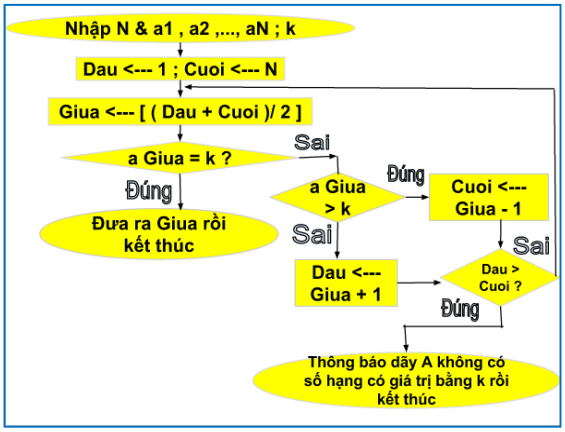
SỞ GD&ĐT BẠC LIÊU

TRƯỜNG THPT CHUYÊN BẠC LIÊU

Chuyên đề bồi dưỡng HSG môn Tin học:

ỨNG DỤNG KỸ THUẬT CHẶT NHỊ PHÂN TRONG VIỆC GIẢI CÁC BÀI TOÁN TIN HỌC



GV Biên soạn : **NGUYỄN CHÁNH TÍN**

Đơn vị: **Trường THPT Chuyên Bạc Liêu**

Tháng 02/2018

MỤC LỤC

LỜI MỞ ĐẦU

Hiện nay trong các kỳ thi, đề thi luôn yêu cầu vận dụng và phối hợp nhiều thuật toán một cách linh hoạt để giải bài toán. Với mỗi bài toán học sinh không chỉ phải đưa ra được thuật toán đúng mà thuật toán đó còn phải tốt đáp ứng yêu cầu về thời gian để có thể "ăn" hết các test lớn. Phần lớn khi giải bài toán, chúng ta chỉ tập trung ở mức độ tìm thuật toán đúng cho bài toán.

Chẳng hạn với bài toán sau: Cho một dãy số nguyên gồm N phần tử, hãy sắp xếp để dãy số đã cho trở thành dãy không giảm?

Trước tiên chúng ta ai cũng nghĩ ngay đến thuật toán sắp xếp tráo đổi (thuật toán sắp xếp "nổi bọt") có độ phức tạp $O(N^2)$. Đây là thuật toán đúng nhưng chỉ trong trường hợp $N \leq 10^3$. Nếu tăng số lượng phần tử lên khoảng 10^5 phần tử thì ta cần một thuật toán sắp xếp tốt hơn như: QuickSort, MergeSort có độ phức tạp $O(NlogN)$.

Từ vấn đề thực tiễn trên thì kỹ thuật "Chặt nhị phân" là một kỹ thuật sẽ giúp làm giảm thời gian thực hiện thuật toán từ O(K) xuống còn O(logK).

PHẦN NỘI DUNG

I- Cơ sở lý thuyết:

Bản chất của kỹ thuật "Chặt nhị phân" là dựa trên ý tưởng của thuật toán tìm kiếm nhị phân. Thuật toán tìm kiếm nhị phân đã được học trong chương trình lớp 10 THPT *(trang 42 – SGK Tin học 10)*, chương trình Tin học lớp 11 đã thiết kế và cài đặt chương trình bằng pascal.

Trước khi trình bày kỹ thuật "Chặt nhị phân" ta tìm hiểu lại bài toán sau đây:

Bài toán: Cho dãy A là dãy tăng gồm N số nguyên khác nhau **a1, a2 ..,an** và số nguyên K. Hỏi trong dãy A có phần tử nào có giá trị bằng K không?(SGK – Tin học 10)

Để giải quyết bài toán trên ngoài thuật toán tìm kiếm tuần tự có độ phức tạp O(N), thuật toán tìm kiếm nhị phân có độ phức tạp ***O(logN)***:

1) Giới thiệu:

Trong khoa học máy tính, thuật toán **tìm kiếm nhị phân** là một thuật toán dùng để tìm kiếm phần tử trong một danh sách đã được sắp xếp. Thuật toán hoạt động như sau. Trong mỗi bước, so sánh phần tử cần tìm với phần tử nằm ở chính giữa danh sách. Nếu hai phần tử bằng nhau thì phép tìm kiếm thành công và thuật toán kết thúc. Nếu chúng không bằng nhau thì tùy vào phần tử nào lớn hơn, thuật toán lặp lại bước so sánh trên với nửa đầu hoặc nửa sau của danh sách. Vì số lượng phần tử trong danh sách cần xem xét giảm đi một nửa sau mỗi bước, nên thời gian thực thi của thuật toán là hàm lôgarit.

