BÁO CÁO CHUYÊN ĐỀ

ỨNG DỤNG HÀM TÌM KIẾM NHỊ PHÂN ĐỂ TĂNG TỐC CHƯƠNG TRÌNH

CHUYÊN ĐỀ ỨNG DỤNG HÀM TÌM KIẾM NHỊ PHÂN ĐỂ TĂNG TỐC CHƯƠNG TRÌNH

PHẦN I: MỞ ĐẦU

Trong các kỳ thi học sinh giỏi các cấp, các môn thi như Toán, Lý, Hóa, Sinh chỉ cần đưa ra cách giải đúng là được điểm tối đa. Nhưng đặc thù của môn Tin học thì ngoài việc viết chương trình để đưa ra kết quả đúng còn phải phải có tốc độ chạy của chương trình nhanh. Như vậy ngoài việc tìm ra được thuật toán để giải bài toán, học sinh còn phải lựa chọn thuật toán tối ưu để ăn điểm các Test với dữ liệu lớn. Kĩ thuật tìm kiếm nhị phân các thầy cô đều đã phải trang bị cho học sinh khi vào học đầu năm lớp 10. Tuy nhiên để phong phú hệ thống bài tập cho các thầy cô và các em học sinh, Trong chuyên đề này tôi xin phép chia sẻ với các thầy cô lớp các bài toán sử dụng kĩ thuật tìm kiếm nhị phân để giải quyết bài toán được tối ưu. Tăng tốc chương trình.

PHẦN II: NỘI DUNG

I. LÝ THUYẾT

Có nhiều tài liệu đã trình bày về Giải thuật tìm kiếm nhị phân (Binary Search) . Tôi xin phép trình bày ngắn gọn như sau:

Tìm kiếm nhị phân (Binany Search) là một giải thuật tìm kiếm nhanh với độ phức tạp thời gian chạy là O(log n). Giải thuật tìm kiếm nhị phân làm việc dựa trên nguyên tắc chia để trị (Divide and Conquer). Để sử dụng giải thuật này thì tập dữ liệu thường phải trong dạng đã được sắp xếp theo một trật tự nào đó.

Tìm kiếm nhị phân (Binany Search) là tìm kiếm một phần tử cụ thể bằng cách so sánh phần tử tại vị trí giữa nhất của tập dữ liệu. Nếu tìm thấy kết nối thì chỉ mục của phần tử được trả về. Nếu phần tử cần tìm là lớn hơn giá trị của phần tử đứng giữa thì phần tử cần tìm được sẽ chỉ cần tìm trong mảng con nằm ở bên phải phần tử giữa, khi đó phạm vi tìm kiếm thu hẹp lại còn một nửa; nếu phần tử cần tìm là giá trị nhỏ hơn phần tử đứng giữa thì sẽ tìm ở trong mảng con nằm ở bên trái phần tử giữa. quá trình cũng thu hẹp lại phạm vi tìm kiếm. Tiến trình sẽ tiếp tục như vậy trên mảng con cho tới khi tìm hết mọi phần tử trên mảng con này.

Trước kia khi dạy học sinh sử dụng ngôn ngữ lập trình pascal thì Giáo viên cần trang bị cho học sinh hai hàm tìm kiếm firt và last. Tuy nhiên ngày nay chúng ta đều khai thác lợi ích của C++ nên trong C++ đã có sẵn các hàm dùng để khai thác. Ta lần lượt xét các bài toán sau để hiểu về cách sử dụng các hàm đó:

Bài toán 1: Cho một dãy số nguyên a1, a2, …,an và một số nguyên x. Giả sử dãy số đã được sắp theo thứ tự $a1 \le a2 \le \ldots \le an$. Hãy tìm vị trí đầu tiên trong mảng có giá trị $\ge x$?

Rào $a[0]=-\infty$; $a[n+1]=+\infty$;

Khi đó để biết được chỉ số của phần tử đầu tiên có giá trị ≥x ta sử dụng các hàm:

lower_bound(a+1,a+n+1,x)-a;

Bài toán 2: Cho một dãy số nguyên a1, a2, …,an và một số nguyên x. Giả sử dãy số đã được sắp theo thứ tự a1≤a2≤…≤an. Hãy tìm vị trí cuối cùng trong mảng có giá trị ≤x?

Rào a[0]=-∞; a[n+1]=+∞;

upper_bound(a+1,a+n+1,x)-a-1;

Bài toán 3: Cho một dãy số nguyên a1, a2, …,an và một số nguyên x. Giả sử dãy số đã được sắp theo thứ tự a1≤a2≤…≤an. Tìm chỉ số đầu tiên của phần tử trong dãy a>x là hàm:

Rào a[0]=-∞; a[n+1]=+∞;

upper_bound(a+1,a+n+1,x)-a;

Nếu tất cả các phần tử trong mảng a đều nhỏ hơn x thì nó trả về n+1;

Hệ quả 1: Để đếm số lượng các phần tử trong dãy số a1, a2, …,an có giá trị bằng x. Giả sử dãy số đã được sắp theo thứ tự a1≤a2≤…≤an. Ta chỉ cần gọi:

t:= upper_bound(a+1,a+n+1,x)-lower_bound(a+1,a+n+1,x);

Khi đó t là giá trị cần tìm.

Hệ quả 2: Để đếm xem có bao nhiêu phần tử trong dãy số nguyên a1, a2, …,an (a1≤a2≤…≤an) có giá trị thỏa mãn x<a[i]<y (x, y là hai giá trị cho trước) khi đó ta gọi

u:=lower_bound(a+1,a+n+1,y)-upper_bound(a+1,a+n+1,x);

Khi đó u là giá trị cần tìm.

Hệ quả 3: Cho một dãy số nguyên a1, a2, …,an và một số nguyên x. Giả sử dãy số đã được sắp theo thứ tự a1≤a2≤…≤an. Đếm xem có bao nhiêu phần tử có giá trị >x**:**

v:= n+1-(upper_bound(a+1,a+n+1,x)-a);

Khi đó v là giá trị cần tìm.

CHẶT NHỊ PHÂN

1. Chặt nhị phân trên miền nguyên

Bài toán: Cho hàm f(n) không giảm trên đoạn [a, b] và số nguyên y_0. Hãy tìm số nguyên nhỏ nhất $n_0 \in [a,b]$ sao cho:

$$f(n_0) \geq y_0$$

Hàm dưới đây thực hiện điều trên:

```
int calc() {
    if (f(a)≥y0) return a;
    if (f(b)<y0) return b+1; // Không tìm thấy
```

```
    int lo=a, hi=b;
    while (hi-lo>1) {
        int mid=(lo+hi)/2;
        if (f(mid)≥y0) hi=mid; else lo=mid;
    }
    return hi;
}
```

2. Chặt nhị phân trên miền thực

Bài toán: Cho hàm f(x) liên tục, không giảm trên đoạn [a,b] và số thực y_0 Hãy tìm số thực $x_0 \in$ [a,b] nhỏ nhất sao cho:

$$f(x_0) \geq y_0$$

Hàm dưới đây thực hiện điều trên

```
double calc() {
    if (f(a)≥y₀) return a;
    if (f(b)<y₀) return b+1; // Không tìm thấy
    double lo=a, hi=b;
    int k=int(log2((hi-lo)/EP))+1;
    for(int i=1;i≤k;++i) {
        double mid=(lo+hi)/2;
        if (f(mid)≥y₀) hi=mid; else lo=mid;
    }
    return hi;
}
```

Chú ý EP là sai số dược định nghĩa tùy theo đề bài.

