INTERVAL TREE,
BINARY INDEX TREE

1. Lý do chọn đề tài

- Ưu điểm của việc sử dụng interval tree và binary index tree:
  - Bộ nhớ thấp
  - Cài đặt đơn giản
  - Có thể giải được nhiều bài toán về dãy số
  - Thời gian chạy: O(logN)

2. Một số bài toán

Bài 1: INVTRI - Bộ Ba Nghịch Thế
Cho dãy số nguyên A độ dài N, đếm số bộ ba (i, j, k) thỏa mãn $1 \le i < j < k \le N$ và $A_i > A_j > A_k$?

Input

Dòng 1: Số nguyên dương N ($1 \le N \le 10^5$).

Dòng 2: N số nguyên dương biểu diễn dãy A ($1 \le A_i \le 10^9$).

Output

Số bộ ba (i, j, k) đếm được.

Example

Input:

6
3 1 11 9 4 8

Output:

2

Solution

Với mỗi I (I = 1 -> N) ta sẽ tìm số lượng bộ 2 nghịch thế rồi lấy kết quả vừa tìm được để tính được số bộ 3 nghịch thế tại mỗi vị trí.

Để tìm số lượng bộ 2 nghịch thế ta dùng cấu trúc BIT, rồi sử dụng tiếp BIT 1 lần nữa để tìm ra số bộ 3 nghịch thế tại mỗi vị trí.

Kết quả bài toán là tổng số bộ 3 nghịch thế tại mỗi vị trí.

```
#include <bits/stdc++.h>

using namespace std;
```

```
int N, a[1000001], Lim, BIT[1000001], L[1000001], R[1000001];
vector <int> Zip;
long long ans;

main()
{
    scanf("%d", &N);
    for (int i = 1; i <= N; i++)
    {
        scanf("%d", &a[i]);
        Zip.push_back(a[i]);
    }

    sort(Zip.begin(), Zip.end());
    Zip.resize(Lim = distance(Zip.begin(), unique(Zip.begin(), Zip.end())));
    for (int i = 1; i <= N; i++)
        a[i] = lower_bound(Zip.begin(), Zip.end(), a[i]) - Zip.begin() + 1;

    memset(BIT, 0, sizeof BIT);
    for (int i = 1; i <= N; i++)
    {
        for (int x = a[i] + 1; x <= Lim; x += x & -x)
            L[i] += BIT[x];
        for (int x = a[i]; x; x -= x & -x)
            BIT[x]++;
    }
    memset(BIT, 0, sizeof BIT);
    for (int i = N; i; i--)
```

```
    {
        for (int x = a[i] - 1; x; x -= x & -x)
            R[i] += BIT[x];
        for (int x = a[i]; x <= Lim; x += x & -x)
            BIT[x]++;
    }

    for (int i = 1; i <= N; i++)
        ans += 1LL * L[i] * R[i];
    printf("%lld", ans);
}
```

Bài 2: VPBUS - Xe Bus

Trên mặt phẳng tọa độ cho K điểm là tọa độ của K trạm xe bus, trạm i có tọa độ (X_i, Y_i) và có Z_i người đang chờ ở đó. Tính số người đón được nhiều nhất có thể khi đi xe bus từ ô (1, 1) đến ô (N, M) biết xe bus chỉ có thể đi lên trên hoặc sang phải (giả sử không hạn chế số người lên xe) ?

Input

Dòng đầu tiên ghi 3 số nguyên dương N, M và K ($1 \le N, M \le 10^9$, $1 \le K \le 10^5$).

Trong K dòng sau, dòng thứ i chứa 3 số nguyên X_i, Y_i và Z_i ($1 \le X_i \le N$, $1 \le Y_i \le M$, $1 \le Z_i \le 10^6$).

Output

Gồm một dòng ghi số lượng người đón được nhiều nhất.

Example

Input:

```
6 7 6
1 1 13
1 2 10
6 4 9
1 3 5
2 2 5
1 6 5
```

Output:

Solution:

Ta sort lại N điểm theo tọa độ. Một lộ trình đi đúng thì tung độ của dãy sẽ tạo thành 1 dãy không giảm. Ta gọi f[i] là số lượng người MAX có thể chở được nếu kết thúc tại i.

F[i] = MAX(f[j] , với j < i và tung độ điểm j < tung độ điểm i.)

