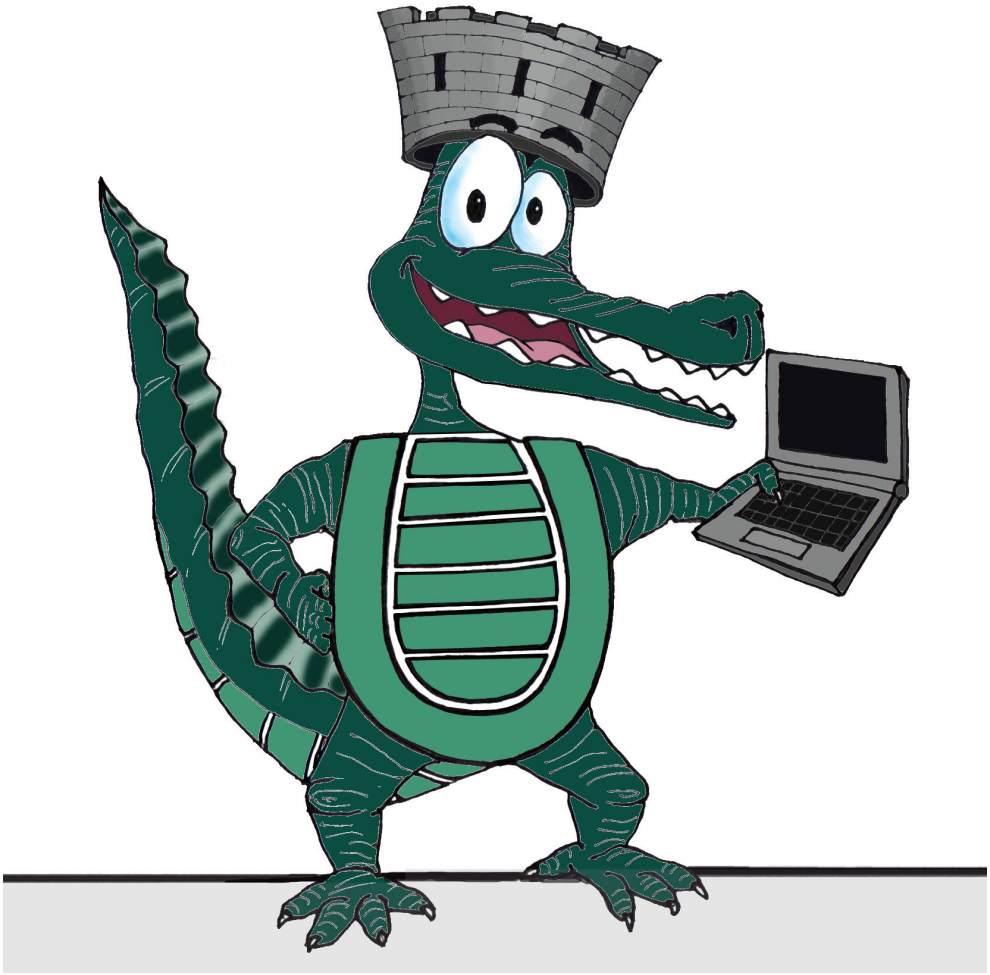


GATOR: Testi e Soluzioni dei problemi

Ministero dell'Istruzione dell'Università e Ricerca

AICA
Associazione Italiana per l'Informatica ed il Calcolo Automatico

Problemi a cura di
Giuseppe F. Italiano, Luigi Laura

Coordinamento
Monica Gati

Logo della gara
Manuela Pattarini

Testi dei problemi
William Di Luigi, Giuseppe F. Italiano, Luigi Laura

Soluzioni dei problemi
Alessandro Bugatti, Giovanni Campagna, William Di Luigi, Luca Versari

Gestione della gara online
William Di Luigi, Luca Versari

Sistema di gara
Contest Management System (CMS) http://cms-dev.github.io/.

Supervisione a cura del Comitato per le Olimpiadi di Informatica

Indice

1 Introduzione

La prima edizione della Gara di Allenamento TOR vergata (GATOR) si è svolta sabato 29 e domenica 30 marzo 2014 (http://www.disp.uniroma2.it/users/italiano/gator/). La GATOR è una gara online di programmazione, con 4 problemi da svolgere in 5 ore. I problemi sono stati concepiti in modo da poter essere affrontati dagli studenti delle scuole secondarie superiori selezionati per la fase Territoriale delle Olimpiadi di Informatica. Ben 254 persone da tutta Italia si sono registrate alla prima edizione della GATOR; di queste, 180 si sono collegate al sistema di gara e 139 hanno ottenuto punti su almeno uno dei quattro problemi proposti.

