

**Solució de l'Examen d'Ordinador EDA**

11/01/2021

**Torn 1****Proposta de solució al problema 1**

```
#include <iostream>
#include <vector>
#include <stdlib.h>

using namespace std;

bool compatible(int left, int right, int d) {
    return abs(left - right) ≤ d;
}

void write_balanced_permutations(int n, int d, vector<int>& partial_sol, vector<bool>& used) {
    if (int(partial_sol.size()) == n) {
        cout << "(";
        for (int i = 0; i < n; ++i) cout << (i == 0 ? "" : ",") << partial_sol[i];
        cout << ")" << endl;
    } else {
        for (int k = 1; k ≤ n; ++k) {
            if (!used[k]) {
                if (partial_sol.size() == 0 || compatible(partial_sol.back(), k, d)) {
                    partial_sol.push_back(k);
                    used[k] = true;
                    write_balanced_permutations(n, d, partial_sol, used);
                    used[k] = false;
                    partial_sol.pop_back();
                }
            }
        }
    }
}

void write_balanced_permutations(int n, int d) {
    vector<bool> used(n+1, false);
    vector<int> partial_sol;
    write_balanced_permutations(n, d, partial_sol, used);
}

int main(){
    int n, d;
    cin >> n >> d;
    write_balanced_permutations(n, d);
}
```

**Seqüències equilibrades**

X40596\_ca

Diem que una seqüència de nombres és *d-equilibrada* si la diferència en valor absolut entre qualsevol parell de nombres consecutius és com a molt *d*. Formalment,  $(x_1, x_2, \dots, x_n)$  és *d-equilibrada* si per a tot  $1 \leq i < n$  es compleix que  $|x_i - x_{i+1}| \leq d$ . Feu un programa que, donat un enter  $n \geq 1$  i un enter  $d \geq 0$ , escrigui totes les seqüències *d-equilibrades* que es poden formar reordenant la seqüència  $(1, 2, \dots, n)$ .

**Entrada**

L'entrada consisteix en un enter  $n \geq 1$  seguit d'un altre enter  $d \geq 0$ .

**Sortida**

Escriviu totes les seqüències *d-equilibrades* que es poden formar reordenant la seqüència  $(1, 2, \dots, n)$ . Podeu escriure les seqüències en qualsevol ordre.

**Exemple d'entrada 1**

3 1

**Exemple de sortida 1**(1, 2, 3)  
(3, 2, 1)**Exemple d'entrada 2**

4 2

**Exemple de sortida 2**(1, 2, 3, 4)  
(1, 2, 4, 3)  
(1, 3, 2, 4)  
(1, 3, 4, 2)  
(2, 1, 3, 4)  
(2, 4, 3, 1)  
(3, 1, 2, 4)  
(3, 4, 2, 1)  
(4, 2, 1, 3)  
(4, 2, 3, 1)  
(4, 3, 1, 2)  
(4, 3, 2, 1)**Exemple d'entrada 3**

1 0

**Exemple de sortida 3**

(1)

## Proposta de solució al problema 2

```

#include <iostream>
#include <vector>

using namespace std;

void dfs(int u, const vector<vector<int>>& g, vector<int>& vis) {
    if (vis[u]) return;
    vis[u] = true;
    for (int v : g[u])
        dfs(v, g, vis);
}

int main() {
    int n, u, v, m;
    while (cin >> n >> u >> v >> m) {
        vector<vector<int>> g(n);
        vector<vector<int>> i(n); // Inverted graph (flipped edges)
        while (m--) {
            int x, y;
            cin >> x >> y;
            g[x].push_back(y);
            i[y].push_back(x);
        }
        vector<int> fwd(n, false);
        dfs(u, g, fwd);
        if (not fwd[v]) cout << 0 << endl;
        else {
            vector<int> bwd(n, false);
            dfs(v, i, bwd);
            int sum = 0;
            for (int x = 0; x < n; ++x) {
                if (fwd[x] and bwd[x]) cout << x << endl; Vèrtexs intermedis
                sum += (fwd[x] and bwd[x]);
            }
            cout << sum - 2 << endl;
        }
    }
}

```

X34137\_ca

Donats un graf dirigit i dos vèrtexs  $u$  i  $v$  diferents, calculeu quants vèrtexs  $x$  que no siguin ni  $u$  ni  $v$  hi ha tals que existeix algun camí d' $u$  a  $v$  que passa per  $x$ .

### Entrada

L'entrada consisteix en diversos casos. Cada cas comença amb  $n, u, v$  i  $m$ , seguit d' $m$  parells diferents  $x y$ , amb  $x \neq y$ , que indiquen un arc que va d' $x$  a  $y$ . Suposeu  $2 \leq n \leq 10^4$ ,  $0 \leq m \leq 10n$ , i que els vèrtexs es numeren entre 0 i  $n - 1$ .

### Sortida

Per a cada cas, escriu la quantitat de vèrtexs pels quals es pot passar anant des d' $u$  fins a  $v$  pel camí que sigui.

### Pista

Per a cada cas, la solució esperada bàsicament només fa dos recorreguts, cadascun en el graf adequat.

#### Exemple d'entrada

```

9 7 4 9
8 7
7 1
7 2
7 5
1 3
2 3
3 4
6 4
4 0

2 0 1 0
3 0 1 2
1 2
2 0

4 0 2 3
0 2
2 3
3 0

```

#### Exemple de sortida

```

3
0
0
1

```

**Torn 2****Proposta de solució al problema 1**

```
#include <iostream>
#include <vector>

using namespace std;

bool compatible(int left, int mid, int right, int n) {
    return left + right ≤ 2*mid;
}

void write_no_well_permutations (int n, vector<int>& partial_sol, vector<bool>& used) {
    if (int( partial_sol .size ()) == n) {
        cout << "(";
        for (int i = 0; i < n; ++i) cout << (i == 0 ? "" : ",") << partial_sol [ i ];
        cout << ")" << endl;
    }
    else {
        for (int k = 1; k ≤ n; ++k) {
            if (not used[k]) {
                if ( partial_sol .size () ≤ 1 or
                    compatible( partial_sol [ partial_sol .size ()-2], partial_sol .back (), k, n)){
                    partial_sol .push_back(k);
                    used[k] = true;
                    write_no_well_permutations (n, partial_sol ,used);
                    used[k] = false;
                    partial_sol .pop_back ();
                }
            }
        }
    }
}
```

```
void write_no_well_permutations (int n) {
    vector<bool> used(n+1, false);
    vector<int> partial_sol ;
    write_no_well_permutations (n, partial_sol ,used);
}
```

```
int main(){
    int n;
    cin >> n;
    write_no_well_permutations (n);
}
```

**Seqüències sense pou****X41088\_ca**

Diem que una seqüència de nombres té un pou si conté una tripla de nombres consecutius tals que els dos extrems sumen més del doble del nombre del mig.

