Distancia Barcelona

Se garantiza que:

- Todos los puntos están en una calle, es decir, una de las dos coordenadas es múltiplo de diez.

Constraints

$1 \leq x_i, y_i, x_r, y_r \leq 10^7 \ \forall i$

$1 \leq Q \leq 10^4$

Subcasos

Subcaso 1 (10 pt):

- Todas las coordenadas son múltiplos de 10.
- $0 \leq x_i, x_r \leq 10^3$
- $10^6 \leq y_i, y_r \leq 10^6 + 10^3$

En las anteriores reuniones de la Organización Independiente de Epizootias, también conocida como la Organización Mundial de Sanidad Animal, se reunían todos en Barcelona para discutir sobre los temas más relevantes. Pero cada año tenían varios problemas.

Muchos miembros no conocen la distancia des del lugar de la reunión hasta el hotel, entonces cuando quieren coger un taxi no saben cuánto dinero llevar, está claro que no llevarán más de lo que toca ya que temen perder el dinero. Por esto y otras razones este año la reunión se realizará a distancia.

Concretamente, supondremos que Barcelona es una cuadrícula de longitud infinita de origen $(0, 0)$, con manzanas de lado 10 metros y con una diagonal de longitud también infinita que cruza la ciudad pasando por el $(0, 0)$ y $(10, 10)$. Dado que los taxistas quieren minimizar el tiempo en el trayecto, **el taxímetro no cuenta la distancia recorrida en la Diagonal**. Para la simplicidad del problema supondremos que todas las calles son bidireccionales. Por tanto la distancia Barcelona es la distancia mínima entre dos puntos recorriendo las calles de Barcelona.

Para poder hacer futuras reuniones, la organización te pide que les ayudes a calcular la distancia Barcelona des del lugar de la reunión a distintos hoteles.

Input Format

La entrada consiste en varios casos.

La primera línea de cada caso son 2 enteros x_r, y_r, las coordenadas de el lugar de la reunión, seguido de Q enteros, el número de hoteles a considerar. La siguiente línea consiste en Q parejas de enteros x_i, y_i, las coordenadas del hotel i-ésimo, separado por espacios.

Subcaso 2 (20 pt):

- Todas las coordenadas son múltiplos de 10.
- $0 \leq x_i, x_r \leq 10^4$
- $10^6 \leq y_i, y_r \leq 10^6 + 10^4$

Subcaso 3 (30 pt):

- Todos los puntos son múltiplos de 10.

Subcaso 4 (40 pt):

- No hay restricciones adicionales

Output Format

Para cada caso debes imprimir Q enteros no negativos separados por un cambio de línea al final.

El i-ésimo enteros no negativo es la distancia Barcelona entre el i-ésimo hotel y el lugar de la reunión.

Sample Input 0

```
0 1000000 2
0 1000010 10 1000010
0 1000020 2
0 1000010 10 1000010
```

Sample Output 0

```
10 20
10 20
```

Explanation 0

Sample del primer subcaso

Sample Input 1

```
0 1000000 2
0 1010000 9090 1010000
```

Sample Output 1

```
10000 19090
```

Explanation 1

Sample del segundo subcaso

Sample Input 2

```
0 0 2
10 10 20 10
```

Sample Output 2

```
0 10
```

Explanation 2

Sample del tercer subcaso, nótese que de $(0, 0)$ a $(10, 10)$ podemos ir por la diagonal, y por tanto la distancia total es 0.

Solución

```cpp
#include <iostream>

using namespace std;

//subcaso 4
const int oo = 0x3fffffff;

int d(int a, int b, int c, int d) { return abs(a - c) + abs(b - d); }

int dist(int a, int b) { return abs(b-a); }

int main() {
    int x, y;
    while(cin >> x >> y) {
        int px[2], py[2];
        px[0] = x - x%10;
        px[1] = x - x%10 + 10;

        py[0] = y - y%10;
        py[1] = y - y%10 + 10;
        int Q;
        cin >> Q;
        while(Q--) {
            int a,b;
            cin >> a >> b;
            int cx[2], cy[2];
            cx[0] = a - a%10;
            cx[1] = a - a%10 + 10;

            cy[0] = b - b%10;
            cy[1] = b - b%10 + 10;
            int ans = oo;
            if(a % 10 == 0 && a == x) ans = abs(b-y);
            if(b % 10 == 0 && b == y) ans = abs(a-x);
            for(int I = 0; I < 0xf; ++I) {
                bool i = I&1, j = I&2, k = I&4, l = I&8;
                ans = min(dist(cx[i], cy[j]) + dist(px[k],py[l]) +
                      d(a,b,cx[i],cy[j]) + d(x, y, px[k], py[l]), ans);
                ans = min(d(cx[i], cy[j], px[k], py[l]) + d(a,b,cx[i],cy[j]) +
                      d(x, y, px[k], py[l]), ans);
            }

            cout << ans;
            if(!Q) cout << endl;
            else cout << ' ';
        }
    }
}
```

