Upsolving Escola de Inverno UTFPR - dia 2

Henrique Mendes

30 de julho de 2024

Contents

1 Problema A: Extracting Pollen

1.1 Enunciado - inglês

☆ Extracting Pollen

Gym - 104555E

Spring has arrived, ushering in a season of hard work at the Swarm of Bees Company (SBC). With the blooming of N beautiful flowers in the garden, each flower boasts a certain quantity of pollen grains. The SBC enforces strict rules to keep the bees industrious in their pollen collection.

1. The first rule pertains to the amount of pollen grains collected: when a bee visits a flower, it must gather the sum of the digits in its current pollen quantity. For instance, if a bee visits a flower with 123 pollen grains, it must collect $1 + 2 + 3 = 6$ grains, leaving the flower with $123 - 6 = 117$ grains. Similarly, if the flower holds 201 grains, the bee must gather $2 + 0 + 1 = 3$ grains, leaving 198 grains remaining.
2. All bees must form a queue at the start of the day; the bee at the front of the queue must collect pollen from one of the flowers with the largest amount pollen. If a bee visits a flower with 0 grains of pollen, it collects zero grains. After collecting pollen from a flower, the bee ends its shift and returns to the hive.

Gertrude finds these rules bewildering and seeks help to determine the pollen amount she must collect when it's her turn. Getrude has amazingly sharp sight and noticed that she is currently the K-th bee in the SBC-defined order.

Input

The first line contains two integers N ($1 \le N \le 10^6$) and K ($1 \le K \le 10^9$), representing the number of flowers and Gertrude's position in the bee line, respectively. The second line contains N integers, where the i-th integer F_i ($1 \le F_i \le 10^6$ for $1 \le i \le N$) denotes the initial quantity of pollen grains of the i-th flower.

Output

Output a single integer Q representing the amount of pollen Gertrude will collect.

Examples

Input	copy	Output	copy
5 3 22 15 7 2 1		6	

Input	copy	Output	copy
3 10 21 21 21		0	

Input	copy	Output	copy
3 9 21 21 21		9	

1.2 Enunciado - português

Problema E

Extraindo Pólen

É chegada a primavera, dando início a mais uma temporada de trabalho intenso na Sociedade das aBelhas de Chapecó (SBC)! No jardim da SBC, N lindas flores floresceram, cada uma com uma certa quantidade de grãos de pólen, que serão coletados pelas árduas trabalhadoras da sociedade. Para manter o ambiente de trabalho seguro, a SBC tem regras muito estritas para a coleta do pólen, sendo elas:

1. Quando uma abelha visita uma flor, ela deve coletar uma quantidade de pólen igual à soma dos dígitos do total de pólen atualmente naquela flor. Por exemplo, se uma flor tem 123 grãos de pólen, a abelha que a visitar deve coletar $1+2+3=6$ grãos, deixando a flor com $123-6=117$ grãos. Se a flor tem 201 grãos, a abelha coletará $2+0+1=3$ grãos, deixando a flor com $201-3=198$ grãos de pólen.

2. Todas as abelhas devem formar uma fila no início do dia; aquela que estiver na primera posição da mesma deve coletar pólen de alguma flor com o maior total de pólen. Se a abelha visitar um flor com 0 grãos, ela coleta zero grãos de pólen. Após coletar o pólen de uma flor, a abelha encerra seu turno de trabalho e volta para a colmeia.

A abelha Gertrude achou essas regras muito estranhas e procurou a sua ajuda para saber quanto pólen ela irá coletar no seu turno. Gertrude tem uma visão incrível e descobriu que atualmente está na K-ésima posição da fila.

Entrada

A primeira linha contém dois inteiros N ($1 \leq N \leq 10^6$) e K ($1 \leq K \leq 10^9$), que representam o número de flores e a posição de Gertrude na fila, respectivamente. A segunda linha contém N inteiros, onde o i-ésimo inteiro F_i ($1 \leq F_i \leq 10^6$ para $1 \leq i \leq N$) denota a quantidade inicial de pólen da i-ésima flor.

Saída

Imprima um único inteiro Q, que representa a quantidade de pólen que será coletada por Gertrude.