Formalment,  $(x_1, x_2, \dots, x_n)$  té un pou si existeix almenys una  $i$  amb  $1 \leq i < n - 1$  tal que  $x_i + x_{i+2} > 2 \cdot x_{i+1}$ .

Feu un programa que, donat un enter  $n \geq 1$ , escrigui totes les seqüències que no tinguin cap pou que es poden formar reordenant la seqüència  $(1, 2, \dots, n)$ .

**Entrada**

L'entrada consisteix en un enter  $n \geq 1$ .

**Sortida**

Escriviu totes les seqüències que no tenen cap pou que es poden formar reordenant la seqüència  $(1, 2, \dots, n)$ . Podeu escriure les seqüències en qualsevol ordre.

**Exemple d'entrada 1**

3

**Exemple de sortida 1**

(1, 2, 3)
(1, 3, 2)
(2, 3, 1)
(3, 2, 1)

**Exemple d'entrada 2**

2

**Exemple de sortida 2**

(1, 2)
(2, 1)

**Exemple d'entrada 3**

4

**Exemple de sortida 3**

(1, 2, 3, 4)
(1, 3, 4, 2)
(1, 4, 3, 2)
(2, 3, 4, 1)
(2, 4, 3, 1)
(4, 3, 2, 1)

**Exemple d'entrada 4**

1

**Exemple de sortida 4**

(1)
-----

## Proposta de solució al problema 2

```

#include <iostream>
#include <vector>
#include <queue>
#include <limits.h>

using namespace std;
using P = pair<int,int>;
```

**const int** oo = INT\_MAX;

```

int dijkstra (const vector<vector<P>>& g, int x, int y) {
    int n = g.size ();
    vector<int> dist(n, +oo);
    priority_queue<P, vector<P>, greater<P> q;
    dist [x] = 0;
    q.push({0, x});
    while (not q.empty ()) {
        auto t = q.top ();
        q.pop ();
        int u = t.second;
        int d = t.first ;
        if (u == y) return dist [y];
        if (d == dist [u]) {
            for (auto p : g[u]) {
                int v = p.second;
                int l = p.first ;
                int d2 = max(dist [u], l);
                if (d2 < dist [v]) {
                    dist [v] = d2;
                    q.push({d2, v});
                }
            }
        }
        return +oo;
    }
}

int main() {
    int n, m;
    while (cin >> n >> m) {
        vector<vector<P>> g(n);
        while (m--) {
            int x, y, l;
            cin >> x >> y >> l;
            g[x].push_back({l, y});
        }
        cout << dijkstra (g, 0, 1) << endl; } }
```

### Carreteres curtes

### X50299\_ca

Considereu el mapa d'un país, amb  $n$  ciutats (numerades entre 0 i  $n - 1$ ) i  $m$  carreteres unidireccionals que les connecten. Cada carretera té una certa longitud. Volem anar de la ciutat 0 a la ciutat 1. Com que viatgem amb persones que es maregen, i no volem parar a estirar les cames a mitja carretera, volem seguir el camí tal que la carretera més llarga que faci servir sigui el més curta possible. És a dir, si el camí usa  $k$  carreteres, amb longituds  $\ell_1, \dots, \ell_k$ , i  $\ell = \max(\ell_1, \dots, \ell_k)$ , volem que  $\ell$  sigui tan petita com sigui possible.

#### Entrada

L'entrada consisteix en diversos casos. Cada cas comença amb  $n$  i  $m$ , seguits d' $m$  triplets  $x \ y \ \ell$ , amb  $x \neq y$ , indicant una carretera que va de  $x$  a  $y$  de longitud  $\ell$ . Suposeu  $2 \leq n \leq 10^4$ ,  $1 \leq m \leq 10n$ , que no hi ha més d'una carretera d' $x$  a  $y$  en aquest mateix ordre, que les longituds es troben entre  $1$  i  $10^5$ , i que sempre hi ha algun camí entre 0 i 1.

#### Sortida

Per a cada cas, escriviu la longitud màxima de les carreteres del millor camí possible. La segona línia de l'exemple de sortida es correspon al camí  $0 \rightarrow 4 \rightarrow 2 \rightarrow 1$ , el qual té una carretera (la  $0 \rightarrow 4$ ) de longitud màxima 80.

#### Pista

Considereu una variant de l'algorisme de Dijkstra.

#### Exemple d'entrada

```

2 2
0 1 100000
1 0 42

5 6
0 1 90
0 4 80
4 1 60
0 2 100
2 1 60
4 2 70
```

#### Exemple de sortida

```

100000
80
```

**Solució de l'Examen d'Ordinador EDA**

08/06/2021

**Proposta de solució al problema 1**

```
#include <iostream>
#include <vector>

using namespace std;

void write_words(const vector<string>& words, vector<bool>& used, vector<int>& sol) {
    if (sol.size() == words.size()) {
        for (int idx : sol) cout << words[idx];
        cout << endl;
    }
    else {
        for (uint i = 0; i < words.size(); ++i) {
            if (not used[i] and (sol.size() == 0 or words[sol.back()].back() != words[i][0])) {
                used[i] = true;
                sol.push_back(i);
                write_words(words, used, sol);
                sol.pop_back();
                used[i] = false;
            }
        }
    }
}

void write_words(const vector<string>& words) {
    int n = words.size();
    vector<bool> used(n, false);
    vector<int> sol; // solution will be a sequence of ints,
                    // representing the indices in words
    write_words(words, used, sol);
}
```

```
int main(){
    int n;
    cin >> n;
    vector<string> words(n);
    for (string& s : words) cin >> s;

    write_words(words);
}
```

**Paraules concatenades**

X67572\_ca

Donades  $n$  paraules, escriviu totes les maneres de concatenar-les sense que hi hagi dues lletres adjacents iguals.

