Bài đếm dãy con tăng dài nhất:

Thuật toán như này:

từ dãy ban đầu a1, a2, a3 …. ai … an, ta duyệt từ i đến n;

ta gọi dãy f[j] là dãy của các phần tử cuối của dãy con có j phần từ f[j] = ai

ta dùng tìm kiếm nhị phân để tính số phần tử của dãy con tăng theo công thức vẫn làm: gọi j là số phần tử của dãy con tăng nếu lấy ai làm phần tử cuối

j = lowbound (f, ai); f[j] = a[i]

ta thấy rằng các dãy con mà có cùng số phần tử j mà kết thúc tại các giá trị ai thì các giá trị cuối ai này tạo thành một dãy không tăng lập luận logic tương tự bài chia dãy con tăng thì bằng cách tìm dãy con không tăng lớn nhất ấy;

ví dụ dãy: 1 2 1 2 -1 1 4 2 2 3 ta duyệt từ đầu đến cuối tìm các dãy con

xét dãy con tăng mà có 2 phần tử thì nó sẽ gồm 1 2; 1 2; -1 1 như vậy dãy này có phần tử cuối không tăng (chỉ nhỏ hơn hoặc bằng);

tại mỗi vị trí này ta có thể tính đc số dãy con tương ứng có j phần tử được tạo ra:

ta tạo struct cach{} hoặc dùng pair <int,int> để lưu giá trị first = ai; second = số dãy con có j phần tử tính đến ai;

ta lưu vào mảng vector <vector <cach> > sc(n+1)

như vậy sc[j] sẽ lưu tất cả các cách có điểm kết thúc mà có số lượng phần tử dãy con tăng bằng j, trong đó sc[j].[t].first = a[i], sc[j].[t].second = số lượng dãy con

Như ví dụ trên sc[2] là vector chứa các cach mà tạo nên dãy con có 2 phần tử: các vị trí tạo ra dãy con tăng có 2 phần tử chính là 1 2; 1 2; -1 1; số cách tương ứng tạo ra dãy con:

tại vị trí i =2, a[i] = 2 số cách là 1

tại vị trí i =4, a[i] = 2 số cách là 3

tại vị trí i =6 , a[i] = 1 số cách là 4

vậy véctor sc[2] sẽ có các phần tử là {(2,1) (2,3) (1,4)} đây là dãy giảm theo chỉ số first

bây giờ ta tính số dãy con có j phần tử tạo thành khi duyệt theo ai từ 1 đến n; dãy con kết thúc tại ai sẽ tạo ra các dãy con với các dãy con trước đó có j-1 phần tử mà có phần tử cuối nhỏ hơn ai

như vậy số lượng dãy con mới tạo ra ta phải tìm ở sc[j-1]. và phần tử đó phải nhỏ hơn cach(ai, 0); số cách tạo thành sẽ là tổng các cách ở sc[j-i] mà có a_{jt} < ai;

để tính toán cho nhanh hơn không phải cộng lần lượt các phần tử sc[j-i] mà có $a_{jt} <$ ai; thì các mảng sc[j] này ta cộng dồn số cách vào phần tử đứng sau, nên các dãy này có chỉ số first là giảm dần nhưng chỉ số second là cộng dồn nên tăng dần;

kết quả với dãy : 1 2 1 2 -1 1 4 2 2 3 thì các mảng sc[j] như sau

```
i=1 j= 1
: a =1 sodaycon = 1
i=2 j= 2
: a =2 sodaycon = 1
i=3 j= 1
: a =1 sodaycon = 2
i=4 j= 2
: a =2 sodaycon = 3
i=5 j= 1
: a =1 sodaycon = 2
: a =-1 sodaycon = 3
i=6 j= 2
: a =2 sodaycon = 3
: a =1 sodaycon = 4
i=7 j= 3
: a =4 sodaycon = 4
i=8 j= 3
: a =4 sodaycon = 4
: a =2 sodaycon = 5
i=9 j= 3
: a =4 sodaycon = 4
: a =2 sodaycon = 6
i=10 j= 4
: a =3 sodaycon = 2
```

ví dụ ta xét đến giá trị i = 7: a[7] = 4 ta tìm đc dãy con tăng lớn nhất ở đây j = 3