Ta đễ dàng tính được f[N] (kết quả bài toán) bằng cấu trúc cây BIT.

```
#include <bits/stdc++.h>
using namespace std;

int N, M, K;
struct data
{
   int x, y, z;
   data (int x = 0, int y = 0, int z = 0): x(x), y(y), z(z) {}
   bool operator < (const data & op) const
   {
      return make_pair(x, y) < make_pair(op.x, op.y);
   }
} bus_stop[100001];
vector <int> Zip;
long long BIT[100001];

void update(int x, long long v)
{
   for (; x <= K; x += x & -x)
      BIT[x] = max(BIT[x], v);
}
```

```
long long getmax(int x)
{
   long long v = 0;
   for (; x; x -= x & -x)
      v = max(v, BIT[x]);
   return v;
}

main()
{
   scanf("%d %d %d", &N, &M, &K);
   for (int i = 1; i <= K; i++)
      scanf("%d %d %d", &bus_stop[i].x, &bus_stop[i].y, &bus_stop[i].z);

   sort(bus_stop + 1, bus_stop + K + 1);
   for (int i = 1; i <= K; i++) Zip.push_back(bus_stop[i].y);
   sort(Zip.begin(), Zip.end());
   Zip.resize(distance(Zip.begin(), unique(Zip.begin(), Zip.end())));
   for (int i = 1; i <= K; i++)
   {
      int y = lower_bound(Zip.begin(), Zip.end(), bus_stop[i].y) - Zip.begin() +
1;
      update(y, getmax(y) + bus_stop[i].z);
   }

   printf("%lld", getmax(K));
}
```

Bài 3: VPKQUERY - K-Query

Cho dãy số nguyên A độ dài N và Q truy vấn: Mỗi truy vấn là một bộ ba (i, j, k) - yêu cầu đếm số lượng phần tử lớn hơn k nằm trong dãy con A_i, A_{i+1}, ..., A_j ?

Input

Dòng đầu tiên chứa 2 số nguyên dương N và Q ($1 \le N, Q \le 10^5$).

Dòng thứ hai chứa N số nguyên biểu diễn dãy A ($|A_i| \le 10^9$).

Trong Q dòng tiếp theo, mỗi dòng chứa 3 số nguyên i, j, k biểu diễn một truy vấn ($1 \le i \le j \le N$, $|k| \le 10^9$).

Output

Gồm Q dòng, mỗi dòng là kết quả của một truy vấn tương ứng.

Example

Input:

5 3
5 1 2 3 4
2 4 1
4 4 4
1 5 2

Output:

2
0
3

Solution:

Ta sử dụng cấu trúc cây IT, với mỗi nút ta lưu lại 1vector gồm toàn bộ các số trong đoạn mà nó bao phủ và đã được sắp xếp.

Để trả lời các truy vấn ta sẽ gộp các đoạn lại sồi binary search vị trí nó trong đoạn(kết quả bài toán).

```
#include <bits/stdc++.h>

using namespace std;


int N, Q, BIT[100001], ans[100001];

pair <int, int> a[100001];

struct node

{

   int l, r, k, i;
```

```
    node(int l = 0, int r = 0, int k = 0, int i = 0): l(l), r(r), k(k), i(i) {}
    bool operator < (const node & op) const
    {
        return k > op.k;
    }
} query[100001];

void update(int x)
{
    for (; x <= N; x += x & (-x)) BIT[x]++;
}

int get(int x)
{
    int ret = 0;
    for (; x; x -= x & (-x)) ret += BIT[x];
    return ret;
}

main()
{
    scanf("%d %d", &N, &Q);
    for (int i = 1; i <= N; i++)
    {
        scanf("%d", &a[i].first);
        a[i].second = i;
    }
    for (int i = 1; i <= Q; i++)
    {
```

```
        scanf("%d %d %d", &query[i].l, &query[i].r, &query[i].k);
        query[i].i = i;
    }

    sort(a + 1, a + N + 1);
    sort(query + 1, query + Q + 1);
    for (int i = 1, j = N; i <= Q; i++)
    {
        while (a[j].first > query[i].k && j) update(a[j--].second);
        ans[query[i].i] = get(query[i].r) - get(query[i].l - 1);
    }

    for (int i = 1; i <= Q; i++) printf("%d\n", ans[i]);
}
```

Bài 4: VPPALIPALI - Đối Xứng Đối Xứng

Một xâu được gọi là dx nếu nó có dạng XX^R với X là một xâu khác rỗng và X^R là xâu đảo ngược của X. Cho xâu S chỉ gồm các ký tự Latin thường, đếm số xâu con là xâu dx của S.