II. BÀI TẬP

Bài 1. Pair.cpp

Cho một dãy số nguyên a1, a2,…,an. Hãy đếm xem trong dãy số đã cho có bao nhiêu cặp số ai,aj với i<j thỏa mãn ai+aj=0?

- Input: File PAIR.INP gồm có dòng đầu tiên ghi số nguyên dương n ($n<=10^5$) các dòng tiếp theo lần lượt ghi các số a1, a2, ..,an có $|ai| \leq 2.10^9$
- Output: File PAIR.OUT một số nguyên duy nhất là số lượng cặp số tìm được.
- Ví dụ:

PAIR.INP	PAIR.OUT
5 1 -2 -1 3 2	2

Thuật toán: Đây là bài toán đơn giản chỉ cần duyệt 2 vòng lặp và kiểm tra xem cặp số (i,j) nào mà ai+aj=0 thì đếm.

```
int  dem=0;
 for (int  i=1; i<=n; i++)
   for (int j=i+1; j<=n ; j++)  if (a[i]+a[j]==0)  dem++;
cout≪dem;
```

Ta ngay với n lớn chương trình chạy chậm vì độ phức tạp là $O(n^2)$ để khử vòng lặp for bên trong ta cố định j thì a[j]=-a[i] do vậy ta chỉ cần tìm các giá trị x=-a[i] xuất hiện bao nhiêu lần trong mảng a thì chứng tỏ có bấy nhiêu cặp thỏa mãn a[i]+a[j]=0 . Để biết có bao nhiêu giá trị x=-a[i] ta dùng hàm tìm kiếm nhị phân để tìm kiếm trong mảng a giúp tốc độ xử lí nhanh, với điều kiện mảng a đã sắp theo thứ tự.

```
void xuli()
  {
    int  res=0;
   sort(a+1,a+n+1);
   a[0]=-oo; a[n+1]=oo;
  for (int i=1; i<=n; i++)
     {
           int x=-a[i];
           int u=lower_bound(a+1,a+n+1,x)-a;
           int v=upper_bound(a+1,a+n+1,x)-a-1;
            if (u<=v)
               if (u<=i &&  i<=v) ds+=v-u ; // xét trường hợp a[i]=0
                    else ds+=v-u+1;
       }
          printf("%d",res/2);   // do mỗi cặp duyệt hai lần nên kq /2.
   }
```

Từ bài toán trên ta phát triển thành bài toán sau dễ dàng cho học sinh làm được và sẽ hứng thú

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 2. Cặp đôi hoàn hảo

Hai số nguyên được gọi là một "Cặp số hoàn hảo" nếu như tổng của chúng bằng giá trị S cho trước. Hãy đếm xem trong dãy số nguyên a1, a2, …, an có bao nhiêu cặp số hoàn hảo.

- Input: capso.inp gồm có:
  - Dòng thứ nhất ghi số nguyên dương n($n\le10^5$) và số nguyên S ($|S|\le10^9$).
  - Các dòng tiếp theo lần lượt ghi các số a1, a2, ..,an ($|ai|\le10^9$).

- Output: capso.out một số nguyên duy nhất là số lượng cặp hoàn hảo.
- Ví dụ:

capso.inp	capso.out
10 7 5 2 5 3 4 3 1 6 4 0	7

Thuật toán: Bài toán này giống như bài 1, chỉ khác là tổng ai+aj=S (S cho trước)

Như vậy dễ dàng học sinh chỉnh sửa một chút là đã giải quyết được bài toán

```
void xuli()
  {
    sort(a+1,a+n+1);
    a[0]=-oo; a[n+1]=oo;
   int res=0;
  for (int i=1; i<=n; i++)
     {
            int k=S-a[i];
            int u=lower_bound(a+1,a+n+1,k)-a;
            int v=upper_bound(a+1,a+n+1,k)-a-1;
            if (u<=v)
              {
                 if (u<=i && i<=v) res+=v-u;
                     else res+=v-u+1;
                }
     printf("%d",res/2);
    }
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 3 Tổng bằng x

Cho hai dãy số nguyên a1, a2, a3, …., am và b1, b2, b3, …, bn và một số nguyên x. Hãy đếm xem có bao nhiêu cặp (i,j) thỏa mãn ai+bj=x.

Input: File sumx.inp:

- Dòng đầu tiên ghi ba số nguyên dương m, n, x ($1 \le m,n \le 100000$).
- Dòng thứ hai ghi các số nguyên a1, a2, a3, .., am ($|ai| \le 10^9$).
- Dòng thứ ba ghi các số nguyên b1, b2, …, bn (($|bj| \le 10^9$).

Output: File sumx. out một số nguyên duy nhất là số cặp (i,j) tìm được

Example:

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

Thuật toán đơn giản chỉ cần duyệt 2 vòng lặp for độ phức tạp của thuật toán là O(m.n). Ta thấy ý nghĩa của vòng lặp j với a[i] cố định vòng lặp j đếm xem có bao nhiêu phần tử của mảng b có giá trị =x-a[i]. Để giảm độ phức tạp của thuật toán trước tiên các em sắp xếp mảng a sau đó thực hiện việc đếm nhị phân xem có bao nhiêu phần tử=x-b[i]

```
void xuli()
    {
      sort(a+1,a+n+1);
      a[0]=-oo; a[n+1]=oo;
     int res=0;
      for (int i=1; i<=m; i++)
       {
            double  s=x-b[i];
             int u=lower_bound(a+1,a+n+1,x)-a;// chi so dau tien>=x
             int v=upper_bound(a+1,a+n+1,x)-a;// chi so dau tien >x
            res+=v-u;
        }
       printf("%d",res);
    }
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 4 seagame **(Bài 3.18 sách chuyên Tin quyển 1- VOI2008)**

Hai bạn học sinh lúc ngồi nhàn rỗi nghĩ ra trò chơi sau đây, Mỗi bạn chọn trước một dãy số gồm n số nguyên, Giải sử dãy số mà bạn thư nhất chọn là b1, b2, ..,bn, còn dãy số mà bạn chọn thứ 2 là c1, c2, …,cn. Mỗi lượt chơi, mỗi bạn đưa ra một số hạng trong dãy số của mình. Nếu bạn thứ nhất chọn số bi ($0<i\leq n$), bạn thứ 2 chọn số cj ($0<j\leq n$)thì giá trị của lượt chơi sẽ là |bi+cj|. Hãy tìm giá trị nhỏ nhất của một lượt chơi trong số các lượt chơi có thể.