**Entrada**

L'entrada consisteix en una  $n$  entre 1 i 8, seguida d' $n$  paraules, cadascuna amb entre 1 i 10 lletres minúscules. Cap paraula té dues lletres adjacents iguals.

**Sortida**

Escriviu totes les maneres correctes de concatenar les  $n$  paraules, una per línia.

**Informació sobre el corrector**

Podeu escriure les solucions d'aquest exercici en qualsevol ordre.

**Exemple d'entrada 1**

```
2
z
abcdz
```

**Exemple de sortida 1**

```
zabcdz
```

**Exemple d'entrada 2**

```
3
adefghijkabxyzc
adefghijkacxyzc
badefghijkacxyzc
bcxyzcadefghijk
cxyzcadefghijkab
cxyzcbadefghijk
```

**Exemple de sortida 2**

```
adefghijkabxyzc
adefghijkacxyzc
badefghijkacxyzc
bcxyzcadefghijk
cxyzcadefghijkab
cxyzcbadefghijk
```

**Exemple d'entrada 3**

```
2
ie
ei
```

**Exemple de sortida 3**

```
bab
bab
```

**Exemple d'entrada 4**

```
3
a
b
b
```

**Exemple de sortida 4**

```
bab
bab
```

## Proposta de solució al problema 2

```

#include <iostream>
#include <vector>
#include <queue>
#include <limits>

using namespace std;

typedef pair<int,int> Cell;
typedef vector<vector<int>> Matrix;

const int inf = numeric_limits<int>::max();
const vector<pair<int,int>> dirs = {{0,-1},{0,1},{1,0}, {-1,0}};

bool on_border (Cell& c, int n){
    return c.first == 0 || c.second == 0 || c.first == n-1 || c.second == n-1;
}

int cost (Matrix& M) {
    int n = M.size();
    Cell center = {n/2,n/2};
    Matrix dist(n,vector<int>(n,inf));
    Matrix removed(n,vector<int>(n,0));

    priority_queue<pair<int,Cell>,vector<pair<int,Cell>>,greater<pair<int,Cell>>> Q;
    dist [center . first ][ center . second ] = M[center . first ][ center . second ];
    Q.push({dist [center . first ][ center . second ], center });

    while (not Q.empty()) {
        Cell c = Q.top().second;
        Q.pop();
        if (not removed[c . first ][c . second ]) {
            removed[c . first ][c . second ] = 1;
            if (on_border(c,n)) return dist [c . first ][c . second ];
            for (auto& d : dirs ) {
                Cell neigh = {c . first + d . first , c . second + d . second };
                if (dist [c . first ][c . second ] + M[neigh . first ][neigh . second ] < dist [neigh . first ][neigh . second ]) {
                    dist [neigh . first ][neigh . second ] = dist [c . first ][c . second ] + M[neigh . first ][neigh . second ];
                    Q.push({dist [neigh . first ][neigh . second ], neigh });
                }
            }
        }
    }
    return -1;
}

// Note that we can guarantee that neigh is always inside the
// grid. This is because we can only get out of the grid by visiting a
// neighbour of a border cell, but the algorithm stops as soon as we

```

```
// find such a cell

int main () {
    int n;
    while (cin >> n) {
        Matrix M(n, vector<int>(n));
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; ++j)
                cin >> M[i][j];
        cout << cost(M) << endl;
    }
}
```

**Camí en un tauler****X13208\_ca**

Considereu un tauler  $n \times n$ , amb  $n$  senar. Des de cada casella, ens podem moure a qualsevol de les (com a molt) quatre caselles adjacents horitzontalment o verticalment. Cada casella té un cert cost positiu que cal pagar quan s'hi passa. Calculeu el cost mínim d'anar des del centre del tauler fins a qualsevol casella de la perifèria.

**Entrada**

L'entrada consisteix en diversos casos, cadascun amb  $n$ , seguida d'una matriu  $n \times n$ . Podeu suposar que  $n$  és un nombre senar entre 1 i 499, i que tots els costs són nombres enters entre 1 i 1000.

**Sortida**

Per a cada cas, escriviu el cost mínim d'anar des del centre del tauler fins a qualsevol casella de la vora del tauler.

**Exemple d'entrada**

```
1
42
3
1 2 3
4 5 6
7 8 9

9
999 1 999 999 999 999 999 999 999
999 2 999 6 5 4 3 2 999
999 3 999 7 999 999 999 1 999
999 4 999 8 999 999 999 9 999
999 5 999 9 1 999 999 8 999
999 6 999 999 999 999 999 7 999
999 7 999 999 999 999 999 6 999
999 8 9 1 2 3 4 5 999
999 999 999 999 999 999 999 999 999
```

**Exemple de sortida**

```
42
7
136
```

**Solució de l'Examen d'Ordinador EDA**

07/01/2022

**Torn 1****Proposta de solució al problema 1**

```

#include <iostream>
#include <vector>
#include <limits>
using namespace std;
int infinite = numeric_limits<int>::max();

// Returns size of connected component
// PRE: marked[u] = false
int size_con_component(const vector<vector<int>>& G, vector<bool>& marked, int u) {
    marked[u] = true;
    int total = 1;
    for (int v : G[u])
        if (not marked[v]) total += size_con_component(G,marked,v);
    return total;
}

// Returns (min,max)
pair<int,int> size_con_component(const vector<vector<int>>& G) {
    int n = G.size();
    pair<int,int> result = { infinite, 0 };
    vector<bool> marked(n, false);
    for (int u = 0; u < n; ++u) {
        if (not marked[u]) {
            int s = size_con_component(G,marked,u);
            result . first = min(result . first ,s);
            result . second = max(result . second ,s);
        }
    }
    return result;
}

int main() {
    int n, m;
    while (cin >> n >> m) {
        vector<vector<int>> G(n);
        for (int i = 0; i < m; ++i) {
            int x, y;
            cin >> x >> y;
            G[x].push_back(y);
            G[y].push_back(x);
        }
        pair<int,int> p = size_con_component(G);
        cout << p.first << " " << p.second << endl;
    }
}

```

**Component connex mínim i màxim**

X68591\_ca

Donat un graf no dirigit, calculeu-ne el nombre de vèrtexs tant del component connex més petit com del component connex més gran.

