** Su input

```
30
0 0 22 0 8 0 0 0 0 0 26 0 0 0 0 0 4 0 7 29 9 0 0 0 0 0 1 0 0 0
```

(che rappresenta la solita griglia disegnata in figura), il programma deve stampare:

```
3
```

**Suggerimento.** Anche se l'algoritmo da implementare è definito in termini di grafi, non è detto che sia necessario mantenere in memoria tutti i dati relativi a nodi e archi; nel caso in cui il grafo abbia una struttura molto regolare, infatti, può essere più comodo calcolare questi dati durante l'esecuzione dell'algoritmo stesso.

**Note per la consegna.** Scrivete il vostro programma in un file con nome `es4-min.c`.

## Esercizio 5: scale e serpenti - comandi (facoltativo)

Estendete il programma che avete sviluppato per l'esercizio 4, implementando anche gli altri algoritmi che avete progettato e descritto nell'esercizio 3.

Il programma deve leggere da standard input una sequenza di istruzioni, scritte una per riga, secondo il formato nella seguente tabella. Il particolare il comando *R* legge da standard input la descrizione sintetica della griglia di gioco formata come specificata a pagina 2.

| Istruzione | Operazione |
|---|---|
| `R` $n\ a_1\ a_2 \cdots a_n$ | Legge la descrizione sintetica della griglia di gioco |
| `x` $c\ \ d$ | Stampa la casella che si raggiunge partendo dalla casella $c$ e avendo ottenuto $d$ con il lancio del dado (se la mossa non è valida, non stampa niente). |
| `m` $i$ | Stampa il minimo numero di lanci necessari per vincere la partita partendo dalla casella $i$. |
| `s` $i$ | Stampa una sequenza di lanci di lunghezza minima che consente di vincere la partita partendo dalla casella $i$. |
| `m-` $i$ | Stampa il minimo numero di lanci necessario per vincere la partita partendo dalla casella $i$, ma senza usare né scale né serpenti. |
| `s-` $i$ | Stampa una sequenza di lanci di lunghezza minima che consente di vincere la partita partendo dalla casella $i$, ma senza usare né scale né serpenti. |
| `f` | Termina l'esecuzione |

**Note per la consegna.** Scrivete il vostro programma in un file con nome `es5-comandi.c`.

**Esempio di esecuzione** Su input

```
R 30 0 0 22 0 8 0 0 0 0 0 26 0 0 0 0 0 4 0 7 29 9 0 0 0 0 0 1 0 0 0
x 1 2
x 22 6
x 28 2
m 1
s 1
m- 1
s- 1
f
```

il programma deve stampare:

```
22
28
30
3
2 2 6
5
5 6 6 6 6
```