File input sgame.inp gồm có:

- Dòng đầu tiên chứa số nguyên dương n ($-<n<=10^5$)
- Dòng thứ hai chúa dãy số nguyên b1, b2, …,bn ($|bi|\leq 10^9$,i=1,…,n);
- Dòng thứ ba chứa dãy số nguyên c1, c2, …,cn ($|cj|\leq 10^9$, j=1,2,…,n);

File ouput sgame.out một số nguyên duy nhất là giá trị nhỏ nhất tìm được.

Ví dụ:

sgame.inp	sgame.out
2 1 -2 2 3	0

Thuật toán: Nhận xét thấy rằng bi+cj càng nhỏ khi cj càng gần –bi như vậy việc đầu tiên ta sẽ sắp xếp lại mảng c1≤c2≤c3≤…≤cn.

Sau đó dùng hàm lower_bound() để tìm vị trí xuất hiện đầu tiên của –b[i] trong mảng c mà –b[i]<=C do vậy nếu k=lower_bound(c+1,c+n+1,–b[i])-c thì , $c_{k-1}<-b_i\leq c_k$

```
void xuli()
{
  sort(c+1,c+n+1);
  int res=2000000001;
  int t;
  for (int i=1; i<=n; i++)
    {
     int x=-b[i];
     int k=lower_bound(c+1,c+1+n,x)-c;
     if(k==1) t=abs(c[1]+b[i]);
          else if (k==n+1) t=abs(c[n]+b[i]);
               else t=min(abs(c[k-1]+b[i]),abs(c[k]+b[i]));
     res=min(res,t)  ;
     }
     cout<<res;
}
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 5 Trắc nghiệm tâm lí

Trắc nghiệm tâm lí là phương pháp thông dụng để có thể đoán nhận được tính cách của mỗi người trong cuộc sống và cũng là một trò chơi khá phổ biến trên truyền hình. Trong một tò chơi như vậy được phát trên kênh NTTV. Trước tiên, ban tổ chức phát cho mỗi khán giả ngồi xem trực tiếp một phiếu thăm dò trong đó có các câu hỏi trắc nghiệm. Tất cả các phương án trả lời đều có điểm và mỗi người sau khi trả lời xong sẽ được tổng điểm là một số nguyên dương. Có m người tham gia cuộc chơi trên sân khấu. Với người chơi thứ i , sau khi nghe speaker đọc các câu hỏi trắc nghiệm sẽ đưa ra hai số

nguyên si và fi với ý nghĩa rằng những khán giả có tổng điểm nằm trong đoạn [si, fi] sẽ là những người có tính cách phù hợp với mình nhất.

Viết chương trình tính xem mỗi người chơi sẽ tìm thấy bao nhiêu khán giả có tính cách phù hợp với mình nhất.

Input: File prefer.inp:

+ Dòng đầu tiên chứa số nguyên n ($0<n<=10^5$) là số khán giả.
+ Dòng thứ hai chứa n số nguyên dương a1, a2, ..,an ($ai<=10^9$) là tổng điểm của mỗi khán giả.
+ Dòng thứ 3 ghi số nguyên m ($1<m<=10^5$) là số người chơi
+ M dòng tiếp theo, dòng thứ i ghi hai số nguyên si, fi ($1<=si<=fi<=10^9$) là khoảng điểm của những người có tính cách phù hợp nhất với người i nhất.

Output: File prefer.out gồm m dòng, dòng thứ i ghi số lượng khán giả có tính cách phù hợp với người thứ i nhất.

Ví dụ:

PREFER	PREFER
5 7 2 4 5 3 2 1 4 3 10	3 4

Thuật toán: Ta quan sát thấy nếu mảng a được sắp xếp theo thứ tự khi đó thì việc đếm số lượng khán giả có tính cách phù hợp với người thứ i sẽ rất đơn giản chỉ cần sử dụng hàm upper_bound() để tìm vị trí xuất hiện cuối cùng của <= f[i] trong a, hàm lower_bound() để tìm vị trí xuất hiện đầu tiên mà <=s[i] trong mảng a.

```
void xuli(){
  sort(a+1,a+n+1);
 int res;
 for (int i=1; i<=m; i++)    {
          int u=upper_bound(a+1,a+n+1,f[i])-a-1;
          int v=lower_bound(a+1,a+n+1,s[i])-a;
          if (u>=v) res=u-v+1;else res=0;
        cout<<res<<endl;
       }
}
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 6. ESEQ

Cho dãy số nguyên *A* gồm *n* phần tử *a1, a2, ..., an*. Hãy đếm số cặp chỉ số *i, j* thỏa mãn

$$\sum_{p=1}^{i} a_p = \sum_{q=j}^{n} a_q \text{ Với } 1 \le i < j \le n.$$

Input*:* Vào từ file ESEQ.INP có dạng:

+ Dòng đầu ghi số nguyên dương *n* ($2 \le n \le 10^5$)
+ Dòng tiếp theo ghi *n* số nguyên *a1, a2, ..., an* (*ai* <= 10^9) các số cách nhau bởi dấu cách

Output*:* Ghi ra file văn bản ESEQ.OUT một số duy nhất là số cặp tìm được

Ví dụ:

eseq.inp	eseq.out
3 1 0 1	3

Thuật toán: Việc đầu tiên ta đi tìm mảng tổng tiến tố s[i]=a[i]+s[i-1]

Sau đó để tính tổng trên đoạn i…j ta chỉ cần lấy s[j]-s[i-1]