**Entrada**

L'entrada consisteix en diversos grafs. Cadascun comença amb el nombre de vèrtexs  $n$  i el nombre d'arestes  $m$ , seguits d' $m$  parells  $x$   $y$  que indiquen una aresta entre els vèrtexs  $x$  i  $y$ . Suposeu  $1 \leq n \leq 10^4$ ,  $0 \leq m \leq 5n$ , que els vèrtexs es numeren entre 0 i  $n - 1$ , i que no hi ha arestes repetides ni de tipus  $x$   $x$ .

**Sortida**

Per a cada graf, escriviu les mides mínima i màxima dels seus components connexos.

**Exemple d'entrada**

```

3 1 0 2
1 0
6 5 0 1 4 2 2 1 5 3 4 0

```

**Exemple de sortida**

```

1 2
1 1
2 4

```

## Proposta de solució al problema 2

```

#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

bool is_vowel (char c) {
    return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u';
}

// partial_sol[0...k-1] already filled
void write_words (const vector<char>& letters, vector<char> partial_sol, int k) {
    if (k == int( partial_sol . size ()) ) {
        for (char c: partial_sol ) cout << c;
        cout << endl;
    }
    else {
        for (char c : letters ) {
            if (not is_vowel(c) || k == 0 || not is_vowel( partial_sol [k-1])) {
                partial_sol [k] = c;
                write_words( letters , partial_sol ,k+1);
            }
        }
    }
}

int main ( ){
    int n, m;
    while (cin >> n >> m) {
        vector<char> letters(m);
        for (int i = 0; i < m; ++i) cin >> letters [i];
        sort ( letters .begin (), letters .end ());
        vector<char> partial_sol(n);
        write_words( letters , partial_sol , 0);
        cout << string(10,'-') << endl;
    }
}

```

---

**Runes**
**X18624\_ca**

Excavacions recents han descobert una antiga llengua ja extinta. A partir de les runes trobades, s'ha deduït que hi havia vocals i consonants, i que se'n podien formar totes les paraules, amb una sola excepció: No hi podia haver dues o més vocals consecutives. Per exemple, amb les dues vocals a i e i la consonant b es podien formar 11 paraules amb tres lletres: aba, abb, abe, bab, bba, bbb, bbe, beb, eba, ebb, ebe.

Quines paraules de mida  $n$  es podien formar amb  $m$  lletres donades?

**Entrada**

L'entrada consisteix en diversos casos, cadascun amb  $n$  i  $m$ , seguida d' $m$  lletres minúscules diferents. Podeu assumir  $n \geq 1$ ,  $2 \leq m \leq 26$ , i que cada cas té almenys una vocal i una consonant.

**Sortida**

Per a cada cas, escriviu en ordre lexicogràfic totes les paraules de longitud  $n$  que es poden construir amb les  $m$  lletres donades. Escriviu una línia amb 10 guions després de cada cas.

**Exemple d'entrada**

3 3
aeb
1 2
az
3 2
pe

**Exemple de sortida**

aba
abb
abe
bab
bba
bbb
bbe
beb
eba
ebb
ebe
-----
a
z
-----
epe
epp
pep
ppe
ppp
-----

**Torn 2****Proposta de solució al problema 1**

```
#include <iostream>
#include <vector>
#include <queue>
#include <limits>
using namespace std;
const int UNDEF = -1;

int farthest (const vector<vector<int>>& G) {
    int n = G.size ();
    vector<int> dist(n,UNDEF);
    queue<int> Q;
    Q.push(0);
    dist [0] = 0;
    while (not Q.empty()) {
        int u = Q.front ();
        Q.pop();
        for (int v : G[u]) {
            if (dist [v] == UNDEF)
                dist [v] = dist [u] + 1;
            Q.push(v);
        }
    }
}
```

// This postprocessing could have been avoided by updating the max  
// distance every time we compute the distance for a new vertex. We  
// do it this way for simplicity and readability of the code.

```
int result = 0;
for (int v = 1; v < n; ++v)
    if (dist [v] != UNDEF and dist[v] > dist[result])
        result = v;
return result ;
}
```

```
int main() {
    int n, m;
    while (cin >> n >> m){
        vector<vector<int>> G(n);
        for (int i = 0; i < m; ++i) {
            int x, y;
            cin >> x >> y;
            G[x].push_back(y);
            G[y].push_back(x);
        }
        cout << farthest (G) << endl;
    }
}
```

**Vèrtex més allunyat**

X83283\_ca

Donat un graf no dirigit, calculeu el vèrtex més allunyat del vèrtex 0.

**Entrada**

L'entrada consisteix en diversos grafs. Cadascun comença amb el nombre de vèrtexs  $n$  i el nombre d'arestes  $m$ , seguits d' $m$  parells  $x\ y$  que indiquen una aresta entre els vèrtexs  $x$  i  $y$ . Suposeu  $1 \leq n \leq 10^4$ ,  $0 \leq m \leq 5n$ , que els vèrtexs es numeren entre 0 i  $n - 1$ , i que no hi ha arestes repetides ni de tipus  $x\ x$ .

**Sortida**

Per a cada graf, escriviu el vèrtex més allunyat del vèrtex 0. En cas d'empat, escolliu el vèrtex més petit. Ignoreu els vèrtexs als quals no es pot arribar des de 0.