La GATOR è una iniziativa svolta nell'ambito della convenzione tra il Comitato Italiano delle Olimpiadi di Informatica e il Dipartimento di Ingegneria Civile e Ingegneria Informatica dell'Università di Roma "Tor Vergata. Le Olimpiadi di Informatica sono nate con l'intento di selezionare e formare, ogni anno, una squadra di atleti che rappresenti il nostro paese alle International Olympiad in Informatics (IOI), indette dall'UNESCO fin dal 1989. L'organizzazione delle Olimpiadi di Informatica è gestita dal Ministero dell'Istruzione, dell'Università e della Ricerca e dall'AICA, con l'obiettivo primario di stimolare l'interesse dei giovani verso la scienza dell'informazione e le tecnologie informatiche.

In questo documento trovate i testi dei quattro problemi proposti nella gara e una proposta di soluzione. Vista l'imminenza dei Campionati Mondiali di Calcio 2014 in Brasile, il tema dei problemi proposti nella GATOR è stato il calcio.

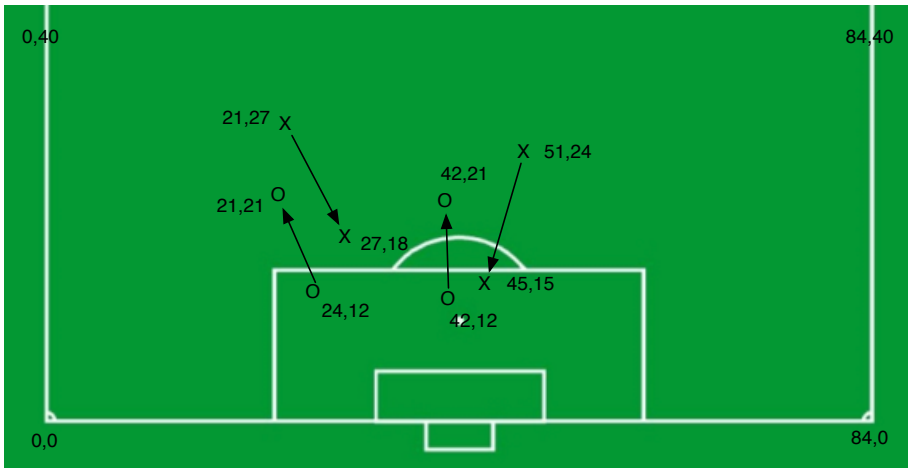
Ringraziamo Manuela Pattarini, Monica Gati, Luca Versari e William Di Luigi per la loro collaborazione, senza la quale non sarebbe stato possibile organizzare la GATOR.

Giuseppe F. Italiano e Luigi Laura

2 Fuorigioco (`fuorigioco`) [Difficoltà D=2]

2.1 Descrizione del problema

Il fuorigioco è una delle regole più discusse del gioco del calcio, e spesso è complicato riuscire a capire se un giocatore è partito in posizione regolare. Grazie alle nuove tecnologie, viene scattata una immagine in ogni secondo della partita, localizzando le posizioni dei giocatori in una griglia, e siamo in grado di dire quando è partito il pallone con una precisione del decimo di secondo. Per esempio, nell'immagine qui sotto possiamo vedere una tipica situazione di gioco, con due attaccanti (rappresentati dalle X) che avanzano, e due difensori (rappresentati dai cerchi O) che cercano di metterli in fuorigioco. A fianco a ogni giocatore vediamo la sua posizione, in un sistema di coordinate X, Y, in metri, che ha la sua origine nella bandierina del calcio d'angolo in basso a sinistra.