Sau đó sắp xếp mảng s tăng dần. Trong quá trình sắp xếp mảng s dẫn đến chỉ số thay đổi do đó ta phải dùng một mảng thứ hai để lưu chỉ số trước khi sắp xếp. Dùng kiểu dữ liệu cặp để khi mảng s thay đổi thì kéo theo cả chỉ số luôn.

```
typedef pair<long long, int> LI;
LI c[maxn];
int main()
{
  freopen("eseq.inp","r",stdin);
  freopen("eseq.out","w",stdout);
  cin>>n;
  for (int i=1; i<=n; i++) cin>>a[i];
  c[0].first=0;
  int res=0;
  for (int i=1; i<=n; i++) { c[i].first=c[i-1].first+a[i]; c[i].second=i;}
  long long t=c[n].first;
  sort(c+1,c+n+1);
```

```
    LI x;
    for (int i=1; i<=n; i++)
    {
       x.first=t-c[i].first;
       x.second=c[i].second;
       int u=lower_bound(c+1,c+1+n,x)-c;
       x.second=n-1;
       int v=upper_bound(c+1,c+n+1,x)-c-1;
       if (u<=v) res+=v-u+1;
       }
    cout<<res;
    return 0;
}
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 7.Nối điểm

"Trên Hai đường thẳng song song L1 và L2, Người ta đánh dấu trên mỗi đường N Điểm, Các điểm trên đương thẳng L1 Được đánh số từ 1 đến N, từ trái qua phải, còn các điểm trên đường thẳng L2 được đánh số bởi P[1], P[2], . . . P[N] cũng từ trái qua phải, trong đó P[1], P[2], . . P[N] là một hoán vị của các số 1, 2, . . . N

Ta gọi các số gán cho các điểm là số hiệu của chúng. Cho phép nối hai điểm trên 2 đường thẳng có cùng số hiệu. **Yêu cầu**: Tìm cách nối được nhiều cặp điểm nhất với điều kiện các đoạn nối không được cắt nhau.

Dữ liệu: Vào từ File WIRES.INP:

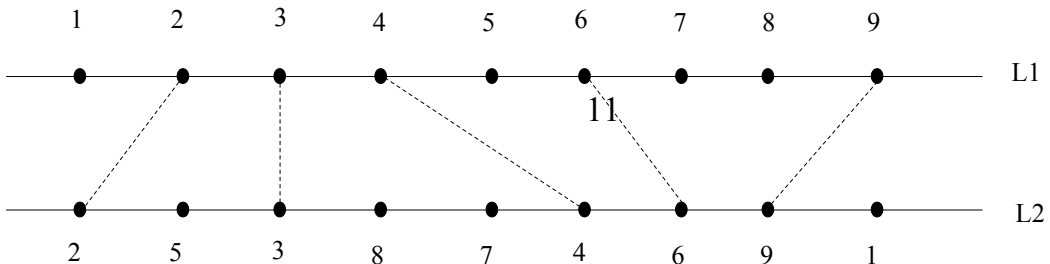
- Dòng Đầu tiên chứa số Nguyên Dương N ($N<=10^5$)
- Dòng thứ hai chứa các số P[1], P[2], ... P[N]

Kết quả Ghi ra File:WIRES.OUT

- Dòng Đầu tiên chứa K là số lượng đoạn nối tìm được
- Dòng tiếp theo chứa K số hiệu của các đầu mút của các đoạn nối được ghi theo thứ tự tăng dần.

Ví dụ:

WIRES. INP	WIRES. OUT
9 2 5 3 8 7 4 6 9 1	5 2 3 4 6 9



Thuật toán: Gọi f[i] là số đoạn thắng tối đa của các cặp nối của các điểm có số hiệu<=i.

Ta có công thức quy hoạch động như sau:

F[i]=max{f[j]+1; với j:=1 ….i-1; đk p[j]<p[i];}

Sau đó tìm max{f[i], i=1..n} là đáp số cần tìm. Để truy vết ta dùng một mảng tr[i] để lưu các điểm nối là j; Với cách này ta phải 2 vòng lặp for nên độ phức tap O (n^2) nên không được full điểm. Để được điểm tối đa ta sử dụng tìm kiếm nhị phân để tăng tốc chương trình

```
int main() {
   freopen("wires.inp","r",stdin);
    freopen("wires.out","w",stdout);
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d",a+i);
    for(int i=1;i<=n;i++) h[i]=n+1; h[0]=-1; idx[0]=0;
    for(int i=1;i<=n;i++) {
       int u=lower_bound(h,h+n+1,a[i])-h-1;
       f[i]=u+1; prev[i]=idx[u];
       if (a[i]<h[u+1]) {h[u+1]=a[i]; idx[u+1]=i;}
    }
    int u=1;
    for(int i=1;i<=n;i++) if (f[i]>f[u]) u=i;
    int res=f[u];
    while (u>0) {x[++slx]=a[u]; u=prev[u];}
    printf("%d\n",res);
    for(int i=slx;i>=1;i--) printf("%d ",x[i]);
}
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 8. Bán kính phủ sóng

Quốc lộ 5 đi qua địa phận Hưng Yên, Hải Dương, Hải Phòng được mô hình hóa như một trục tọa độ (mỗi điểm trên đường thẳng tương ứng với một số thực được gọi là tọa độ của điểm này). Dọc theo Quốc lộ có n điểm dân cư, mỗi điểm dân cư có thể được mô hình như một điểm trên trục tọa độ. Điểm dân cư thứ i có a_i người dùng điện thoại mạng ABC và đặt tại điểm có tọa độ x_i.

Hãng ABC cần đặt một trạm BTS thu sóng điện thoại từ các điểm dân cư. Trạm BTS có thể đặt tại một điểm bất kỳ trên trục tọa độ mô tả Quốc lộ 5 và có bán kính phủ sóng là k (các máy điện thoại có khoảng cách đến trạm thu không vượt quá k thì thu và nhận được tín hiệu từ trạm BTS này).

Hãy tìm số máy điện thoại lớn nhất mà trạm BTS có thể phục vụ nếu đặt tại một vị trí thích hợp.

Dữ liệu: Vào từ file văn bản ABCTEL.INP

- Dòng đầu tiên ghi hai số nguyên dương n, k ($n \le 10^5, k \le 2.10^6$)
- n dòng tiếp theo, dòng thứ i ghi hai số nguyên $a_i, x_i (1 \le a_i \le 10000, 0 \le x_i \le 10^6)$

Các số liên tiếp trên cùng một dòng cách nhau ít nhất một dấu trống

Kết quả: Ghi ra file văn bản ABCTEL.OUT một số nguyên duy nhất là số máy điện thoại lớn nhất mà trạm BTS có thể phục vụ nếu đặt tại một vị trí thích hợp.

Ví dụ

ABCTEL.INP	ABCTEL.OUT
4 3 4 7 10 15 2 2 5 1	11

Thuật toán: Trước tiên không mất tổng quát ta có thể coi $x_1 \le x_2 \le \ldots \le x_n$. Thuật toán đơn giản là thử tất cả các vị trí x_i là điểm đầu mút bên phải của vùng phủ sóng (như vậy $x_i - 2k$ sẽ là đầu mút bên trái) như vậy ta sẽ duyệt từ cao xuống thấp. Trường hợp ít nhất là dem=b[i].sc; với mỗi giá trị i ta duyệt từ i-1 về 1. Nếu b[j].sc>b[i].fi-2*k thì dem+=b[j].sc; sau đó ds=max(danh sách,dem);

Ver1.0

```
scanf("%d%d",&n,&k);
for(int i=1;i<=n;i++) scanf("%d%d",&b[i].second,&b[i].first);
sort(b+1,b+n+1);    long long ds=0;
for(int i=n;i>=1;i--){
   long long dem=(long long)(b[i].second);
   for(int j=i-1;j>=1;j--) if(b[j].first>=b[i].first-2*k) dem+=(long long)
(b[j].second);
   ds=max(ds,dem);
}
printf("%I64d",ds);
}
```

