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

Phan Thị Huệ

Trường THPT chuyên Nguyễn Trãi- Hải Dương

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≤a2≤…≤an. Hãy tìm vị trí đầu tiên trong mảng có giá trị ≥x?

Rào a[0]=-∞; a[n+1]=+∞;

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($n \le 10^5$) và số nguyên S ($|S| \le 10^9$).
  - Các dòng tiếp theo lần lượt ghi các số a1, a2, ..,an ($|ai| \le 10^9$).
- 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 ($0<i\le n$), bạn thứ 2 chọn số cj ($0<j\le n$)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|≤10^9 ,i=1,…,n);
- Dòng thứ ba chứa dãy số nguyên c1, c2, …,cn (|cj|≤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 , $c_{k-1}<-b_i\le c_k$

```
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 trò 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 ($a_i<=10^9$) 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/1zUpiV82fgElDwpr_FSEkfgMxIWaVAra/view?usp=sharing