Il vostro compito è quello di scrivere un programma che, ricevute in ingresso le posizioni dei giocatori nelle due immagini, scattate prima e dopo il lancio, e a quale decimo di secondo (compreso tra 1 e 9 inclusi) è avvenuto il lancio, calcoli se era fuorigioco oppure no. Per valutare la posizione di un giocatore sull'asse Y al decimo di secondo d, con $1 \leqslant d \leqslant 9$ è possibile usare la formula seguente.

$$y(d) = (y_{finale} - y_{iniziale}) * \frac{d}{10} + y_{iniziale}$$

In particolare, se la applichiamo alle posizioni dei giocatori in figura, e riportate nella seguente tabella:

Giocatore	Posizione iniziale	Posizione finale
Attaccante 1	$(21, 27)$	$(27, 18)$
Attaccante 2	$(51, 24)$	$(45, 15)$
Difensore 1	$(24, 12)$	$(21, 21)$
Difensore 2	$(42, 12)$	$(42, 21)$

possiamo verificare che, nel caso di $d = 7$, l'attaccante 2 è in fuorigioco (si trova ad $y = 17, 7$, mentre i due difensori sono entrambi a $y = 18, 3$. Invece, ad esempio, nel caso di $d = 5$, i difensori (entrambi a $y = 16, 5$) tengono in gioco entrambi gli attaccanti.

Dati di input

Il file `input.txt` contiene $1 + A + D$ righe. La prima riga contiente tre interi separati da spazio: d, il decimo di secondo in cui avviene il lancio, A, il numero di attaccanti, e D, il numero di difensori. Le successive A righe contengono, per ognuno degli attaccanti, quattro interi separati da spazio: le coordinate iniziali e finali. Le ultimi D righe contengono, per ognuno dei difensori, quattro interi separati da spazio: le coordinate iniziali e finali.

Dati di output

Nel file `output.txt` dovrai stampare un solo carattere, indicante se c'era o meno fuorigioco; i valori ammessi sono:

- `F`: fuorigioco;
- `R`: azione regolare.

Assunzioni

- $1 \leqslant A, D \leqslant 3$
- Come da regolamento del calcio, se attaccante e difensore sono alla stessa altezza (intesa come coordinata y) al momento del lancio, la posizione è regolare.
- $0 \leqslant X \leqslant 84$, $0 \leqslant Y \leqslant 40$ per le posizioni di tutti i giocatori nel campo.

Esempi di input/output

File `input.txt`	**File** `output.txt`

`7 2 2`
`21 27 27 18`
`51 24 45 15`
`24 12 21 21`
`42 12 42 21`
	`F`

File `input.txt`	**File** `output.txt`

`5 2 2`
`21 27 27 18`
`51 24 45 15`
`24 12 21 21`
`42 12 42 21`
	`R`

2.2 Descrizione della soluzione

Per risolvere il problema bisogna decidere se almeno un attaccante è oltre la linea dell'ultimo difensore. Il codice si calcola la coordinata y di ogni attaccante, e la confronta con il minimo corrente (per gli attaccanti), aggiornandolo se necessario. Subito dopo calcola la coordinata y di ogni difensore, e la confronta con il minimo corrente (per i difensori). Infine confronta i due minimi, e stampa il messaggio opportuno (F oppure R).

2.3 Codice della soluzione (C++)

```cpp
#include <iostream>
#include <fstream>
#include <algorithm>

using namespace std;

int d, A, D;
int xi, yi, xf, yf;
float YA = 100, YD = 100;

int main()
{
    ifstream in("input.txt");
    ofstream out("output.txt");
    in >> d >> A >> D;
    for (int i = 0; i < A; i++)
    {
        in >> xi >> yi >> xf >> yf;
        float y = (float)(yf-yi)*d/10.0 + yi;
        if (y < YA) YA = y;
    }
    for (int i = 0; i < D; i++)
    {
        in >> xi >> yi >> xf >> yf;
        float y = (float)(yf-yi)*d/10.0 + yi;
        if (y < YD) YD = y;
    }
    if (YA < YD)
        out << "F" << endl;
    else
        out << "R" << endl;
    return 0;
}
```

3 Cannoniere (`cannoniere`) [Difficoltà D=1]

3.1 Descrizione del problema

Vincere la classifica dei cannonieri è sempre una grande soddisfazione per un attaccante, e in particolare vincerla ai mondiali è la massima aspirazione! Il vostro compito è quello di calcolare il vincitore della classifica cannonieri, a partire da un elenco di tutti i marcatori, in ordine sparso, in tutte le partite. Ogni giocatore è rappresentato da un numero intero, che lo identifica, e in ogni riga del file ci sono due interi: il numero che rappresenta il giocatore e il numero di reti che ha segnato. Per esempio, se il file contiene le seguenti righe (la prima riga contiene un solo intero, il numero di righe rimanenti del file):

```
7
23 2
11 1
67 3
45 1
11 2
23 1
11 1
```

il vostro programma dovrà stampare, in una riga,

```
11 4
```

ovvero l'identificativo del capocannoniere e il numero totale dei suoi gol.