để tính số lượng dãy con có 3 phần tử mà kết thúc tại ai = 4 thì ta tìm về mảng các dãy con có 2 phần tử đó chính là dãy sc[2] ứng với j = 2

trong sc[2] thì có 2 phần tử là {(2,3) (1,4)} các phần tử này có chỉ số first đều nhỏ hơn 4 nên số dãy con có 3 phần tử tạo nên bởi a7=4 chính là toàn bộ các dãy con có 2 phần tử : đó chính là phần tử cuối cùng (1,4) là 4 phần tử:

Tại sao số dãy con có 2 phần tử lại là 4 vì ta dùng cộng dồn vào phần tử cuối lên tất cả các dãy con có 2 phần tử đã được cộng vào chỉ số second phần từ cuối của mảng sc[2] tức là cách(1,4) tương ưng a[6] = 1;

khi có được số các dãy con có 3 phần tử vừa tạo thành bởi a7 thì ta cộng dồn với các phần tử đã có 3 phần tử trước đó

vì lúc này sc[3] = null; nên ta sc[3]. pushback (cach(a7, 4)) như vậy các dãy con có 3 phần tử bây giờ có một vị trí kết thúc là a7=4 và có 4 dãy con như vậy tạo ra;

xét tiếp i=8; a[8] = 2; ta tìm đc j tương ứng cũng là 3

ta lại xét mảng sc[2] nó vẫn có 2 giá trị là {(2,3) (1,4)} ta phải tỉm xem trong mảng sc[2] này các dãy mà kết thúc bởi các giá trị <2. vì sc[2] là giảm nên ta có thể tìm kiếm ngược lại sc[2].rbegin về rend với giá trị tìm kiếm là cach (a[8],0)

ta tìm cach (a[8],0) chính tìm phần từ đầu tiên có a = a[8], phần tử này sẽ không thể cùng với a[8] tạo nên dãy có 3 phần tử được; ta tìm đc vị trí này là jt

int jt = lower_bound(sc[j-1].rbegin(),sc[j-1].rend(), cach(a[i], 0))-sc[j-1].rbegin();

vì đây là mảng cộng dồn nên ta chỉ cần lấy chỉ số second phàn tử cuối trừ phần tử ở jt là là ra số dãy con mà có a <a[i] , ta gọi số này là d:

int d=sc[j-1].rbegin()->second; //nếu mà tất cả các giá trị first ở sc[j-1] đều nhỏ hơn ai thì d = d=sc[j-1].rbegin()->second;

int ns = sc[j-1].size();

if (jt<ns) d-=sc[j-1][ns-1-jt].second; // còn nếu có phần tử lớn hơn thì phải trừ đi giá trị second tại điểm đó - sc[j-1][ns-1-jt].second; vì đây là tổng cộng dồn

if (d<0) d+= M;

tại sao có dòng này: đây là chỗ bố không để ý phải tìm nguyên nhân mãi mới ra

vì ta lấy phần dư cho M lên có thể số lớn hơn khi lấy dư với M lại cho số nhỏ hơn

ví dụ 10%7 =3; và 6%7 = 6;

rõ ràng nếu 10>6 và 10 – 6 =4 nhưng khi lấy dư xong thì 10%7 - 6%7 = -3;

vì vậy để bù vào khi trừ âm thì ta -3+7 = 4 thì trở về đúng giá trị phép trừ

như ở trên mảng sc[j] ở trên là cộng dồn vào chỉ số second, các phần tử phía sau co second là cộng dồn của các chỉ số phía trc, nhưng vì là lấy dư cho nên có thể phần tử trước lại lớn hơn phần tử sau mặc dù cộng dồn số dươgn vì vậy để trả về kết quả đúng nếu d<0 ta phải bù vào M