Input

Một dòng duy nhất chứa xâu S ($1 \leq |S| \leq 10^5$).

Output

Số lượng xâu con là xâu dx của S.

Example

Input:

bbaabbbbaabb

Output:

2

```
#include <bits/stdc++.h>
using namespace std;
```

```
int N, R[100010], BIT[100010];
long long myhash[2][100010], Pw[100010];
char S[100010];
pair <int, int> data[100010];
struct query
{
    int l, r, k;
    bool operator < (const query & op) const
    {
        return k < op.k;
    }
} Q[100010];

bool Palin(int l, int r)
{
    return myhash[0][r] - myhash[0][l - 1] * Pw[r - l + 1]
            == myhash[1][l] - myhash[1][r + 1] * Pw[r - l + 1];
}

void update(int x)
{
    for (; x <= N; x += x & -x)
        BIT[x]++;
}

int get(int x)
{
    int y = 0;
```

```
    for (; x; x -= x & -x)
        y += BIT[x];
    return y;
}

main()
{
    Pw[0] = 1;
    for (int i = 1; i <= 100000; i++) Pw[i] = Pw[i - 1] * 1000000007;

    scanf("%s", S + 1);     N = strlen(S + 1);

    for (int i = 1; i <= N; i++)
    {
        myhash[0][i] = myhash[0][i - 1] * 1000000007 + S[i];
        myhash[1][N - i + 1] = myhash[1][N - i + 2] * 1000000007 + S[N - i + 1];
    }
    for (int i = 1; i <= N; i++)
    {
        int &k = R[i] = 0;
        for (int l = 1, r = min(i, N - i); l <= r; )
        {
            int m = (l + r) / 2;
            if (Palin(i - m + 1, i + m)) k = max(k, m), l = m + 1;
                else r = m - 1;
        }
    }

    for (int i = 1; i <= N; i++)
```

```
    {
        data[i] = make_pair(2 * i - R[i], i);
        Q[i] = {i + 1, i + R[i], 2 * i};
    }
    sort(data + 1, data + N + 1);
    sort(Q + 1, Q + N + 1);

    long long ans = 0;
    for (int i = 1, j = 1; i <= N; i++)
    {
        while (data[j].first <= Q[i].k && j <= N) update(data[j++].second);
        ans += get(Q[i].r) - get(Q[i].l - 1);
    }
    printf("%lld\n", ans);
}
```

Bài 5: SCMQUERY – Xử Lý Truy Vấn

Cho dãy số nguyên dương a độ dài N, xử lý Q truy vấn gồm 1 trong 3 dạng:

- 1 l r: Tính $\sum_{i=l}^{r} a[i]$ ($1 \le l \le r \le N$).
- 2 p x: Gán a[p] = x ($1 \le p \le N$, $1 \le x \le 10^9$).
- 3 l r x: Gán a[i] = a[i] mod x ($l \le i \le r$, $1 \le x \le 10^9$).

Dữ liệu:

Input:

- Dòng 1: Số nguyên dương N và Q ($1 \le N, Q \le 10^5$).
- Dòng 2: N số nguyên dương biểu diễn dãy a ($1 \le a[i] \le 10^9$).
- Dòng 3..Q+2: Mỗi dòng là một truy vấn.

Output:

- Với mỗi truy vấn loại 1, in kết quả trên 1 dòng.

Input	**Output**

5 5 5 4 3 2 1 3 3 5 2 2 3 3 1 1 5 3 1 5 1 1 2 4	13 0

Solution:

Ta sử dụng cấu trúc cây IT. Ta dễ dàng sử lý được truy vấn 1 và truy vấn 2. Với truy vấn 3 ta cần xử lý khéo léo để tăng tốc. Ta sẽ update toàn bộ những nút trong đoạn từ L -> R bằng cách chỉ đi xuống những nút có tổng lớn hơn X ta giảm độ phúc tạp xuống O(log2(n)).

```
#include <bits/stdc++.h>

using namespace std;


int N, Q;

pair <long long, int> IT[400004];


void Modify(int k, int l, int r, int pos, int val)

{

   if (pos < l || r < pos) return;

   if (l == r) IT[k] = make_pair(val, val);

   else

   {

      int mid = (l + r) / 2;

      Modify(2 * k, l, mid, pos, val);

      Modify(2 * k + 1, mid + 1, r, pos, val);

      IT[k] = make_pair(IT[2 * k].first + IT[2 * k + 1].first, max(IT[2 *
k].second, IT[2 * k + 1].second));

   }

}
```

```
void Mod(int k, int l, int r, int x, int y, int mod)
{
   if (y < l || r < x || IT[k].second < mod) return;
   if (l == r) IT[k].first %= mod, IT[k].second %= mod;
   else
   {
      int mid = (l + r) / 2;
      Mod(2 * k, l, mid, x, y, mod);
      Mod(2 * k + 1, mid + 1, r, x, y, mod);
      IT[k] = make_pair(IT[2 * k].first + IT[2 * k + 1].first, max(IT[2 * k].second, IT[2 * k + 1].second));
   }
}

long long Sum(int k, int l, int r, int x, int y)
{
   if (y < l || r < x) return 0;
   if (x <= l && r <= y) return IT[k].first;
   int mid = (l + r) / 2;
   return Sum(2 * k, l, mid, x, y) + Sum(2 * k + 1, mid + 1, r, x, y);
}

main()
{
   scanf("%d %d", &N, &Q);
   for (int A, i = 1; i <= N; i++)
   {
      scanf("%d", &A);
      Modify(1, 1, N, i, A);
```

```
    }

    for (int cmd, l, r, k, x; Q--;)
    {
        scanf("%d", &cmd);
        if (cmd == 1)
        {
            scanf("%d %d", &l, &r);
            printf("%lld\n", Sum(1, 1, N, l, r));
        }
        else if (cmd == 2)
        {
            scanf("%d %d", &k, &x);
            Modify(1, 1, N, k, x);
        }
        else
        {
            scanf("%d %d %d", &l, &r, &x);
            Mod(1, 1, N, l, r, x);
        }
    }
}
```

Bài 6: VPBRACKET – Dãy Ngoặc Đúng

Một dãy ngoặc đúng được định nghĩa như sau:

- Xâu rỗng là 1 dãy ngoặc đúng.

- Nếu A là 1 dãy ngoặc đúng thì (A) là 1 dãy ngoặc đúng.

- Nếu A và B là dãy ngoặc đúng thì AB là 1 dãy ngoặc đúng.

Cho dãy ngoặc S độ dài N và Q truy vấn biểu diễn bởi 2 số nguyên l và r ($1 \le l \le r \le N$): Kiểm tra dãy ngoặc con S[l..r] có là dãy ngoặc đúng hay không ?

Dữ liệu:

Input:

- Dòng đầu tiên chứa 2 số nguyên dương N và Q ($1 \le N, Q \le 10^5$).
- Dòng thứ hai chứa dãy ngoặc S.
- Trong Q dòng tiếp theo, mỗi dòng chứa một truy vấn.

Output:

- Với mỗi truy vấn, in trên một dòng "YES" nếu đúng, ngược lại in "NO".

Input	Output
7 3 (((())() 3 4 1 5 2 7	YES NO YES

Solution:

Ta định nghĩa a[i] = 1 nếu s[i] = '(' vào a[i] = -1 nếu s[i] = ')'
mảng tổng tiền tố đuộc tính như sau
Sum[i] = Sum[i – 1] + a[i];
Ta thấy dãy con [L ... R] là dãy ngoặc đúng khi và chỉ khi
+ tổng dãy a[i] từ [L ... R] bằng 0;
+ tổng dãy a[i] từ [L ... j] lớn hơn bằng 0 với mọi j <= R;
Ta có thể dùng cây IT để lưu mảng giá trị nhỏ nhất của mảng tổng dồn

```
#include <bits/stdc++.h>

using namespace std;


int N, Q, l, r, s[100001], IT[400004];

char c;


void build(int k, int l, int r)

{

   if (l == r) IT[k] = s[l];

   else

   {

      int m = (l + r) / 2;

      build(2 * k, l, m);
```

```
        build(2 * k + 1, m + 1, r);
        IT[k] = min(IT[2 * k], IT[2 * k + 1]);
    }
}

int getmin(int k, int l, int r, int x, int y)
{
    if (r < x || y < l) return INT_MAX;
    if (x <= l && r <= y) return IT[k];
    else
    {
        int m = (l + r) / 2;
        return min(getmin(2 * k, l, m, x, y), getmin(2 * k + 1, m + 1, r, x, y));
    }
}

main()
{
    scanf("%d %d\n", &N, &Q);
    for (int i = 1; i <= N; i++)
    {
        scanf("%c", &c);
        s[i] = s[i - 1] + ((c == '(') ? 1 : -1);
    }
    build(1, 1, N);
    while (Q--)
    {
        scanf("%d %d", &l, &r);
```

```
        printf("%s\n", (s[l - 1] == s[r] && getmin(1, 1, N, l, r) >= s[r]) ? "YES" :
"NO");
    }
}
```