Exemplo de entrada 1	**Exemplo de saída 1**
5 3 22 15 7 2 1	6

Explicação do exemplo 1:

A primeira abelha coletará o pólen da primeira flor, deixando-a com $22-(2+2)=18$ grãos restantes. A segunda abelha também coletará da primeira flor, deixando-a com $18-(1+8)=9$ grãos restante. Por fim, Gertrude, a terceira abelha da fila, irá coletar pólen da segunda flor, coletando um total de $1+5=6$ grãos, que será a resposta para este caso de teste.

Exemplo de entrada 2	**Exemplo de saída 2**
3 10 21 21 21	0

1.3 Dicas

1.3.1 Dica 1

Atenção quanto ao limite do valor de k ($1 \leq k \leq 10^9$). Iterar abelha por abelha é uma opção viável dentro dos limites do problema?

1.3.2 Dica 2

Atenção quanto ao limite dos valores F_i ($1 \leq F_i \leq 10^6$). Esse valor parece bem mais razoável, mas como usá-lo?

1.3.3 Dica 3

Considere um vetor de frequências.

1.4 Ideia geral da solução

Para conseguirmos uma solução viável dentro dos limites de tempo, já de cara devemos descartar a iteração abelha por abelha (10^9 operações). Ao observar que os limites dos valores de quantidade de pólen por flor são mais baixos (10^6), podemos manipular a solução para que ela dependa desse número. Para isso, criamos um vetor de frequências, contando quantas vezes o valor de cada flor aparece.
Também computamos as transições, pois há 10^6 elementos, e suas transições são calculadas em d operações, sendo d a quantidade de dígitos do número, de modo que temos algo próximo de $5 \cdot 10^6$ operações para calcular todas as transições.
Então iteramos de "cima pra baixo", começando pelas flores com mais pólen e contando quantas abelhas irão coletar flores com essa quantidade de pólen, usando o vetor de frequências. Lembre de atualizar as frequências

1.5 Código

```
#define MAXN 1123456

vi freq(MAXN), nxt(MAXN);

int sum(int n) {
    int ans = 0;
    while (n > 0) {
        ans += n % 10;
        n /= 10;
    }
    return ans;
}

void solve() {
    int n, k, x;
    cin >> n >> k;

    for (int i = 0; i < n; i++) {
        cin >> x;
        freq[x]++;
    }

    for (int i = 0; i < MAXN; i++)
        nxt[i] = i - sum(i);
```

```
    int ans = 0;
    for (int i = MAXN - 1; i >= 0; i--){
        k -= freq[i];
        freq[nxt[i]] += freq[i];
        freq[i] = 0;
        if (k <= 0) {
            ans = sum(i);
            break;
        }
    }
    cout << ans << endl;
}
```

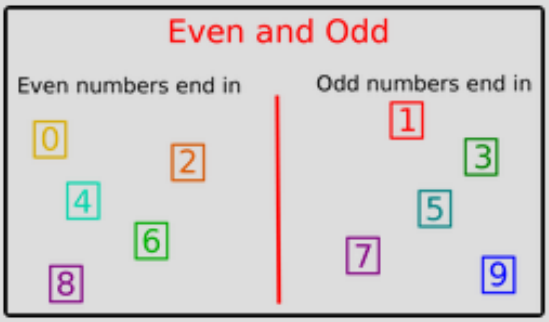
2 Problema B: Some sum

2.1 Enunciado do problema

☆ Some Sum Kattis - somesum

Your friend has secretly picked N consecutive positive integers between 1 and 100, and wants you to guess if their sum is even or odd.

If the sum must be even, output 'Even'. If the sum must be odd, output 'Odd'. If the sum could be even or could be odd, output 'Either'.



Input

The input is a single integer N with $1 \leq N \leq 10$.

Output

Output a single word. The word should be 'Even', 'Odd', or 'Either', according to the rules given earlier.

Sample 1

Input	copy	Output	copy
1		Either	

Sample 2

Input	copy	Output	copy
2		Odd	

2.2 Dicas

2.2.1 Dica 1

Note que apenas a quantidade de ímpares importa para definir a paridade da solução. Com uma quantidade ímpar de ímpares, a soma é ímpar, com uma quantidade par de ímpares, a soma é par

2.3 Código

```
void solve(){
    int n; cin >> n;
    if (n % 2 != 0) cout << "Either" << endl;
    else if (n%4 == 0) cout << "Even" << endl;
    else cout << "Odd" << endl;
}
```

Outra versão do código

```
 void solve(){
    int n; cin >> n;
    if (n & 1) cout << "Either" << endl;
    else cout << ((n%4) ? "Odd" : "Even") << endl;
}
```

3 Problema C: Valuable cards

3.1 Enunciado do problema

☆ Valuable Cards CodeForces - 1992F

In his favorite cafe Kmes once again wanted to try the herring under a fur coat. Previously, it would not have been difficult for him to do this, but the cafe recently introduced a new purchasing policy.