**Exemple d'entrada**

3	2	0	2	0	1								
1	0												
7	6	0	1	4	2	6	3	2	1	2	5	4	0

**Exemple de sortida**

1
0
5

## Proposta de solució al problema 2

```

#include <iostream>
#include <vector>

using namespace std;

vector<char> letters = {'x', 'y', 'z'};

void write_words (int n, int c, vector<char>& partial_sol, int consec, int k) {
    if (k == n) {
        for (char aux : partial_sol ) cout << aux;
        cout << endl;
    }
    else {
        for (int i = 0; i < 3; ++i) {
            bool repeated = (k > 0 and partial_sol [k-1] == letters [i]);
            int new_consec = (repeated ? consec + 1 : 1);
            if (new_consec ≤ c) {
                partial_sol [k] = letters [i];
                write_words(n,c, partial_sol ,new_consec,k+1);
            }
        }
    }
}

int main () {
    int n, c;
    while (cin >> n >> c) {
        vector<char> partial_sol (n);
        write_words(n,c, partial_sol ,0,0);
        cout << string(20,'-') << endl;
    }
}

```

### Paraules amb x, y i z

X20680\_ca

En aquest problema considerem paraules de mida  $n$ , formades només amb les lletres 'x', 'y' i 'z', i sense més de  $c$  lletres iguals consecutives. Feu un programa que escrigui totes les paraules que compleixen aquestes restriccions.

#### Entrada

L'entrada consisteix en diversos casos, cadascun amb una  $n$  entre 1 i 15 i una  $c$  entre 1 i  $n$ .

#### Sortida

Per a cada cas, escriui en ordre alfàbetí totes les paraules de longitud  $n$  formades amb 'x', 'y' i 'z' que no contenen  $c + 1$  lletres iguals consecutives. Escriui una línia amb 20 guions al final de cada cas.

#### Exemple d'entrada

```

1 1
2 2
3 1

```

#### Exemple de sortida

```

x
y
z
-----
xx
xy
xz
yx
yy
yz
zx
zy
zz
-----
xyx
xyz
xzx
xzy
yxy
yxz
yzx
yzy
zxy
zxz
zyx
zyz
-----

```

## Solució de l'Examen d'Ordinador EDA

### Proposta de solució al problema 1

```
#include <iostream>
#include <map>

using namespace std;

int main() {
    int n;
    while (cin >> n) {

        // Read proposals and create map: author to proposal
        map<string,string> author2proposal;
        for (int i = 0; i < n; ++i) {
            string person, proposal ;
            cin >> person >> proposal;
            author2proposal [proposal] = person;
        }

        // Read all votes and create map: (person,proposal) to bool
        int m;
        cin >> m;
        map<pair<string,string>,bool> all_votes; // pair is (person,proposal)
        for (int i = 0; i < m; ++i) {
            string name, vote, proposal ;
            cin >> name >> vote >> proposal;
            all_votes [{name,proposal}] = vote == "SI";
        }

        // Count votes for every proposal and store in map: proposal to (votes SI, votes NO)
        map<string,pair<int,int>> votes_for_proposal ; // pair is (votes SI,votes NO)
        for (auto& elem : all_votes ) {
            if (elem.second) ++ votes_for_proposal [elem. first .second ]. first ;
            else ++ votes_for_proposal [elem. first .second ].second;
        }

        // Write result
        for (auto& p : votes_for_proposal )
            if (p.second. first > p.second. second)
                cout << p.first << ", de " << author2proposal[p. first ] << ", que ha guanyat "
                << p.second. first << "-" << p.second.second << endl;
            cout << string(10,'-') << endl;
    }
}
```

La Laura està organitzant alguna activitat a la FME, ja sigui del Club d'Esports, de la Sortida de la Dèla, de Festes FME, etc. Per descomptat, hi ha  $n$  persones que no hi estan d'accord, així que cadaquest formula una proposta, i s'organitza un sistema de vots perquè tothom pugui votar sobre totes les propostes. Si algú no vota una proposta voldrà dir que li és indiferent, i si algú vota una mateixa proposta dos o més cops només se li computarà l'últim vot sobre la proposta.

La Laura implementarà les propostes que tinguin estrictament més SIs que NOs, ignorant els indiferents. Podeu dir quines propostes seran?

#### Entrada

L'entrada consisteix en diversos casos. Cada cas comença amb  $n$ , seguida d' $n$  parells amb un nom i una proposta. Tots són strings diferents entre si formats només amb ' ' i lletres minúscules. Segueix el nombre de vots  $m$ , seguit d' $m$  triplets nom, SI/NO, proposta. Tant el nom com la proposta existeixen. Podeu suposar  $1 \leq n \leq 100$  i  $1 \leq m \leq 1000$ .

#### Sortida

Per a cada cas, escriviu en ordre lexicogràfic les propostes que s'implementaran, junt amb els vots obtinguts. Escriviu una línia amb 10 guions al final de cada cas.

#### Exemple d'entrada

```
4
luis que_el_club_de_runners_sigui_oficial
javier que_les_rotacions_de_la_gimnasta_siguin_un_quadrat_llati
fontana estar_amb_el_seus_amics_a_1_equip
miguel que_el_seu_equip_es_digui_team_tryhards
5
fontana SI estar_amb_el_seus_amics_a_1_equip
javier NO que_el_seu_equip_es_digui_team_tryhards
miguel NO que_el_seu_equip_es_digui_team_tryhards
luis SI que_el_seu_equip_es_digui_team_tryhards
miguel SI que_el_seu_equip_es_digui_team_tryhards
1
jordi qualsevol_bestiesa
1
jordi NO qualsevol_bestiesa
2
anna estudiar
ivet jugar
2
anna SI jugar
ivet SI jugar
4
a z
b y
c x
d w
0
d SI y
a SI y
b SI w
b NO y
c SI y
b SI w
a SI z
c SI z
a NO z
b SI y
```

#### Exemple de sortida

```
estar_amb_el_seus_amics_a_1_equip, de fontana, que ha guanyat 1-0
que_el_seu_equip_es_digui_team_tryhards, de miguel, que ha guanyat 2-1
-----
jugar, de ivet, que ha guanyat 2-0
w, de d, que ha guanyat 1-0
y, de b, que ha guanyat 4-0
-----
```