Bài 7: INCSUBSTR – Xâu Không Giảm

Cho xâu S độ dài N chỉ chứa các ký tự '0' và '1' và Q truy vấn có 2 dạng:

- switch x: Thay đổi S[x] (thay '0' bằng '1' và ngược lại).

- count: Tính độ dài xâu con (không cần liên tiếp) không giảm dài nhất của S.

Dữ liệu:

Input:

- Dòng 1: Số nguyên dương N và Q ($1 \le N, Q \le 10^5$).
- Dòng 2: Xâu S chỉ gồm '0' hoặc '1'.
- Dòng 3..Q+2: Mỗi dòng biểu diễn một truy vấn.

Output:

- Với mỗi truy vấn loại 2, in trên một dòng độ dài xâu con không giảm dài nhất của S.

Input	Output
7 5 1101001 count switch 4 count switch 1 count	4 5 6

Solution:

Nhận thấy dãy con không giảm sẻ ở dạng 0..01..1;

Vì thế ta dùng cây IT để đếm số số 0 và số số 1 từ đoạn L - > R;

Để tính độ dài dãy con không giảm dài nhất của nốt cha ta chỉ cần lấy max của các cách

ghép (0, 0) hoặc (0, 1) hoặc (1, 1) của 2 nút con

```
#include <bits/stdc++.h>

using namespace std;
```

```
int N, M, x;
char S[100001], cmd[11];
struct node
{
   int cnt00, cnt01, cnt10, cnt11;
   bool Switch;
   node (int cnt00 = 0, int cnt01 = 0, int cnt10 = 0, int cnt11 = 0, bool Switch =
0) :
      cnt00(cnt00), cnt01(cnt01), cnt10(cnt10), cnt11(cnt11), Switch(Switch) {};
} IT[400004];

node Combine(node A, node B)
{
   node C;
   C.cnt00 = A.cnt00 + B.cnt00;
   C.cnt01 = max(A.cnt01 + B.cnt11, A.cnt00 + B.cnt01);
   C.cnt10 = max(A.cnt10 + B.cnt00, A.cnt11 + B.cnt10);
   C.cnt11 = A.cnt11 + B.cnt11;
   return C;
}

void Switch_node(node &X)
{
   X.Switch ^= 1;
   swap(X.cnt00, X.cnt11);
   swap(X.cnt01, X.cnt10);
}

void build(int k, int l, int r)
```

```
{
    int m = (l + r) / 2;
    if (l == r)
    {
        IT[k].cnt01 = IT[k].cnt10 = 1;
        if (S[m] == '0') IT[k].cnt00 = 1;
            else IT[k].cnt11 = 1;
    }
    else
    {
        build(2 * k, l, m);
        build(2 * k + 1, m + 1, r);
        IT[k] = Combine(IT[2 * k], IT[2 * k + 1]);
    }
}

void update(int k, int l, int r, int x)
{
    if (r < x || x < l) return ;
    if (l == r) Switch_node(IT[k]);
    else
    {
        int m = (l + r) / 2;
        update(2 * k, l, m, x);
        update(2 * k + 1, m + 1, r, x);
        IT[k] = Combine(IT[2 * k], IT[2 * k + 1]);
    }
}
```

```
main()
{
   scanf("%d %d\n%s", &N, &M, S + 1);
   build(1, 1, N);
   while (M--)
   {
      scanf("\n%s", cmd + 1);
      if (cmd[1] == 'c') printf("%d\n", IT[1].cnt01);
      else
      {
         scanf("%d", &x);
         update(1, 1, N, x);
      }
   }
}
```

Bài 8: MAXQUERY – Tổng Lớn Nhất

Cho dãy số nguyên A độ dài N và Q truy vấn có 2 dạng:

1 i v: Gán A[i] = v ($1 \le i \le N$, $|v| \le 10^4$).

