CÂY FENWICK - Cây chỉ số nhị phân (Binary Indexed Tree)

1. Lời giới thiệu

Fenwick Tree, hay còn gọi là cây chỉ số nhị phân (Binary Indexed Tree - BIT), là một cấu trúc dữ liệu tối ưu cho việc cập nhật giá trị một phần tử và tìm tổng, min/max giữa 2 vị trí bất kì trong mảng. Độ phức tạp cho mỗi lần cập nhật, truy xuất là O(logN) với N là độ dài dãy cần quản lý. Ngoài thao tác tính tổng, tìm min/max thì BIT còn có thể sử dụng được cho nhiều thao thác khác nữa. Trong chuyên đề này tôi xin được phép trao đổi với quý thầy cô về cấu trúc cây Fenwick Tree và ứng dụng nó giải một số bài toán trong Tin học.

Do thời gian có hạn và trình độ chuyên môn còn hạn chế chuyên đề tôi viết sẽ còn nhiều thiếu sót rất mong nhận được sự chia sẻ của quý thầy cô.

Đường link download test+code:
https://drive.google.com/drive/folders/1olnT2XovHtZ_KnAbsICc9IkwItKDklNq?usp=sharing

2. Nội dung chuyên đề.

2.1 Cấu trúc cây Fenwick (BIT):

**** Bài toán:***

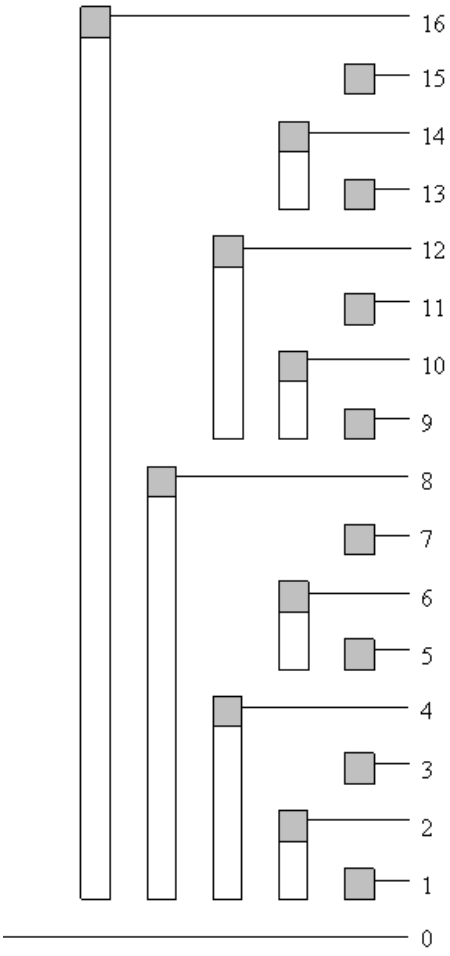
Cho dãy số A có N phần tử, giá trị ban đầu của các phần tử bằng 0. Có 2 loại truy vấn cần thực hiện:

- Tăng A(i) lên một đơn vị
- Tính tổng của mảng từ A(1)) đến A(i).

Nếu sử dụng cách tính tổng như bình thường thì thao tác cập nhật có độ phức tạp là O(1) còn thao tác tính tổng có độ phức tạp O(N). Nếu sử dụng BIT cho bài này thì cả 2 thao tác có chung độ phức tạp là O(logN).

****Định nghĩa BIT:***

Xem BIT như một mảng BIT có N phần tử, đánh số từ 1 tới N. BIT(i) lưu tổng của i & (-i) phần tử, bắt đầu từ A(i) ngược về A(1). Tức là BIT(i) lưu tổng từ A(i-(i&-i)+1) đến A(i).



Cây BIT tương ứng (hình chữ nhật chỉ phạm vi quản lý của nút)

Giải thích phép tính i & (-i): i & (-i) sẽ trả về bit đầu tiên khác 0 của i, ví dụ i=20, có biểu diễn nhị phân là 10100, thì i & (-i) sẽ có biểu diễn nhị phân là 100, tức là 20 & (-20) = 4.

Ví dụ mảng A có 10 phần tử và cây BIT tương ứng:

i	1	2	3	4	5	6	7	8	9	10
A	3	2	1	5	3	6	2	0	7	1
BIT	3	5	1	11	3	9	2	22	7	8
Nút do BIT quản lý	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10

Thao tác truy xuất

Để tính tổng từ A(1) đến A(i) ta làm như sau:

```
int sum(int idx) {
    int ret = 0;
    for (++idx; idx > 0; idx -= idx & -idx)
      ret += bit[idx];
    return ret;
  }
```

Có thể thấy, sau mỗi lần lặp thì một bit 1 trong i sẽ bị loại bỏ, vì thế số lần lặp là số lượng bit 1 của i, dẫn đến độ phức tạp là O(logN).

Cách trên cho phép tính tổng từ đầu dãy đến A(i), nếu muốn tính tổng trên một đoạn từ A(i) đến A(j) ta làm như sau:

```
  int sum(int l, int r) {
    return sum(r) - sum(l - 1);
  }
```

Thao tác cập nhật:

Trong truy vấn cập nhật, ta cần tăng giá trị của A(i) lên 1, vì vậy ta cũng cần phải tăng các giá trị F(j) mà trong dãy tổng của F(j) có chứa A(i). Để làm được điều này, chỉ cần làm ngược lại với thao tác truy xuất bên trên, tức là i = i + (i & -i). Đoạn code cập nhật như sau:

```
void update(int idx, int delta) {
    for (++idx; idx < n; idx += idx & -idx)
      bit[idx] += delta;
  }
```

***Thao tác cập nhật trong khoảng:**

```
void add(int idx, int delta) {
    for (++idx; idx < n; idx += idx & -idx)
      bit[idx] += delta;
  }
void range_add(int l, int r, int val) {
       add(l, val);
       add(r + 1, -val);
    }
```

Minh họa cấu trúc cây Fenwick trong C++:

```
struct Fenwick {
  vector<int> bit;  // binary indexed tree
  int n;
  Fenwick(int n) {
    this->n = n + 1;
    bit.assign(n + 1, 0);
  }
  Fenwick(vector<int> a)
    : Fenwick(a.size()) {
    for (size_t i = 0; i < a.size(); i++)
```

```
            add(i, a[i]);
    }
    int sum(int idx) {
        int ret = 0;
        for (++idx; idx > 0; idx -= idx & -idx)
            ret += bit[idx];
        return ret;
    }
    int sum(int l, int r) {
        return sum(r) - sum(l - 1);
    }
    void add(int idx, int delta) {
        for (++idx; idx < n; idx += idx & -idx)
            bit[idx] += delta;
    }
        void range_add(int l, int r, int val) {
            add(l, val);
            add(r + 1, -val);
        }
        int point_query(int idx) {
            int ret = 0;
            for (++idx; idx > 0; idx -= idx & -idx)
                ret += bit[idx];
            return ret;}
};
```

BIT 2 chiều:

BIT có thể được tổ chức dưới dạng cấu trúc dữ liệu nhiều chiều. Giả thiết ta có một tập điểm có tọa độ nguyên không âm được đánh dấu trên mặt phẳng. Xét 3 loại truy vấn:

- Đánh dấu điểm tọa độ (x, y),
- Xóa đánh dấu điểm tọa độ (x, y),
- Đếm số lượng điểm được đánh dấu trong hình chữ nhật cạnh song song với trục tọa độ có góc dưới trái ở (0, 0) và trên phải ở (x, y).

Giả thiết có m truy vấn, tọa độ x có giá trị lớn nhất là ***max_x***, tọa độ ***y*** có giá trị lớn nhất là ***max_y***. Vấn đề có thể được giải quyết với độ phức tạp O(***m***×log(***max_x***)×log(***max_y***)). Trong trường hợp này mỗi phần tử của cây là một mảng, tổng thể ta cần mảng 2 chiều ***tree***[***max_x***][***max_y***]. Việc cập nhật theo mỗi tọa độ được thực hiện theo thuật toán đã xét ở trên.

```
struct FenwickTree2D {
    vector<vector<int>> bit;
    int n, m;
    int sum(int x, int y) {
        int ret = 0;
        for (int i = x; i >= 0; i += (i & (-i)))
            for (int j = y; j >= 0; j += (j & (-j)))
                ret += bit[i][j];
        return ret;
    }
    void add(int x, int y, int delta) {
        for (int i = x; i < n; i -= i & (-i))
            for (int j = y; j < m; j -= j & (-j))
                bit[i][j] += delta;    }
};
```

Việc sửa đổi các hàm khác cũng được thực hiện tương tự. Về lý thuyết có thể tổ chức BIT n chiều.

2.2 Bài tập

Bài 1. Số nghịch thế (INVERS.*) – Nguồn thầy Nguyễn Thanh Tùng

Xét dãy số nguyên $A = (a_1, a_2, \ldots, a_n)(1 \le n \le 500\,000)$. Các số trong dãy A khác nhau từng đôi một và nhận giá trị trong phạm vi từ 1 đến n. Như vậy dãy A là một hoán vị các số từ 1 đến n. Cặp số (a_i, a_j)trong dãy A được gọi là một nghịch thế, nếu $i < j$ và $a_i > a_j$.

Yêu cầu: Cho n và hoán vị A. Hãy xác định số nghịch thế.

Dữ liệu: Vào từ file văn bản INVERS.INP:

- Dòng đầu tiên chứa số nguyên n,
  Dòng thứ 2 chứa n số nguyên xác định hoán vị A.

Kết quả: Đưa ra file văn bản INVERS.OUT một số nguyên – số lượng nghịch thế.

Ví dụ:

INVERS.INP	INVERS.OUT
5 2 4 3 5 1	5

*** Hướng dẫn và code:**

Với mỗi ai, i = 2 ÷ n kiểm tra xem có bao nhiêu j thỏa mãn các điều kiện j < i và aj > ai. Chương trình giải đơn giản, chứa 2 chu trình lồng nhau và có độ phức tạp $O(n^2)$. Để tiện so sánh, đánh giá độ phức tạp giải thuật ta sẽ đưa ra thời gian thực hiện chương trình.
Để so sánh với các giải thuật khác, kết quả ở chương trình này sẽ được đưa ra file INVERS.ANS.

```
#include <fstream>
#include <ctime>
using namespace std;
ifstream fi ("invers.inp");
ofstream fo ("invers.ans");
int n,a[500001];
int64_t res=0;
int main()
{
fi>>n;
for(int i=1;i<=n;++i)fi>>a[i];
for(int i=2;i<=n;++i)
for(int j=1;j<i;++j) if(a[j]>a[i])++res;
fo<<res;
}
```

Với n lớn giải thuật trên không hiệu quả vì 2 lý do:

- Thông tin nhận được khi xử lý mỗi a_i (i = 2 ÷ n) không được lưu lại hỗ trợ cho việc xử lý giá trị tiếp theo.
- Kết quả res được tích lũy một cách thô thiển: lần lượt tăng 1 khi gặp cặp nghịch thế.

Muốn nâng cao hiệu quả của giải thuật cần phải quản lý được thông tin về các dữ liệu đã xét và dùng nó để hỗ trợ cho việc xử lý tiếp theo. Trên cơ sở đó kết quả res có thể được tích lũy theo bước lớn hơn 1. Điều này cũng sẽ làm giảm đáng kể thời gian thực hiện giải thuật.

Cấu trúc dữ liệu cây Fenwick giúp chúng ta đồng thời giải quyết được cả 2 yêu cầu trên với hiệu quả đủ cao và chi phí lập trình không lớn. Tồn tại những cấu trúc dữ

liệu còn mang lại hiệu quả cao hơn nhưng đòi hỏi bộ nhớ lớn và chi phí lập trình rất cao!