## Proposta de solució al problema 2

```
#include <iostream>
#include <vector>

using namespace std;

int first_occurrence (double x, const vector<double>& v, int l, int r) {
    if (l > r) return -1;
    else {
        int m = (l+r)/2;
        if (v[m] == x) {
            int p_left = first_occurrence (x,v,l,m-1);
            if (p_left == -1) return m;
            else return p_left ;
        }
        else if (v[m] > x) {
            int p_left = first_occurrence (x,v,l,m-1);
            if (p_left == -1) {
                if (m+1 ≤ r and v[m+1] == x) return m+1;
                else return -1;
            }
            else return p_left ;
        }
        else { // v[m] < x
            if (m-1 ≥ l and v[m-1] == x) return m-1;
            return first_occurrence (x,v,m+1,r);
        }
    }
}

int first_occurrence (double x, const vector<double>& v){
    return first_occurrence (x,v,0,int(v.size ())-1);
}
```

```
int main() {
    int n;
    while (cin >> n) {
        vector<double> V(n);
        for (int i = 0; i < n; ++i) cin >> V[i];
        int t;
        cin >> t;
        while (t--) {
            double x;
            cin >> x;
            cout << first_occurrence (x, V) << endl;
        }
    }
}
```

### Primera aparició en vector quasi ordenat

X77076\_ca

Escriviu una funció eficient

`int first_occurrence (double x, const vector<double>& v);`  
que retorna la posició de la primera aparició de  $x$  dins del vector  $v$ . Si  $x$  no apareix a  $v$ , retorna un -1.

#### Precondició

El vector  $v$  està “quasi ordenat” en ordre no decreixent, en el sentit que hi pot haver com a molt un parell d’índexos  $i$  i  $j$  tals que  $0 \leq i < j < n$  i  $V[i] > V[j]$ .

#### Observació

Podeu definir funcions auxiliars si us calen.

#### Observació

Només cal enviar el procediment demandat; el programa principal serà ignorat.

**Solució de l'Examen d'Ordinador EDA**

09/01/2023

**Torn 1****Proposta de solució al problema 1**

```

#include <iostream>
#include <map>
using namespace std;

int main() {
    map<string, int> M;
    char c;
    while (cin >> c) {
        if (c == 'N') {
            cout << "number: " << M.size() << endl;
        }
        else if (c == 'D') {
            string nif;
            int money;
            cin >> nif >> money;
            M[nif] += money;
        }
        else if (c == 'Q') {
            string nif;
            cin >> nif;
            auto it = M.find(nif);
            cout << (it == M.end() ? -1 : it->second) << endl;
        }
        else if (c == 'P') {
            bool primer = true;
            for (auto x : M)
                if ((x.first[7] - '0')%2 == 0) {
                    if (primer) primer = false;
                    else cout << ' ';
                    cout << x.first ;
                }
            cout << endl;
        }
        else { // c == 'L'
            if (M.size() < 1) cout << "NO LAST NIF" << endl;
            else {
                auto it = M.end();
                --it;
                cout << it->first << ' ' << it->second << endl;
            }
        }
    }
}

```

## Proposta de solució al problema 2

```
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

vector<pair<int,int>> dirs = {{2,1},{2,-1},{-2,1},{-2,-1},
{1,2},{1,-2},{-1,2},{-1,-2}};

bool ok_cell (const vector<vector<char>>& T, int i, int j) {
    return i >= 0 and i < T.size() and j >= 0 and j < T[0].size() and T[i][j] != 'X';
}

int pastanaga_propera (const vector<vector<char>>& T, int ini_i, int ini_j ) {
    int n = T.size(), m = T[0].size();
    queue<pair<int,int>> Q;
    vector<vector<int>> dist(n, vector<int>(m,-1));
    Q.push({ini_i, ini_j });
    dist[ini_i][ini_j] = 0;
    while (not Q.empty()) {
        int i = Q.front().first;
        int j = Q.front().second;
        Q.pop();
        for (auto& d : dirs){
            int n_i = i + d.first;
            int n_j = j + d.second;
            if (ok_cell(T,n_i,n_j) and dist[n_i][n_j] == -1) {
                dist[n_i][n_j] = dist[i][j] + 1;
                if (T[n_i][n_j] == 'p') return dist[n_i][n_j];
                Q.push({n_i,n_j });
            }
        }
    }
    return -1;
}

int main ( ){
    int n, m;
    while (cin >> n >> m){
        vector<vector<char>> T(n,vector<char>(m));
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < m; ++j) cin >> T[i][j];
        int ini_i, ini_j ;
        cin >> ini_i >> ini_j ;
        int result = pastanaga_propera(T, ini_i - 1, ini_j - 1);
        if (result == -1) cout << "no" << endl;
        else cout << result << endl;
    }
}
```

### El cavall afamat

Examen extraordinari d'Algorísmia, FME (2013-07-03)

P17866\_ca

Un cavall es troba sobre un tauler d'escacs. El cavall només es pot moure segons les regles habituals (és a dir, modificant en dues unitats una de les seves coordenades, i en una unitat l'altra coordenada, per a un total de vuit possibles moviments), sempre que no surti del tauler ni visiti cap casella amb un obstacle. Quin és el mínim nombre de passos que ha de fer per arribar a una pastanaga?

### Entrada

L'entrada consisteix en diversos casos. Cada cas comença amb el nombre de files  $f$  i el nombre de columnes  $c$  del tauler. Tant  $f$  com  $c$  estan entre 3 i 200. Segueixen  $f$  files amb  $c$  caràcters cadascuna. Una 'p' indica una pastanaga, una 'X' indica un obstacle, i un punt indica una posició lliure. Cada cas acaba amb la posició (fila i columna) inicial del cavall. La casella més a l'esquerra de la primera fila és la (1, 1). La posició inicial sempre està lliure.

### Sortida

Per a cada cas, si el cavall pot arribar a alguna pastanaga, escriu el mínim nombre de passos. Altres, escriu "no".

