Representação Grafo

Matriz de Adjacência

Inicialização

```
for(i=1;i<=n;i++){
            for(j=1;j<=n;j++){
              g[i][j] = 0;
            }
}
```

Arestas

```
for(i=1;i<=m;i++){
            scanf("%d %d",&a,&b);
            g[a][b] = g[b][a] = 1;
}
```

Lista de Adjacência

Inicialização

```
for(i=1;i<=n;i++){
   d[i]=0;
}
```

Arestas

```
for(i=1;i<=m;i++){
       scanf("%d %d",&a,&b);
       g[a][d[a]++]=b;
       g[b][d[b]++]=a;
}
```

Busca em Profundidade

Inicialização

```
 for(i=1;i<=n;i++)
   marc[i] = 0;
 bp(1);
```

Procedimento

```
void bp(int i){
 int j;
 marc[i] = 1;
 Para cada vértice não marcadoj adjacente a i
     bp(j)
}
```

APlicações

- Conexidade
- Bipartido
- Acíclico

http://br.spoj.pl/problems/OBIDOMIN/

http://br.spoj.pl/problems/DENGUE/

Dicionário Estático

```
#define MAXNUM 101

#define MAXTAM 25

char dict[MAXNUM][MAXTAM];

int icont;
```

```
int find(char * s){

      int i;

      for(i=0;i<icont;i++)

        if(strcmp(dict[i],s)==0)

          return i;

      return -1;

}
```

```
void insere(char *s){

      int x;

      x = find(nome);

      if( x == -1){

            strcpy(dict[icont],nome);

            return icont++;

      }else{

            return x;

      }

}
```

Complexidade

Inserção O(n)

Busca O(n)

```
map<string,int> theMap;

map<string,int>::iterator it1,it2;
```

```
for(i=1;i<=n;i++){

cin >> s1;

theMap.insert( make_pair(s1,i) );


}
```

```
for(i=1;i<=m;i++){

cin >> s1 >> s2;

it1 = theMap.find(s1);

it2 = theMap.find(s2);

g[it1->second][d[it1->second]++]=it2->second;

g[it2->second][d[it2->second]++]=it1->second;
}
```

USANDO STL (map) Problema Eleições

```
#include <stdio.h>

#include <map>

using namespace std;

map <int,int> votos;

map <int,int>::iterator it;

int main(){

      int i,n,maxv,x,maxc;

      scanf("%d",&n);

      for(i=1;i<=n;i++){

            scanf("%d",&x);

            //Complexidade O(lg n)

            votos[x]++;

      }

      maxc = votos.begin()->first;

      maxv = votos.begin()->second;

      for(it=votos.begin(); it != votos.end(); it++){

            if(it->second > maxv ){

                  maxv = it->second;

                  maxc = it->first;

            }

      }

      printf("%d\n",maxc);

      return 0;

}
```

Problem A
Pebble Solitaire

Input: standard input
Output: standard output
Time Limit: 1 second

Pebble solitaire is an interesting game. This is a game where you are given a board with an arrangement of small cavities, initially all but one occupied by a pebble each. The aim of the game is to remove as many pebbles as possible from the board. Pebbles disappear from the board as a result of a move. A move is possible if there is a straight line of three adjacent cavities, let us call them **A**, **B**, and **C**, with **B** in the middle, where **A** is vacant, but **B** and **C** each contain a pebble. The move constitutes of moving the pebble from **C** to **A**, and removing the pebble in **B** from the board. You may continue to make moves until no more moves are possible.

In this problem, we look at a simple variant of this game, namely a board with twelve cavities located along a line. In the beginning of each game, some of the cavities are occupied by pebbles. Your mission is to find a sequence of moves such that as few pebbles as possible are left on the board.

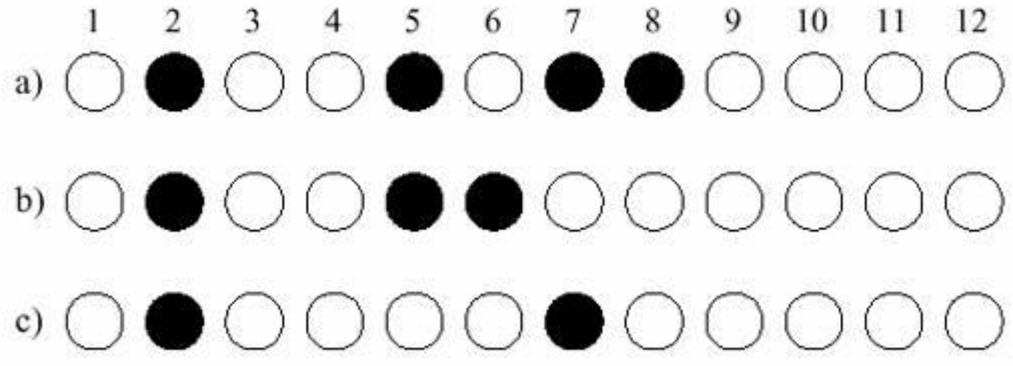

Fig 1. In a) there are two possible moves, namely 8 → 6, or 7 → 9. In b) the result of the 8 → 6 move is depicted, and again there are two possible moves, 5 → 7, or 6 → 4. Making the first of these results in c), from which there are no further moves.

Input

The input begins with a positive integer **n** on a line of its own. Thereafter **n** different games follow. Each game consists of one line of input with exactly twelve characters, describing the twelve cavities of the board in order. Each character is either **'-'** or **'o'** (The fifteenth character of English alphabet in lowercase). A **'-'** (minus) character denotes an empty cavity, whereas a **'o'** character denotes a cavity with a pebble in it. As you will find in the sample that there may be inputs where no moves is possible.

Output

For each of the n games in the input, output the minimum number of pebbles left on the board possible to obtain as a result of moves, on a row of its own.

Sample Input���������������������������� �� Output for Sample Input

5	1
---oo-------	2
-o--o-oo----	3
-o----ooo---	12
oooooooooooo	1
oooooooooo-o	

Swedish National Contest

http://br.spoj.pl/problems/JUNINA/