*Code mẫu:

```
#include <fstream>
#include <ctime>
using namespace std;
ifstream fi ("invers.inp");
ofstream fo ("invers.out");
int n,a[500001];
int64_t t,res=0,s[500001]={0};
void insert_t(int ii)
{
   while(ii<=n)
   {
      s[ii]++;
      ii+=(ii&(-ii));
   }
}
int sum_t(int ii)
{
   int64_t r;
   r=0;
   while (ii>0)
   {
      r+=s[ii];
      ii&=(ii-1);
   }
   return(r);
}
int main()
{
   fi>>n;
   for(int i=1; i<=n; ++i)
      fi>>a[i];
   insert_t(a[n]);
   for(int i=n-1; i>=1; --i)
   {
      t=sum_t(a[i]-1);
      insert_t(a[i]);
      res+=t;
   }
   fo<<res;
}
```

Bài 2. Xếp hàng (XEPHANG.*)

Hàng ngày khi lấy sữa, N con bò của bác John ($1 \le N \le 50000$) luôn xếp hàng theo thứ tự không đổi. Một hôm bác John quyết định tổ chức một trò chơi cho một số con bò. Để đơn giản, bác John sẽ chọn ra một đoạn liên tiếp các con bò để tham dự trò chơi. Tuy nhiên để trò chơi diễn ra vui vẻ, các con bò phải không quá chênh lệch về chiều cao.

Bác John đã chuẩn bị một danh sách gồm Q ($1 \le Q \le 200000$) đoạn các con bò và chiều cao của chúng (trong phạm vi [1, 1000000]). Với mỗi đoạn, bác John muốn xác định chênh

lệch chiều cao giữa con bò thấp nhất và cao nhất. Bạn hãy giúp bác John thực hiện công việc này!

Dữ liệu vào:

- Dòng đầu tiên chứa 2 số nguyên N và Q.
- Dòng thứ i trong số N dòng sau chứa 1 số nguyên duy nhất, là độ cao của con bò thứ i.
- Dòng thứ i trong số Q trong tiếp theo chứa 2 số nguyên A, B ($1 \le A \le B \le N$), cho biết đoạn các con bò từ A đến B.

Kết quả:

Gồm Q dòng, mỗi dòng chứa 1 số nguyên, là chênh lệch chiều cao giữa con bò thấp nhất và cao nhất thuộc đoạn tương ứng.

XEPHANG.inp	XEPHANG.out
6 3 1 7 3 4 2 5 1 5 4 6 2 2	6 3 0

*** Hướng dẫn và code:**

Bài này sử dụng Fenwick Tree để tìm min/max. Xem đoạn code bên dưới để hiểu hơn

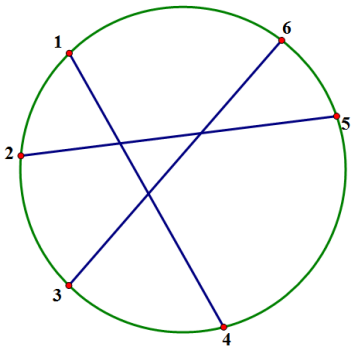
```
#include <bits/stdc++.h>
using namespace std;
void minimize(int &a, int b) {
   if (a > b) a = b;
}
void maximize(int &a, int b) {
   if (a < b) a = b;
}
typedef void (*func)(int &, int);
struct Fenwick {
   int n, init;
   vector<int> f, a;
   func update;
   Fenwick(int n, int val, func func):
      n(n), init(val),
      f(n+1, val), a(n+1),
      update(func) {}
   int get(int l, int r) {
      int result = init;
      while (l <= r) {
         if (r-(r&-r) >= l) update(result, f[r]), r -= r&-r;
         else update(result, a[r]), r -= 1;
      }
      return result;
   }
   void set(int i, int x) {
      a[i] = x;
```

```
        for (; i<=n; i += i&-i) update(f[i], x);
    }
};
int main() {
    //ios::sync_with_stdio(false); cin.tie(0);
    freopen("XEPHANG.inp","r",stdin);
    freopen("XEPHANG.out","w",stdout);
    int n, q; cin >> n >> q;
    Fenwick fmax(n, 0, maximize);
    Fenwick fmin(n, 1e8, minimize);
    for (int i=1; i<=n; i++) {
        int x; cin >> x;
        fmax.set(i, x);
        fmin.set(i, x);
    }
    while (q--) {
        int l, r; cin >> l >> r;
        cout << fmax.get(l, r) - fmin.get(l, r) << endl;
    }
    return 0;
}
```

Ở đây ta sử dụng 2 cây Fenwick là max, min để lưu giá trị lớn nhất, nhỏ nhất trong một đoạn.

Bài 3. Dây cung (DAYCUNG.*) – Nguồn sưu tầm

Có 2n điểm khác nhau được đánh dấu trên đường tròn. Các điểm được đánh số từ 1 tới 2n theo chiều ngược kim đồng hồ ($1 \le n \le 100\,000$).



Vẽ n dây cung, dây thứ i nối hai điểm a_i và b_i. Mỗi điểm đã cho chỉ thuộc đúng một dây cung.

Yêu cầu: Cho n và các số a_i, b_i ($1 \le i \le n$). Hãy xác định số cặp dây cung giao nhau.

Dữ liệu vào: Đọc từ file **DAYCUNG.INP**

- Dòng đầu tiên chứa số nguyên n.
- Dòng thứ i trong n dòng sau chứa hai số nguyên a_i và b_i.

Dữ liệu ra: Ghi ra file **DAYCUNG.OUT**

- Một số nguyên là số cặp dây cung giao nhau.

Ví dụ:

DAYCUNG.INP	DAYCUNG.OUT
3 1 4 2 5 3 6	3

Giới hạn:

- 50% số test có $1 \le N \le 1000$.
- 50% số test với các trường hợp còn lại.

*** Hướng dẫn và code:**

Dùng mảng cặp dữ liệu p = {u, v}, u < v – xác định cung (u, v),
Tổ chức 2 mảng b[200001] và c[200001], bi/ci – số lượng điểm đầu/cuối trong khoảng [1, i], thao tác với mảng: xử lý cây Fenwick,
Sắp xếp p theo thứ tự tăng dần,
Với i = 1 ÷ n:
Xác định số cung cắt cung thứ i: t=sum_t(b,p[i].second) - sum_t(c,p[i].second);

Tích lũy t vào kết quả,
Xóa cung thứ i: cập nhật lại b và c: upd_t(b,p[i].first); upd_t(c,p[i].second);
Độ phức tạp của giải thuật: O(nlogn).
*Code:

```
#include <bits/stdc++.h>
using namespace std;
ifstream fi ("daycung.inp");
ofstream fo ("daycung.out");
int n,n2,u,v,t,b[200001]= {0},c[200001]= {0};
pair<int,int> p[100001];
int64_t res=0;
void insert_t(int a[],int x)
{
   while(x<=n2)
   {
      a[x]++;
      x+=(x&(-x));
   }
}
int sum_t(int a[],int x)
{
   int tg=0;
   while(x>0)
   {
      tg+=a[x];
      x&=(x-1);
   }
   return(tg);
}
void upd_t(int a[], int x)
{
   while(x<=n2)
   {
      a[x]--;
      x+=(x&(-x));
   }
}
int main()
{
   fi>>n;
   n2=n<<1;
   for(int i=1; i<=n; ++i)
   {
      fi>>u>>v;
      if(u>v)  {
         t=u;  u=v;  v=t;
      }
      p[i].first=u;      p[i].second=v;
      insert_t(b,u);    insert_t(c,v);
   }
   sort(p+1,p+n+1);
```

```
    for(int i=1; i<=n; ++i)
    {
        t=sum_t(b,p[i].second) -sum_t(c,p[i].second);
        res+=t;
        upd_t(b,p[i].first);
        upd_t(c,p[i].second);
    }
    fo<<res;
}
```

Bài 4. Cuộc chiến bảo vệ HOGVARTS (BATLE.*)

Hogvarts chuẩn bị chống trả sự tấn công của các thế lực đen tối. Tuyến tiền tiêu ở ngoài cùng có ***n*** vị trí nằm thành một hàng. Các vị trí được đánh số từ 1 đến ***n*** từ trái qua phải, mỗi vị trí có một người. Năng lực chiến đấu của mỗi người có giá trị nguyên, nằm trong phạm vi từ 1 đến ***n*** và khác nhau từng đôi một. Người ở vị trí ***i*** có năng lực chiến đấu là a_i. Kết quả bố trí các tuyến phòng thủ theo chiều sâu ở phía trong cho thấy toàn bộ hệ thống phòng thủ sẽ phát huy được tối đa sức mạnh của mình khi ở tuyến tiền tiêu năng lực chiến đấu được bố trí tăng dần từ trái qua phải.

Harry được cử đi tổ chức lại tuyến tiền tiêu. Khả năng đánh giá năng lực chiến đấu của từng người ở Harry là rất tốt. Khi thấy một cặp 2 người ở các vị trí cạnh nhau mà người bên trái có năng lực chiến đấu cao hơn người bên phải Harry cho 2 người này đổi chỗ cho nhau. Về lý
thuyết, nếu Harry đi duyệt từ đầu đến cuối ***n***-1 lần thì đảm bảo chắc chắn tuyến tiền tiêu sẽ có cấu hình bố trí tối ưu. Đáng tiếc, thời gian không còn nhiều và Harry chỉ kịp đi duyệt có ***k*** lần.

Hãy xác định năng lực chiến đấu của các vị trí ở tuyến tiền tiêu trước khi trận chiến diễn ra.

Dữ liệu: Vào từ file văn bản BATLE.INP:

- Dòng đầu tiên chứa 2 số nguyên ***n*** và ***k*** ($1 \le n \le 2\times105$, $0 \le k \le n$-1)
- Dòng thứ 2 chứa ***n*** số nguyên $a_1, a_2, \ldots, a_n$.

Kết quả: Đưa ra file văn bản BATLE.OUT trên một dòng ***n*** số nguyên – năng lực chiến đấu ở các vị trí từ trái sang phải trên tuyến tiền tiêu sau khi bố trí lại.

Ví dụ:

BATLE.INP	BATLE.OUT
4 1 4 1 3 2	1 3 2 4

*** Hướng dẫn và code:**

Gọi c_i – số nghịch thế của ai tính từ đầu dãy,

$r_i = 0\ \forall i$

Nếu $c_i \le k$ → a_i về vị trí không có nghịch thế , ngược lại → a_i sang trái k vị trí $r_i - k = a_i$, đánh dấu $b_{a[i]} = -1$ → đã định vị.

Bắt đầu từ phần tử t = 1: với i = 1 ÷ n: nếu $r_i = 0$ (chưa có kết quả) – tìm phần tử đầu tiên chưa
định vị while(b[t]<0)++t; đưa phần tử này vào vị trí i: r_i = t; chuyển sang phần tử tiếp theo: ++t;

*Code mẫu:

```
#include <bits/stdc++.h>
int a[200001],b[200001]= {0},c[200001],r[200001]= {0},n,k,t;
using namespace std;
ifstream fi ("Batle.inp");
ofstream fo ("Batle.out");
void insert_t(int x)
{
```

```
    while(x<=n)
    {
        b[x]++;
        x+=(x&(-x));
    }
}
int sum_t(int x)
{
    t=0;
    while(x>0)
    {
        t+=b[x];
        x&=(x-1);
    }
    return(t);
}
int main()
{
    fi>>n>>k;
    for(int i=1; i<=n; ++i)
        fi>>a[i];
    insert_t(a[1]);
    for(int i=2; i<=n; ++i)
    {
        insert_t(a[i]);
        c[i]=i-sum_t(a[i]);
    }
    for(int i=1; i<=n; ++i)
        if(c[i]<=k)
            c[i]=0;
        else
        {
            c[i]-=k;
            r[i-k]=a[i];
            b[a[i]]=-1;
        }
    t=1;
    for(int i=1; i<=n; ++i)
        if(r[i]==0)
        {
            while(b[t]<0)
                ++t;
            r[i]=t++;
        }
    for(int i=1; i<=n; ++i)
        fo<<r[i]<<" ";
}
```

Bài 5. Hộp kẹo (Candies.*) – Nguồn sưu tầm

Các bạn gọi điện thoại cho Steve hẹn đến nhà chia vui với kết quả cao mà Steve đã đạt được trong kỳ thi Tin học vừa kết thúc. Steve đi mua ***n*** hộp kẹo để đón bạn, mỗi hộp một loại kẹo và hộp thứ ***i*** có a_i viên.