con phải ghi nhớ nội dung này khi trừ các số mà chia dư sau khi có d chính là các dãy con có j phần tử mới tạo thêm rồi thì ta tính lại mảng sc[j];

nêu f[j] = ai thì phần tử này bằng với phần tử đã có trong sc[j] (nó là phần tử cuối vì dãy này luôn nhỏ hơn hoặc bằng) thì ta chỉ cộng dồn phần tử mới tạo thành vào phần tử cuối

if (f[j] == a[i]) sc[j].rbegin()->second = (sc[j].rbegin()->second+d)%M;

còn nếu nó nhỏ hơn thì đây là phần tử mới tạo thành của sc[j] ta phải chèn thêm nó vào cuối; trước khi chèn ta cộng d vào giá trị second của phân tử cuối hiện tại (vì là cộng dồn mà)

```
else
        {
            if (!sc[j].empty()) d= (sc[j].rbegin()->second + d)%M;
            sc[j].push_back(cach(a[i],d));
        }
```

kết thúc ta chỉ cần xuất ra sc[jmax].rbegin()->second là kết quả

Bố dùng vector <vector <cach>> scvà tìm kiểm ngược như vừa xong

con có thể không tìm kiếm ngược nhưng phải định nghĩa làm hàm comp để tính so sánh giảm dần vì vị trí tìm kiếm phải là cach (ai, M+1) để nó chỉ về phần tử đầu tiên nhỏ hơn ai, sau đó ta lấy d=sc[j-1].rbegin(). second – sc[j-1][jt-1].second ta lùi lại một vị trí để trừ

Cách khác ta dùng vector <list <cach> >sc

khi tìm kiếm ta tìm kiếm xuôi bình thường với

iterator it=sc[j-1].lowbound(cach(ai,0))

d = sc[j-1].front().second - it->second // chú ý nếu d âm

mỗi khi tính được giá trị sc[j] thì không pushback mà pust_front hoặc thay thế phàn tử sc[j].front().second bằng giá trị mới;

cách nữa là dùng vector <map<int, int> >sc cũng ok

dùng map cũng dễ tìm kiếm hơn ta tìm kiếm lowbound bằng a[i] và cộng dồn cũng viết dễ hơn sc[j][a[i]] += d;

con có thể thử list và map bố thấy hay hơn

code bố pass con đọc tham khảo, đừng copy

```
#include <bits/stdc++.h>
#define M 1000000007
using namespace std;
typedef pair <int, int> cach;
int main()
{
   int n;
   cin >>n;
   int *a = new int [n+1];
   for (int i = 1; i<=n; i++) cin >>a[i];
   vector <int> f(n+1, 1000000007);
   f[0] = -1000000007;
   vector <vector <cach> > sc(n+1);
   sc[0].push_back(cach (-M,1));
   for (int i = 1;i<=n; i++ )
   {
      int j = lower_bound (f.begin(), f.end(), a[i]) - f.begin();
      int jt = lower_bound(sc[j-1].rbegin(),sc[j-1].rend(), cach(a[i], 0))-sc[j-
1].rbegin();
      int d=sc[j-1].rbegin()->second;
      int ns = sc[j-1].size();
      if (jt<ns) d-=sc[j-1][ns-1-jt].second;
      if (d<0) d+= M;
      if (f[j] == a[i]) sc[j].rbegin()->second = (sc[j].rbegin()->second+d)%M;
      else
      {
         if (!sc[j].empty()) d= (sc[j].rbegin()->second + d)%M;
         sc[j].push_back(cach(a[i],d));
```

```
        }
        f[j] = a[i];
    }
sc[j][k].second<<endl;
    int i = lower_bound(f.begin(), f.end(), 1000000007) - f.begin()-1;
    cout << sc[i].rbegin()->second;
    return 0;
}
```