Dati di input

Come detto in precedenza, nel file `input.txt` sono presenti $N + 1$ righe di testo: nella prima c'è un singolo numero intero positivo N che ci dice quanti sono le linee dell'elenco dei marcatori; le restanti N righe del file contengono due coppie di interi (positivi): G, il numero che rappresenta il giocatore, e R il numero di reti che ha segnato (in una data partita).

Dati di output

Nel file `output.txt` dovrai stampare due interi positivi: il numero che rappresenta il cannoniere, e il numero di reti totali che ha segnato.

Assunzioni

- $1 \leqslant N \leqslant 50$.
- $1 \leqslant G \leqslant 100$.

Esempi di input/output

File `input.txt`	File `output.txt`
7 23 2 11 1 67 3 45 1 11 2 23 1 11 1	11 4

File `input.txt`	File `output.txt`
5 10 3 9 1 10 1 7 2 10 1	10 5

3.2 Descrizione della soluzione

In questo caso, sfruttando il fatto che il numero che rappresenta il giocatore è minore di 100, l'idea per risolvere il problema è la seguente: alloco un array di dimensione 100, in cui ogni elemento rappresenta il numero di gol segnato dal giocare con l'identificativo corrispondente; inizializzo i valory dell'array a 0 e poi, per ogni riga letta dal file, incremento il numero di gol del giocatore corrispondente. Alla fine della lettura del file, scansiono l'array alla ricerca del massimo, e lo stampo in output, insieme con l'indice dell'array.

3.3 Codice della soluzione (C++)

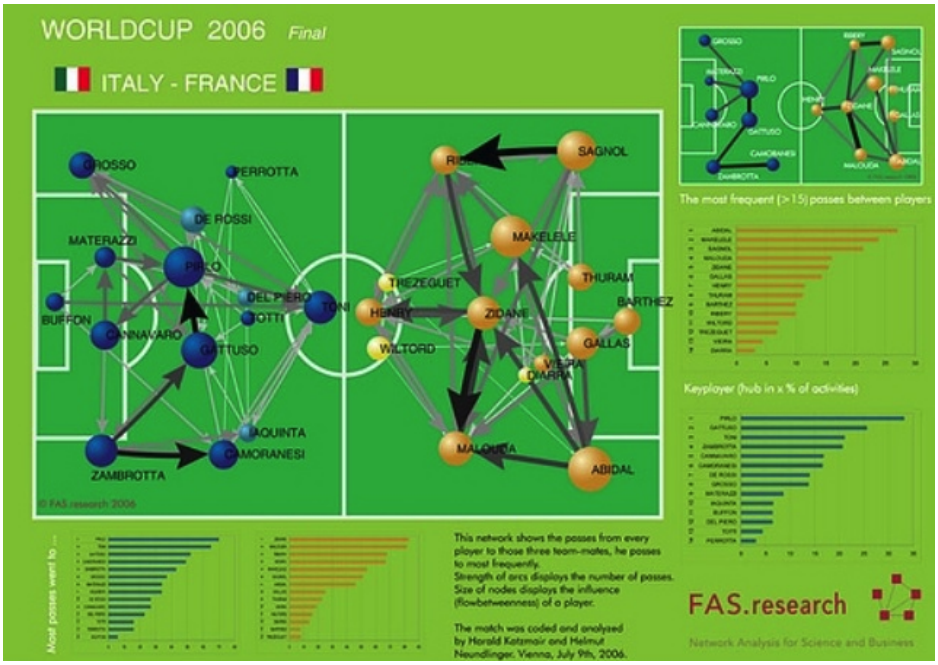
```cpp
#include <iostream>
#include <fstream>

using namespace std;

int N;
int calciatori[101];

int main()
{
    ifstream in("input.txt");
    ofstream out("output.txt");
    in >> N;
    for (int i = 0; i < N; i++)
    {
        int g, r;
        in >> g >> r;
        calciatori[g] += r;
    }
    int cannoniere = 0;
```

```
21      for (int i = 0; i <= 100; i++)
22          if (calciatori[i] > calciatori[cannoniere])
23              cannoniere = i;
24      out << cannoniere  << " " << calciatori[cannoniere] << endl;
25  }
```