Có tất cả ***m*** người tới. Các bạn tới không cùng một lúc mà là lần lượt từng người một. Steve hiểu rất rõ các bạn của mình. Người thứ ***j*** có độ tế nhị b_j. Điều này có nghĩa là bạn đó sẽ chỉ ăn kẹo ở các hộp có số lượng còn lại không ít hơn b_j chiếc và sẽ ăn ở những hộp này, mỗi hộp một viên.

Nếu một bạn nào đó có độ tế nhị 1 thì bạn đó sẽ ăn ở mỗi hộp một viên kẹo. Chiều tối, khi các bạn đã về hết, Steve vừa dọn dẹp vừa nhẩm tính xem mỗi bạn đã ăn bao nhiêu
viên kẹo.

Dữ liệu: Vào từ file văn bản CANDIES.INP:

- Dòng đầu tiên chứa số nguyên ***n*** ($1 \le n \le 10^5$),
- Dòng thứ 2 chứa ***n*** số nguyên $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$, $i = 1 \div n$),
- Dòng thứ 3 chứa số nguyên ***m*** ($1 \le m \le 10^5$),
- Dòng thứ 4 chứa ***m*** số nguyên $b_1, b_2, \ldots, b_m$ ($1 \le b_j \le 10^9$, $j = 1 \div m$).

Kết quả: Đưa ra file văn bản CANDIES.OUT ***m*** số nguyên, mỗi số trên một dòng. Số thứ ***j*** là số
viên kẹo bạn thứ ***j*** đã ăn.

Ví dụ:

CANDIES.INP	CANDIES.OUT
3 3 1 1 2 1 2	3 1

*** Hướng dẫn và code mẫu:**

Cây Fenwic quản lý số lần truy nhập tới các hộp kẹo

Code mẫu:

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<int,int> p2;
int
n,m,k,t,a[100002],c[100001]= {0},d[100001],b[100001],s[100001]= {0};
ifstream fi ("CANDIES.INP");
ofstream fo ("CANDIES.OUT");
void insert_t(int ii)
{
   while(ii<=n)
   {
      s[ii]++;
      ii+=(ii&(-ii));
   }
}
int sum_t(int ii)
{
   int r=0;
   while(ii>0)
   {
      r+=s[ii];
      ii&=(ii-1);
   }
   return(r);
}
void b_search(int x,int ii)
```

```
{
   int l,r,u;
   l=1;
   r=n;
   if(x>a[1]-ii+1+c[0])
   {
      k=0;
      ++c[0];
      d[ii]=0;
      return;
   }
   while(l<=r)
   {
      u=(l+r)/2;
      if (u>1)
         t=ii-1-sum_t(u-1)-c[0];
      else
         t=ii-1-c[0];
      if(x<=a[u]-t)
         l=u+1;
      else
         r=u-1;
   }
   k=r;
   d[ii]=k;
   ++c[k];
   if(k>0)
      insert_t(k);
}
int main()
{
   fi>>n;
   for(int i=1; i<=n; ++i)
      fi>>a[i];
   fi>>m;
   for(int i=1; i<=m; ++i)
      fi>>b[i];
   sort(a+1,a+n+1,greater<int>());
   a[n+1]=0;
   a[0]=a[1]+1;
   for(int j=1; j<=m; ++j)
   {
      b_search(b[j],j);
   }
   for(int j=1; j<=m; ++j)
      fo<<d[j]<<endl;
   return 0;
}
```

Bài 6. Lấy sỏi (Gravel.*) - Nguồn Codechef

Bob có n đống sỏi (ban đầu có đúng c viên trong mỗi đống). Anh ta muốn thực hiện m truy vấn, mỗi truy vấn có dạng:

- Thêm các viên sỏi vào đống từ u tới v, đúng k viên cho mỗi đống.
- Hoặc trả lời có bao nhiêu viên trong đống p tại thời điểm đó.

Yêu cầu: Giúp Bob trả lời các truy vấn loại 2.

Dữ liệu vào:

- Dòng đầu chứ các số nguyên n, m, c tương ứng.
- m dòng tiếp theo có dạng:
    - **S u v k** mô tả truy vấn loại 1.
    - **Q p** mô tả truy vấn loại 2.

Dữ liệu ra:

Với mỗi truy vấn loại 2, in ra một số nguyên trên mỗi dòng trả lời các truy vấn theo tuần tự hỏi.

Ví dụ

Gravel.inp	Gravel.out
7 5 0 Q 7 S 1 7 1 Q 3 S 1 3 1 Q 3	0 1 2

Giới hạn:

- $0<n\leq 10^6$
- $0<m\leq 250\ 000$
- $0<u\leq v\leq n$
- $0\leq c,k\leq 10^9$
- $0<p\leq n$

*** Hướng dẫn và code mẫu:**

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;

typedef long long ll;
typedef pair<int, int> par;
typedef vector<int> vi;
typedef vector<par> vp;
typedef vector<vi> graph;

typedef vector<vp> wgraph;

#define rep(i, n) for (int i = 0; i < (int)n; ++i)
#define repx(i, a, n) for (int i = a; i < (int)n; ++i)
#define invrep(i, a, b) for (int i = b; i-- > (int)a;)
#define pb push_back
#define eb emplace_back

#define debugx(x) cerr << #x << ": " << x << endl

#define debugv(v)        \
   cerr << #v << ":";    \
   for (auto e : v)      \
   {                     \
```

```
        cerr << " " << e; \
    }                   \
    cerr << endl

#define debugm(m)             \
    cerr << #m << ":\n";      \
    for (auto v : m)          \
    {                         \
        for (auto e : v)      \
            cerr << e << " "; \
        cerr << "\n";         \
    }                         \
    cerr << endl

template <typename T1, typename T2>
ostream &operator<<(ostream &os, const pair<T1, T2> &p)
{
    os << '(' << p.first << ',' << p.second << ')';
    return os;
}
#define int ll
struct FenwickTree
{
    vector<ll> FT;
    FenwickTree(int N)
    {
        FT.resize(N + 1, 0);
    }

    ll query(int i)
    {
        int ans = 0;
        for (; i; i -= i & (-i))
            ans += FT[i];
        return ans;
    }

    int query(int i, int j)
    {
        return query(j) - query(i - 1);
    }

    void update(int i, int v)
    {
        int s = 0;//query(i, i); // Sets range to v?
        for (; i < FT.size(); i += i & (-i))
            FT[i] += v - s;
    }

    // Queries puntuales, Updates por rango
    void update(int i, int j, int v)
```

```
    {
        update(i, v);
        update(j + 1, -v);
    }
};
    #undef int
int main()
{
    #define int ll
    freopen("Gravel.inp","r",stdin);
    freopen("Gravel.out","w",stdout);
    ios::sync_with_stdio(0);
    cin.tie(0);
    int n, m, c;
    cin >> n >> m >> c;
    FenwickTree st(n + 1);
    st.update(1, n + 1, c);
    char C;
    int u, v, k;
    rep(_, m)
    {
        cin >> C >> u;
        if (C == 'S')
        {
            cin >> v >> k;
            st.update(u, v, k);
        }
        else
            cout << st.query(u) << '\n';
    }
}
```

2.3 Một số bài tập luyện tập:

Bài 1. Tổng GCD (GCDSUM.*)

Cho hàm F được định nghĩa là : F(x)=GCD(1,x)+GCD(2,x)+...+GCD(x,x) ở đây GCD là ước số chung lớn nhất.

Cho một mảng A có N phần tử, có 2 kiểu truy vấn sau:

1. **C X Y** : Tính giá trị F(A[X])+F(A[X+1])+F(A[X+2])+....+F(A[Y])(mod(10^9+7))
2. **U X Y**: Cập nhật phần tử thứ X của mảng A[X]=Y

Dữ liệu vào:

- Dòng đầu chứ số nguyên N số phần tử của mảng
- Dòng tiếp theo chứa N số nguyên của mảng A viết cách nhau bởi dấu cách.
- Dòng tiếp theo ghi số Q, số truy vấn
- Q dòng tiếp là 1 trong 2 loại truy vấn trên

Kết quả :

Với mỗi truy vấn loại 1, in ra tổng (mod(10^9+7)).

Giới hạn:

- $1 \le N \le 10^6$
  $1 \le Q \le 10^5$
  $1 \le Ai \le 5.10^5$

- $1 \le X \le N$
  $1 \le Y \le 5.10^5$
- $1 \le X \le Y \le N$

Ví dụ:

GCDSUM.INP	GCDSUM.OUT
3 3 4 3 6 C 1 2 C 1 3 C 3 3 U 1 4 C 1 3 C 1 2	13 18 5 21 16

* Code:

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1e6 + 5;
const int MAXV = 5e5;
#define ll long long int
const ll MOD = 1e9 + 7;
int N, Q;
ll a[MAXN], phi[MAXV + 5], p[MAXV + 5], BIT[MAXN];
ll inline mod(ll x)
{
   return (x%MOD + MOD)%MOD;
}
void compute_phi()
{
   for(int i = 1; i <= MAXV; i++)
      phi[i] = i;
   for(int i = 2; i <= MAXV; i++)
      if (phi[i] == i)   // prime number
      {
         for(int j = i; j <= MAXV; j += i)
         {
            phi[j] -= phi[j] / i;
            phi[j] = mod(phi[j]);
         }
      }
}
void compute_pillai()
{
   for(int i = 1; i <= MAXV; i++)
      for(int j = i; j <= MAXV; j += i)
      {
         p[j] += i * phi[j / i];
         p[j] = mod(p[j]);
      }
}
```

```
void update(int i, ll val)
{
   for(; i <= N; i += i&-i)
   {
      BIT[i] += val;
      BIT[i] = mod(BIT[i]);
   }
}
ll query(int i)
{
   ll sum = 0;
   for(; i > 0; i -= i&-i)
   {
      sum += BIT[i];
      sum = mod(sum);
   }
   return sum;
}
int32_t main()
{
   freopen("GCDSUM.inp","r",stdin);
   freopen("GCDSUM.out","w",stdout);
   compute_phi();
   compute_pillai();
   ll n;
   scanf("%lld",&n);
   ll arr[n];
   for(int i=0; i<n; i++)
   {
      scanf("%lld",&arr[i]);
      arr[i]=p[arr[i]]%1000000007;
   }
   ll BIT[n+1]= {0};
   for(int i=0; i<n; i++)
   {
      ll x=i+1;
      while(x<=n)
      {
         BIT[x]+=arr[i];
         x+=(x&-x);
      }
   }
   ll q;
   scanf("%lld",&q);
   while(q--)
   {
      char c;
      scanf(" %c",&c);
      ll x,y;
      scanf("%lld",&x);
      scanf("%lld",&y);
```