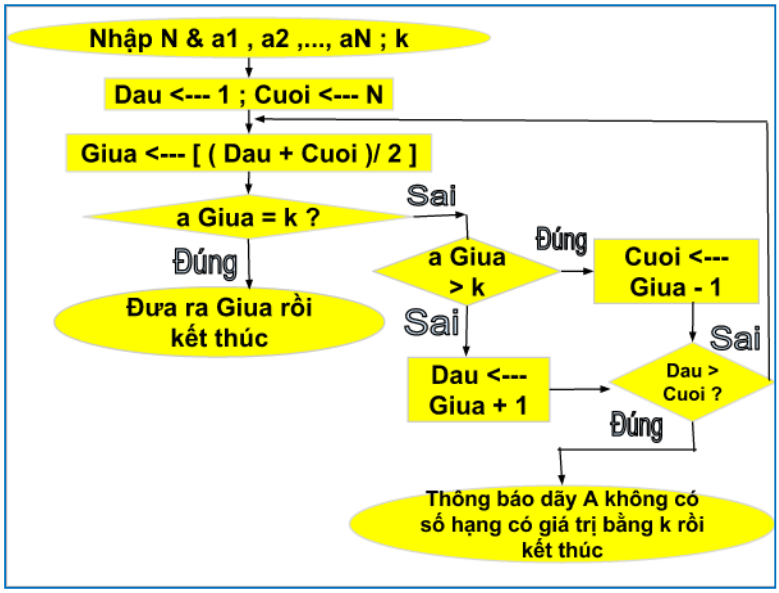
Thuật toán tìm kiếm nhị phân dùng để tìm kiếm phần tử trong một danh sách đã được sắp xếp, ví dụ như trong một danh bạ điện thoại sắp xếp theo tên, có thể tìm kiếm số điện thoại của một người theo tên người đó.

Thuật toán tìm kiếm nhị phân chạy nhanh hơn tìm kiếm tuần tự nhưng cũng có một số nhược điểm. Tìm kiếm nhị phân có thể chậm hơn bảng băm. Nếu nội dung danh sách bị thay đổi thì danh sách phải được sắp xếp lại trước khi sử dụng tìm kiếm nhị phân. Thao tác này thường tốn nhiều thời gian.

2) Ý tưởng thuật toán:

- Vì dãy A là dãy tăng nên ta tìm cách thu hẹp phạm vi tìm kiếm sau mỗi lần so sánh khóa với số hạng được chọn. Để làm được điều đó, ta chọn số hạng Agiua ở "giữa dãy" để so sánh với k, trong đó ***Giua*** =[(N+1)/2]
  Khi đó xảy ra một trong ba trường hợp:
- Nếu A_{giua}=k thì ***giua*** là là phần tử cần tìm, thông báo có phần tử bằng K rồi kết thúc thuật toán.
- Nếu A_{giua}>k thì việc tìm kiếm tiếp theo chỉ xét trên dãy ***a1,a2 ..,agiua-1***
- Nếu A_{giua}<k thì việc tìm kiếm tiếp theo chỉ xét trên dãy ***agiua+1,agiua+2 ..,aN*** Quá trình trên sẽ được lặp đi lặp lại một số lần cho tới khi hoặc đã tìm thấy khóa K trong dãy A hoặc phạm vi tìm kiếm bằng rỗng.

3) Sơ đồ khối:



4) Cài đặt:

Hàm ***BinarySearch*** sau đây sẽ trả về chỉ số phần tử tìm thấy nếu giá trị tìm ***x*** tồn tại trong dãy A. Ngược lại sẽ trả về -1 nếu không tìm thấy.

a) Hàm không đệ quy:

*** Cài đặt bằng Pascal:**

```
function BinarySearch(n,x: integer): integer;
var  l,r: integer;
     mid: integer;
begin
     l:=1; r:=n;
     while l <= r do
     begin
          mid := (l+r)div 2;
          if a[mid]=x then exit(mid);
          if a[mid]<x then l:= mid+1
            else r:= mid-1;
     end;
     exit(-1);
end;
```