Với cách này qua được 6 test vì độ phức tạp là $O(n^2)$

Cải tiến: x_i là điểm đầu mút bên trái của vùng phủ sóng như vậy x_i+2k sẽ là đầu mút bên phải việc phủ sóng và với mỗi x_i ta chỉ việc tính tổng các a_j thỏa mãn $x_i \le x_j \le x_i+2k$. Điều này có thể thực hiện trong O(log n) bằng cách thực hiện tìm kiếm nhị phân trên mảng x và sử dụng mảng tổng:

$$s[0]=0, s[i]=s[i-1]+a[i] \forall i=1,2,\ldots,n$$

Độ phức tạp giải thuật là $O¿$

```
int main()
{
   freopen("abctel.inp","r",stdin);
   freopen("abctel.out","w",stdout);
   ios::sync_with_stdio(false);
   cin>>n>>k;
   for (int i=1; i<=n;i++)        cin>>b[i].sc>>b[i].fi;
   sort(b+1,b+n+1);
   for (int i=1; i<=n; i++) x[i]=b[i].fi;
   s[1]=b[1].sc;
   for (int i=2; i<=n; i++)s[i]=s[i-1]+b[i].sc;
   int kq=0;
   for (int i=1; i<=n; i++)    {
      int j=upper_bound(x+1,x+n+1, x[i]+2*k)-x-1;
      kq=max(kq,s[j]-s[i-1]);
   }
   cout<<kq;
}
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 9. ĐẾM BI

Sống tại nhà Phú Ông từ nhỏ, Bờm và Cuội là đôi bạn rất thân. Một hôm, do làm việc siêng năng, chăm chỉ nên hai bạn đã được Phú Ông thưởng cho MxN viên bi. Có bi rồi, nhưng vấn đề nảy sinh hết sức phức tạp là hai bạn không làm sao chia nhau số viên bi được. Bởi vì ai cũng muốn giành được nhiều viên bi về mình. Băn khoăn suốt mấy ngày đêm, cuối cùng hai bạn đành dắt nhau đến nhờ Phú Ông chia giúp, nghĩ mãi nghĩ mãi rồi Phú Ông cũng tìm ra được một cách chia mà có lẽ theo ông hai bạn sẽ vui vẻ sau khi nhận được số viên về mình. Dắt hai bạn ra sân đình, Phú Ông yêu cầu Bờm vẽ N đường thẳng khác nhau song song với trục Oy có hoành độ X_i(0<i<=n), Cuội vẽ M đường thẳng khác nhau song song với trục Ox có tung độ Y_i(0<i<=m), rồi đặt vào các

giao điểm của các đường thẳng, mỗi giao điểm 1 viên bi. Sau đó Phú Ông vẽ một đường tròn với toạ độ tâm (U;V) bán kính R, cho Bờm lấy số viên bi ở phía ngoài đường tròn, Cuội lấy số viên bi còn lại.

Yêu cầu : Tính số viên bi chênh lệch của hai bạn.

Input: Vào file marbles.inp

- Dòng đầu tiên theo thứ tự là 5 số N, M, U, V, R.
- Dòng thứ hai ghi N số mô tả hoành độ các đường thẳng Bờm vẽ.
- Dòng thứ ba ghi M số mô tả tung độ các đường thẳng Cuội vẽ.

Output: Ghi ra file marbles.out

Một số nguyên duy nhất là số viên bi chênh lệch của Bờm và Cuội.

Giới hạn : 0<N, M ≤50000; Các số nguyên U, V, R, X_i, Y_i có giá trị tuyệt đối <=10^7. Có 60% test N, M ≤2000

Example:

marbles.inp	marbles.inp
3 4 3 3 3 1 5 6 1 7 6 3	2

Thuật toán: Những viên bi nằm trong đường tròn là những viên bi có tọa độ (x_i,y_i) thỏa mãn phương trình sau: $(x_i-u)^2+(y_i-v)^2 \le R^2 \Leftrightarrow (yi-v) \le R^2-(x_i-u)$

$\Leftrightarrow$ v-$\sqrt{R^2-¿¿}$

$$y_{min} \le y_j \le y_{max}$$

Như vậy ta sẽ tính ymin= v-$\sqrt{R^2-¿¿}$

Khi đó số viên bi trong hình tròn sẽ là p, trong đó

p=upper_bound(y+1,y+m+1,ymax)-lower_bound(y+1,y+m+1,ymin);

Như vậy độ chênh lệch cần tìm sẽ là mn-2*p

Code chương trình :

```
int main(){
 freopen("marbles.inp","r",stdin);
 freopen("marbles.out","w",stdout);
 cin>>n>>m>>u>>v>>r;
 for (int i=1; i<=n; i++)  cin>>x[i];
 for (int i=1; i<=m; i++)  cin>>y[i];
 sort(y+1,y+m+1);
 for (int i=1; i<=n; i++)
 {
```

```
        if ((long long)(r)*r-(long long)(x[i]-u)*(x[i]-u)>=0)
        {
           ymax=v+sqrt((long long)(r)*r-(long long)(x[i]-u)*(x[i]-u));
           ymin=v-sqrt((long long )(r)*r-(long long)(x[i]-u)*(x[i]-u));
           int u=lower_bound(y+1,y+m+1,ymin)-y;
          int v=upper_bound(y+1,y+m+1,ymax)-y-1;
          p+=v-u+1;
          }
     }
      cout<< (long long )(m)*n-2*p;
     return 0;
  }
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 10. NỐI DÂY

Đề thi vào trường mẫu giáo SuperBabies khá đơn giản: Có n đoạn dây xanh, n đoạn dây đỏ, n đoạn dây tím và n đoạn dây vàng. Độ dài các đoạn dây được cho trước. Mỗi bé được cho một số nguyên L và cần cho biết có bao nhiêu cách chọn đúng 1 đoạn dây xanh, 1 đoạn dây đỏ, 1 đoạn dây tím và 1 đoạn dây vàng để nối lại thành một sợi dây trang trí có độ dài bằng L. Hai cách chọn được gọi là khác nhau nếu có đoạn dây được chọn trong một cách nhưng không được chọn trong cách còn lại.

Yêu cầu: Viết chương trình tìm đáp án để chấm cho các bé.

Dữ liệu: Vào từ file văn bản TERA.INP

- Dòng 1 chứa hai số nguyên dương $n \le 1000$; $L \le 10^9$
- Dòng 2 chứa n số nguyên dương là độ dài n đoạn dây xanh.
- Dòng 3 chứa n số nguyên dương là độ dài n đoạn dây đỏ.
- Dòng 4 chứa n số nguyên dương là độ dài n đoạn dây tím.
- Dòng 5 chứa n số nguyên dương là độ dài n đoạn dây vàng.

Các số trên một dòng của input file được ghi cách nhau bởi dấu cách, độ dài các đoạn dây không quá 10^9

Kết quả: Ghi ra file văn bản TERA.OUT một số nguyên duy nhất là số cách chọn tính được

Ví dụ

TERA.INP	TERA.OUT
3 28 1 1 1 1 1 1 10 11 12 13 14 15	**18**

Chú ý: Ít nhất 40% số điểm ứng với các test có $n \le 50$

Thuật toán: Gọi 4 mảng a, b, c, d lần lượt chứa độ dài của n đoạn xanh, n đoạn đỏ, n đoạn tím và n đoạn vàng.