Now, in order to make a purchase, Kmes needs to solve the following problem: n cards with prices for different positions are laid out in front of him, on the i-th card there is an integer a_i, among these prices there is no whole positive integer x.

Kmes is asked to divide these cards into the minimum number of *bad* segments (so that each card belongs to exactly one segment). A segment is considered *bad* if it is impossible to select a subset of cards with a product equal to x. All segments, in which Kmes will divide the cards, must be *bad*.

Formally, the segment (l, r) is *bad* if there are no indices $i_1 < i_2 < \ldots < i_k$ such that $l \le i_1, i_k \le r$, and $a_{i_1} \cdot a_{i_2} \ldots \cdot a_{i_k} = x$.

Help Kmes determine the minimum number of *bad* segments in order to enjoy his favorite dish.

Input

The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases.

The first line of each set of input data gives you 2 integers n and x ($1 \le n \le 10^5, 2 \le x \le 10^5$) — the number of cards and the integer, respectively.

The second line of each set of input data contains n integers a_i ($1 \le a_i \le 2 \cdot 10^5, a_i \ne x$) — the prices on the cards.

It is guaranteed that the sum of n over all sets of test data does not exceed 10^5.

Output

For each set of input data, output the minimum number of *bad* segments.

Examples

Input	copy	Output	copy
8 6 4 2 3 6 2 1 2 9 100000 50000 25000 12500 6250 3125 2 4 8 16 5 2 1 1 1 1 1 8 6 4 3 4 3 4 3 4 3 7 12 6 11 1 3 11 10 2 10 5 2 4 4 2 4 4 4 3 1 1 7 8 4 6 5 1 2 4 1 8 27 3 9 17 26 2 20 9 3		3 2 1 1 2 1 3 3	

3.2 Dicas

3.2.1 Dica 1

Analisando os limites, não temos tempo para uma solução quadrática.

3.2.2 Dica 2

Em quanto tempo (complexidade) conseguimos computar os divisores de um número?

3.3 Ideia geral da solução

Para a solução, a ideia é realmente simular todos os divisores que podem ser gerados dentro do subset que estamos considerando.
Quando vamos adicionar um novo elemento ao set, percorremos todos os divisores, e adicionamos os novos múltiplos deles com o novo número, desde que sejam divisores também. Nesse processo de inserção de um novo divisor, verificamos se não estamos inserindo o próprio x, se estivermos, marcamos a troca para um novo set.

3.4 Código

```
void solve() {
 int n, x;
 cin >> n >> x;
 vi a(n);
 vi divisors;
 for (int d = 1; d * d <= x; d++) {
  if (x % d == 0) {
   divide[d] = 1;
   divisors.push_back(d);
   if (d * d < x) {
       divisors.push_back(x / d);
    divide[x / d] = 1;
   }
  }
 }
 for (int i = 0; i < n; i++) cin >> a[i];

 int ans = 1;
 used[1] = 1;
 vi cur = { 1 };
 for (int i = 0; i < n; i++) {
  if (!divide[a[i]])
   continue;
  vi ncur;
  bool ok = true;
  for (int d : cur)
   if (d * a[i] <= x && divide[d * a[i]] && !used[d * a[i]]) {
    ncur.push_back(d * a[i]);
```

```
    used[d * a[i]] = 1;
    if (d * a[i] == x)
     ok = false;
   }
  for (int d : ncur)
   cur.push_back(d);
  if (!ok) {
   ans++;
   for (int d : cur)
    used[d] = 0;
            cur.clear();
   used[1] = 1;
   used[a[i]] = 1;
   cur = { 1, a[i] };
  }
 }
 for (int d : divisors) {
     divide[d] = 0;
     used[d] = 0;
 }
 cout << ans << endl;
}
```

4 Problema D: Choosing Cubes

4.1 Enunciado do problema

☆ Choosing Cubes CodeForces - 1980B

Dmitry has n cubes, numbered from left to right from 1 to n. The cube with index f is his favorite.