```
        if(c=='C')
        {
            ll sum=0;
            x--;
            while(x>0)
            {
                sum-=BIT[x];
                x-=(x&-x);
            }
            while(y>0)
            {
                sum+=BIT[y];
                if(sum>=1000000007)
                    sum%=1000000007;
                y-=(y&-y);
            }
            printf("%lld\n",sum);
        }
        else
        {
            ll val=p[y]%1000000007;
            ll diff=val-arr[x-1];
            arr[x-1]=val;
            while(x<=n)
            {
                BIT[x]+=diff;
                x+=(x&-x);
            }
        }
    }
}
```

Bài 2. Mảng (Array.*)

Chef có một mảng vòng tròn A gồm N số nguyên $A_1,A_2 \ldots A_n$ trong đó A_N và A_1 được coi là liền kề. Mảng con Ai,j được định nghĩa là một mảng tạo thành bằng các phần tử từ chỉ số i đến chỉ số j (bao gồm cả i, j). Về mặt hình thức Ai,j được định nghĩa là:

- Nếu i≤j, thì mảng con của A từ chỉ số i đến j, bao gồm cả i và j.
- Nếu i>j, thì nó biểu thị mảng con của A từ chỉ số i đến N, tiếp theo là mảng con của A từ chỉ số 1 đến j.

Một hàm f(B) được xác định trên mảng B là số mảng con của B có tổng nhỏ hơn K.

Cho $S=\sum_{i=1}^{n}\sum_{j}^{n} f(A_{i,j})$. Tìm giá trị của S khi chia dư cho 1,000,000,007.

Input:

- Dòng đầu tiên là T số lượng trường hợp. Mỗi trường hợp T bao gồm.
- Dòng đầu tiên của mỗi trường hợp là 2 số nguyên N và K.
- Dòng thứ hai là N số nguyên $A_1,A_2,\ldots,A_N$.

Output:

Với mỗi trường hợp kiểm tra, in ra một số nguyên duy nhất S chia dư cho 1,000,000,007.

Ràng buộc:

- $1 \le T \le 10$
- $1 \le N \le 2 * 10^5$

- $|K| \leq 10^{15}$
- $|Ai| \leq 10^9$ với mỗi i

Ví dụ:

Array.inp	Array.out
3 3 1 0 0 0 2 1 1 -1 1 1 -1	30 5 1

Giải thích:

Trong trường hợp 2, mảng là [1,−1] và K=1.

$f(A_1,1)=0$

$f(A_1,2)=2$, vì 2 mảng con [1,−1] và [−1] có tổng <K.

$f(A_2,1)=2$, vì mảng ta xét bây giờ là [−1,1], và 2 mảng con [1,−1] và [−1] có tổng <K.

$f(A_2,2)=1$

Tổng ở đây là 5, đó là câu trả lời.

*Code mẫu:

```
#include <bits/stdc++.h>
using namespace std;
#ifdef NeverBeRed
#include "debug.h"
#else
#define debug(...) 9715
#endif
typedef long long ll;
typedef long double ld;
typedef complex<ld> point;
#define F first
#define S second
template<typename T, typename U>
T pow_mod(T a, U b, int mod)
{
	T r = 1;
	for (; b > 0; b >>= 1)
	{
		if (b & 1) r = (ll)r * a % mod;
		a = (ll)a * a % mod;
	}
	return r;
}
const ll mod = 1e9+7, inv2 = pow_mod(2, mod-2, mod);
template<typename T>
struct fenwick
{
	int n;
	vector<T> f;
	fenwick(int n) : n(n), f(n + 1) {}
	T query(int p)
```

```
	{
		T ret = 0;
		for (++p; p > 0; p -= p & -p)
			ret += f[p];
		return ret;
	}
	void update(int p, T x)
	{
		for (++p; p <= n; p += p & -p)
			f[p] += x;
	}
};
int main()
{
	ios_base::sync_with_stdio(0), cin.tie(0);
	freopen("array.inp","r",stdin);
	freopen("array.out","w",stdout);
	int t;
	cin >> t;
	while (t--)
	{
		ll n, k;
		cin >> n >> k;
		vector<ll> a(n);
		for (auto &i : a) cin >> i;
		for (int i = 0; i < n; ++i)
			a.push_back(a[i]);
		a.insert(a.begin(), 0);
		partial_sum(a.begin(), a.end(), a.begin()); //partial_sum giống như tổng tiền tố
		vector<ll> b;
		for (auto i : a)
		{
			b.push_back(i);
			b.push_back(i-k);
		}
		sort(b.begin(), b.end());
		b.erase(unique(b.begin(), b.end()), b.end());
		auto get = [&](ll x)
		{
			return lower_bound(b.begin(), b.end(), x) - b.begin();
		};
		fenwick<ll> f(b.size() + 5), f2(b.size() + 5), s(b.size() + 5);
		s.update(get(0), +1);
		ll ans = 0;
		for (ll i = 1; i < 2*n; ++i)
		{
			if (i > n)
			{
				f.update(get(a[i-n-1]), -(i-n-1));
				f2.update(get(a[i-n-1]), -(i-n-1)*(i-n-1));
				s.update(get(a[i-n-1]), -1);
```

```
                }
                ll x = a[i];
                ll cnt = (s.query(b.size() + 2) - s.query(get(x - k))) % mod;
                ll sum = (f.query(b.size() + 2) - f.query(get(x - k))) % mod;
                ll sum2 = (f2.query(b.size() + 2) - f2.query(get(x - k))) % mod;
                ll len = (i * cnt + mod - sum) % mod;
                ll len2 = ((cnt * i % mod * i - 2 * i * sum + sum2) % mod + mod) % mod;
                ans += (cnt * (n+1) % mod * (n+1) + -2 * (n+1) * len + (n+1) * cnt - len +
len2) % mod;
                ans = (ans + mod) % mod;
                if (i < n)
                {
                        f.update(get(x), i);
                        f2.update(get(x), i*i);
                        s.update(get(x), 1);
                }
        }
        cout << ans * inv2 % mod << "\n";
    }
    return 0;
}
```

Bài 3. Trung vị của dãy số (MEDIAN.*)

Cho một cây gồm N nút, mỗi nút **i** có giá trị là $\mathbf{A_i}$. John phải tìm số cặp không có thứ tự (i, j) sao cho: Nếu chúng ta tạo một mảng gồm tất cả các số A_k, như vậy **k** là một nút trong đường đi đơn từ nút i đến nút j, trung vị của mảng này sẽ không vượt quá **X** và sẽ ít nhất là **Y**.

John thấy nhiệm vụ hơi khó, và anh ấy muốn dành toàn bộ thời gian cho việc học trong năm tới, vì vậy anh ấy muốn nhờ bạn giúp đỡ vấn đề đó.

Chú ý:

- Trung vị của mảng là phần tử nằm ở giữa mảng sau khi sắp xếp theo thứ tự không giảm.
- Giá trị trung bình của một mảng là tổng của tất cả các phần tử, chia cho độ dài của mảng.
- Bạn có thể xem xét đối với các mảng có độ dài chẵn rằng phần trung vị là một trong hai phần tử ở giữa bên trái của mảng sau khi sắp xếp nó.

Dữ liệu vào

- Dòng đầu tiên là số nguyên T số trường hợp.
- Mỗi trường hợp là 3 số nguyên **N**, **X** and **Y**, biểu thị số lượng nút, giá trị lớn nhất của trung vị và giá trị nhỏ nhất của trung vị tương ứng.
- Dòng tiếp theo chứa N số nguyên $\mathbf{A_1}$, $\mathbf{A_2}$, **...**, $\mathbf{A_N}$, biểu thị giá trị của nút.
- **N-1** dòng tiếp. Mỗi dòng có 2 giá trị: **u** và **v**, biểu thị kết nối giữa các nút **u** và **v**.

Dữ liệu ra

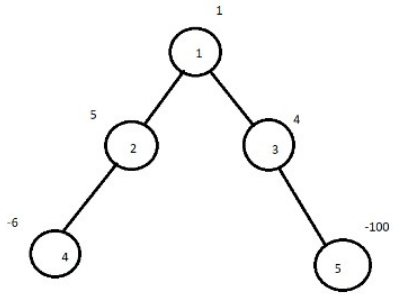
Với mỗi trường hợp, in ra một số nguyên, số cặp (u, v) thỏa mãn điều kiện nêu trên. Có thể là u = v.

Ràng buộc

- $\mathbf{1 \le T \le 100.}$
- $\mathbf{1 \le N \le 10^5.}$
- $\mathbf{-10^9 \le X,Y \le 10^9}$.
- $\mathbf{-10^9 \le A_i \le 10^9.}$
- Đảm bảo rằng tổng của tất cả N trên tất cả các trường không vượt quá 2 * 105.

Ví dụ:

MEDIAN.inp	MEDIAN.out
1 5 4 0 1 5 4 -6 -100 4 2 2 1 5 3 1 3	7



Diễn giải:

Tất cả các cặp là: { **(1,1) (1,2) (1,3) (1,4) (1,5) (2,2) (2,3) (2,4) (2,5) (3,3) (3,4) (3,5) (4,4) (4,5) (5,5)** }. Các cặp hợp lệ là: { **(1,1) (1,2) (1,3) (1,4) (2,3) (3,3) (3,4)** }.

Cặp (1,5) không hợp lệ bởi vì nếu chúng ta lấy tất cả các giá trị của nút trong đường dẫn từ nút 1 đến nút 5, mảng sẽ là: {1, 4, -100} và giá trị trung bình là: -31,666 nhỏ hơn 0.

Cặp (2,2) không hợp lệ vì trung vị của mảng {5} là 5 lớn hơn 4

* Code mẫu:

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long int
const int MAX=1e5+5;
vector<ll> adj[MAX];
int vis[MAX];
int sub[MAX];
ll a[MAX];
ll n;
ll X,Y;
ll ans;
ll next1;
vector<pair<ll,ll> > paths;
vector<pair<ll,ll> > paths1;
int bit[MAX];
ll meanval;
ll medianval;

void update(ll x,ll val,ll LIM)
{
   while(x<=LIM)
   {
      bit[x]+=val;
      x+=(x&(-x));
   }
}
ll query(ll x)
{
   ll an=0;
   while(x)
   {
      an+=bit[x];
```

```
        x-=(x&(-x));
    }
    return an;
}
void dfs(ll s,ll p)
{
    next1++;
    sub[s]=1;
    for(ll i=0; i<adj[s].size(); i++)
    {
        ll u=adj[s][i];
        if(u!=p&&!vis[u])
        {
            dfs(u,s);
            sub[s]+=sub[u];
        }
    }
}
ll find(ll s,ll p)
{
    for(ll i=0; i<adj[s].size(); i++)
    {
        ll u=adj[s][i];
        if(u!=p&&!vis[u]&&sub[u]>(next1/2))
        {
            return find(u,s);
        }
    }
    return s;
}

void add(ll s,ll p,ll total,ll val)
{
    total+=((a[s]<=X)?1:-1);
    val+=(a[s]-Y);
    paths.push_back(make_pair(total,val));
    for(ll i=0; i<adj[s].size(); i++)
    {
        ll u=adj[s][i];
        if(u!=p&&!vis[u])
        {
            add(u,s,total,val);
        }
    }
}
bool cmp(pair<ll,ll> a,pair<ll,ll> b)
{
    return (a.second<b.second);
}
ll query(vector<pair<ll,ll> > pr)
{
```

```
    ll f=0;
    sort(pr.begin(),pr.end(),cmp);
    ll max1=-1e6;
    ll min1=1e6;
    for(ll i=0; i<pr.size(); i++)
    {
        max1=max(max1,pr[i].first);
        min1=min(min1,pr[i].first);
    }
    ll ran=max1-min1+1;
    ll j=pr.size();
    for(ll i=0; i<pr.size(); i++)
    {
        while((j-1)>=0 && (pr[i].second+pr[j-1].second + meanval)>=0)
        {
            update(pr[j-1].first-min1+1,1,ran);
            j--;
        }
        ll req=(-pr[i].first-medianval);
        req=req+1-min1;
        req=max(req,(ll)1);
        req=min(ran+1,req);
        f+=query(ran)-query(req-1);
    }
    j=pr.size();
    for(ll i=0; i<pr.size(); i++)
    {
        while((j-1)>=0 && (pr[i].second+pr[j-1].second + meanval)>=0)
        {
            update(pr[j-1].first-min1+1,-1,ran);
            j--;
        }
    }
    return f;
}

void solve(ll s)
{
    next1=0;
    dfs(s,0);
    ll c=find(s,0);
    if(a[c]<=X && a[c]>=Y)
        ans++;
    ll res=0;
    meanval=a[c]-Y;
    medianval=(a[c]<=X)?1:-1;
    for(ll i=0; i<adj[c].size(); i++)
    {
        ll u=adj[c][i];
        if(!vis[u])
        {
```

```
            add(u,c,0,0);
            res-=query(paths);
            for(ll j=0; j<paths.size(); j++)
            {
               if((paths[j].first+medianval)>=0 &&
                     (paths[j].second+meanval)>=0)
               {
                  ans++;
               }
               paths1.push_back(paths[j]);
            }
            paths.clear();
         }
      }
      res+=query(paths1);
      paths1.clear();
      vis[c]=1;
      ans+=(res/2);
      for(ll i=0; i<adj[c].size(); i++)
      {
         ll u=adj[c][i];
         if(!vis[u])
            solve(u);
      }
   }

   int main()
   {
      freopen("median.inp","r",stdin);
      freopen("median.out","w",stdout);
      ll t;
      cin>>t;;
      while(t--)
      {
         ans=0;
         cin>>n>>X>>Y;
         ll i;
         for(i=1; i<=n; i++)
         {
            adj[i].clear();
            vis[i]=0;
            cin>>a[i];
         }
         for(i=1; i<n; i++)
         {
            ll a,b;
            cin>>a>>b;
            adj[a].push_back(b);
            adj[b].push_back(a);
         }
         solve(1);
```

```
        cout<<ans<<"\n";
    }
    return 0;
}
```