2 x y: Tính $\max\{A_i + A_{i+1} + ... + A_j \mid x \le i \le j \le y\}$.

Dữ liệu:

Input:

- Dòng đầu tiên chứa 2 số nguyên dương N và Q ($1 \le N, Q \le 10^5$).
- Dòng thứ hai chứa N số nguyên biểu diễn dãy A ($|A_i| \le 10^4$).
- Trong Q dòng tiếp theo, mỗi dòng chứa một truy vấn.

Output:

- Với mỗi truy vấn dạng 2, in kết quả cần tìm trên một dòng.

Input	Output
4 3 1 2 3 4 2 2 4 1 3 -3 2 2 4	9 4

Solution:

Bài này chúng ta có thể dùng cây IT để giải như sau

Với mỗi nút của cây IT ta lưu lại 4 trường là

Tổng tất cả các số

Tổng tiền tố có giá trị lớn nhất

Tổng hậu tố có giá trị lớn nhất

Tổng đoạn con liên tiếp có giá trị lớn nhất

Ta có thể dễ dàng cập nhật là truy suất ra kết quả của đoạn bất kì từ L -> R;

```
#include <bits/stdc++.h>
#define lim -10000000
using namespace std;

int N, Q, a[100001], cmd, x, y;
struct node
{
    int L, R, M, sum;
    node (int L = lim, int R = lim, int M = lim, int sum = 0): L(L), R(R), M(M), sum(sum) {}
    node operator + (const node & B) const
    {
        node C;
        C.sum = sum + B.sum;
        C.L = max(L, sum + B.L);
        C.R = max(B.R, R + B.sum);
        C.M = max(max(M, B.M), max(max(C.L, C.R), R + B.L));
```

```
        return C;
    }
} IT[400004];

void update(int k, int l, int r, int i, int v)
{
    if (r < i || i < l) return;
    if (l == r) IT[k] = {v, v, v, v};
    else
    {
        int m = (l + r) / 2;
        update(2 * k, l, m, i, v);
        update(2 * k + 1, m + 1, r, i, v);
        IT[k] = IT[2 * k] + IT[2 * k + 1];
    }
}

node get(int k, int x, int y, int l, int r)
{
    if (l <= x && y <= r) return IT[k];
    if (r < x || y < l) return {lim, lim, lim, lim};
    int m = (x + y) / 2;
    return get(2 * k, x, m, l, r) + get(2 * k + 1, m + 1, y, l, r);
}
```

```
main()
{
   scanf("%d %d", &N, &Q);
   for (int i = 1; i <= N; i++)
   {
      scanf("%d", &a[i]);
      update(1, 1, N, i, a[i]);
   }
   while (Q--)
   {
      scanf("%d %d %d", &cmd, &x, &y);
      if (cmd == 1) update(1, 1, N, x, y);
         else printf("%d\n", get(1, 1, N, x, y).M);
   }
}
```

Bài 9: INCSUBSEQS – Dãy Con Tăng

Cho dãy số nguyên A độ dài N và số nguyên dương K. Yêu cầu đếm số lượng dãy con tăng độ dài K của A.

Dữ liệu:

Input:

- Dòng đầu tiên chứa 2 số nguyên dương N và K ($1 \le N \le 10^5$, $1 \le K \le 20$).
- Dòng tiếp theo chứa N số nguyên biểu diễn dãy A ($1 \le A_i \le N$).

Output:

- Số lượng dãy con tăng độ dài K đếm được sau khi chưa dư 10^9+7.