*** Cài đặt bằng C++:**

```
int BinarySearch(int a[], int n, int x)
{
      int left= 0;
      int right= n - 1;
      while (left <= right)
      {
            int mid = (left + right) / 2;
            if (x == a[mid]) return mid;
            else if (x < a[mid]) right = mid - 1;
            else if (x > a[mid]) left = mid + 1;
      }
      return -1;
}
```

a) Hàm không đệ quy:

*** Cài đặt bằng Pascal:**

```
function BinarySearch(l,r:integer):integer;
var  mid : integer;
begin
     if l>r then exit(-1);
     mid := (l+r) div 2;
     if a[mid]= x then exit(mid);
     if a[mid]< x then exit(BinarySearch(mid+1,r))
       else exit(BinarySearch(l,mid-1));
end;
```

*** Cài đặt bằng C++:**

```
int BinarySearch(int a[], int left, int right, int x)
 {
     if (left > right) return -1;
     int mid = (left + right) / 2;
     if (x == a[mid]) return mid;
     if (x < a[mid])
          return BinarySearch(a,left,mid-1,x);
     if (x > a[mid])
          return BinarySearch(a,mid+1,right,x);
 }
```

Sau mỗi phép so sánh, số lượng phần tử trong danh sách cần xét giảm đi một nửa. Thuật toán kết thúc khi số lượng phần tử còn không quá 1. Vì vậy thời gian thực thi của thuật toán là *O(logn)*.

* **Chương trình** sau đây tạo ngẫu nhiên mảng ***data***, sắp xếp tăng dần bằng thuật toán Quicksort và nhập vào giá trị tìm ***k***, dùng thuật toán tìm kiếm nhị phân ***BS*** để thực hiện tìm kiếm:

```
const max = 1000000;
type  tlist = array[1..max] of longint;
var   data : tlist;
      f:text;
      tim,k, i,N : longint;

procedure qsort(var a : tlist);

    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;

       begin
          sort(1,N);
       end;

function BS(k: longint): longint;
var dau,cuoi,giua: longint;
begin
        dau:=1;
        cuoi:=N;
        repeat
                giua:= (dau+cuoi) div 2;
                if data[giua]=k then exit(giua)
                  else
                    if data[giua]>k then cuoi:=giua-1
                      else dau:=giua+1;
        until dau>cuoi;
        exit(0);
end;

BEGIN
        //tao ngau nhien mang data
        randomize;
        write('Nhap N: ');
        readln(N);
        for i:=1 to N do
          data[i]:=random(1000);
        //sap xep
        qsort(data);
        //ghi ra file
        assign(f,'SO.INP');
        rewrite(f);
        for i:=1 to N do  writeln(f,data[i]);
        close(f);
        //Tim kiem nhi phan
        write('Nhap k: ');
        readln(k);
        tim:=BS(k);
        if tim=0 then writeln('Khong tim thay')
          else writeln('Tim thay tai vi tri : ', tim);
        readln;
END.
```

Dựa vào ý tưởng của thuật toán tìm kiếm nhị phân ở trên chúng ta xây dựng kỹ thuật "***Chặt nhị phân***" và áp dụng kỹ thuật này giải một số bài toán.

Kỹ thuật Chặt nhị phân có thể chia thành hai dạng:

- Chặt nhị phân dựa theo kết quả.
- Chặt nhị phân trên dãy số.

II- Chặt nhị phân dựa theo kết quả:

1) Xét bài toán

Cho n đoạn dây điện ($1 \le n \le 10^5$). Đoạn dây thứ ***i*** có độ dài a_i ($0 < a_i \le 10^9$). Cần phải cắt các đoạn đã cho thành các đoạn sao cho có được K đoạn dây bằng nhau có độ dài nguyên. Có thể không cần cắt hết các đoạn dây đã cho. Mỗi đoạn dây bị cắt có thể có phần còn thừa khác 0.

Yêu cầu: Xác định độ dài lớn nhất của đoạn dây có thể nhận được. Nếu không có cách cắt thì đưa ra số 0.

Dữ liệu: file văn bản WIRES.INP có cấu trúc:

+ Dòng đầu tiên chứa hai số nguyên N, K

+ Dòng thứ ***i*** trong N dòng tiếp theo chứa số nguyên ***ai***

Kết quả: Đưa ra file văn bản WIRES.OUT,

Một dòng duy nhất ghi độ dài lớn nhất có thể nhận được.