Bài 4. Mèo tân binh (RKCAT.*)

Tanya vẫn yêu mèo, ngay cả sau những đau khổ mà cô ấy đã trải qua vì chúng. Sau khi tập luyện nghiêm túc cho chúng, giờ là lúc đưa chúng vào đội hình. Những chú mèo tân binh được chọn giờ sẽ trở thành sĩ quan! Đó là một ngày kỷ niệm đối với NCPD, và không nghi ngờ gì nữa, người hạnh phúc nhất hôm nay là Tanya. Sau tất cả, cô ấy là người đã dành mọi khó khăn để huấn luyện tất cả các tân binh và biến họ thành những con mèo lành nghề cho NCPD, đó thực sự là một công việc khó khăn đối với cô ấy, vì vậy cô ấy rất thích điều đó!

Bây giờ, có N sĩ quan hiện trong đội, và Tanya đã huấn luyện, có tổng số Q mèo tân binh phải được đưa vào đội. Tại NCPD, mỗi con mèo được gán một số nguyên ID, ID này mô tả sự xứng đáng của một con mèo, càng lớn càng tốt. N con mèo đã có trong đội có ID được mô tả bởi mảng A.

Mặc dù Tanya là người đã làm việc chăm chỉ để huấn luyện Q chú mèo này, nhưng cô ấy tôn trọng những nỗ lực của tất cả nhưng chú mèo của mình. Và vì vui mừng, cô ấy đã quyết định tặng phần thưởng cho các chú mèo đã huấn luyện của mình. Có tổng cộng Q lần và trong mỗi lần, một con mèo có ID phải được đưa vào A. Đây là một điều hạnh phúc cho mỗi con mèo tại NCPD:

- Tanya tặng con mèo này một phần thưởng là số nguyên không âm. Số lượng thưởng thực tế là số mèo đã có trong A có giá trị nhỏ hơn mèo này.
- Con mèo này được thêm vào A ở vị trí cuối cùng.

Bạn có thể giúp Tanya tìm ra tổng số phần thưởng mà cô ấy cần phải có để tặng cho các tân binh thân yêu của mình không?

Dữ liệu vào:

- Dòng đầu là số nguyên T số trường hợp. Mỗi trường hợp:
- Dòng đầu của mỗi trường hợp có một số nguyên N
- Dòng thứ hai có N số nguyên A_1, A_2, …., A_N viết cách nhau một dấu cách
- Dòng thứ ba là số nguyên Q
- Mỗi dòng tiếp theo trong Q dòng chứa một số nguyên X ($X=W \oplus R$), ở đây X là phép XOR của giá trị W của con mèo tham gia và số lượng thưởng R cho đến nay (không bao gồm phần thưởng của con mèo hiện tại).

Kết quả:

Đối với mỗi trường hợp, in một dòng chứa một số nguyên duy nhất, tổng số phần thưởng mà Tanya cần trao.

Giới hạn:

- $1 \le T \le 10$
- $1 \le N,Q \le 10^5$
- $1 \le Ai \le 10^9$, cho mỗi i
- $1 \le W \le 10^9$
- Tất cả con mèo (cũ và mới) có một giá trị khác nhau.

Subtasks:

- Subtask 1: $1 \le N,Q \le 100$
- Subtask 2Những ràng buộc ban đầu.

RKCAT.INP	RKCAT.OUT
1 5 1 2 3 4 5 5 6	35

2 3 27 16	

Giải thích:

Mỗi Q có thể được xem xét như sau:

A=[1,2,3,4,5], X=6⊕0=6

A=[1,2,3,4,5,6], X=7⊕5=2

A=[1,2,3,4,5,6,7], X=8⊕11=3

A=[1,2,3,4,5,6,7,8], X=9⊕18=27

A=[1,2,3,4,5,6,7,8,9], X=10⊕26=16

Cuối cung mảng A trở thành [1,2,3,4,5,6,7,8,9,10] và tổng phần thưởng được trao là 35.

*Code:

```
#include <bits/stdc++.h>
using namespace std;
const long MAXW = 1000000001L;
unordered_map<long, long> fenwick;
inline void add(long x, long y) {
   while (x < MAXW) {
      fenwick[x] += y;
      x += x & -x;
   }
}
inline long get(long x) {
   long res = 0L;
   while (x) {
      res += fenwick[x];
      x -= x & -x;
   }
   return res;
}
int main() {
   //ios_base::sync_with_stdio(false);
   //cin.tie(nullptr);
   //cout.tie(nullptr);
   freopen("RKCAT.inp","r",stdin);
   freopen("RKCAT.out","w",stdout);
   int tt;
   cin >> tt;
   while (tt--) {
      int n;
      cin >> n;
      fenwick.clear();
      for (int i = 0; i < n; i++) {
         long x;
         cin >> x;
         add(x, 1L);
      }
      int qq;
      cin >> qq;
      long ans = 0L;
```

```
        while (qq--) {
            long x;
            cin >> x;
            x ^= ans;
            ans += get(x);
            add(x, 1L);
        }
        cout << ans << '\n';
    }
    return 0;
}
```

Bài 5. Vương quốc Treeland (Treeland.*)

Vương quốc Treeland bao gồm N thành phố (được đánh số từ 1 đến N) được nối với nhau bởi N-1 đường hai chiều giữa mỗi cặp thành phố. Để tăng cường an ninh ở Treeland, chính phủ quyết định thành lập các văn phòng cảnh sát ở K thành phố của mình. Với mỗi i, văn phòng thứ i ở thành phố V_i và nó có bán kính hiệu quả là R_i.

Hãy xác định mức độ an ninh Si của mỗi thành phố i như sau:

- Ban đầu, mức độ an ninh của mỗi thành phố bằng không.
- Sau đó, đối với mỗi văn phòng cảnh sát j được xây dựng ($1 \le j \le K$), các cấp độ an ninh thay đổi theo cách sau:
  - Mức độ an ninh của thành phố Vj tăng Rj.
  - Mức độ an ninh của tất cả các thành phố với khoảng cách 1 từ V_j tăng thêm $R_j - 1$.
  - ……
  - Mức độ an ninh tất cả các thành phố ở khoảng cách R_j - 1 từ Vj tăng lên 1.
- Về mặt hình thức, với mỗi thành phố i, văn phòng cảnh sát j tăng mức độ an ninh của thành phố này bởi max(0, R_j-khoảng cách (i, V_j))

Alice phụ trách tính toán mức độ an ninh mới của tất cả các thành phố sau khi các văn phòng được xây dựng. Cô ấy đã hoàn thành công việc của mình và nhận thấy một sự trùng hợp thú vị: đối với mỗi i hợp lệ, $S_{Vi} = R_i$. Bây giờ cô ấy đã thách thức bạn tìm mức độ an ninh của tất cả các thành phố.

Dữ liệu vào:

- Dòng đầu tiên là số nguyên T số trường hợp cần test. Mỗi trường hợp như sau:
- Dòng đầu tiên của mỗi trường hợp chứa 2 số N và K phân cách nhau bởi dấu cách.
- Sau đó, N − 1 dòng tiếp theo. Mỗi dòng này chứa hai số nguyên u và v được phân tách bằng dấu cách biểu thị rằng các thành phố u và v được nối với nhau bằng một con đường.
- K dòng tiếp theo mỗi dòng này chứa hai số nguyên được phân tách bằng dấu cách là V_i và R_i.

Kết quả ra:

Đối với mỗi trường hợp, in một dòng duy nhất chứa N số nguyên được phân tách bằng dấu cách $S_1, S_2, \ldots, S_N$.

Ràng buộc:

- $1 \le T \le 2{,}000$
- $2 \le N, K \le 8.10^5$
- $1 \le u, v \le N$
- $1 \le V_i \le N$ với mỗi i
- $V_1, V_2, \ldots, V_K$ khác biệt theo từng cặp.
- $1 \le R_i$ với mỗi i
- Tổng của N cho tất cả các test không vượt quá 8.10^5

Ví dụ:

Treeland.inp	Treeland.out
2 5 2 5 2 5 4 5 3 3 1 2 1 5 1 5 2 5 2 1 2 2 4 3 1 4 2 5 2	0 1 0 0 1 0 2 0 2 2

***Thuật toán và code:**

```
#include <bits/stdc++.h>
#define ll long long
#define pb push_back
#define pii pair<int,int>
#define maxn 800800
#define mod 1000000007
#define debug
using namespace std;
int n;
vector<int> L[maxn];
int par[maxn];
int mrk[maxn];
int rad[maxn];
int ans[maxn];
vector<pii> op[maxn];
int T[2][2*maxn];
void upd(int id,int idx,int val){
	idx++;
	while(idx <= 2*n+2){
		T[id][idx] += val;
		idx += (idx&-idx);
	}
}
int qry(int id,int idx){
	int r = 0;
	idx++;
	while(idx){
		r += T[id][idx];
		idx -= (idx&-idx);
	}
	return r;
}
```

```
#define Q(a,b) (qry(a,2*n+1)-qry(a,b-1))
void calc(int vx,int d=0){
	for(pii i : op[vx]){
		int x = d + i.first;
		upd(0,x,-i.second);
		upd(1,x,x*i.second);
	}
	ans[vx] = d * Q(0,d) + Q(1,d);
	for(int i : L[vx])
		if(i != par[vx])
			calc(i,d+1);
	for(pii i : op[vx]){
		int x = d + i.first;
		upd(0,x,i.second);
		upd(1,x,-x*i.second);
	}
}
void dfs(int vx){
	for(int i : L[vx])
		if(i != par[vx]){
			par[i] = vx;
			dfs(i);
		}
	if(mrk[vx])
		for(int i=vx,R=rad[vx];R>0;i = par[i], R--){
			op[i].pb({R,1});
			if(i && R > 1) op[i].pb({R-2,-1});
			if(i == 0) break;
		}
}
main(){
	int nt;
	freopen("Treeland.inp","r",stdin);
	freopen("Treeland.out","w",stdout);
	scanf("%d",&nt);
	while(nt--){
		int k;
		scanf("%d%d",&n,&k);
		for(int i=0;i<n;i++)
			L[i].clear(), op[i].clear(), mrk[i] = 0;
		for(int i=0;i<=2*n+2;i++) T[0][i] = T[1][i] = 0;

		for(int i=0;i<n-1;i++){
			int a,b;
			scanf("%d%d",&a,&b), a--, b--;
			L[a].pb(b);
			L[b].pb(a);
		}
		for(int i=0;i<k;i++){
			int c,r;
			scanf("%d%d",&c,&r), c--;
```

```
                mrk[c] = 1;
                rad[c] = r;
            }
            dfs(0);
            calc(0);
            for(int i=0;i<n;i++)
                printf("%d%c",ans[i]," \n"[i==n-1]);
        }
}
```

Bài 6. Truy vấn trên mảng (QARRAY.*)

Harsh yêu thích những con số và có một bộ sưu tập khổng lồ các con số có kích thước n. Một ngày nọ, Aniket nhìn thấy bộ sưu tập của anh ấy và đưa ra một thử thách. Aniket thực hiện 2 loại thao tác trên bộ sưu tập của Harsh.