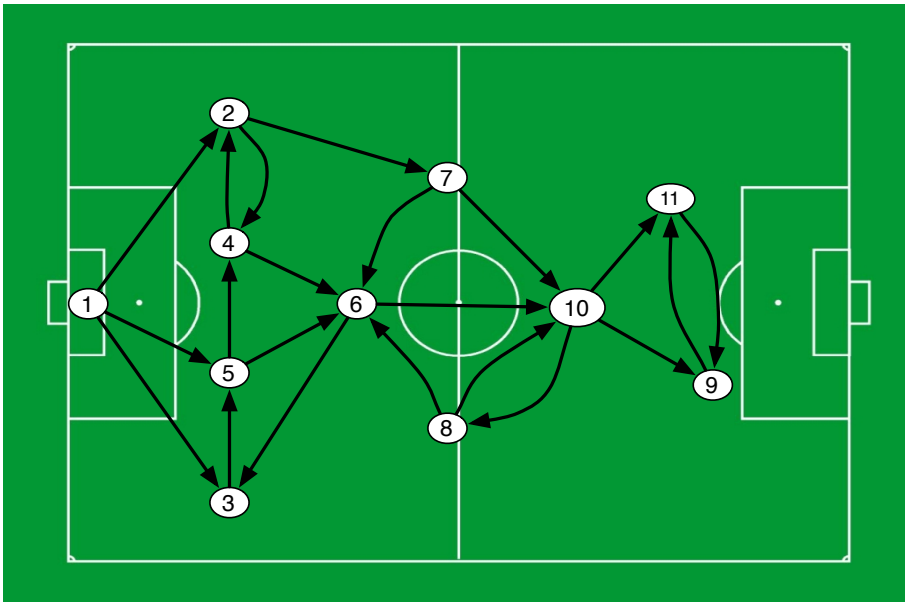
4 Fulcro del gioco (`fulcrodelgioco`) [Difficoltà D=3]

4.1 Descrizione del problema

Una partita di calcio può essere analizzata (anche) in funzione della rete di passaggi che vengono effettuati tra i giocatori. Per esempio, qui sotto vediamo l'analisi della finale dei mondiali 2006: il verso delle frecce rappresenta la direzione dei passaggi, e la dimensione delle frecce è proporzionale al numero di passaggi tra due giocatori.



Prandelli, il CT della nazionale italiana, sta studiando il modo di pressare i giocatori avversari quando la palla è in possesso del portiere (avversario). Come dati a disposizione abbiamo lo schema dei passaggi tra i giocatori, e siamo interessati a capire quale sia il giocatore da marcare per impedire al maggior numero di giocatori di ricevere il pallone. Ad esempio, nella figura qui sotto, si vede che se si impedisce al 10 di prendere palla, non ci sono modi di farla arrivare al numero 8, al 9 e all'11. Possiamo pensare alla marcatura come la rimozione del giocatore dal campo: noi siamo interessati a capire chi sia il giocatore avversario che, se rimosso (mediante marcatura) danneggi maggiormente la squadra avversaria, come numero di giocatori che non riescono ad essere raggiunti dal pallone!



Il vostro compito è quello di scrivere un programma che aiuti Prandelli a determinare quale giocatore avversario sia il fulcro del gioco. Ad esempio, nello schema qui sopra (ricordandosi che non si può rimuovere il portiere):

- rimuovendo il numero 2 il numero 7 non è più raggiungibile;
- rimuovendo un solo giocatore qualsiasi, scelto tra i numeri 3, 4, 5, 6, 7, 8, 9, 11 non ci sono conseguenze;
- rimuovendo il numero 10 i numeri 8, 9 e 11 non sono più raggiungibili.

Dati di input

Come detto in precedenza, nel file `input.txt` $M + 1$ righe di testo: la prima riga contiene M, il numero di linee di passaggio (ovvero le frecce nella figura!) tra i giocatori. Le successive M linee contengono due interi A e B, a denotare che il giocatore A passa la palla al giocatore B.

Dati di output

Nel file `output.txt` dovrai stampare un solo intero: il numero del fulcro del gioco della squadra avversaria. Se ci sono due o più giocatori ugualmente importanti (ovvero tali che rimuovendoli non è raggiungibile lo stesso numero di giocatori) restituire quello con il numero di maglia più piccolo.

Assunzioni

- $1 \leqslant A, B \leqslant 11$.