// Autor de la solución: Dean Zhu (Miembro del comité organizador)

El examen muy difícil

Después de pasar una hora y media en el examen final de Matemáticas elucubrando en vano las respuestas a las preguntas tan diabólicas que han salido, decides que es hora de rendirte, inventarte las respuestas que quedan, entregar y rezar por el aprobado. Afortunadamente para ti, el examen es de tipo test y cada pregunta tiene tres posibles opciones: A, B o C y algunas de las respuestas ya las conoces. Cada acierto vale un punto y los fallos no restan. Recuerdas de los exámenes de prueba que hiciste ayer que el profesor jamás pone dos preguntas consecutivas con la misma respuesta, es decir, que si la pregunta 1 es la A, entonces la pregunta 2 no puede ser la A. Tu objetivo es rellenar el examen de tal forma que maximices el número de puntos que sacarás en el peor de los casos.

Input Format

La entrada consistirá de un entero t, seguido de t casos de prueba. Cada caso tendrá en una línea N, el número de preguntas en el examen, y en la siguiente línea un string con N caracteres. El caracter i-ésimo será 'A', 'B' o 'C' si conoces la respuesta a la pregunta i y '?' si no la conoces.

Constraints

33 puntos: $1 \leq N \leq 8, t = 10$

11 puntos: $1 \leq N \leq 10^5, t = 50$, no hay dos '?' consecutivas.

56 puntos: $1 \leq N \leq 10^5, t = 50$

Output Format

Una línea por caso con el número de puntos que puedes asegurar en el examen.

Sample Input 0

```
2
6
A??B?C
5
A???A
```

Sample Output 0

```
5
3
```

Explanation 0

En el primer caso, si respondes ABABAC, el peor de los casos es que aciertes 5 preguntas: Si la solución es ABABAC, aciertas 6 preguntas. Si la solución es ABCBAC o ACABAC, aciertas 5 preguntas. En el segundo caso, responder ACABA garantiza almenos 3 respuestas correctas, y no existe una respuesta que asegure 4 respuestas correctas.

Solución

```cpp
#include <iostream>

using namespace std;

int main() {
    int ccc;
    cin >> ccc;
    for (int cas = 0; cas < ccc; ++cas) {
        int n;
        cin >> n;
        int res = 0;
        bool primer = true;
        int cont = 0;
        char ant;
        for (int i = 0; i < n; ++i) {
            char a;
            cin >> a;
            if (primer and a == '?') continue;

            if (a == '?') ++cont;
            else if (!primer) {
                ++res;
                if (ant == a) {
                    if (cont > 1) ++res;
                    cont = 0;
                } else {
                    if (cont) ++res;
                    cont = 0;
                    ant = a;
                }
            }
            else {
                ++res;
                primer = false;
                ant = a;
                cont = 0;
            }
        }
        cout << res << endl;
    }
}
```

// Autor: Cesc Folch (Miembro del comité organizador)

Greed

Antes de la batalla contra el jefe final, vas a comprar algunos objetos para mejorar tus posibilidades. Pero esta tienda es un poco distinta a las que estás acostumbrado: hay N pilas, la i-ésima pila contiene m_i objetos y el j-ésimo objeto de la i-ésima pila suma a_{ij} unidades de ataque a tu personaje. Inicialmente solo puedes comprar el primer objeto de cada pila. Cuando compras un objeto, el siguiente en la pila se desbloquea y puedes comprarlo también. Además, todos los objetos tienen el mismo precio: 15 monedas. Tu objetivo es gastarte las C monedas que tienes con tal de maximizar tu ataque.

Input Format

La entrada consistirá en un entero t, seguido de t casos de prueba. En la primera línea de cada caso estarán N y C. A continuación vienen N líneas, la i-ésima de las cuales describe la pila i: primero viene m_i, la cantidad de objetos en esa pila, y luego m_i enteros, el j-ésimo de los cuáles es a_{ij}.

Constraints

Para todos los casos de prueba se cumple $-10^9 \leq a_{ij} \leq 10^9$.