### Exemple d'entrada

```
3 3
...
...
1 1
```

```
4 6
XXXXXX
X...XX
XXXX.X
pX.XX
Z..3
```

```
3 4
XXX
XXX.
XXXX
2 4
```

```
3 4
ppXX
P..P
PPP.
3 4
```

### Exemple de sortida

```
1
4
no
no
```

**Torn 2****Proposta de solució al problema 1**

```

#include <iostream>
#include <map>
using namespace std;

struct Info {
    string code;
    int price ;
};

int main() {
    map<int, Info> M;
    char c;
    while (cin >> c) {
        if (c == 'n') {
            cout << "num: " << M.size() << endl;
        }
        else if (c == 'u') {
            string code;
            int length , price ;
            cin >> code >> length >> price;
            M[length] = { code , price };
        }
        else if (c == 'q') {
            int length ;
            cin >> length;
            auto it = M.find(length );
            cout << (it == M.end() ? -1 : it->second.price) << endl;
        }
        else if (c == 'p') {
            cout << string(10, '-') << endl;
            for (auto x : M) cout << x.second.code << ' ' << x.first << ' ' << x.second.price << endl;
            cout << string(10, '*') << endl;
        }
        else { // c == 's'
            if (M.size() < 2) cout << "no" << endl;
            else {
                auto it = M.begin();
                ++it;
                cout << it->second.code << ' ' << it->first << ' ' << it->second.price << endl;
            }
        }
    }
}

```

Feu un programa que, donat un mapa amb tresors i obstacles, digui a quina distància es troba, de tots els tresors accessibles des d'una posició inicial donada, el segon tresor més llunyà. Els moviments permeten són horizontals o verticals, però no diagonals. Si cal, es pot passar per sobre dels tresors.

## Solucions d'Exàmens d'Ordinador

### Proposta de solució al problema 2

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;

vector<pair<int,int>> dirs = {{1,0}, {-1,0}, {0,1}, {0,-1}};

bool ok_cell (const vector<vector<char>>& T, int i, int j) {
    return i >= 0 and i < T.size() and j >= 0 and j < T[0].size() and T[i][j] != 'X';
}

int segon_mes_llunya (const vector<vector<char>>& T, int ini_i, int ini_j ) {
    vector<int> distancies_tresors ;
    int n = T.size(), m = T[0].size();
    queue<pair<int,int>> Q;
    vector<vector<int>> dist(n, vector<int>(m,-1));
    Q.push({ini_i, ini_j});
    dist[ini_i][ini_j] = 0;
    while (not Q.empty()) {
        int i = Q.front().first;
        int j = Q.front().second;
        Q.pop();
        for (auto& d : dirs){
            int n_i = i + d.first;
            int n_j = j + d.second;
            if (ok_cell(T,n_i,n_j) and dist[n_i][n_j] == -1) {
                dist[n_i][n_j] = dist[i][j] + 1;
                if (T[n_i][n_j] == 't') distancies_tresors .push_back(dist[n_i][n_j]);
                Q.push({n_i,n_j});
            }
        }
    }
    if (distancies_tresors .size() < 2) return -1;
    else return distancies_tresors [distancies_tresors .size()-2];
}

int main ( ){
    int n, m;
    cin >> n >> m;
    vector<vector<char>> T(n, vector<char>(m));
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) cin >> T[i][j];
    int ini_i, ini_j;
    cin >> ini_i >> ini_j;
    int result = segon_mes_llunya(T, ini_i - 1, ini_j - 1);
    if (result == -1) cout << "no es pot arribar a dos o mes tresors" << endl;
    else cout << "segona distància maxima:" << result << endl;
}
```

#### Entrada

L'entrada comença amb el nombre de files  $n > 0$  i de columnes  $m > 0$  del mapa. Segueixen  $n$  files amb  $m$  caràcters cadascuna. Un punt indica una posició buida, una 'X' indica un obstacle, i una 't' indica un tresor. Finalment, un parell de nombres  $i:j$  indiquen la fila i columna inicials (ambdues començant en 1) des de les quals cal començar a buscar tresors. Podeu suposar que  $i$  està entre 1 i  $n$ , que  $j$  està entre 1 i  $m$ , i que la posició inicial sempre està buida.

#### Sortida

Escriviu el nombre mínim de passos des de la posició inicial fins al segon tresor més llunyà. Si no es pot arribar a dos o més tresors, cal indicar-ho.

#### Exemple d'entrada 1

```
7 6
...t...
....XXX...
...t...
tX...X.
.X...Xt
...XX...
....t...
5 3
```

#### Exemple de sortida 1

segona distància maxima: 5

#### Exemple d'entrada 2

```
4 10
...t...
....X...
....X.t.
XXXXXX.X...
.....X.t
4 3
```

#### Exemple de sortida 2

no es pot arribar a dos o mes tresors

#### Exemple d'entrada 3

```
5 7
.....
....X...
...X.X...
...X.XX...
...X.XX
5 5
```

#### Exemple de sortida 3

segona distància maxima: 19

#### Exemple d'entrada 4

```
1 3
t.t
1 2
```

#### Exemple de sortida 4

segona distància maxima: 1

**Solució de l'Examen d'Ordinador EDA**

07/06/2023

**Proposta de solució al problema 1**

```

#include <iostream>
#include <vector>
#include <queue>
#include <limits>

using namespace std;

vector<pair<int,int>> dirs = { {1,2}, {-1,2}, {1,-2}, {-1,-2},
{2,1}, {2,-1}, {-2,1}, {-2,-1}};

static const int inf = numeric_limits<int>::max();

bool pos_ok (const vector<vector<char>>& T, int i, int j) {
    return (i ≥ 0 and i < int(T.size ()) and j ≥ 0 and j < int(T[0].size ()) and T[i][j] ≠ 'a');
}

pair<int,double> cerca (const vector<vector<char>>& T, int ini_i, int ini_j ) {
    int flors = 0;
    int suma_dist = 0;
    vector<vector<int>> dist(T.size(), vector<int>(T[0].size (), inf));
    queue<pair<int,int>> Q;
    Q.push({ini_i , ini_j });
    dist [ini_i ][ini_j ] = 0;
    while (not Q.empty()) {
        pair<int,int> pos = Q.front ();
        Q.pop();
        for (auto& d : dirs ) {
            int n_i = pos . first + d. first ;
            int n_j = pos . second + d. second;
            if (pos_ok(T,n_i ,n_j ) and dist [n_i ][n_j ] == inf) {
                dist [n_i ][n_j ] = dist [pos . first ][pos . second ] + 1;
                Q.push({n_i ,n_j });
                if (T[n_i ][n_j ] == 'F') {
                    ++flors;
                    suma_dist += dist [n_i ][n_j ];
                }
            }
        }
        if (flors ≠ 0) return {flors ,double(suma_dist)/flors };
        else return {0,0};
    }