Input	**Output**
4 2	5

1 2 4 3	

Solution:

Gọi f[i][j] là số lượng dãy con tăng độ dài j và phần tử cuối là a[i];

Kết quả trả về f[i][k];

Ta có f[i][j] = f[i][j] + f[h][j – 1] (với a[j] > a[h])

Ta dùng cây BIT để tính tổng các f[h][j]

```
#include <bits/stdc++.h>
using namespace std;

int N, K, BIT[22][101101];

void update(int x, int y, int v)
{
   for (y++; y <= N + 1; y += y & -y)
   {
      BIT[x][y] += v;
      if (BIT[x][y] >= 1000000007) BIT[x][y] -= 1000000007;
   }
}

int get(int x, int y)
{
   long long ret = 0;
   for (y++; y; y -= y & -y)
      ret += BIT[x][y];
   ret %= 1000000007;
   return ret;
}
```

```
main()
{
   scanf("%d %d", &N, &K);
   update(0, 0, 1);
   for (int i = 1; i <= N; i++)
   {
      int a;
      scanf("%d", &a);
      for (int k = K; k; k--)
         update(k, a, get(k - 1, a - 1));
   }
   printf("%d", get(K, N));
}
```

Bài 10: SORTARR - Sắp Xếp Dãy Số

Cho dãy số nguyên A độ dài N. Tính chi phí nhỏ nhất để sắp xếp dãy A thành một dãy số không giảm biết chi phí đổi chỗ 2 phần tử ở vị trí i và j trong dãy là $(i-j)^2$.

Dữ liệu:

Input:

- Dòng đầu tiên chứa số nguyên dương N ($1 \le N \le 10^5$).
- Dòng tiếp theo chứa N số nguyên biểu diễn dãy A ($|A_i| \le 10^9$).

Output:

- Chi phí nhỏ nhất cần tìm.

Input	Output
4 1 1 3 2	1

Solution:

Ta có nếu phải đổi chỗ a[i] với a[j] thì ta đổi lần lượt a[i] với a[i + 1] .....

a[j – 1] với a[j], a[j – 1] với a[j – 2] ... a[i + 1] với a[i];

ta nhận thấy số phép biến đổi sẽ là 2 * (j - i) - 1 <= (j - i)^2;

bài toán quy về tìm chi phí ít nhất để biến đổi dãy A thành 1 dãy tăng với mỗi bước di chuyển a[i] vào vị trí j với chi phí là abs(i – j);

ta dùng cây BIT có thể dễ dàng gải quyết bài toán này.

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 100000;
const int INF = 1000000000;

int a[MAXN], backup[MAXN], n;
int tree[MAXN + 1];

void add(int x)
{
   for (int y = x; y <= n; y += y & -y) {
      ++ tree[y];
   }
}

int getSum(int x)
{
   int sum = 0;
   for (int y = x; y > 0; y -= y & -y) {
      sum += tree[y];
   }
   return sum;
}
```

```
main()
{
   scanf("%d", &n);
   for (int i = 0; i < n; ++ i) {
      scanf("%d", &a[i]);
      backup[i] = a[i];
   }
   sort(backup, backup + n);
   int m = unique(backup, backup + n) - backup;

   long long answer = 0;
   for (int i = 0; i < n; ++ i)
   {
      int x = m - 1 - (lower_bound(backup, backup + m, a[i]) - backup);
      answer += getSum(x);
      add(x + 1);
   }
   cout << answer << endl;
}
```

Bài 11: VPMAXSUMSUB – Tổng Đoạn Lớn Nhất

Cho dãy số nguyên A độ dài N. Thực hiện Q truy vấn, mỗi truy vấn được biểu diễn bởi một số nguyên K: Tính tổng đoạn con lớn nhất trong những đoạn con của A có các phần tử không vượt quá K ?

Input:

- Dòng đầu tiên chứa 2 số nguyên dương N và Q.
- Dòng thứ hai chứa N số nguyên biểu diễn dãy A.
- Trong Q dòng tiếp theo, mỗi dòng chứa một số nguyên K biểu diễn một truy vấn.

Output:

- Gồm Q dòng, nếu tồn tại đoạn con có các phần tử không vượt quá K, in trên một dòng giá trị tổng đoạn con lớn nhất tìm được, ngược lại in "No Solution".

Example:

Input	Output
5 6 1 2 3 4 5 4 0 5 2 1 3	10 No Solution 15 3 1 6

Giới hạn:

- $1 \leq N, Q \leq 10^5$.
- $|A_i|, |K| \leq 10^9$.

Solution:

Sử dụng interval tree giống bài http://www.spoj.com/problems/GSS1/

Sắp xếp truy vấn theo giá trị của K.

Với mỗi truy vấn i, cập nhật vào cây interval các số A[j] có giá trị không vượt quá K[i] và lớn hơn K[i-1] (nên dùng duyệt tuyến tính).

Lúc này kết quả của truy vấn i chính bằng giá trị max của nút 1 trong cây interval.

