MỘT SỐ BÀI TẬP ỨNG DỤNG
THUẬT TOÁN TÌM KIẾM NHỊ PHÂN ĐỂ GIẢI

Bài tập 1: Dãy tổng đối xứng

Một dãy các số nguyên không âm A[1], A[2], …, A[N] được gọi là dãy tổng đối xứng nếu ta có thể tách dãy đó làm 2 dãy có tổng các giá trị bằng nhau. Nghĩa là tồn tại một số k trong đoạn [1..N] sao cho tổng A[1] + A[2] + ... + A[k] = A[k+1] + A[k+2] + ... + A[N].

*** Yêu cầu:** Cho một dãy gồm N số nguyên không âm. Hãy tìm dãy con gồm các phần tử liên tiếp dài nhất mà cũng là dãy tổng đối xứng.

*** Dữ liệu vào: Từ tệp văn bản BAI1.INP có cấu trúc:**

- Dòng đầu tiên chứa số nguyên N ($2 \le N \le 5000$).
- N dòng tiếp theo, dòng thứ i trong N dòng chứa giá trị của phần tử A[i] của dãy (i = 1, 2, …, N). Các số trong dãy không âm và nhỏ hơn 200000

*** Dữ liệu ra: Ghi vào tệp văn bản BAI1.OUT có cấu trúc:**

Gồm một dòng: Ghi 1 số là độ dài lớn nhất của dãy con gồm các phần tử liên tiếp dài nhất là dãy tổng đối xứng. Nếu không có kết quả thì ghi số 0.

Ví dụ:

BAI1.INP	BAI1.OUT
6 2 10 3 2 5 1	4

*** Ý tưởng**

Gọi B[i] = A[1]+A[2]+…+A[i]. Với i = 1, 2, ..., N

Cần tìm đoạn dài nhất dãy tổng đối xứng.

- Ta có tổng các phần tử từ vị trí L đến R bằng: B[R]-B[L-1].

- Xét đoạn [L,R], L≤R, ta tìm vị trí K sao cho tổng các phần tử trong đoạn [L,K] bằng tổng các phần từ trong đoạn [K+1,R], nếu có K thì [L,R] là dãy tổng đối xứng.

Nghĩa là tìm K để: $B[K]-B[L-1]=B[R]-B[K]\Rightarrow B[K]=\frac{B[R]-B[L-1]}{2}$

Nhận xét 1: Nếu $B[R]-B[L-1]$ *chẵn thì ta mới tìm vị trí K.*

Nhận xét 2: Tìm vị trí K sao cho $B[K]=\frac{B[R]-B[L-1]}{2}$*. Vì B sắp tăng, nghĩa là* $B[L-1]<B[L]<...<B[R]$*, nên dùng thuật toán tìm kiếm nhị phân tìm vị trí K nhanh hơn và thỏa dữ liệu vào của bài toán.*

Thuật toán kiểm tra đoạn [L, R] là dãy tổng đối xứng không

Đặt $gt=\frac{B[R]-B[L-1]}{2}$

```
function np(gt,L,R:longint):boolean;
var g:longint;
begin
        while (L<=R) do
        begin
             g:=(L+R) div 2;
             if b[g]=gt then exit(true);
             if gt>b[g] then L:=g+1
             else R:=g-1;
        end;
        exit(false);
end;
```

Nhận xét 3: Nếu độ dài dãy cần tìm có, khi đó độ dài lớn hơn hoặc bằng 2.

Gọi d là độ dài cần tìm, 2 ≤ d ≤ n, lần lượt xét từng đoạn có độ dài d và kiểm tra đoạn đó có là dãy tổng đối xứng không.

Bài toán cần tìm độ dài lớn nhất nên ta duyệt đoạn có độ dài d từ N xuống 2.

+ Với mỗi đoạn, ta xác định vị trí đầu đoạn L và vị trí cuối đoạn R. Kiểm tra đoạn này có là dãy tổng đối xứng không bằng thuật toán tìm kiếm nhị phân. Nếu thỏa d là kết quả cần tìm và dừng.