code làm bit của họ

```
#include <algorithm>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <deque>
#include <fstream>
#include <iostream>
#include <map>
#include <set>
#include <sstream>
#include <stack>
#include <queue>
#include <vector>
#include <utility>
#define BASE 1000000007
using namespace std;

vector< pair<int,int> > v;
```

```
vector<int> len[100010];
int tx[100010],a[100010],f[100010],g[100010],st[100010];
int n;

void update(int x,int v)
{
//    cout << x << ' ' << v << endl;
    while (x <= 100000)
    {
        tx[x] += v;
        if (tx[x] < 0) tx[x] += BASE;
        tx[x] %= BASE;
        x += (x & -x);
    };
};

int calc(int x)
{
    int r = 0;
    while (x)
    {
        r = (r + tx[x]) % BASE;
        x -= (x & -x);
    };
    return r;
};

int main()
{
```

```
//    freopen("nt.in","r",stdin);
//    freopen("nt.ou","w",stdout);
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
    {
        scanf("%d", &a[i]);
        v.push_back(make_pair(a[i],i));
    };
    sort(v.begin(),v.end());
    a[v[0].second] = 1;
    for (int i = 1; i < v.size(); i++)
      a[v[i].second] = (v[i].first == v[i - 1].first) ? a[v[i - 1].second] : (i + 1);

    for (int i = 1; i <= 100000; i++) st[i] = BASE;  st[0] = 0;
    int maxlen = 0;
    for (int i = 1; i <= n; i++)
    {
        int inf = 0;  int sup = maxlen;
        while (inf <= sup)
        {
            int med = (inf + sup) / 2;
            if (st[med] < a[i])
            {
                f[i] = med;  inf = med + 1;
            }
            else sup = med - 1;
        };
        f[i]++;  maxlen = max(maxlen,f[i]);  st[f[i]] = min(st[f[i]],a[i]);
len[f[i]].push_back(i);
```

```
//      cout << i << ' ' << f[i] << endl;
    };
//  for (int i = 1; i <= n; i++) cout << f[i] << endl;
//  cout << len[1].size() << ' ' << len[2].size() << endl;
    memset(tx,0,sizeof(tx));
    memset(g,0,sizeof(g));
    for (int i = 0; i < len[1].size(); i++) g[len[1][i]] = 1;
    for (int i = 2; i <= maxlen; i++)
    {
        int f1 = 0;  int f2 = 0;
        while (f1 < len[i - 1].size() || f2 < len[i].size())
        {
//          cout << f1 << ' ' << f2 << endl;
            if (f2 == len[i].size() || (f1 < len[i - 1].size() && len[i - 1][f1] < len[i][f2]))
            {
                update(a[len[i - 1][f1]] + 1,g[len[i - 1][f1]]);
//              cout << len[i - 1][f1] << ' ' << g[len[i - 1][f1]] << endl;
                f1++;
            }
            else
            {
                g[len[i][f2]] = calc(a[len[i][f2]]);
//              cout << len[i][f2] << ' ' << g[len[i][f2]] << endl;
                f2++;
            };
        };
        for (f1 = 0; f1 < len[i - 1].size(); f1++)
          update(a[len[i - 1][f1]] + 1,-g[len[i - 1][f1]]);
    };
```

```
    int ret = 0;
    for (int i = 1; i <= n; i++)
      if (f[i] == maxlen) ret = (ret + g[i]) % BASE;
    printf("%d", ret);
};
```

3(i^2+j^2+k^2)>= (i+j+k)^2

vì i<=j<=k => i <= căn(n/3)

2*(i^2+j^2) > =(i+j)^2

2(n-k^2) >= (i+j)^2

i^2 + (i+t)^2 = n-k^2 = d

2i^2 +2it + t^2-d = 0

i^2 – 2i^2 +d = d-i^2

d-i^2 là một số chính phương

- i +căn(d-i^2) >=0 -> i^2< =d/2