Esempi di input/output

File `input.txt`	File `output.txt`
21 1 2 1 5 1 3 3 5 5 4 4 2 2 4 4 6 5 6 6 3 2 7 7 6 6 10 8 6 8 10 7 10 10 8 10 9 10 11 9 11 11 9	10

4.2 Descrizione della soluzione

Per risolvere il problema bisogna saper fare una visita di un grafo. In particolare, nella soluzione qui sotto, si utilizza una funzione `visita` che prende come parametro il giocatore da *escudere* e restituisce il numero di giocatori che non sono raggiunti. Con questa funzione, per risolvere il problema basta chiamarla per ogni giocatore (diverso dal portiere), memorizzando il massimo.

4.3 Codice della soluzione (C++)

```cpp
#include <fstream>
#include <list>
#include <stack>

using namespace std;

int M;

struct giocatore{
    list <int> passaggi;
};

giocatore giocatori[12];

bool visitato[12];
```

```
16
17  int visita(int n)
18  {
19      int contatore = 0;
20      fill(visitato, visitato + 12,false);
21      visitato[n] = true; //escludo il giocatore n dal gioco
22      stack <int> pila;
23      pila.push(1); //parto sempre dal portiere
24      while(!pila.empty())
25      {
26          int corrente = pila.top();
27          pila.pop();
28          if (visitato[corrente] == false)
29          {
30              visitato[corrente] = true;
31              list <int>::iterator i;
32              for (i = giocatori[corrente].passaggi.begin();
33                  i!=giocatori[corrente].passaggi.end();i++)
34                  pila.push(*i);
35          }
36      }
37      for (int i = 2; i<12; i++)
38          if (visitato[i] == false)
39              contatore++;
40      return contatore;
41  }
42
43
44  int main()
45  {
46      ifstream in("input.txt");
47      ofstream out("output.txt");
48      in >> M;
49      for (int i = 0; i < M; i++)
50      {
51          int a, b;
52          in >> a >> b;
53          giocatori[a].passaggi.push_back(b);
54      }
55      int numero_maglia = 0, giocatori_esclusi = -1;
56      for (int i = 2; i < 12; i++)
57      {
58          int temp = visita(i);
59          if (temp > giocatori_esclusi)
60          {
61              giocatori_esclusi = temp;
62              numero_maglia = i;
63          }
64      }
65      out << numero_maglia << endl;
66      return 0;
67  }
```

5 Pilota il sorteggio (`sorteggio`) [Difficoltà D=3]

5.1 Descrizione del problema

Siamo tutti rimasti senza parole avendo visto il risultato del sorteggio dei mondiali: Francia e Svizzera sono finite in un girone facile, mentre l'Italia avrà da soffrire in un girone con Uruguay e Inghilterra. A pensar male, come hanno fatto molti giornali e tanti tifosi italiani, sembrerebbe che lo svizzero Blatter, presidente della FIFA, e il francese Platini, presidente dell'UEFA, si siano messi d'accordo e siano riusciti a pilotare il sorteggio.