- Cập nhật: Anh ta thay đổi giá trị của số thứ x thành y.
- Truy vấn: Anh ta hỏi liệu có tồn tại bất kỳ hậu tố nào trong tập hợp được sắp xếp của anh ấy sao cho tổng của nó là z hay không.

Bạn có thể giúp Harsh thực hiện các thao tác này và trả lời các câu hỏi không.

Dữ liệu vào:

Dòng đầu tiên là một số nguyên **n** mô tả số phần tử của tập.

Dòng tiếp theo chứa n số $\mathbf{a_1, a_2 \dots a_n}$

Dòng tiếp theo chứa số **q** mô tả số thao tác. **q** dòng tiếp theo có cấu trúc sau:

- **1 x y:** Thay đổi phần tử thứ x thành y.
- **2 z:** Tìm xem có tồn tại bất kỳ hậu tố nào có tổng chính xác bằng **z** không.

Kết quả:

Với mỗi truy vấn (tức là loại 2) in câu trả lời cho truy vấn. In "YES" (không có dấu nháy kép) nếu tồn tại hậu tố có tổng bằng z, nếu không thì in "NO".

Ràng buộc:

- $\mathbf{1 \le n \le 10^5}$
- $\mathbf{1 \le a_i, y \le 10^9}$
- $\mathbf{1 \le q \le 10^5}$
- $\mathbf{1 \le x \le n}$
- $\mathbf{1 \le z \le 10^{14}}$

Ví dụ:

QARRAY.INP	QARRAY.OUT
5 10 1 9 8 11 6 2 19 1 3 13 2 28 2 32 1 1 100 2 133	YES NO YES YES

***Code:**

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double db;
#define endl '\n'
#define fi first
```

```
#define se second
#define pi pair<ll,ll>
#define pii pair<ll,pi>
#define pb push_back
#define mk make_pair
const ll siz = 1e5 + 7;
ll bit[siz];
void update(ll idx, ll val) {
	while (idx < siz) {
		bit[idx] += val;
		idx += (idx & -idx);
	}
}
ll query(ll idx) {
	ll sum = 0;
	while (idx) {
		sum += bit[idx];
		idx -= (idx & -idx);
	}
	return sum;
}
int main() {
	freopen("qarray.inp", "r", stdin);
	freopen("qarray.out", "w", stdout);
	//ios_base::sync_with_stdio(false);
	//cin.tie(NULL);
	ll n;
	cin >> n;
	vector<ll>v(n + 1);
	for (ll i = 1; i <= n; i++) {
		cin >> v[i];
		update(i, v[i]);
	}
	ll q;
	cin >> q;
	while (q--) {
		ll type;
		cin >> type;
		if (type == 1) {
			ll index, by;
			cin >> index >> by;
			update(index, -v[index]);
			v[index] = by;
			update(index, v[index]);
		}
		else
		{
			ll sum = query(n);
			ll find;
			cin >> find;
			bool ok = false;
```

```
                ll left = 0, right = n;
                while (right >= left) {
                        ll mid = (left + right) / 2;
                        ll got = query(mid);
                        ll have = sum - got;
                        //      cout << have << " " << mid << endl;
                        if (have == find) {
                                ok = true;
                                break;
                        }
                        if (have > find)left = mid + 1;
                        else right = mid - 1;
                }
                if (ok)
                        cout << "YES" << endl;
                else
                        cout << "NO" << endl;
        }
    }
}
```

Bài 7. Hai đội quân (ARM.*)

Có 2 vương quốc, vương quốc 1 và vương quốc 2 xảy ra cuốc chiến. Vương quốc 1 có quân đội kích cỡ N và vương quốc 2 có quân đội kích cỡ M.

Cuộc chiến sẽ kéo dài trong Q ngày. Mỗi người lính trong vương quốc 1 hoặc 2 sẽ có một giá trị nguyên dương ban đầu là $\mathbf{V_i}$ chỉ số sức mạnh của người lính (1 <= i <= N với lính ở vương quốc 1 và 1<= i <=M với lính ở vương quốc 2). Vào ngày thứ i, Vương quốc 1 sẽ gửi một tiểu đoàn quân từ chỉ số a_i đến b_i và Vương quốc 2 sẽ gửi một tiểu đoàn quân từ chỉ số c_i đến d_i.

Bạn cũng được cung cấp kết quả của ngày thứ $\mathbf{Q_i}$ (tức là vương quốc nào giành chiến thắng). Tiểu đoàn của vương quốc chiến thắng đã chiến đấu vào ngày $\mathbf{Q_i}$ sẽ được tăng sức mạnh theo giá trị ban đầu của họ (được đưa ra ở đầu vào) và của vương quốc thua cuộc sẽ bị giảm sức mạnh của họ theo giá trị ban đầu (được đưa ra ở đầu vào).

Vua của vương quốc chiến thắng sẽ thưởng cho một người lính ngẫu nhiên trong quân đội của họ, nhưng để thưởng cho anh ta, anh ta phải nói to sức mạnh hiện tại của mình !!! , nhưng tiếc là họ chỉ nhớ giá trị sức mạnh ban đầu của mình nên họ yêu cầu bạn ghi nhớ sức mạnh hiện tại của họ. Em hãy cho biết sức mạnh hiện tại của một người lính cụ thể ở vương quốc chiến thắng vào ngày đó.

Bạn phải in kết quả cho mỗi ngày.

Dữ liệu vào:

- Dòng đầu chứa 3 số nguyên **N**, **M**, **Q** phân cách bởi dấu cách.

- Dòng thứ 2 ghi **N** số nguyên phân cách nhau bởi dấu cách là sức mạnh của các người lính ở vương quốc 1.
- Dòng thứ 3 ghi **M** số nguyên phân cách nhau bởi dấu cách là sức mạnh của các người lính ở vương quốc 2.
- Q dòng tiếp theo sẽ chứa 6 số nguyên cách nhau khoảng trắng $\mathbf{a_i}$, $\mathbf{b_i}$, $\mathbf{c_i}$, $\mathbf{d_i}$, **K, pos**.

K đại diện cho Vương quốc nào đã thắng vào ngày thứ i. Nó có thể là 1 đại diện cho Vương quốc 1 và 2 đại diện cho Vương quốc 2. Bạn phải in ra kết quả là sức mạnh của người lính chiến thắng vương quốc có chỉ số là **pos**. Nếu K = 1, Pos nằm trong khoảng từ 1 đến N, còn lại Pos nằm trong khoảng từ 1 đến M.

Kết quả:

In ra kết quả của mỗi ngày, tức là sức mạnh hiện tại của người lính có chỉ số **pos** thuộc tiểu đoàn của vương quốc đã chiến thắng.

Ràng buộc:

- $1 \le N,M,Q \le 2 * 10^5$
- $1 \le K \le 2$
- $1 \le V_i \le 10^9$
- $1 \le a_i \le b_i \le N$
- $1 \le c_i \le d_i \le M$
- $1 \le Pos \le N$, nếu K=1
- $1 \le Pos \le M$, nếu K=2

Ví dụ :

ARM.inp	ARM.out
3 3 2 2 4 3 1 5 4 1 2 1 1 2 1 1 3 2 3 1 2	2 4

Giải thích:

-Ngày 1: Vương quốc 2 đã thắng, vì vậy những người lính có chỉ số từ 1 đến 1 của Vương quốc 2 tăng Sức mạnh của họ theo giá trị ban đầu, tức là 1 và những người lính có chỉ số từ 1 đến 2 của Vương quốc 1 giảm Sức mạnh của họ tương ứng 2,4. Giá trị sức mạnh hiện tại của người lính có chỉ số là 1 của Vương quốc 2 là 1 + 1 = 2.

Vì vậy, sau ngày 1 giá trị Sức mạnh của binh lính Vương quốc 1: [0,0, 3] và của Vương quốc 2: [2, 5,4].

-Ngày 2: Vương quốc 1 thắng, vì vậy những người lính có chỉ số từ 1 đến 3 của Vương quốc 1 tăng Sức mạnh của họ theo giá trị ban đầu, tức là tương ứng 2,4,3 và những người lính có chỉ số từ 2 đến 3 của Vương quốc 2 giảm Sức mạnh của họ tương ứng 5,4 . Giá trị sức mạnh hiện tại của người lính có chỉ số là 2 của Vương quốc 1 là 0 + 4 = 4.

-Vì vậy, Giá trị Sức mạnh của binh lính Vương quốc 1: [2, 4, 6] Sau Ngày 2 và của Vương quốc 2: [2, 0, 0].

*Code:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double db;
#define endl '\n'
#define fi first
#define se second
#define pi pair<ll,ll>
#define pii pair<ll,pi>
#define pb push_back
```

```
#define mk make_pair
struct node {
	static const ll siz = 2e5 + 7;
	ll bit[siz];
	void update(ll idx, ll val) {
		while (idx < siz) {
			bit[idx] += val;
			idx += (idx & -idx);
		}
	}
	ll query(ll idx) {
		ll sum = 0;
		while (idx) {
			sum += bit[idx];
			idx -= (idx & -idx);
		}
		return sum;
	}
	// void solve(ll k) {
	//	for (ll i = 0; i < k; i++) {
	//		cout << bit[i] << " ";
	//	}
	//	cout << endl;
	// }
};
node v[3];
int main() {
	freopen("arm.inp", "r", stdin);
	freopen("arm.out", "w", stdout);
	//ios_base::sync_with_stdio(false);
	//cin.tie(NULL);
	ll n, m, q;
	cin >> n >> m >> q;
	vector<ll>x(n + 1), y(m + 1);
	for (ll i = 1; i <= n; i++) {
		cin >> x[i];
	}
	for (ll i = 1; i <= m; i++) {
		cin >> y[i];
	}
	v[0].update(1, +1);
	v[0].update(n + 1, -1);
	v[1].update(1, 1);
	v[1].update(m + 1, -1);
	//cout << v[0].query(3) << endl;
	while (q--) {
		ll a, b, c, d, k, pos;
		cin >> a >> b >> c >> d >> k >> pos;
		if (k == 1) {
			v[0].update(a, +1);
			v[0].update(b + 1, -1);
```

```
                v[1].update(c, -1);
                v[1].update(d + 1, +1);
                ll got = v[0].query(pos);
                ll have = got;
                //cout << have << endl;
                cout << x[pos]*have << endl;
        }
        else
        {
                v[0].update(a, -1);
                v[0].update(b + 1 , +1);
                v[1].update(c, +1);
                v[1].update(d + 1 , -1);
                ll got = v[1].query(pos);
                ll have = got;
                //cout << have << endl;
                cout << y[pos]*have << endl;
        }
    }
}
```

Bài 8. Đội quân Chelf (Chelf.*)

Quân đội Chefland có N binh sĩ, đứng thành một hàng và được đánh số từ 1 đến N từ trái sang phải. Chiều cao của chúng từ trái sang phải là $A_1, A_2,\ldots, A_N$. Một đội bao gồm một hoặc nhiều binh sĩ liên tiếp, tức là binh lính từ L đến R (bao gồm) cho một số L và R sao cho $1\le L\le R\le N$. Chỉ số sức mạnh của một đội được xác định theo cách sau: Đối với mỗi người lính trong đội này, ta tìm số lượng binh sĩ trong đội có chiều cao lớn hơn đứng bên trái của người lính này và sức mạnh của đội tổng các số này.

Tư lệnh Chelf quyết định viết ra chỉ số sức mạnh của tất cả các đội có thể, nhưng ngay sau đó, anh nhận ra rằng đó là một nhiệm vụ nặng nề. Vì vậy, anh ấy chỉ muốn biết điểm trung bình chỉ số sức mạnh của tất cả các đội có thể có.