INPUT	OUTPUT
4 11 802 743 547 539	200

2) Giải quyết bài toán

Ta dùng một mảng **A** lưu độ dài các đoạn dây, biến ***Res*** lưu kết quả bài toán, biến ***sum*** là tổng độ dài của N đoạn dây.

Bài toán trên có thể được giải bằng giải thuật như sau: ta thử lần lượt độ dài ***x*** (***x*** nhận giá trị từ **1** tới (***sum*** div ***k***)), sau đó kiểm tra xem có cắt được K đoạn có độ dài ***x*** không?.

Hàm kiểm tra xem có cắt được K đoạn có độ dài ***x*** như sau:

```
Function Check(x: Longint): Boolean;
Var i,count:longint;
Begin
  Count:=0;
  For i:=1 to n do
   Count:= Count + A[i] div x;
  Check:=(Count>=K);
End;
```

Đoạn chương trình xét lần lượt các giá trị của ***x*** như sau:

```
  Res:=0;// Res là biến lưu kết quả bài toán
  For x:=1 to (sum div k) do
    If Check(x) then Res:=x;
```

Ta thấy hàm Check(***x***) có độ phức tạp ***O(n)***, như vậy đoạn chương trình xét từng giá trị của ***x*** sẽ có độ phức tạp ***O(nL)*** với L=min(***sum*** div **k**, max(***ai***)). Theo đề bài thì $n \le 10^5$ và $a_i \le 10^9$, do đó giải thuật trên chỉ giải quyết được bài toán với những bộ dữ liệu nhỏ. Để giải quyết trọn vẹn bài toán ta cần giảm độ phức tạp thuật toán. Chi phí của hàm

Check(***x***) phụ thuộc vào N, ta chỉ có thể giảm số phép thử của x như sau:

Giả sử phạm vi giá trị là [dau .. cuoi], xét một giá trị x:=(dau+cuoi) div 2, nếu Check(x) = true → kết quả bài toán là x hoặc nằm trong đoạn [x+1 .. cuoi], ngược lại kết quả bài toán nằm trong đoạn [dau .. x-1], quá trình này lặp đi lặp lại cho tới khi dau>=cuoi thì dừng.

Đoạn chương trình xét lần lượt các giá trị của ***x*** viết lại như sau:

```
Res:=0;
Dau:=0; cuoi:=sum div k+1;
While (dau<cuoi) do
 Begin
    X:= (dau+cuoi) div 2;
    If Check(x) then
      Begin
       Res:=x;//ghi nhận kết quả sau mỗi lần Check(x)=true
        Dau:=x+1;
       End
     Else cuoi:=x-1;
 End;
```

Đoạn chương trình này chi phí xét các giá trị của x chỉ còn:***O(log(cuoi-dau+1))***, vậy độ phức tạp của thuật toán là ***O(nlogL)*** đáp ứng được yêu cầu bài toán.

Tư tưởng giảm chi phí xét các khả năng của ***x*** trong thuật toán trên hoàn toàn giống với quá trình tìm khóa trong thuật toán tìm kiếm nhị phân.

Các bài toán áp dụng kỹ thuật chặt nhị phân theo kết quả thường có yêu cầu là tìm giá trị nhỏ nhất hoặc lớn nhất. Tùy vào từng yêu cầu mà ta có cấu trúc đoạn chương trình chặt nhị phân khác nhau.

❖ ***Trường hợp tìm giá trị lớn nhất:***

```
Dau:=a-1; cuoi:=b+1;
While ( dau<cuoi ) do
 Begin
  giua:= (dau+cuoi) div 2;
  If Check(giua) then
   Begin
    Res:=giua;//ghi nhận kết quả sau mỗi lần Check()=true
    Dau:=giua+1;
   End
  Else cuoi:=giua-1;
 End;
```

❖ ***Trường hợp tìm giá trị nhỏ nhất:***

```
Dau:=a-1; cuoi:=b+1;
While ( dau<cuoi ) do
 Begin
  giua:= (dau+cuoi) div 2;
  If Check(giua) then
   Begin
    Res:=giua;//ghi nhận kết quả sau mỗi lần Check()=true
    Cuoi:=giua-1;
   End
  Else Dau:=giua+1;
 End;
```

Trong đó, ban đầu kết quả bài toán nằm trong đoạn **[a..b]**, biến Res dùng để lưu kết quả bài toán, hàm Check() kiểm tra giá trị được thử có thỏa mãn không.