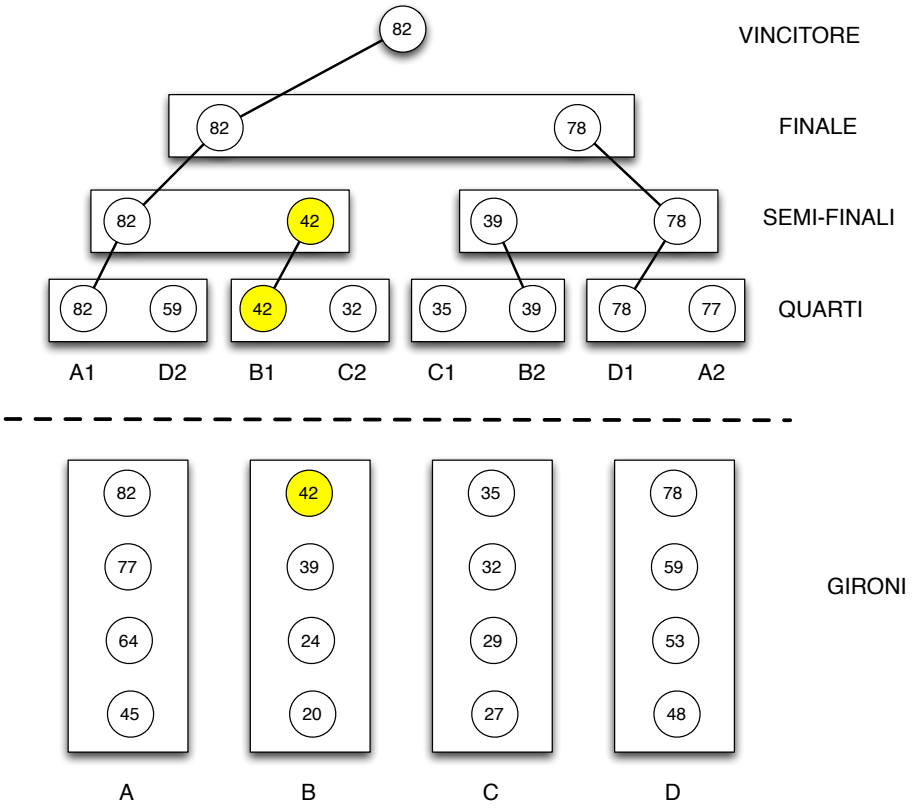
A questo proposito, se fossimo stati noi italiani in grado di pilotare il sorteggio, che gironi avremmo disegnato? Quanto è complicato riuscire a spingere la propria squadra il più in alto possibile? Assumendo che ogni squadra abbia un valore, rappresentato da un numero intero, che rappresenta la propria forza, e che quando due squadre si incontrano vinca quella con il valore maggiore, come bisogna disporre le squadre nei gironi per assicurarsi che la propria squadra arrivi più in alto possibile?

Il tuo compito è quello di scrivere un programma che, ricevute in ingresso m=16, 32 o 64 squadre, le piazzi in $m/4$ gironi, ognuno con 4 squadre. Di ogni girone passano le prime due squadre, che accedono alla fase ad eliminazione diretta. Le squadre vengono messe nel tabellone in questo modo: in ordine di girone, abbiamo da sinistra verso destra le squadre che sono arrivate prime nel proprio girone, e da destra verso sinistra le squadre che sono arrivate seconde. Per esempio, in un torneo con 16 squadre, divise in quattro gironi che chiameremo A, B, C e D, il tabellone vedrà in ordine, i seguenti scontri: A1 - D2, B1 - C2, C1 - B2 e D1 - A2. La prima semifinale sarà tra le vincenti delle prime due partite, e la seconda tra le vincenti delle altre due.

Ad esempio, supponendo che una squadra abbia un valore di forza pari a 42, e le altre quindici squadre abbiano valori, in ordine, rispettivamente pari a:

```
82, 78, 77, 64, 59, 53, 48, 45, 39, 35, 32, 29, 27, 24, 20
```

è possibile vedere, nella figura qui sotto, come si riesca a far arrivare la squadra di valore 42 fino alle semifinali, nonostante ci siano ben otto squadre più forti di lei.



Nella figura qui sopra vediamo, infatti, una possibile divisione delle squadre nei gironi (la nostra squadra ha il fondo giallo) e, dopo i gironi, il tabellone del torneo.

Dati di input

Come detto in precedenza, nel file `input.txt` sono presenti 2 righe di testo: nella prima c'è un singolo intero m che vale 16, 32 o 64 e rappresenta il numero di squadre che partecipano al torneo. Nella seconda riga ci sono m interi che rappresentano la forza delle squadre che partecipano al torneo: la tua squadra, che deve arrivare più in alto possibile, è la prima, seguita, in ordine di forza, dalle altre.

Dati di output

Nel file `output.txt` dovrai stampare un solo carattere che rappresenta quanto in alto ti riesce di spingere, con una disposizione opportuna delle squadre nei gironi, la tua squadra; i valori ammessi sono:

- `V`: vincitrice del torneo;
- `F`: finalista;
- `H`: semi-finalista;
- `Q`: quarti di finale;
- `O`: ottavi (solo per tornei con 32 o 64 squadre!);

- `S`: sedicesimi (solo per tornei con 64 squadre!);
- `G`: gironi - la tua squadra non riesce ad accedere alla fase ad eliminazione diretta!

Assunzioni

- m vale 16, 32 o 64.
- La forza di ogni squadra è un intero positivo compreso tra 1 e 100 (inclusi). Le forze delle squadre sono tutte distinte: non esistono due squadre con la stessa forza.