Dmitry threw all the cubes on the table, and the i-th cube showed the value a_i ($1 \le a_i \le 100$). After that, he arranged the cubes in non-increasing order of their values, from largest to smallest. If two cubes show the same value, they can go in any order.

After sorting, Dmitry removed the first k cubes. Then he became interested in whether he removed his favorite cube (note that its position could have changed after sorting).

For example, if $n = 5$, $f = 2$, $a = [4, 3, 3, 2, 3]$ (the favorite cube is highlighted in green), and $k = 2$, the following could have happened:

- After sorting $a = [4, 3, 3, 3, 2]$, since the favorite cube ended up in the second position, it will be removed.
- After sorting $a = [4, 3, 3, 3, 2]$, since the favorite cube ended up in the third position, it will not be removed.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. Then follow the descriptions of the test cases.

The first line of each test case description contains three integers n, f, and k ($1 \le f, k \le n \le 100$) — the number of cubes, the index of Dmitry's favorite cube, and the number of removed cubes, respectively.

The second line of each test case description contains n integers a_i ($1 \le a_i \le 100$) — the values shown on the cubes.

Output

For each test case, output one line — `"YES"` if the cube will be removed in all cases, `"NO"` if it will not be removed in any case, `"MAYBE"` if it may be either removed or left.

You can output the answer in any case. For example, the strings `"YES"`, `"nO"`, `"mAyBe"` will be accepted as answers.

Examples

Input (copy)	Output (copy)
12 5 2 2 4 3 3 2 3 5 5 3 4 2 1 3 5 5 5 2 5 2 4 1 3 5 5 5 1 2 5 4 3 5 5 4 3 1 2 4 5 5 5 5	MAYBE YES NO YES YES YES MAYBE MAYBE YES YES YES NO

4.2 Dicas

4.2.1 Dica 1

Guarde o valor no índice f.

4.2.2 Dica 2

Ao ordenar o vetor, o que acontece se o k-ésimo elemento for igual ao valor do favorito? E e for maior? E se for menor?

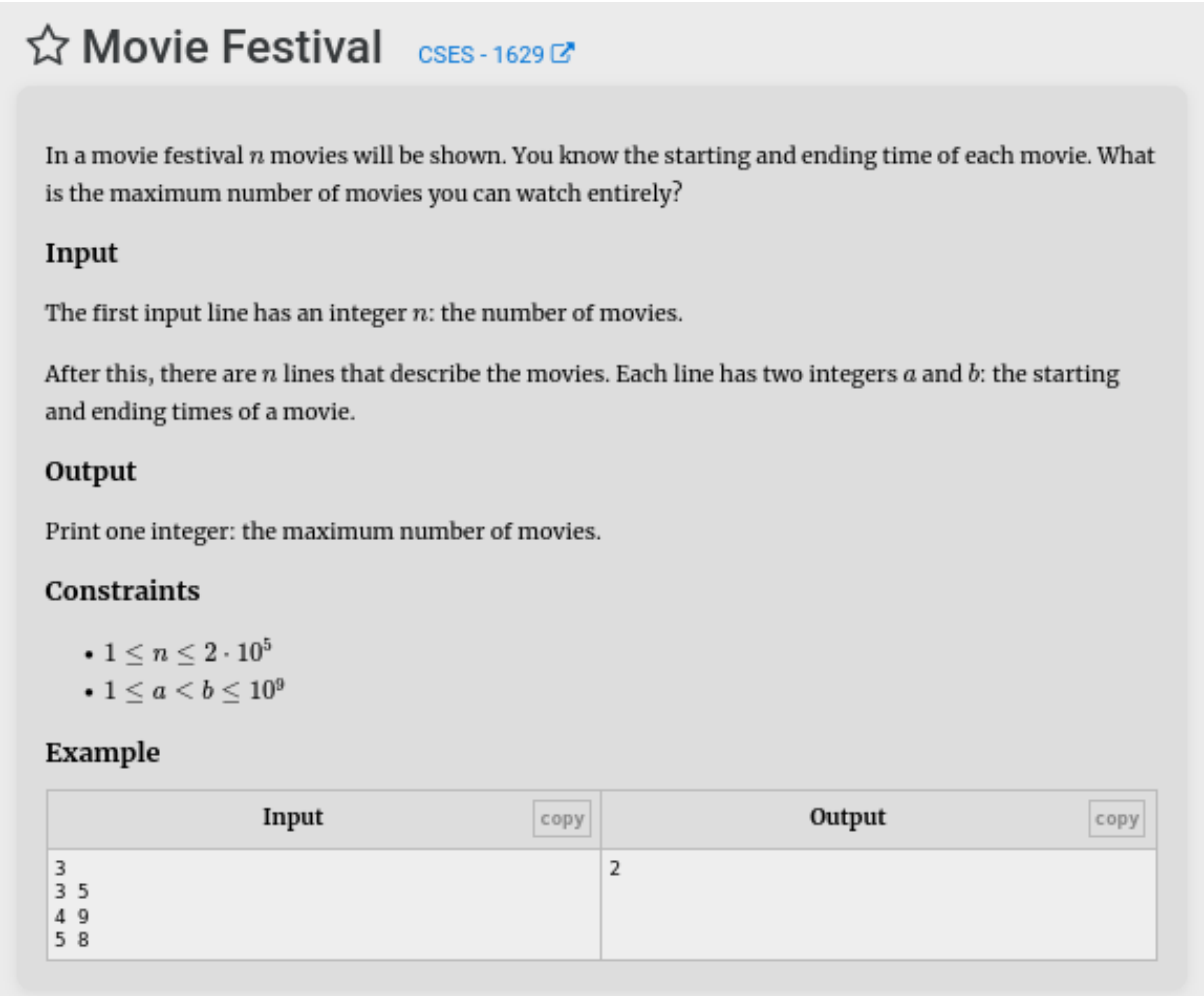
4.3 Código

```
#define endl '\n'
#define vi vector<int>

void solve(){
    int n, f, k;
    cin >> n >> f >> k;
    vi a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    f = a[f-1];
    sort(a.rbegin(), a.rend());

    if (a[k-1] < f) cout << "YES" << endl;
    else if (a[k-1] > f) cout << "NO" << endl;
    else { // a[k-1] == f
        if(k < n && a[k] == f) cout << "MAYBE" << endl;
        else cout << "YES" << endl;
    }
}

int main(){
    cin.tie(0); cout.tie(0);
    ios_base::sync_with_stdio(0);
    int t; cin >> t;
    while(t--) solve();
    return 0;
}
```