Để hiểu hơn về kỹ thuật này ta xét một số ví dụ sau:

3) Bài tập áp dụng:

Bài 1: XẾP TRỨNG

Cho N quả trứng được đưa vào dây chuyền theo thứ tự (quả trứng thứ ***i*** có thể tích là ***ai***). Ở cuối dây chuyền đã có sẵn M thùng chứa trứng. Các thùng này nhận trứng theo quy tắc: chứa trứng cho đến khi đầy thì chuyển sang thùng khác. Hãy tính sức chứa K tối thiểu của mỗi thùng để M thùng này có thể chứa hết trứng theo quy trình trên.

Dữ liệu:

+ Dòng đầu: Ghi 2 số nguyên ***n, m*** ($0 < n, m \le 10^9$)
+ Các dòng tiếp theo: dãy a_i ($0 < a_i \le 10^6$).

Kết quả : Một số duy nhất là số k tìm được.

Ví dụ:

INPUT	OUTPUT	GIẢI THÍCH
5 3 6 5 4 8 9	12	Thùng 1: ***a1, a2*** Thùng 2: ***a3, a4*** Thùng 3: ***a5***

Thuật toán:

Gọi ***sum*** là tổng các a_i, ***dmin*** là giá trị a_i nhỏ nhất, vậy kết quả bài toán nằm trong đoạn [***dmin*** .. ***sum***]. Dãy a_i được chọn phải theo thứ tự, với mỗi giá trị của phép chặt nhị phân ta kiểm tra xem có thể xếp vào đúng M thùng được không.

```
Function Check(val:longint):Boolean;
Var dem,i:longint;
        s:int64;
Begin
  dem:=1; s:=0;
  for i:=1 to n do
   if s+a[i]<=val then s:=s+a[i]
   else
     begin
      if a[i]>val then
        exit(false);
      inc(dem);
      if dem>M then exit(false);
      s:=a[i];
     end;
   Check:=true;
End;
```

Đoạn chương trình chặt nhị phân trong trường hợp tìm giá trị nhỏ nhất

```
Procedure solve;
Var Dau,Cuoi,Giua:longint;
Begin
    Res:=dmax+1;
    {Chat nhi phan ket qua}
    Dau:=dmin-1; Cuoi:=sum+1;
    while  (dau<cuoi)  do
    begin
       Giua:=(Dau+Cuoi)div 2;
       if Check(Giua)  then
         begin
            Res:=Giua;
            Cuoi:=Giua-1;
         end
      else Dau:=Giua+1;
    end;
End;
```

Độ phức tạp thuật toán là ***O(Nlog(sum))***

Bài 2: MOUNTAIN WALKING (MW)

Cho một bản đồ kích thước NxN ($2 \le N \le 100$), mỗi ô mang giá trị là độ cao của ô đó ($0 \le$ độ cao ≤ 110). Bác John và bò Bessie đang ở ô trên trái (dòng 1, cột 1) và muốn đi đến cabin (dòng N, cột N). Họ có thể đi sang phải, trái, lên trên và xuống dưới nhưng không thể đi theo đường chéo. Hãy giúp bác John và bò Bessie tìm đường đi sao cho chênh lệch giữa điểm cao nhất và thấp nhất trên đường đi là nhỏ nhất.

Dữ liệu : + Dòng 1: Chứa số N.

+ N dòng tiếp theo chứa N số nguyên, mỗi số cho biết cao độ của một ô.

Kết quả: Một số nguyên là chênh lệch cao độ nhỏ nhất.

Ví dụ:

INPUT	OUTPUT
5 1 1 3 6 8 1 2 2 5 5 4 4 0 3 3 8 0 2 3 4 4 3 0 2 1	2

Thuật toán:

Chặt nhị phân độ chênh lệch, với mỗi độ chênh lệch mid trong lúc chặt , kiểm tra xem có tồn tại đường đi từ (1,1) đến ô (n,n) mà chênh lệch không quá mid hay không. Dùng BFS loang tìm đường đi từ (1,1) đến (n,n). Tuy nhiên nếu chỉ đơn thuần là xét các đường đi với việc lưu max, min thì bài toán sẽ không hề đơn giản, vì mỗi ô lúc đó sinh ra thêm 110*110 đỉnh khác, tương ứng với max, min tại mỗi ô. Do đó ta tạo hai biến L, R lưu khoảng giá trị mặc định của các ô, vì chênh lệch cao nhất và thấp nhất là trong khoảng mid nên ta chỉ cần tạo khoảng (L,R) mà R-L = mid, khi đó chỉ xét các ô trong khoảng này thì đảm bảo tính chất bài toán.