- Đầu tiên:
  + Tính số cách nối n đoạn xanh và n đoạn đỏ lưu vào mảng p có n^2 phần tử
  + Tính số cách nối n đoạn tím và n đoạn vàng lưu vào mảng q có n^2 phần tử.
- Sắp xếp hai mảng p và q theo thứ tự tăng dần.
- Sử dụng tìm kiếm nhị phân ứng với mỗi giá trị p[i] ta có k=L-p[i] . Tìm k trong mảng q. Nếu xuất hiện thì cộng dồn vào dem

Code chương trình:

```
int main(){
        cin>>n>>l;
        for(int i=1;i<=n;i++)cin>>x[i];
        for(int i=1;i<=n;i++){
              cin>>u;
              for(int j=1;j<=n;j++)  a[(i-1)*n+j]=u+x[j];
              }
        for(int i=1;i<=n;i++) cin>>x[i];
        for(int i=1;i<=n;i++){
              cin>>u;
              for(int j=1;j<=n;j++)  b[(i-1)*n+j]=u+x[j];
              }
        sort(b+1,b+n*n+1);
        for(int i=1;i<=n*n;i++){
        ds+=upper_bound(b+1,b+n*n+1,l-a[i])-lower_bound(b+1,b+n*n+1,l-a[i]);
       }
    cout<<ds;;
} Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing
```

Bài 11. Chạy đua

Bờm được Cuội mời lên mặt trăng để tham quan cơ ngơi của mình. Đã lâu không gặp đôi bạn quyết định tổ chức chạy thi tranh giải. Do đã sống lâu trên mặt trăng nên Cuội luôn chạy nhanh gấp đôi Bờm (tuy vậy nhanh hơn nữa là điều không thể!). Biết ưu thế đó của chồng mình, Hằng Nga tổ chức chạy thi như sau:

Có *n* vị trí phân biệt được đánh dấu dọc theo một đường thẳng. Bờm xuất phát từ vị trí X và chạy về bên phải còn Cuội xuất phát từ vị trí Z chạy về phía bên trái. Tại một vị trí Y nào đó nằm giữa hai vị trí trên Hằng Nga đặt một phần thường và vì là vợ của Cuội nên tất nhiên cô ta sẽ chọn vị trí Y sao cho Bờm không thể tới Y trước Cuội. Tuy vậy, Cuội luôn muốn mình không xuất phát ở điểm ở gần đích hơn so với Bờm.

Hãy đếm xem có bao nhiêu cặp vị trí (X,Y,Z) với X<Y<Z thỏa mãn điều kiện trên?

Input: vào file running .inp

- Dòng đầu tiên ghi số nguyên N - số vị trí đánh dấu (N≤1000)
- N dòng tiếp theo, mỗi dòng chứa một số nguyên là một vị trí đánh dấu (có giá trị nằm trong khoảng [0...10^9])

Output: Ghi ra file running .out

Một dòng ghi một số nguyên duy nhất là số bộ (X, Y, Z) tìm được

Example

running .inp	running .out
5 3 1 10 7 4	4

Thuật toán: Không mất tính tổng quát ta sắp xếp lại dãy số theo thứ tự không giảm.Ver 1. Do Cuội luôn xuất phát ở vị trí không gần hơn đích so với Bờm nên cách đơn giản là duyệt các cặp số thoả mãn ay-ax<=az-ay (trong đó az là vị trí của Cuội, ax là vị trí của Bờm, ay là vị trí phần thưởng) và do cuội luôn chạy nhanh gấp đôi so với Bờm. nên az-ay<=2*(ay-ax). Như vậy cứ đếm bao nhiêu cặp thoả mãn thì dem++.

Code chương trình :

```
sort(a=1,a+n+1);
dem=0;
for (int i=1; i<=n-2; i++)
  for (int j=i+1; j<=n-1; j++)
    for (int z=i+1; j<=n; j++)
      if ( a[j]-a[i]<=a[z]-a[j]   && (a[z]-a[j]<=2*(a[j]-a[i])) dem++;
```

Với thuật toán này chỉ được 50% số điểm. Để được full điểm cần xử lí bớt vòng lặp for bên trong. Đặt d=aj-ai

Ta nhận thấy do aj-ai<=az-aj && az-aj<=2*(aj-ai)

=> d<=az-aj<2*d => d+aj<=az<=aj+2d

Như vậy ta sẽ dùng hai hàm tìm kiếm nhị phân đề tìm các vị trí thoả mãn

```
Dem=0;
For (int i=1; i<=n-1; i++)
  For (int j=i+1; j<=n; j++)  {  d=a[j]-a[i];
     ans+=upper_bound(a+1,a+n+1,a[j]+2*d)-lower_bound(a+1,a+n+1,a[j]+d);}
  cout<<ans;
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

Bài 12. Giá trị lớn nhất

Bạn được cho một dãy số nguyên $A=(a_1,a_2,\ldots,a_n)$. Hãy tìm giá trị lớn nhất của $a_i mod\ a_j$ (phần dư của phép chia số nguyên a_i cho a_j) với $1\le i,j\le n$ và $a_i\ge a_j$.

Input:vào file MAXVAL.INP

- Dòng đầu tiên chứa số nguyên dương n - độ dài của dãy ($1\le n\le 2.10^5$¿
- Dòng thứ hai chứa n số nguyên cách nhau bởi dấu trống $a_i(1\le a_i\le 10^6)$

Output: Ghi ra file MAXVAL.OUT Kết quả tìm được

Example:

MAXVAL.INP	MAXVAL.OUT
3 **2 4 5**	**1**

Ghi chú: 50% số test có $n\le 5000$

Thuật toán: Để tìm giá trị lớn nhất của phép chia dư a[i] mod a[j] ta dễ dàng tìm được khi sử dụng 2 vòng lặp for.

```
for (int i=1; i<=n; i++)
   for (int j=1; j<=n; j++)        {
      if (a[i]>=a[j]) r=a[i] %a[j];
      res=max(res,r);         }
```

Với cách này chỉ được 50% số điểm. Để cải tiến chương trình ta nhận thấy nếu các giá trị trong mảng a được sắp xếp theo thứ tự: a1<a2<….<an

Thì ta chỉ cần xét những giá trị gần 2*a[i], 3*a[i], …., ka[i] sẽ luôn cho giá trị chia dư lớn nhất. Như vây tại sử dụng tìm kiếm nhị phân tìm vị trí cuối cùng<=2*a[i] khi đó đây là phần tử luôn cho giá trị lớn nhất khi chia dư cho a[i]

Code chương trình :