    int main ( ){
        cout.setf(ios :: fixed );

```

```

cout.precision(4);
int n, m;
cin >> n >> m;
vector<vector<char>> T(n, vector<char>(m));
int ini_i = -1, ini_j = -1;
for (int i = 0; i < n; ++i)
for (int j = 0; j < m; ++j) {
    cin >> T[i][j];
    if (T[i][j] == 'C') { ini_i = i; ini_j = j; }
}
pair<int,double> res = cerca(T, ini_i, ini_j);
if (res.first > 0) {
    cout << "flors accessibles: " << res.first << endl;
    cout << "distància mitjana: " << res.second << endl;
}
else cout << "el cavall no pot arribar a cap flor" << endl;
}

```

**I2P02. El cavall menja-flors****P39058.ca**

Un cavall, dins d'un prat rectangular, vol saber a quantes flors pot arribar, i quina és la distància mitjana a aquestes flors des de la seva posició inicial. El cavall només pot fer els salts típics dels escacs (és a dir, modificar en dues unitats una component, i en una unitat l'altra component), i no pot sortir mai del prat ni trepitjar basses d'aigua.

Feu un programa que llegeixi la descripció d'un prat, i que calculi i escrigui el nombre de flors a les quals pot arribar el cavall, i la distància mitjana a aquestes flors, mesurada per a cada flor com el nombre mínim de salts d'escacs des de la posició inicial del cavall.

**Entrada**

L'entrada comença amb el nombre de files  $n$  i de columnes  $m$  del mapa. Segueixen  $n$  files amb  $m$  caràcters cadascuna. Un punt indica una posició buida, una 'a' indica una bassa d'aigua, una 'F' indica una flor, i una 'C' indica la posició inicial del cavall. Podeu suposar que hi haurà exactament una 'C' dins del mapa.

**Sortida**

Escriviu el nombre de flors accessibles des de la posició inicial del cavall, així com la distància mitjana, amb quatre decimals. Si no es pot arribar a cap flor, cal indicar-ho. Seguiu el format dels exemples.

**Observació**

Escriviu aquestes dues línies al principi del vostre *main()*:

```

cout.setf(ios::fixed);
cout.precision(4);

```

**Exemple d'entrada 1**

```

5 5
aFaFa
Faaaf
aaCaa
FaaaF
aFaFa

```

**Exemple de sortida 1**

```

flors accessibles: 8
distància mitjana: 1.0000

```

**Exemple d'entrada 2**

```

4 6
..F..
aa.C..
.Fa.F.
F.a...

```

**Exemple de sortida 2**

```

flors accessibles: 3
distància mitjana: 2.3333

```

**Exemple d'entrada 3**

```

3 5
FaafF
CF.F.
FF.FF

```

**Exemple de sortida 3**

```

el cavall no pot arribar a cap flor

```

**Exemple d'entrada 4**

```

1 1
C

```

**Exemple de sortida 4**

```

el cavall no pot arribar a cap flor

```

**Exemple d'entrada 5**

```

12 10
Caaa.aaa.a
aa.aaa.aaa
aaaaaaaa.
aaaaaaaaa.
aaaaaaaaa.
Faaaaaaaa.a
aaaaaaaaa.
a.aaaaaaaa.
aaaaaaaaa.
aaaaaaaaa.
aaaaaaaaa.
aaaaaaa.aa
aaaaaaaaa.
a.aaaaaaa.
aaaaaaaaa.
aaaaaaa.aa
aaaaaaaaa.
a.aaa.aaa.
aaa.aaa.aa

```

**Exemple de sortida 5**

```

flors accessibles: 1
distància mitjana: 16.0000

```

## Proposta de solució al problema 2

```

#include <iostream>
#include <vector>

using namespace std;

void escriu_solicions (vector<int>& rosers, vector<int>& sol_parcial, bool consec, int idx) {
    if (idx == sol_parcial . size ()) {
        if (consec) {
            for (int x : sol_parcial ) cout << x;
            cout << endl;
        }
    } else {
        for (int r = 1; r ≤ 3; ++r) {
            if ( rosers [r] > 0 and (not consec or idx == 0 or sol_parcial [idx-1] ≠ r)) {
                sol_parcial [idx] = r;
                --rosers[r];
                escriu_solicions (rosers , sol_parcial ,consec or (idx > 0 and sol_parcial [idx-1] == r) ,idx+1);
                ++rosers[r];
            }
        }
    }
}

void escriu_solicions (int n) {
    vector<int> sol_parcial (3*n);
    vector<int> rosers(4,n); // rosers[0] no utilitzat
    escriu_solicions (rosers , sol_parcial ,false ,0);
}

int main ( ){
    int n;
    while (cin >> n){
        escriu_solicions (n);
        cout << string(10,'*') << endl;
    }
}

```

### Rosers i testos (1)

Desè Concurs de Programació de la UPC - Final (2012-09-15)

P88410\_ca

Teniu rosers de tres colors diferents. En particular, teniu  $n$  rosers de cada color. Escriviu totes les maneres de plantar tots els rosers en una filera de  $3n$  testos, un roser per test, de forma que hi hagi exactament un parell de rosers adjacents del mateix color. Els rosers del mateix color són indistingibles entre si.

#### Entrada

L'entrada consisteix en diversos casos, cadascun amb  $n$ . Assumiu  $1 \leq n \leq 6$ .

#### Sortida

Per a cada cas, escriviu totes les solicions en ordre lexicogràfic. Identifiqueu cada color usant nombres començant en 1. Escriviu una línia amb 10 asteriscs després de cada cas.

#### Exemple d'entrada

1  
2

#### Exemple de sortida

\*\*\*\*\*  
112323  
113232  
121233  
121332  
122313  
122312  
123312  
123321  
131223  
131322  
132213  
132231  
133212  
211323  
212133  
212331  
213312  
213321  
231123  
231132  
232113  
232311  
233121  
311232  
312213  
312231  
313122  
313221  
321123  
321132  
322131  
323112  
323211  
331212  
332121  
\*\*\*\*\*