Esempi di input/output

File `input.txt`	**File** `output.txt`
16 42 82 78 77 64 59 53 48 45 39 35 32 29 27 24 20	H

File `input.txt`	**File** `output.txt`
16 80 88 78 77 63 62 61 45 43 39 35 32 29 27 24 20	F

File `input.txt`	**File** `output.txt`
16 11 88 73 71 69 61 55 48 47 41 37 34 31 27 26 22	G

5.2 Descrizione della soluzione

Per risolvere questo problema il modo più semplice, visto che i dati in ingresso si riducevano ai valori di forza di 16, 32 o 64 squadre, tra cui una speciale che è quella che dobbiamo far salire il più in alto possibile nel tabellone. In pratica, i due parametri che ci interessano sono il numero di squadre (16, 32 o 64) e in che posizione è *nostra* squadra se ordiniamo la squadra per valore di forza. Questi due dati sono sufficienti a rispondere alla domanda. A questo punto, per rispondere alla domanda la strada complicata è quella di scrivere il codice che risolve effettivamente il problema, mentre la strada semplice è quella di risolvere il problema su carta, e codificare le risposte all'interno del codice. Il codice che segue, definito *quick and dirty* da Alessandro Bugatti, che ne è l'autore, rappresenta proprio questa idea, ed è una possibilità da non trascurare durante le gare. Presentiamo anche una seconda versione del codice, che codifica la risposta all'interno di una stringa.

5.3 Codice della soluzione (C++)

```cpp
#include <fstream>
#include <algorithm>

using namespace std;

int M;
int squadre[65];

int main()
```

```
{
    ifstream in("input.txt");
    ofstream out("output.txt");
    int valore_mia_squadra;
    in >> M;
    in >> valore_mia_squadra;
    squadre[0] = valore_mia_squadra;
    for (int i = 1; i < M; i++)
    {
        int valore;
        in >> valore;
        squadre[i] = valore;
    }
    sort(squadre, squadre+M);
    reverse(squadre, squadre+M);
    int posizione_in_classifica;
    for (int i = 0; i < M; i++)
    {
        if (squadre[i] == valore_mia_squadra)
            posizione_in_classifica = i+1;
    }
    //out << posizione_in_classifica << endl;
    if (M == 16)
    {
        if (posizione_in_classifica >= 15)
            out << "G" << endl;
        else if (posizione_in_classifica >= 11)
            out << "Q" << endl;
        else if (posizione_in_classifica >= 4)
            out << "H" << endl;
        else if (posizione_in_classifica >= 2)
            out << "F" << endl;
        else
            out << "V" << endl;
    }
    if (M == 32)
    {
        if (posizione_in_classifica >= 31)
            out << "G" << endl;
        else if (posizione_in_classifica >= 27)
            out << "O" << endl;
        else if (posizione_in_classifica >= 20)
            out << "Q" << endl;
        else if (posizione_in_classifica >= 6)
            out << "H" << endl;
        else if (posizione_in_classifica >= 2)
            out << "F" << endl;
        else
            out << "V" << endl;
    }
    if (M == 64)
    {
        if (posizione_in_classifica >= 63)
            out << "G" << endl;
        else if (posizione_in_classifica >= 59)
```

```
            out << "S" << endl;
        else if (posizione_in_classifica >= 52)
            out << "O" << endl;
        else if (posizione_in_classifica >= 38)
            out << "Q" << endl;
        else if (posizione_in_classifica >= 10)
            out << "H" << endl;
        else if (posizione_in_classifica >= 2)
            out << "F" << endl;
        else
            out << "V" << endl;
    }
    return 0;
}
```

5.4 Una soluzione un po' più corta (C)

```
#include <stdio.h>

char sol16[] = "VFFHHHHHHHQQQQGG";
char sol32[] = "VFFFFHHHHHHHHHHHHHHQQQQQQQOOOOGG";
char sol64[] = "VFFFFFFFHHHHHHHHHHHHHHHHHHHHHHHHHHHHQQQQQQQQQQQQQQQOOOOOOOOSSSSGG";

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    int M, me, i, idx;
    idx = 0;
    scanf("%d", &M);
    scanf("%d", &me);
    for(i=1; i<M; i++){
        int a;
        scanf("%d", &a);
        if(a>me) idx++;
    }
    if(M==16) printf("%c\n", sol16[idx]);
    else if(M==32) printf("%c\n", sol32[idx]);
    else if(M==64) printf("%c\n", sol64[idx]);
    return 0;
}
```