```
int main() {
   freopen("maxval.inp","r",stdin);
```

```
        freopen("maxval.out","w",stdout);
        ios::sync_with_stdio(false);
        cin>>n;
        for (int i=1; i<=n; i++)cin>>a[i];
        sort(a+1, a+n+1);
        for (int i=1; i<=n; i++)       {
            int x=2*a[i];
            while (x<=2*a[n])     {
                    int u=lower_bound(a+1, a+n+1,x)-a-1;
                    res=max(res,a[u]%a[i]);
                    x+=2*a[i];
                }
            }
             printf("%d",res) ;
             return 0;
}
```

Link test: https://drive.google.com/file/d/1-zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing

III. BÀI TẬP THAM KHẢO

BÀI 1.TRẠM THU SÓNG BTS

Vịnh Hạ Long là một trong những thắng cảnh thiên nhiên nổi tiếng không chỉ ở Việt Nam mà còn trên Toàn thế giới. Trên Vịnh có rất nhiều hòn đảo đẹp là nơi tham quan của khách du lịch mỗi khi đến thăm vịnh. Để phục vụ nhu cầu của các khách tham quan, công ty viễn thông VNPT quyết định đặt một trạm thu sóng điện thoại (BTS) sao cho mọi khách tham quan trên các đảo có thể sử dụng điện thoại thông minh để liên lạc cũng như ghi nhận những cảm xúc của mình trên các mạng xã hội.

Để đơn giản, ta có thể coi bờ biển là đường thẳng trùng với trục Ox trên măt phẳng tọa độ Đề-các. Mỗi hòn đảo như là một điểm có tung độ dương trên mặt phẳng này. Có tất cả n hòn đảo như vậy. Trạm BTS cần xây dựng là một điểm nằm ngay trên bờ biển (trục hoành) và vùng phủ sóng của trạm là hình tròn bán kính R với tâm là điểm đặt trạm. Tất cả các hòn đảo nằm trong hình tròn này (kể cả trên biên) đều bắt được sóng. Tất nhiên, để giảm chi phí đầu tư thì bán kính phủ sóng R càng nhỏ càng tốt

Yêu cầu: Hãy tìm vị trí đặt trạm BTS sao cho bán kính phủ sóng tối thiểu để phủ sóng toàn bộ số đảo trên vịnh là nhỏ nhất?. Để đơn giản, bạn chỉ cần tính giá trị nhỏ nhất này.

Input:

- Dòng 1: Ghi số nguyên dương $n(1 \le n \le 10^5)$

- Dòng 2...$n+1$: Dòng $i+1$ ghi hai số nguyên $x_i, y_i (|x_i| \le 10^9, 0 < y_i \le 10^9)$ là tọa độ của hòn đảo thứ i.

Output: Ghi một số thực R duy nhất là bán kính phủ sóng tối thiểu tìm được. R được ghi với 2 chữ số sau phần thập phân.

Example:

Input	Output
2 **1 2** **2 2**	**2.06**

Bài 2. TÌM VỊ TRÍ

Một đại lý kinh doanh xăng dầu có n trạm bán xăng dầu (gọi tắt là cây xăng) đánh số từ 1 đến n trên một đường cao tốc muốn tìm vị trí đặt k bể chứa xăng để cung cấp xăng cho các cây xăng. Trên đường cao tốc người ta đặt các cột mốc cây số, bắt đầu từ cột mốc số 0. Biết vị trí của cây xăng thứ i là ở cột cây số d_i (i=1,2,…,n), $d_1<d_2<\ldots<d_n$.

Yêu cầu: Tìm vị trí đặt k bể chứa xăng tại k trong số n cây xăng sao cho khoảng cách lớn nhất từ cây xăng không có bể chứa xăng đến cây xăng có bể chứa xăng gần nó nhất là nhỏ nhất.

Input:

- Dòng đầu tiên ghi hai số nguyên dương n, k ($n \le 10000$, $k \le n$)
- Dòng thứ hai ghi hai ghi các số $d_1, d_2, \ldots, d_n$ (d_i là các số nguyên dương không quá 2.10^9). Các số nguyên trên cùng một dòng ghi cách nhau ít nhất một dấu cách.

Output: Gồm k dòng, mỗi dòng ghi chỉ số của cây xăng đặt bể chứa xăng.

Example:

vitri.inp	vitri.out
6 3 5 6 12 19 20 27	2 4 6

Ghi chú:

- Trong cách đặt bể xăng như trên khoảng cách xa nhất từ cây xăng không có bể chứa xăng đến cây xăng có bể chứa xăng gần nó nhất là 6
- Có 50% số test có $N \le 10$, 30% số test có $10 < n \le 200$, $k \le 30$

Bài 3. LỐI ĐI BỘ CHO KHÁCH DU LỊCH (OAKTREE.*)

Chính phủ có kế hoạch xây dựng một lối đi bộ dành cho khách du lịch ở giữa một khu rừng sồi. Khu rừng có thể được mô tả như là mặt phẳng có N điểm đặc biệt mô tả các các cây sồi.

Lối đi cho khách du lịch là một đường vòng có dạng hình chữ nhật có các cạnh song song với các trục tọa độ. Nếu như cạnh của hình chữ nhật này đi qua cây sồi nào thì cây sồi đó phải được đốn bỏ. Các cây sồi không nằm trên các cạnh của hình chữ nhật thì không cần phải đốn hạ.

Một dạnh sách P lối đi bộ cho khách du lịch được đệ trình lên chính phủ. Ljuko, một quan chức cao cấp của Bộ Lâm nghiệp - người rất yêu cây cối quan tâm đến việc với kế hoạch xây dựng được đệ trình thì có bao nhiêu cây sồi bị đốn hạ trong quá trình xây dựng các lối đi bộ.

Viết chương trình xác định với mỗi lối đi bộ được xây dựng thì có bao nhiêu cây sồi bị đốn hạ.

Input: File OAKTREE.INP

- Dòng đầu tiên ghi số N là số lượng cây sồi ($N \leq 300000$)
- N dòng tiếp theo, dòng thứ i ghi hai số nguyên X_i, Y_i là tọa độ của một cây sồi ($1 \leq X_i, Y_i \leq 10^9$)
- Dòng kế tiếp ghi P là số lối đi bộ trên kế hoạch ($P \leq 100000$)
- P dòng tiếp theo, mỗi dòng ghi 4 số nguyên X1, Y1, X2, Y2 với ý nghĩa (X1,Y1) là tọa độ của góc trái dưới và (X2, Y2) là tọa độ của góc phải trên của 1 lối đi bộ lần lượt được xây dựng ($1 \leq X1 \leq X2 \leq 10^9$, $1 \leq Y1 \leq Y2 \leq 10^9$)

Output: File OAKTREE.OUT

Gồm P dòng, dòng thứ i ghi số lượng cây sồi bị đốn hạ khi xây dựng lối đi bộ thứ i (theo trình tự trong file input)

Example:

OAKTREE.INP	OAKTREE.OUT
6 1 2 3 2 2 3 2 5 4 4 6 3 4 2 2 4 4 2 2 6 5	3 4 0 1

3 3 5 6 5 1 6 6	

Bài 4. DÃY CON DÀI NHẤT

Cho n số nguyên $a_1, a_2, \ldots, a_n$ ($|a_i| < 10^9$, $0 \le n \le 100\,000$). Hãy xác định dãy con nhiều phần tử nhất từ dãy đã cho, sao cho không có hai phần tử nào của dãy con có tổng chia hết cho m ($2 \le m \le 100\,000$).

Input:

- Dòng thứ nhất chứa 2 số nguyên n và m,
- Dòng thứ 2 chứa n số nguyên $a_1, a_2, \ldots, a_n$.