Do đó độ phức tạp : O(log(N)* N*N^2).

*** Chương trình tham khảo:**

```
program MTWALK;
uses Math;
const maxMove =4;
  X :Array[1..maxMove] of shortInt=(0,0,-1,1);
  Y :Array[1..maxMove] of shortInt=(1,-1,0,0);
  maxN =100;

var  n,minV,maxV,res :ShortInt;
     A :Array[1..maxN,1..maxN] of shortInt;

procedure Enter;
var i,j : shortInt;
begin
    Read(n); minV:=110; maxV:=0;
    for i:=1 to n do
      for j:=1 to n do
        begin
          read(A[i,j]);
          minV:=Min(minV,A[i,j]);
          maxV:=Max(maxV,A[i,j]);
        end;
end;

function Check(l,r : shortInt) : boolean;
var ok :Boolean;
    Free :Array[1..maxN,1..maxN] of boolean;

    procedure Visit(u,v : shortInt);
    var
      i,xx,yy :ShortInt;
    begin
      if (u=n) and (v=n) then
        begin
          ok:=true; Exit;
        end;
      Free[u,v]:=false;
      for i:=1 to maxMove do
        begin
          xx:=u+X[i]; yy:=v+Y[i];
          if (xx>0) and (xx<=n) and
             (yy>0) and (yy<=n) and
             (A[xx,yy]>=l) and (A[xx,yy]<=r) and
             (Free[xx,yy]) then Visit(xx,yy);
          if (ok) then Exit;
        end;
    end;

begin
    FillChar(Free,SizeOf(Free),true);
    ok:=false;
    Visit(1,1);
    Exit(ok);
end;

procedure Solve;
var  i,left,right,mid : shortInt;
```

```
begin
    res:=maxV-minV;
    for i:=minV to A[1,1] do
      begin
        left:=0; right:=res;
        while (left<right) do
          begin
            mid:=(left+right) div 2;
            if (Check(i,i+mid)) then right:=mid else left:=mid+1;
          end;
        res:=left;
      end;
end;

Begin
  assign(Input,'MTWALK.INP'); reset(Input);
  assign(Output,'MTWALK.OUT'); rewrite(Output);
  Enter;
  Solve;
  write(res);
  close(Input); close(Output);
End.
```

Bài 2: BẢO TỒN ĐỘNG VẬT HOANG DÃ

Một khu bảo tồn động vật có n địa điểm và các đường đi hai chiều nối các địa điểm đó, địa điểm thứ i có nhiệt độ là t_i, giữa hai địa điểm bất kỳ có nhiều nhất là một đường đi nối chúng.

Người ta muốn di chuyển một loài động vật quý hiếm từ địa điểm A tới địa điểm B, tuy nhiên nếu chênh lệch về nhiệt độ giữa hai địa điểm liên tiếp trên đường đi là quá cao thì loài động vật này rất có thể bị chết.

Yêu cầu: Hãy chỉ ra một hành trình mà độ lệch nhiệt độ lớn nhất giữa hai địa điểm liên tiếp bất kỳ trên đường đi là cực tiểu.

Dữ liệu: Vào từ file văn bản **MOVE.INP**

+ Dòng 1: Chứa ba số N, A, B ($2 \le n \le 200$; $A \ne B$)

+ Dòng 2: Chứa n số tự nhiên $t_1, t_2, ..., t_n$ ($\forall i: 0 \le t_i \le 20000$)

Các dòng tiếp theo, mỗi dòng chứa hai số nguyên dương u, v cho biết giữa hai địa điểm u và v có đường đi nối chúng.

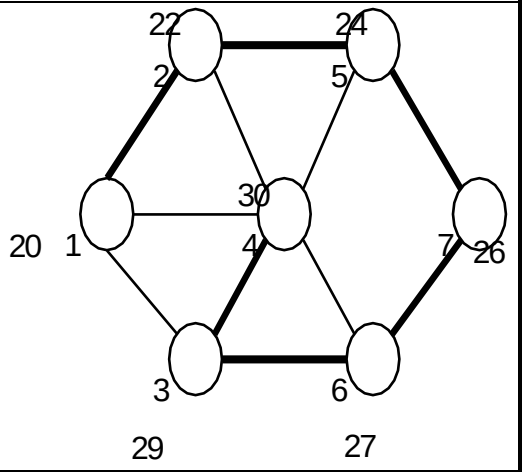
Kết quả: Ghi ra file văn bản **MOVE.OUT**

Dòng 1: Ghi độ lệch nhiệt độ lớn nhất giữa hai địa điểm liên tiếp bất kỳ trên đường đi tìm được, nếu không tồn tại đường đi thì dòng này ghi số -1.