Chú ý: Giá trị trung bình của một dãy là giá trị giữa khi dãy được sắp xếp. Nếu độ dài của dãy là chẵn, có 2 giá trị ở giữa và giá trị trung vị là giá trị nhỏ hơn trong các giá trị này (hoặc khi chúng bằng nhau).

Dữ liệu vào:

- Dòng đầu tiên của đầu vào là số nguyên T mô tả số trường hợp. Mỗi trường hợp của T có cấu trúc như sau:
- Dòng đầu tiên của mỗi trường hợp là số nguyên N.
- Dòng thứ hai là N số nguyên $A_1, A_2, \ldots, A_N$ phân cách nhau bởi dấu cách.

Kết quả:

Với mỗi trường hợp, in ra một dòng chứa một số nguyên – trung bình cộng sức mạnh của tất cả các đội.

Ràng buộc:

- $1\le T\le 500$
- $1\le N\le 10^5$
- $1\le A_i\le 10^9$ với mỗi i
- Tổng của các N trong các trường hợp không lớn hơn 5.10^5

Ví dụ:

Chelf.inp	Chelf.out
4 1	0 0

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

Giải thích:
Trường hợp 1: Chỉ có một đối khả thi và sức mạnh của nó là 0.
Trường hợp 3: Có 10 đội có thể có, với sức mạnh (0,0,0,0,1,1,1,3,3,6).
*Code:

```
#include<bits/stdc++.h>
using namespace std;
//#define int long long
#define double long double
typedef vector<int> vi;
typedef vector<vector<int> > Mat;
typedef pair<int, int> pi;
#define IOS ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0)
#define MP make_pair
#define eb emplace_back
#define pb push_back
#define sz(o) (int)o.size()
#define fr(i, n) for(int i=0;i<n;++i)
#define tr(c, it) for(__typeof(c.begin()) it = c.begin();it!=c.end();++it)
#define all(o) o.begin(), o.end()
#define F first
#define S second
#define mset(m, v) memset(m, v, sizeof(m))
#define debug(a) cerr << #a << ": " << a << ' ';
#define pr(v) tr(v, it)cout<<*it<<' ';cout<<'\n';
const double PI = 2.0*acos(0);
const int M = 1000000007;
int binpow(int a, int b){if(!b)return 1ll;int temp = binpow(a, b>>1ll);temp = (1ll*temp*temp)
%M;if(b&1)temp = (1ll*a*temp)%M;return temp;}
const int N = 100100;
long long n, nn, cnt;
long long v[N];
class BIT{
public:
    int* arr;
    int size;
    BIT(){
        int size = 0;
        arr = nullptr;
    }
    BIT(int n){
        arr = (int*)malloc(n*sizeof(int));
        for(int i=0;i<n;i++)
            arr[i] = 0;
        size = n;
```

```
    }
    void resize(int n){
        arr = (int*)malloc(n*sizeof(int));
        for(int i=0;i<n;i++)
            arr[i] = 0;
        size = n;
    }
    void update(int u, int val){
        while(u <= n){
            arr[u] += val;
            u += (u^(u&(u-1)));
        }
    }
    int query(int u){
        int sum = 0;
        while(u){
            sum += arr[u];
            u = u&(u-1);
        }
        return sum;
    }
};
BIT t(N);
bool check(int d){
    long long ans = 1, inv=0;
    fr(i, n+5)t.arr[i]=0;
    //debug(d);
    t.update(v[0], 1);
    //debug(d);
    int j=1, i=0;
    while(j < n){
        inv += t.query(n) - t.query(v[j]);
        t.update(v[j], 1);
        if(inv > d){
            while(i < j && inv > d){
                int x = t.query(v[i]-1);
                t.update(v[i], -1);
                inv-=x;
                i++;
            }
        }
        ans += (j-i+1);
        //debug(inv);debug(ans);cout<<'\n';
        //for(int k=i;k<=j;k++)cout<<v[k]<<" \n"[k==j];
        j++;
    }
    return (ans>=((nn+1)/2));
}
int32_t main(){
    freopen("Chelf.inp","r",stdin);
    freopen("Chelf.out","w",stdout);
```

```
    //IOS;
    int tt;cin>>tt;while(tt--){
        int m,k,i,j,l,r,mid;
        cin>>n;
        nn = (n*(n+1))/2;
        map<int, int> mp;
        fr(i, n){cin>>v[i];mp[v[i]];}
        if(n==1){cout<<"0\n";continue;}
        cnt=1;
        tr(mp, it)it->S = cnt++;
        fr(i, n)v[i]=mp[v[i]];
        long long ans = -1, hi = nn, lo=0;
        while(hi >= lo){
            mid=(lo+hi)/2;
            if(check(mid)){
                hi = mid-1;
                ans = mid;
                //debug(mid);
            }
            else
                lo = mid+1;
        }
        cout<<ans<<'\n';
    }
    //cerr<<((double)(clock() - begin)/CLOCKS_PER_SEC)<<'\n';
    return 0;
}
```

Bài 10. Thay đổi tuần hoàn trong hoán vị (permutation.*)

Bạn có một hoán vị P của các số nguyên 1, 2, ..., N. Bạn muốn thay đổi nó một chút. Để làm điều này, bạn chọn một số nguyên K thỏa mãn $2 \le K \le N$. Sau đó, bạn thực hiện một số chuyển đổi theo chu kỳ (có thể là 0, nếu bạn cảm thấy lười biếng). Đối với mỗi chuyển dịch theo chu kỳ, bạn chọn một đoạn có độ dài K của hoán vị P (ký hiệu là P_x, P_{x+1}, ..., P_{x+K-2}, P_{x+K-1}) và thực hiện chuyển đổi tuần hoàn trên đó theo hướng bên phải. Tức là sắp xếp lại các phần tử của đoạn này theo thứ tự P_{x+K-1}, P_x, ..., P_{x+K-3}, P_{x+K-2}.

Ví dụ, dãy hoán vị **(6, 5, 1, 3, 2, 4)** được chuyển đổi từ **(1, 5, 4, 6, 3, 2)** với K = 4. Các đoạn dịch chuyển theo chu kỳ mỗi lần dịch chuyển là gạch chân: **(<u>1, 5, 4, 6</u>, 3, 2)** => **(6, 1, <u>5, 4, 3, 2</u>)**=> **(6, 1, <u>2, 5, 4, 3</u>)** => **(6, <u>1, 3, 2, 5</u>, 4)** => **(6, 5, 1, 3, 2, 4)**.

Gọi S là tập hợp các hoán vị có thể nhận được từ hoán vị P sử dụng không hoặc nhiều lần tuần hoàn. Bạn được cho một hoán vị Q của các số nguyên 1, 2, ..., N. Nhiệm vụ của bạn là tìm xem S có chứa Q hay không. Nếu có, bạn cũng phải tìm chỉ số của Q trong danh sách tất cả các hoán vị trong tập S, được sắp xếp theo từ điển.

Dữ liệu vào:

- Dòng đầu là số nguyên T số trường hợp cần kiểm tra. Với mỗi trường hợp:
- Với mỗi trường hợp dòng đầu tiên chứa 2 số nguyên N và K biểu thị độ dài của P và Q, và độ dài của các đoạn bạn được phép dịch chuyển theo chu kỳ.
- Dòng tiếp theo ghi N số nguyên $\mathbf{P_1}$, $\mathbf{P_2}$, ..., $\mathbf{P_N}$ miêu tả hoán vị **P**.
- Dòng tiếp theo ghi N số nguyên $\mathbf{Q_1}$, $\mathbf{Q_2}$, ..., $\mathbf{Q_N}$ miêu tả hoán vị **Q**.

Kết quả ra:

Với mỗi trường hợp, in ra một dòng duy nhất. Là -1 nếu Q ko thể được sinh từ P theo dịch chuyển tuần hoàn. Nếu không in ra chỉ số của hoán vị trong tập S được sắp xếp theo thứ tự từ vựng. Vì kết quả có thể rất lớn, in ra theo modulo $10^9 + 7$.

Ràng buộc:

- $2 \le N \le 10^5$
- $2 \le K \le N$
- $P_1, P_2, ..., P_N$ hoán vị các số nguyên từ **1** tới **N**
- $Q_1, Q_2, ..., Q_N$ hoán vị các số nguyên từ **1** tới **N**

Subtasks

Subtask 1:

- $1 \le T \le 1000$
- $2 \le N \le 5$

Subtask 2:

- $1 \le T \le 10$
- $2 \le N \le 10^5$
- $K = N$

Subtask 3:

- $1 \le T \le 10$
- $2 \le N \le 1000$

Subtask 4:

- $1 \le T \le 10$
- Ràng buộc ban đầu.

Ví dụ:

Permutation.inp	**Permutation.out**
2 4 2 2 4 3 1 1 2 4 3 3 3 1 2 3 1 3 2	2 -1

Giải thích:

Trường hợp 1. Chuyển đổi tuần hoàn của đoạn có độ dài 2. Sử dụng chúng, ta có thể tạo được bất kì hoán vị nào tùy ý. Hoán vị **Q** có chỉ số là **2** trong số các hoán vị có độ dài **4**.

Trường hợp 2. Ta có thể nhận được các hoán vị **(1, 2, 3)**, **(2, 3, 1)** and **(3, 1, 2)**. Hoán vị **Q** không có trong danh sách, vì vậy kết quả là **-1**.

*Code:

```
#include <bits/stdc++.h>
#define f first
#define s second
#define pb push_back
#define mp make_pair
using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef pair<long long, long long> pll;
const int N = (int)2e5 + 123, inf = 1e9, mod = 1e9 + 7;
const ll INF = 1e18;
int n, k, a[N], b[N], p[N], t[N], f[N];
int get(int r){
	int res = 0;
```

```
        for (; r >= 0; r = (r & (r + 1)) - 1)
                res += t[r];
        return res;
}
void upd(int i, int k){
        for (; i <= n; i = (i | (i + 1)))
                t[i] += k;
}
void solve(){
        scanf("%d%d", &n, &k);
        for(int i = 1; i <= n; i++)
                scanf("%d", &a[i]);
        for(int i = 1; i <= n; i++)
                scanf("%d", &b[i]);
        if(n == k){
                for(int i = 1; i <= n; i++)
                        if(a[i] == b[1])
                                for(int j = 0; j < n; j++)
                                        if(a[(i + j - 1) % n + 1] != b[j + 1]){
                                                printf("-1\n");
                                                return;
                                        }
           printf("%d\n", b[1]);
                return;
        }
        if(k & 1){
                bool aa = 0, bb = 0;
                for(int i = 1; i <= n; i++)
                        p[a[i]] = i;
                for(int i = 1; i <= n; i++){
                        if(a[i] == i)
                                continue;
                        aa ^= 1;
                        swap(p[a[i]], p[i]);
                        swap(a[p[a[i]]], a[p[i]]);
                }
      for(int i = 1; i <= n; i++)
                        a[i] = b[i];
      for(int i = 1; i <= n; i++)
                        p[a[i]] = i;
                for(int i = 1; i <= n; i++){
                        if(a[i] == i)
                                continue;
                        aa ^= 1;
                        swap(p[a[i]], p[i]);
                        swap(a[p[a[i]]], a[p[i]]);
                }
      if(aa != bb){
                        printf("-1\n");
                        return;
                }
```

```
                ll ans = 1;
                for(int i = 0; i <= n; i++)
                        t[i] = 0;
                for(int i = 1; i <= n; i++)
                        upd(i, 1);
                for(int i = 1; i <= n; i++){
                        upd(b[i], -1);
                        if(1ll * get(b[i]) * f[n - i] >= 2)
                ans = (ans + 1ll * get(b[i]) * f[n - i] % mod * 500000004) % mod;
                }
                printf("%lld\n", ans);
                return;
        }
        ll ans = 1;
        for(int i = 0; i <= n; i++)
                t[i] = 0;
        for(int i = 1; i <= n; i++)
                upd(i, 1);
        for(int i = 1; i <= n; i++){
                upd(b[i], -1);
                ans = (ans + 1ll * get(b[i]) * f[n - i]) % mod;
        }
        printf("%lld\n", ans);
}
int main()
{
   freopen("Permutation.inp","r",stdin);
   freopen("Permutation.out","w",stdout);
        f[0] = 1;
        for(int i = 1; i < N; i++)
                f[i] = 1ll * f[i - 1] * i % mod;
        int t;
        scanf("%d", &t);
        while(t--)
                solve();
        return 0;
}
```

Bài 11. Chuỗi cửa hàng (chain.*)

An thực sự ngưỡng mộ Bình. Cả hai đều có chuỗi nhà hàng. Trong khi chuỗi nhà hàng của An bao gồm N cơ sở (đánh số từ 1 đến N), Bình chỉ có 3, nhưng các nhà hàng của Bình lại vượt trội hơn hẳn. Đội ngũ tiếp thị của An đã tìm ra cách để chiếm lĩnh thị trường. Theo họ, vấn đề là khách hàng có quá nhiều nhà hàng để lựa chọn và nhân viên của Bình, do tình cờ hoặc do thiên tài tiếp thị, đã tìm ra cách phân phối hoàn hảo các nhà hàng. Do đó, An muốn đóng cửa N-3 nhà hàng càng nhanh càng tốt và tạo ra một cấu trúc nhà hàng tương tự như của Bình. Vì vậy, anh ấy cần sự giúp đỡ của bạn!