*** Cài đặt chương trình**

```
CONST FI='BAI1.INP'; FO='BAI1.OUT';       MAXN=5000;
VAR a, B:array[0..maxn] of longint;
    n,dodaimax:longint;
```

```
    L,R:longint;
    f:text;
procedure doc;
var i:longint;
begin
        assign(f,fi); reset(f);
        readln(f,n);
        for i:=1 to n do readln(f,a[i]);
        fillchar(b,sizeof(b),0);
        for i:=1 to n do b[i]:=b[i-1]+a[i];
        close(f);
        dodaimax:=0;
end;
function np(gt,L,R:longint):boolean;
var g:longint;
begin
        while (L<=R) do
        begin
             g:=(L+R) div 2;
             if b[g]=gt then exit(true);
             if gt>b[g] then L:=g+1
             else R:=g-1;
        end;
        exit(false);
end;
procedure xuly;
var  d,gt:longint;
begin
        for d:=n downto 2 do
           for L:=1 to n-d+1 do
           begin
                R:=L+d-1;
                if (B[L-1]+b[R]) mod 2 = 0 then
                begin
                    gt:=(b[l-1]+b[r]) div 2;
                    if np(gt,L,R) then
                       begin dodaimax:=d; exit; end;
                end;
           end;
end;
procedure ghi;
var i:longint;
begin
        assign(f,fo); rewrite(f);
```

```
        writeln(f,dodaimax);
        close(f);
end;
begin
        doc;
        xuly;
        ghi;
end.
```

Bài 2. Bước nhảy xa nhất

Cho dãy A gồm N số nguyên không âm A_1, A_2,…, A_N. Một bước nhảy từ phần tử A_i đến phần tử A_j được gọi là bước nhảy xa nhất của dãy nếu thỏa mãn các điều kiện sau:

- $1 \le i < j \le N$.
- $A_j - A_i \ge P$.
- $j - i$ lớn nhất

Khi đó $j - i$ được gọi là độ dài bước nhảy xa nhất của dãy.

Yêu cầu: Tìm độ dài bước nhảy xa nhất của dãy A.

Dữ liệu vào : Từ tệp BAI2.INP có cấu trúc như sau:

- Dòng 1: Gồm hai số nguyên N và P ($1 \le N \le 10^5$; $0 \le P \le 10^9$).
- Dòng 2: Gồm N số nguyên A_1, A_2,…, A_N ($0 \le A_i \le 10^9$ với $1 \le i \le N$).

(Các số cách nhau ít nhất 1 dấu cách)

Dữ liệu ra : Ghi vào tệp BAI2.OUT

gồm một số nguyên dương duy nhất là độ dài của bước nhảy xa nhất của dãy (Nếu không có bước nhảy nào thỏa mãn thì ghi kết quả bằng 0).

Ví dụ:

BAI2.INP	BAI2.OUT
6 3 4 3 7 2 6 4	3

*** Ý tưởng**

Gọi L[i] là giá trị nhỏ nhất của dãy A từ phần tử thứ 1 đến phần tử thứ i.

L[1]:=A[1];

L[i]:=Min(L[i-1],A[i]), i=2,3,..., N

i	1	2	3	4	5	6
$\mathbf{A_i}$	4	3	7	2	6	4
$\mathbf{L_i}$	4	3	3	2	2	2

Với mỗi vị trí j = 2, 3 .., n ta tìm vị trí i $\in$[1, j-1] nhỏ nhất sao cho a[j]-P≥L[i]. Khi đó ta được đoạn j-i thỏa mãn điều kiện bài toán, so sánh độ dài đoạn [i,j] với phương án tối ưu.

Vì $1 \le N \le 10^5$, Mảng L là mảng không tăng nên ta có thể áp dụng thuật toán tìm kiếm nhị phân trong đoạn [1, j-1] để tìm vị trí i nhỏ nhất sao cho a[j]-P≥L[i]. Độ phức tạp O(NlogN).

*** Cài đặt chương trình**

```
CONST FI='BAI2.inp'; fo='BAI2.out'; maxn=100000;
var a,L:array[1..maxn] of longint;
    n,p,kq:longint;    f:text;
procedure doc;var i:longint;
begin
        assign(f,fi); reset(f);
        readln(f,n,p);
        for i:=1 to n do read(f,a[i]);
        close(f);
end;
function min(a,b:longint):longint;
begin
        if a<b then exit(a) else exit(b);
end;
function tknp(i,j:longint):longint;
var d,c,m,x:longint;
begin
        d:=i; c:=j-1;x:=j;
        while d<=c do
        begin
                m:=(d+c) div 2;
                if L[m]>a[j]-p then d:=m+1
                else
                begin
                        x:=m;  c:=m-1;
                end;
        end;
        exit(x);
end;
```

```
procedure xuly;var i,j:longint;
begin
        L[1]:=a[1];
        for i:=2 to n do L[i]:=min(L[i-1],a[i]);
        kq:=0;
        for j:=2 to n do
        begin
                i:=tknp(1,j);
                if j-i>kq then kq:=j-i;
        end;
end;
procedure ghi;
begin
        assign(f,fo); rewrite(f);
        writeln(f,kq);
        close(f);
end;
begin
        doc;
        xuly;
        ghi;
end.
```

Bài 3. Chăn bò

Bờm được nhận vào làm việc cho nhà Phú Ông và nhiệm vụ chính của cậu ta là chăn bò. Với bản tính ham chơi nên cậu ta đã quyết định đóng N cái cọc và cột các con bò vào đó, vì vậy cậu ta thỏa thích chơi đùa mà không sợ các con bò đi mất.