25 puntos: $1 \leq N \leq 12, 1 \leq C \leq 150, 1 \leq m_i \leq 12, t = 50$.

25 puntos: $1 \leq N \leq 60, 1 \leq C \leq 1000, 1 \leq m_i \leq 60, t = 100$.

50 puntos: $1 \leq N \leq 250, 1 \leq C \leq 3000, 1 \leq m_i \leq 200, t = 150$.

Output Format

Una línea por caso con la máxima cantidad de ataque que se puede obtener comprando algunos objetos en la tienda.

Sample Input 0

```
3
1 29
4 1 2 3 4
2 30
2 -2 1
3 -1 -1 3
5 90
3 5 1 10
3 6 1 2
3 7 1 3
3 8 1 4
3 9 1 5
```

Sample Output 0

```
1
0
40
```

Explanation 0

En el primer caso, estás obligado a coger el primer objeto y no puedes permitirte nada más. En el segundo caso es mejor no comprar nada, si no pierdes ataque. En el tercer caso puedes comprar los tres objetos de la primera pila y el primer objeto de la tercera, cuarta y quinta pila.

Solución

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

using namespace std;

vector<vector<long long> > dp;
vector<vector<long long> > a;

long long solve(int x, int p) {
    if (p == 0) return 0;
    if (x == dp.size()) return 0;

    long long &d = dp[x][p];
    if (d == -1){
        long long sum = 0;
        d = solve(x+1, p);

        for (int i = 0; i < a[x].size() and p > 0; ++i) {
            sum += a[x][i];
            p--;
            d = max(d, solve(x+1, p)+sum);
        }
    }
    return d;
}

int main(){
    int t;
    cin >> t;
    while (t--){
        int n, c;
        cin >> n >> c;
        c /= 15;
        a = vector<vector<long long> > (n);
        dp = vector<vector<long long> > (a.size(), vector<long long> (c+1, -1));
        for (int i = 0; i < n; ++i){
            int m;
            cin >> m;
            a[i] = vector<long long> (m);
            for (int j = 0;j < m; ++j) cin >> a[i][j];
        }
        cout << solve(0, c) << endl;
    }
}
```

// Autor: Miquel Ortega (Miembro del comité organizador)

abbaaaba

Cuantas palabras de longitud n formadas por 'a' y 'b' hay que no contengan la palabra 'abbaaaba'?

Escribe el resultado módulo $10^9 + 7$

Input Format

Un entero n

Constraints

$1 \leq n \leq 10^{18}$

Hay 100 casos y cada caso vale un punto.

Output Format

Una línea con el resultado.

Sample Input 0

```
8
```

Sample Output 0

```
255
```

Solución

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

using namespace std;

typedef long long ll;
typedef vector<ll> vi;
typedef vector<vi> vvi;

//     _ a b b a a a b a
//     0 1 2 3 4 5 6 7 8
//
//     0 -> a 1 | b 0
//     1 -> a 1 | b 2
//     2 -> a 1 | b 3
//     3 -> a 4 | b 0
//     4 -> a 5 | b 2
//     5 -> a 6 | b 2
//     6 -> a 1 | b 7
//     7 -> a X | b 3

ll M[8][8] = {
```

```
  {1,1,0,0,0,0,0,0},
  {0,1,1,0,0,0,0,0},
  {0,1,0,1,0,0,0,0},
  {1,0,0,0,1,0,0,0},
  {0,0,1,0,0,1,0,0},
  {0,0,1,0,0,0,1,0},
  {0,1,0,0,0,0,0,1},
  {0,0,0,1,0,0,0,0}
};

const ll mod = 1e9+7;

inline vvi operator*(vvi A, vvi B) {
  vvi C = A;
  int n = A.size();
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
      C[i][j] = 0;
      for (int w = 0; w < n; ++w) {
        C[i][j] = (C[i][j] + A[i][w]*B[w][j])%mod;
      }
    }
  }
  return C;
}

vvi exponent(vvi S, ll e) {
  if (e == 1) return S;
  vvi K = exponent(S,e/2);
  K = K*K;
  if (e%2) K = K*S;
  return K;
}

int main() {
  vvi D(8,vi(8));
  for (int i = 0; i < 8; ++i)
    for (int j = 0; j < 8; ++j)
      D[i][j] = M[j][i];
  ll n;
  cin >> n;
  vvi R = exponent(D,n);
  ll res = 0;
  for (int i = 0; i < 8; ++i) res = (res + R[i][0])%mod;
  cout << res << endl;
}
```

// Autor: Cesc Folch (Miembro del comité organizador)