Output: Đưa ra file văn bản MODM.OUT: Ghi ở dòng thứ nhất số nguyên k – số phần tử của dãy con tìm được,

Example:

modm.inp	modm.out
3 2 1 100 10	2

Ghi chú: Có 60% số test có N≤20

Bài 5. TẶNG HOA

Nhân dịp ngày Quốc tế phụ nữ (8-3), các bạn nam trong lớp quyết định mua hoa tặng các bạn nữ trong lớp mình. Tuy nhiên, đây là một kế hoạch tự phát, mỗi bạn nam tự mình đi mua hoa không bàn bạc với bạn khác. Chính vì vậy cuối cùng có M loại hoa khác nhau được đem đến lớp (các loại hoa đánh số từ 1 đến M), loại hoa thứ i có a_i bông hoa.

Một vấn đề đau đầu được đặt ra cho lớp trưởng - một bạn nam đẹp trai nhất trong lớp- là làm thế nào chia hết các bông hoa này cho các bạn nữ trong lớp để **số bông hoa của bạn nữ nhận được nhiều hoa nhất là nhỏ nhất**. Biết rằng mỗi bạn nữ chỉ nhận các bông hoa cùng một loại (hoặc không nhận được bông hoa nào).

Viết chương trình tính số lượng hoa của bạn nữ nhận được nhiều hoa nhất trong phương án trên.

Input:fgift.inp

- Dòng đầu tiên ghi hai số nguyên dương N ($1 \le N \le 10^9$) là số lượng bạn nữ trong lớp và M ($1 \le M \le 3.10^5; M \le N$) là số lượng loại hoa khác nhau
- M dòng tiếp theo, dòng thứ i ghi số a_i là số lượng hoa của loại hoa thứ i ($1 \le a_i \le 10^9$)

Output: fgift.out

Một số nguyên duy nhất là số bông hoa của bạn nữ nhận được nhiều hoa nhất trong phương án tối ưu (là phương án mà số hoa của bạn nữ có nhiều hoa nhất là nhỏ nhất)

Example:

fgift.inp	fgift.out
5 2 7 4	3
7 5 7 1 7 4 4	4

Bài 6. SÓC VÀ HẠT DẺ

Có *n*chú sóc đứng chời dưới gốc của *m* cây hạt dẻ. Cây hạt dẻ thứ *i* sẽ bắt đầu rụng quả đầu tiên sau T_i giây nữa và cứ sau P_i giây nó lại rụng thêm một quả. Sóc mẹ muốn các cậu con trai của mình đem về tổ không ít hơn *k* quả hạt dẻ để chuẩn bị tránh cơn sóng thần dữ dội đến từ biển Đông xa xôi, nhưng cũng phải thật nhanh chóng trước khi cơn sóng ập đến chứ chẳng thể nhởn nhơ! Vậy nên các chú sóc đang bàn nhau xem đứng ở những gốc cây nào để có thể hững đủ số quả cần thiết trong thời gian nhanh nhất. Thời gian để chú sóc đi về vị trí cần đứng không đáng kể, các chú sóc cũng không di chuyển sang gốc cây nào khác trong lúc hứng hạt dẻ.

Yêu cầu: Hãy tìm thời gian sớm nhất (sau bao nhiêu giây nữa) các chú sóc có thể hứng đủ số quả cần thiết.

Input:squirr.inp

- Dòng đầu tiên chứa các số nguyên *m*, *n*, *k* ($0 < m, n \le 50000; 0 < k \le 10^9$)
- Dòng thứ hai chứa lần lượt các số nguyên $T_i \quad (0 < T_i \le 100; i = 1,..., m)$
- Dòng thứ ba chứa lần lượt các số nguyên $P_i \quad (0 < P_i \le 100; i = 1, 2,..., m)$

Output: squirr.out

Ghi ra một số nguyên duy nhất là thời gian sớm nhất tìm được

Example:

squirr.inp	squirr.out
3 2 5 5 1 2 1 2 1	4 (hai chú sóc đứng ở cây 2 và 3

Bài 7. KHỈ HÁI DỪA

Có một khu vườn rộng nằm ngay bên cạnh một sở thú, ở đó có rất nhiều dừa.

Sở thú đã huấn luyện được 2 loại khỉ đặc biệt: Loại thứ nhất có khả năng hái dừa và loại thứ hai có khả năng bổ những quả dừa để chế biến thành sản phẩm (nước, cùi,...).

Có N con khỉ thuộc loại thứ nhất, con khỉ thứ k của loại này sẽ mất A_k thời gian để leo lên cây và bắt đầu hái quả dừa đầu tiên. Cứ sau mỗi B_k giây nó hái được một quả.Có M con khỉ thuộc loại thứ hai, con khỉ thứ k của loại này sẽ mất C_k thời gian để tìm công cụ và bắt đầu chế biến quả dừa đầu tiên, cứ sau D_k giây nó chế biến xong một quả dừa.

Loại khỉ thứ hai là loại khỉ rất hung hãn nên cúng sẽ đuổi đánh loại khỉ thứ nhất nếu như cùng ở trong vườn tại một thời điểm. Chính vì vậy, đầu tiên người ta cho các con khỉ loại thứ nhất vào hái dừa. Sau khi hái hết những quả dừa, người ta mới đưa con khỉ loại thứ hai vào để chế biến. Thời gian đưa các con khỉ vào/ra khỏi vườn xem như bằng 0.

Chúng ta không biết trong vườn có bao nhiêu quả dừa. Tuy nhiên biết rằng T giây là thời gian kể từ khi đưa những con khỉ thứ nhất vào vườn đến khi số dừa được chế biến hoàn toàn.

Hãy xác định sau thời gian bao nhiêu lâu, kể từ khi đưa những con khỉ thứ nhất vào hái dừa đến khi những con khỉ thứ hai được đưa vào vườn.

Input:

- Dòng đầu tiên ghi T là tổng thời gian hai loại khỉ ở trong vườn ($1 \le T \le 10^9$)
- Dòng thứ hai ghi số nguyên dương N ($1 \le N \le 100$)
- N dòng tiếp theo, mỗi dòng ghi hai số A_k, B_k ($1 \le A_k, B_k \le 10^9$)
- Dòng tiếp theo ghi số nguyên dương M ($1 \le M \le 100$)
- M dòng cuối, mỗi dòng ghi hai số C_k, D_k($1 \le C_k, D_k \le 10^9$)

Output:Một số nguyên là khoảng thời gian từ khi đưa con khỉ thứ nhất đến vườn đến khi đưa con khỉ thứ hai đến vườn

Example:

12 1 3 1 1 5 1	5

Giải thích:

Giải thích: Trong vườn có 3 quả dừa

- **Ở giây thứ 3 bắt đầu hái quả dừa đầu tiên**
- **Ở giây 4, 5 hái các quả dừa thứ 2 và thứ 3**
- **Sau 5 giây đưa loại khỉ thứ hai vào vườn**
- **Ở giây thứ 10 bắt đầu chế biến quả dừa đầu tiên**
- **Ở giây 11, 12 bắt đầu chế biến quả dừa thứ 2 và thứ 4**
- **Sau 12 giây đưa con khỉ thứ hai ra khỏi vườn.**

PHẦN III. KẾT LUẬN

Chuyên đề là dạy cho học sinh cách tăng tốc chương trình nhờ các hàm tìm kiếm nhị phân. Các bài tập chuyên đề được sưu tầm từ nhiều nguồn khác nhau, nội dung bài tập chủ yếu áp dụng kiến thức cơ bản và kĩ thuật tăng tốc chương trình phục vụ cho công việc học tập và giảng dạy cho các học sinh nhằm nắm vững hơn về các hàm tìm kiếm nhị phân. Rất mong nhận được ý kiến đóng góp của các thầy cô để chuyên đề được hoàn thiện hơn, Xin trân trọng cảm ơn!

MỤC LỤC- CHUYÊN ĐỀ DUYÊN HẢI 2020