Trong trường hợp tìm được đường đi thì dòng 2 ghi hành trình tìm được, bắt đầu từ địa điểm A, tiếp theo là những địa điểm đi qua, kết thúc là địa điểm B. Các địa điểm phải được liệt kê theo đúng thứ tự đi qua trên hành trình

Các số trên một dòng của Input/ Output file được ghi cách nhau một dấu cách.

Ví dụ:

INPUT	OUTPUT	MINH HỌA
7 1 4 20 22 29 30 24 27 26 1 3 1 4 2 4 2 5 3 4 3 6 4 5 4 6 5 7 6 7	2 1 2 5 7 6 3 4	

Thuật toán: Bài toán yêu cầu tìm một hành trình đi từ A tới B sao cho độ lệch nhiệt độ lớn nhất ***Res*** giữa hai địa điểm liên tiếp bất kỳ trên đường đi là nhỏ nhất. Gọi ***MaxT*** là giá trị nhiệt độ t_i lớn nhất, vậy ***Res*** nằm trong [***0..MaxT***]. Ứng với mỗi giá trị chặt nhị phân ta dùng thuật toán tìm kiếm theo chiều rộng kiểm tra xem có đường đi từ A tới B không.

Đoạn chương trình chặt nhị phân tìm Res

```
Function BFS(Val: Integer): Boolean;
Var u, v: Integer; begin
  InitBFS;
  repeat
    u := Pop;
   for v := 1 to n do
    if a[u,v]and (Trace[v]=0)and (Abs(t[u]-t[v])<=Val)then begin
          Trace[v] := u;
          if v = B then
            Exit(True);
          Push(v);
        end;
  until First > Last; BFS
  := False;
end;
Procedure Solve;
Var Dau, Cuoi, Giua: Integer;
Begin
  Res:=-1;
  Dau := -1; Cuoi := maxT + 1;
  while Dau < Cuoi do
    begin
      Giua := (Dau + Cuoi) div 2;
      if BFS(Giua) then
        begin
         Res:=Giua;
         Cuoi:= Giua-1;
        else
      else Dau:=Giua+1;
    end;
End;
```

Trong trường hợp tồn tại đường đi, ta phải gọi hàm BFS(Res) một lần nữa để truy vết đường đi. Độ phức tạp thuật toán là $O(N^2 logT)$ với N≤ 200 thì chi phí thuật toán vẫn đảm bảo. (T = max(t_i), ***i: 1..N***)

*** Chương trình tham khảo:**

```
const inp='Move.inp';
      out='Move.out';
      maxn=200;
      maxC=maxint;
type mang=array[1..maxn,1..maxn] of boolean;

Var fi,fo:text;
    n,a,b:integer;
    c:mang;
    trace,t:array[1..maxn] of integer;
    Result,first,last,temp:longint;

procedure OpenFile;
begin
   assign(fi,inp);
   reset(fi);
   assign(fo,out);
   rewrite(fo);
end;

Procedure Enter;
var i,j,u,v:integer;
    max:integer;
begin
   readln(fi,n,a,b);
   max:=0;
   for i:=1 to n do Read(fi,t[i]);
   While not eof(fi) do
   begin
           Readln(fi,u,v);
           c[u,v]:=true;
           c[v,u]:=true;
           IF abs(t[u]-t[v])>max then max:=abs(t[u]-t[v]);
   End;
   first:=0;
   last:=max;
End;

function BFS:boolean;
var first2,last2:integer;
    u,v:integer;
    queue:array[1..maxN] of integer;
begin
   first2:=0;
   last2:=1;
   Queue[last2]:=a;
   while first2<last2 do
       begin
           inc(first2);
           u:=Queue[first2];
           for v:=1 to n do
               iF (c[u,v]) and (abs(t[u]-t[v])<=temp) and
 (trace[v]=0) then
                   begin
                       trace[v]:=u;
                       IF v=B then
                           Begin
                               BFS:=true;
```

```
                                    Exit;
                                End;
                            inc(last2);
                            Queue[last2]:=v;
                        end;
            end;
        BFS:=false;
    end;

    procedure Solve;
    begin
        repeat
            temp:=(first+last) div 2;
            fillchar(Trace,sizeof(Trace),0);
            Trace[a]:=n+1;
            if BFS then
                begin
                    last:=temp;
                end
            else
                begin
                    first:=temp;
                end;
        until abs(last-first)<=1;
        temp:=first;
        fillchar(Trace,sizeof(trace),0);
        Trace[a]:=n+1;
        if BFS then Result:=temp
        else
            begin
                temp:=last;
                fillchar(Trace,sizeof(trace),0);
                Trace[a]:=n+1;
                if BFS then Result:=last
                    else result:=-1;
            end;
    end;

    procedure PrintResult;
    var i,sl:integer;
        x:array[1..maxn] of integer;
    begin
        writeln(fo,Result);
        sl:=0;
        i:=b;
        repeat
            inc(Sl);
            x[sl]:=i;
            i:=trace[i];
        until i=a;
        inc(Sl);
        x[sl]:=a;
        for i:=sl downto 1 do Write(fo,x[i],' ');
    end;

    Procedure CloseFile;
    begin
        close(fi);
        close(fo);
```

```
end;

BEGIN
   Openfile;
   Enter;
   Solve;
   PrintResult;
   CloseFile;
END.
```