```
#include <bits/stdc++.h>

const long long INF = 1e14 + 1;

using namespace std;


int N, Q;

struct node

{

   long long L, R, M, sum;

   node operator + (const node & B) const

   {
```

```
        node C;
        C.sum = sum + B.sum;
        C.L = max(L, sum + B.L);
        C.R = max(B.R, R + B.sum);
        C.M = max(max(M, B.M), max(max(C.L, C.R), R + B.L));
        return C;
    }
} IT[400004];
pair <int, int> a[100001], K[100001];
long long ans[100001];

void update(int k, int l, int r, int x, int v)
{
    if (r < x || x < l) return;
    if (l == r) IT[k] = {v, v, v, v};
    else
    {
        int m = (l + r) / 2;
        update(2 * k, l, m, x, v);
        update(2 * k + 1, m + 1, r, x, v);
        IT[k] = IT[2 * k] + IT[2 * k + 1];
    }
}

main()
{
    scanf("%d %d", &N, &Q);
    for (int i = 1; i <= N; i++) scanf("%d", &a[i].first), a[i].second = i;
    for (int i = 1; i <= Q; i++) scanf("%d", &K[i].first), K[i].second = i;
```

```
    sort(a + 1, a + N + 1);
    sort(K + 1, K + Q + 1);
    for (int i = 1; i <= 4 * N; i++) IT[i] = {-INF, -INF, -INF, -INF};

    for (int i = 1, j = 1; j <= Q; j++)
    {
        for (; i <= N && a[i].first <= K[j].first; i++)
            update(1, 1, N, a[i].second, a[i].first);
        ans[K[j].second] = IT[1].M;
    }

    for (int i = 1; i <= Q; i++)
        if (ans[i] != -INF) printf("%lld\n", ans[i]);
            else printf("No Solution\n");
}
```

Bài 12: VPMAXSUBARR – Đoạn Con Có Tổng Lớn Nhất

Cho dãy số A độ dài N và 2 số nguyên dương X, Y. Tính tổng đoạn con lớn nhất có độ dài L thỏa mãn $X \leq L \leq Y$.

Input:

- Dòng đầu tiên chứa 3 số nguyên dương N, X và Y.
- Dòng thứ hai chứa N số nguyên biểu diễn dãy A.

Output:

- Tổng đoạn con lớn nhất tìm được.

Example:

Input	Output
4 3 1 2 -3 4	3

Giới hạn:

- $1 \leq X \leq Y \leq N \leq 10^5$.

- $|A_i| \leq 10^9$.

Solution:

Bài này nên làm theo hướng RMQ để ôn lại. Đpt O(NlogN).

Tính y = log2(x) dùng hàm này nhanh hơn nhiều lần gọi hàm log bình thường:

Nếu x kiểu int: y = 31 - __builtin_clz(x);

Nếu x kiểu long long: y = 63 - __builtin_clzll(x);

```
#include <bits/stdc++.h>
using namespace std;

int N, X, Y, A;
long long S[100010], RMQ[100010][18], ans = LONG_LONG_MIN;

long long get(int L, int R)
{
   int K = 31 - __builtin_clz(R - L + 1);
   return min(RMQ[R][K], RMQ[L + (1 << K) - 1][K]);
}

main()
{
   scanf("%d %d %d", &N, &X, &Y);
   for (int i = 1; i <= N; i++)
   {
      scanf("%d", &A);
      RMQ[i][0] = S[i] = S[i - 1] + A;
      for (int j = 1; (1 << j) <= i; j++)
         RMQ[i][j] = min(RMQ[i][j - 1], RMQ[i - (1 << (j - 1))][j - 1]);
   }
   for (int i = X; i <= N; i++)
```

```
        ans = max(ans, S[i] - get(max(0, i - Y), max(0, i - X)));
    printf("%lld", ans);
}
```

3. Kết luận

Vì thời gian và trình độ tác giả chưa cao nên chuyên đề này còn nhiều hạn chế. Kính mong các vị đồng nghiệp và các độc giả góp ý để tác giả có thể hoàn thiện tốt hơn.