N cái cọc được đặt trên một đường thẳng ở các vị trí $x_1, x_2, \ldots, x_n$. Phú Ông giao cho Bờm chăn thả C con bò. Những con bò này không thích bị buộc vào những chiếc cọc gần các con bò khác. Chúng trở nên hung dữ khi bị buộc gần nhau, vì chúng cho rằng con bò kia sẽ tranh giành cỏ của mình.

Để tránh việc các con bò làm đau nhau, Bờm muốn buộc mỗi con bò vào một cái cọc, sao cho ***khoảng cách nhỏ nhất giữa hai con bò bất kì là lớn nhất có thể***.

Yêu cầu: Hãy tìm giá trị lớn nhất này.

Dữ liệu vào: Cho trong tệp **BAI3.INP** gồm:

- Dòng 1: Chứa hai số nguyên dương N và C ($2 \le C \le N \le 10^5$).
- N dòng tiếp theo, dòng thứ i chứa một số nguyên x_i mô tả vị trí của một cây cọc ($0 \le x_i \le 10^9$). Đương nhiên không có hai cây cọc nào cùng một vị trí.

Dữ liệu ra: Ghi vào tệp **BAI3.OUT**

- In ra giá trị lớn nhất của khoảng cách nhỏ nhất giữa hai con bò bất kì

Ví dụ:

BAI3.INP	BAI3.OUT
5 3 1 2 8 4 9	3

*** Ý tưởng**

Rất khó để xác định khoảng cách nhỏ nhất giữa hai con bò bất kì là lớn nhất. Ta nhận thấy rằng kết quả của bài toán sẽ nằm đâu đó trong đoạn từ 1 đến (x_{max} - x_{min}). Chính vì vậy ta sẽ tìm kiếm nhị phân theo khoảng cách trong đoạn [1.. x_{max} - x_{min}]. Với mỗi giá trị khoảng cách K, ta kiểm tra nếu buộc các con bò với khoảng cách nhỏ nhất lớn hơn hoặc bằng K có thỏa mãn không (buộc đủ C con bò), nếu thỏa thì ghi nhận kết quả hiện tại và thu hẹp không gian tìm kiếm trong đoạn [K+1.. x_{max} - x_{min}], nếu không thỏa thì tìm trong đoạn [1..K-1].

*** Cài đặt chương trình**

```
const fi='BAI3.inp'; fo='BAI3.out';  maxn=100005;
var   f:text;
      a:array [1..maxn] of int64;
      n,c:longint; res:int64; ok:boolean;
procedure nhap;
var  i:longint;
begin
    assign(f,fi);
    reset(f);
    readln(f,n,c);
    for i:=1 to n do readln(f,a[i]);
    close(f);
end;
procedure sort(l,r: longint);
var   i,j,x,y: longint;
begin
         i:=l;
         j:=r;
```

```
          x:=a[(l+r) div 2];
          repeat
            while a[i]<x do inc(i);
            while x<a[j] do dec(j);
            if not(i>j) then
              begin
                 y:=a[i]; a[i]:=a[j];a[j]:=y;
                 inc(i); j:=j-1;
              end;
          until i>j;
          if l<j then sort(l,j);
          if i<r then sort(i,r);
end;
procedure tinh(x:longint);
var h,k:longint; tmp,dem:longint;
begin
        ok:=false;
        tmp:=1; h:=1;k:=2;  dem:=1;
        while (tmp < c) and ( dem <> n ) do
           begin
                if a[k]-a[h] >= x then
                     begin
                        inc(tmp);h:=k;inc(k);
                     end
                else inc(k);
                inc(dem);
           end;
        if tmp=c then ok:=true;
end;
procedure chat;
var   l,r,mid:int64;
begin
    l:=a[1];
    r:=a[n];
    while l<=r do
        begin
            mid:=(l+r) div 2 ;
            tinh(mid);
            if ok  then
              begin
                l:=mid+1;
                res:=mid;
              end
                else r:=mid-1;
```

```
        end;
end;
procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,res);
    close(f);
end;
begin
     nhap;
     sort(1,n);
     chat;
     xuat;
end.
```

Bài tập 4: Trò chơi với dãy số- Đề thi học sinh giỏi quốc gia năm 2008

Hai bạn học sinh trong lúc nhàn rỗi nghĩ ra trò chơi sau đây. Mỗi bạn chọn trước một dãy số gồm n số nguyên. Giả sử dãy số mà bạn thứ nhất chọn là: $b_1, b_2, ..., b_n$ còn dãy số mà bạn thứ hai chọn là $c_1, c_2, ..., c_n$

Mỗi lượt chơi mỗi bạn đưa ra một số hạng trong dãy số của mình. Nếu bạn thứ nhất đưa ra số hạng $b_i (1 \le i \le n)$, còn bạn thứ hai đưa ra số hạng c_j $(1 \le j \le n)$ thì giá của lượt chơi đó sẽ là $|b_i+c_j|$.