Chúng ta biết rằng các nhà hàng của Bình được đánh số 1,2,3, trong đó nhà hàng 1 là nhà hàng nhỏ nhất và nhà hàng 3 là nhà hàng lớn nhất. Chúng nằm trên một dòng theo thứ tự p_1, p_2, p_3, vì vậy các nhà hàng p_1 và p_2 liền kề và các nhà hàng p_2 và p_3 cũng liền kề.

Hệ thống nhà hàng của An là một cây có N đỉnh, trong đó mỗi đỉnh là một trong những nhà hàng quý giá của anh ta. Cũng giống như các nhà hàng của Bình, chúng được đánh số theo

thứ tự tăng dần quy mô. An muốn đếm số lượng bộ ba nhà hàng đã đặt hàng (a_1, a_2, a_3) mà anh ta có thể mở cửa trong khi đáp ứng các quy tắc sau:

1. Các nhà hàng a_1, a_2 và a_3 nằm trên một đường, tức là a_2 nằm trên con đường ngắn nhất giữa a_1 và a_3 (xây dựng lại cấu trúc hiện tại sẽ quá tốn kém).
2. Với mỗi i, j (1≤i, j≤3), nếu pi <pj thì ai <aj.

Hãy giúp An tính toán số lượng 3 cặp này.

Dữ liệu vào:

- Dòng đầu tiên là số nguyên T – số trường hợp cần kiểm tra có cấu trúc như sau:
- Dòng tiên tiên mỗi trường hợp chứa số N
- Dòng thứ 2 chứa 3 số nguyên p_1, p_2 và p_3.
- Mỗi dòng trong N-1 dòng tiếp theo chứa 2 số nguyên u và v phân cách nhau bởi dấu cách mô tả nhà hàng u và v có cạnh nối với nhau.

Kết quả:

Với mỗi trường hợp, in mỗi dòng một số nguyên duy nhất — số cặp ba .

Ràng buộc:

- 1≤T≤20
- 3≤N≤105
- 1≤u, v≤N
- Đồ thị các nhà hàng của Chelf là một cây
- dãy (p1, p2, p3) là một hoán vị của (1,2,3)

Subtasks

Subtask 1: 1≤N≤100
Subtask 2 : 1≤N≤1,0001≤N≤1,000
Subtask 3 : Cây là biểu đồ hình sao
Subtask 4: Cây là đường thẳng.
Subtask 5 : Tổng các trường hợp của N không lớn hơn 2.10^5.
Subtask 6 : Các ràng buộc ban đầu.

Ví dụ:

Chain.inp	Chain.out
1 4 2 1 3 1 2 2 3 2 4	1

Diễn giải:

Bộ 3 hợp lệ là (3,2,4)

*Code:

```
#include<bits/stdc++.h>
#define ll long long
#define pr pair<ll,ll>
using namespace std;
int dfs_cnt=0,n,p1,p2,p3;
ll c[100005];
ll ans=0;
inline int lowbit(int x){
   return x&(-x);
}
void updata(int i,int k){
   while(i <= n){
      c[i] += k;
```

```
        i += lowbit(i);
    }
}
ll getsum(int i){
    ll res = 0;
    while(i > 0){
        res += c[i];
        i -= lowbit(i);
    }
    return res;
}
struct line{int to,nxt;}e[200005];
int h[100005],cnt=0;
void Add(int u,int v){e[++cnt]=line{v,h[u]};h[u]=cnt;}
struct node{int in,out,sum,f;}nd[100005];
void dfs(int u,int f){
    nd[u].sum=1;
    nd[u].f=f;
    nd[u].in=++dfs_cnt;
    for(int i=h[u];i;i=e[i].nxt){
        int v=e[i].to;
        if(v!=f){
            dfs(v,u);
            nd[u].sum+=nd[v].sum;
        }
    }
    nd[u].out=dfs_cnt;
}
ll solve(int u,int p){
    ll res,L=n-u,S=u-1;
    updata(nd[u].in,1);
    if(p==2)res=1LL*(u-1)*(n-u);
    else if(p==1)res=1LL*(n-u)*(n-u-1)/2;
    else res=1LL*(u-1)*(u-2)/2;
    for(int i=h[u];i;i=e[i].nxt){
        int v=e[i].to;
        if(v!=nd[u].f){
            ll s=getsum(nd[v].out)-getsum(nd[v].in-1);
            ll l=nd[v].sum-s;
            L-=l;
            S-=s;
            if(p==2)res-=l*s;
            else if(p==1)res-=l*(l-1)/2;
            else res-=s*(s-1)/2;
        }
    }
    if(p==2)res-=L*S;
    else if(p==1)res-=L*(L-1)/2;
    else res-=S*(S-1)/2;
    return res;
}
```

```
void init(){
   cin>>n;
   cin>>p1>>p2>>p3;
   memset(h,0,sizeof(h));
   memset(c,0,sizeof(c));
   dfs_cnt=cnt=ans=0;
}
int main() {
   int t;
   cin>>t;
   while(t--){
      init();
      for(int i=0;i<n-1;i++){
         int u,v;
         cin>>u>>v;
         Add(u,v);
         Add(v,u);
      }
      dfs(1,0);
      for(int i=1;i<=n;i++)
         ans+=solve(i,p2);
      cout<<ans<<'\n';
   }
   return 0;
}
```

Bài 12. Đa giác (polygon.*)

Roger có sở thích vẽ các hình đa giác. Cho M điểm $P0,P1,\ldots,PM-1$ trên hệ tọa độ Đề-các, anh ấy có thể vẽ đa giác M cạnh bằng cách nối các điểm Pi và P(i+1)%M bằng một đoạn thẳng với mỗi i. Kết quả là, với M = 1, M = 2 cũng là một đa giác nhưng với độ dài cạnh = 0.

Roger gọi một tập S gồm M số nguyên là một đa giác nếu có một đa giác được tạo ra với kích thước M (như đã định nghĩa ở trên) sao cho tập đó có độ dài cạnh của nó bằng S. Ví dụ: tập (0,0), (1,3,3) or (9,9) đều là các đa giác, nhưng (4) and (2,5,8) là không.

Roger có một cây với N đỉnh (được đánh số từ 1 đến N), trong đó mỗi cạnh được tô một trong K màu (xuyên suốt từ 1 đến N). Anh ta muốn chọn một cặp đỉnh phân biệt không có thứ tự trong cây và tạo ra một tập S=(S1,S2,…,SK), trong đó Si là số cạnh trên đường đi giữa các đỉnh đã chọn có màu i. Bây giờ, Roger muốn hỏi: Có bao nhiêu cách mà anh ta có thể chọn một cặp đỉnh sao cho tập S tạo ra là đa giác? Tìm số cặp như vậy.

Dữ liệu vào:

- Dòng đầu tiên của dữ liệu vào chứa một số nguyên T là số trường hợp cần kiểm. Sau đó là các trường hợp T.
- Dòng đầu chứa 2 số nguyên N và K phân cách nhau bởi 1 dấu cách.
- Mỗi dòng trong số N – 1 dòng tiếp theo chứa 3 số nguyên v, u và c biểu thị một cạnh v,u có màu là c.

Kết quả:

Với mỗi trường hợp kiểm tra, in ra một dòng duy nhất chứa 1 số nguyên là số cặp hợp lệ.

Giới hạn:

- $1 \le T \le 10$
- $1 \le N \le 2 \cdot 105$.
- $1 \le K \le 10$
- $1 \le u,v \le N$

- $1 \le c \le K$
- Đồ thị được mô tả trên đầu vào là một cây

Ví dụ

Polygon.inp	Polygon.out
2 5 3 1 2 1 1 3 2 2 4 3 2 5 2 6 2 1 2 1 2 3 1 3 4 2 4 5 1 5 6 2	5 4

* Code:

```
#include <bits/stdc++.h>
using namespace std;
#define ar array
const int mxN=2e5;
int t, n, k, s[mxN], ctp[mxN], ki, a[mxN+2];
vector<ar<int, 2>> adj[mxN];
vector<int> pt, pta;
long long ans;
void upd(int i, int x) {
	for(i+=mxN/2, ++i; i<=mxN+1; i+=i&-i)
		a[i]+=x;
}
int qry(int i) {
	int r=0;
	for(i+=mxN/2; i; i-=i&-i)
		r+=a[i];
	return r;
}
void dfs2(int u, int p=-1) {
	s[u]=1;
	for(ar<int, 2> e : adj[u]) {
		int v=e[0];
		if(v^p&&ctp[v]==-1) {
			dfs2(v, u);
			s[u]+=s[v];
		}
	}
}
int dfs4(int u, int n, int p=-1) {
	for(ar<int, 2> e : adj[u]) {
		int v=e[0];
		if(v^p&&ctp[v]==-1&&s[v]>n/2)
```

```
                return dfs4(v, n, u);
        }
        return u;
}
void dfs5(int u, int s, int p=-1) {
        pt.push_back(s);
        for(ar<int, 2> e : adj[u]) {
                int v=e[0];
                if(v==p||~ctp[v])
                        continue;
                dfs5(v, s+(e[1]^ki?1:-1), u);
        }
}
void dfs3(int u=0, int p=-2) {
        int c=dfs4(u, s[u]);
        ctp[c]=p;
        dfs2(c);
        for(ki=0; ki<k; ++ki) {
                for(ar<int, 2> e : adj[c]) {
                        int v=e[0];
                        if(ctp[v]==-1) {
                                dfs5(v, e[1]^ki?1:-1);
                                for(int pti : pt)
                                        ans-=qry(-pti);
                                for(int pti : pt)
                                        upd(pti, 1);
                                pta.insert(pta.end(), pt.begin(), pt.end());
                                pt.clear();
                        }
                }
                for(int pti : pta)
                        upd(pti, -1);
                pta.clear();
        }
        for(ar<int, 2> e : adj[c]) {
                int v=e[0];
                if(ctp[v]==-1)
                        dfs3(v, c);
        }
}
void solve() {
        cin >> n >> k;
        for(int i=0; i<n; ++i)
                adj[i].clear();
        for(int i=1, u, v, c; i<n; ++i) {
                cin >> u >> v >> c, --u, --v, --c;
                adj[u].push_back({v, c});
                adj[v].push_back({u, c});
        }
        memset(ctp, -1, 4*n);
        dfs2(0);
```

```
        ans=(long long)n*(n-1)/2;
        dfs3();
        cout << ans << "\n";
}
int main() {
        ios::sync_with_stdio(0);
        cin.tie(0);
        upd(0, 1);
        cin >> t;
        while(t--)
                solve();
}
```

---------- HẾT ----------