☆ Movie Festival CSES - 1629

In a movie festival n movies will be shown. You know the starting and ending time of each movie. What is the maximum number of movies you can watch entirely?

Input

The first input line has an integer n: the number of movies.

After this, there are n lines that describe the movies. Each line has two integers a and b: the starting and ending times of a movie.

Output

Print one integer: the maximum number of movies.

Constraints

- $1 \le n \le 2 \cdot 10^5$
- $1 \le a < b \le 10^9$

Example

Input	copy	Output	copy
3 3 5 4 9 5 8		2	

5 Problema E: Movie Festival

O problema clássico da seleção de eventos.

5.1 Enunciado do problema

5.2 Dicas

5.2.1 Dica 1

Uma escolha gulosa vai ter que rolar aqui, qual escolha?

5.2.2 Dica 2

Podemos pensar em algumas ideias:

- Escolher os filmes de menor duração
- Escolher os filmes que começam primeiro
- Escolher os filmes que terminam por último
- Escolher os filmes que começam por último

- Escolher os filmes que terminam por primeiro

Como sabemos que escolhas levam à solução ótima e que escolhas levam à resposta errada?

5.3 Código

```
#define ii pair<int,int>
#define vii vector<ii>

void solve() {
    int n, a, b;
    cin >> n;
    vii v(n);
    for (int i = 0; i < n; i++) {
        cin >> a >> b;
        v[i] = {b,a};
    }
    sort(v.begin(), v.end());
    int ans = 0, last = 0;
    for (auto [b,a] : v){
        if (a >= last) {
            ans++;
            last = b;
        }
    }
    cout << ans << endl;
}
```

Outra versão do código

```
void solve() {
    int n, a, b;
    cin >> n;
    vii v(n);
    for (int i = 0; i < n; i++) {
        cin >> a >> b;
        v[i] = {a,b};
    }
    sort(v.rbegin(), v.rend());
    int ans = 0, first = INF;
    for (auto [a,b] : v){
        if (b <= first) {
            ans++;
            first = a;
        }
    }
    cout << ans << endl;
}
```