Ví dụ: Giả sử dãy số bạn thứ nhất chọn là 1, -2; còn dãy số mà bạn thứ hai chọn là 2, 3. Khi đó các khả năng có thể của một lượt chơi là (1, 2), (1, 3), (-2, 2), (-2, 3). Như vậy, giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể là 0 tương ứng với giá của lượt chơi (-2, 2).

Yêu cầu: Hãy xác định giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể.

Dữ liệu vào: Cho trong tệp BAI4.INP gồm:

- Dòng đầu tiên chứa số nguyên dương n ($n \le 10^5$)
- Dòng thứ hai chứa dãy số nguyên $b_1, b_2, ..., b_n$ ($|b_i| \le 10^9$, i=1, 2, ..., n)
- Dòng thứ hai chứa dãy số nguyên $c_1, c_2, ..., c_n$ ($|c_i| \le 10^9$, i=1, 2, ..., n)

Hai số liên tiếp trên một dòng được ghi cách nhau bởi dấu cách.

Dữ liệu ra : Ghi vào tệp BAI4.OUT

Ghi ra giá nhỏ nhất tìm được.

Ràng buộc

- 60% số tests ứng với 60% số điểm của bài có $1 \le n \le 1000$.

Ví dụ:

BAI4.INP	BAI4.OUT
2 1 -2 2 3	0

*** Ý tưởng**

- Sắp xếp dãy b tăng dần.

- Sắp xếp dãy c tăng dần.

- Với mỗi số b_i , i=1, 2, …, N

  + Tìm trong dãy c một số c_j sao cho $|b_i+c_j|$ nhỏ nhất với mọi j=1,2,…, N.

    Vì c là dãy giảm, sử dụng thuật toán tìm kiếm nhị phân để tìm c_j.

  + Sau khi tìm được c_j ta so sánh $|b_i+c_j|$ với phương án tối ưu.

Độ phức tạp O(NlogN).

Chú ý khi tìm c_j: Nếu ta có b_i thì ta tìm trong dãy c số $-b_i$ hoặc là gần $-b_i$ nhất để có giá trị đạt là nhỏ nhất.

*** Cài đặt chương trình**

```
const fi='BAI4.inp'; fo='BAI4.out'; maxn=100000;
type km=array[1..maxn] of longint;
var b,c:km;  n:longint;  f:text;    min:longint;
procedure doc;
var i:longint;
begin
        assign(f,fi); reset(f);
        readln(f,n);
        for i:=1 to  n do read(f,b[i]); readln(f);
        for i:=1 to n do read(f,c[i]);
        close(f);
end;
procedure sx(L,R:longint; var a:km);
var i,j,g,tg:longint;
begin
        i:=L;
        j:=R;
        g:=a[(l+r) div 2];
```

```
        repeat
                while a[i]<g do inc(i);
                while a[j]>g do dec(j);
                if i<=j then
                begin
                        tg:=a[i];a[i]:=a[j]; a[j]:=tg;
                        inc(i);
                        dec(j);
                end;
        until i>j;
        if i<R then sx(i,R,a);
        if L<j then sx(L,j,a);
end;
function timnp(k:longint):longint;var L,R,M:longint;
begin
        L:=1;
        R:=n;
        while L+1<>R do
        begin
                M:=(L+r) div 2;
                if k>c[m] then L:=m
                else R:=m;
        end;
        if abs(c[L]-k)<abs(C[r]-k) then  timnp:=L
        else timnp:=r;
end;
procedure xuly;var i,j:longint;
begin
        sx(1,n,b);
        sx(1,n,c);
        min:=2000000000;
        for i:=1 to n do
        begin
          j:=timnp(-b[i]);
          if min>abs(c[j]+b[i]) then
               min:=abs(c[j]+b[i]);
          if min=0 then break;
        end;

end;
procedure ghi;var i:longint;
begin
        assign(f,fo); rewrite(f);
        writeln(f,min);
```

```
        close(f);
end;
begin
        doc;   xuly;         ghi;
end.
```