Bài 4: ICEFROG

Chú chó sói Vjekoslav đang chạy trốn khỏi một đám thợ săn khát máu. Những người thợ săn rất thông minh và họ đang nấp sau những cái cây. Vjekoslav biết điều đó, nhưng không biết chính xác cây nào. Con sói muốn về nơi ở của nó một cách an toàn nhất, tức là càng xa cây càng tốt !

Khu rừng có thể được mô tả bằng một hình chữ nhật kích thước N*M. Những ô trống được đánh dấu bằng ký hiệu '.' , những ô có cây là '+' , vị trí ban đầu của Vjekoslav là 'V' và nhà của nó là 'J'. Vjekoslav có thể chạy từ ô nó đang đứng đến 4 ô chung cạnh xung quanh nó đứng.

Nếu Vjekoslav đang ở ô (R,C) và có một cái cây ở ô (A,B) thì khoảng cách được tính theo công thức :|R-A| + |C-B|. Hãy giúp Vjekoslav tìm đường đi an toàn nhất để về nhà . Đường đi an toàn nhất được hiểu là đường đi mà **khoảng cách bé nhất từ một ô nào đó trên đường đi đó đến tất cả các cây** là lớn nhất.

Dữ liệu

+ Dòng đầu tiên là hai số N,M (0<N,M <=500) là kích thước của khu rừng.

+ N dòng sao mỗi dòng gồmN ký tự thuộc tập {'+','.','V','J'} mô tả khu rừng. Input luôn đảm bảo chứa một ký tự 'V', 1 ký tự 'J' và ít nhất một ký tự '+'.

Kết quả: Gồm một dòng duy nhất là kết quả.

Ví dụ

INPUT1	OUTPUT1	INPUT2	OUTPUT2
4 5 ..... .+++. .+.+. V+.J+	0	4 4 +... .... V..J	3

Thuật toán: Cách làm tương tự như 2 bài trên, kết hợp Loang với chặt nhị phân khoảng cách.

Bài 5: BUS

Một xe buýt của công ty có nhiệm vụ đón nhân viên đến trụ sở làm việc. Trên hành trình, xe buýt sẽ tiếp nhận nhân viên đứng chờ ở các điểm hẹn nếu như xe còn chỗ trống. Xe buýt có thể đỗ lại để chờ những công nhân chưa kịp đến điểm hẹn.

Cho biết thời điểm mà mỗi nhân viên đến điểm hẹn của mình và thời điểm qua mỗi điểm hẹn của xe buýt. Giả thiết rằng xe buýt đến điểm hẹn đầu tiên tại thời điểm 0 và thời gian xếp khách lên xe được bằng 0.

Xe buýt cần phải chở một số lượng nhiều nhất các nhân viên có thể được đến trụ sở. Hãy xác định khoảng thời gian ngắn nhất để xe buýt thực hiện công việc.

Dữ liệu:

+ Dòng đầu tiên chứa 2 số nguyên dương n, m theo thứ tự là số điểm hẹn và số chỗ ngồi của xe buýt

+ Dòng thứ i trong số n dòng tiếp theo chứa số nguyên t_i là thời gian cần thiết để xe buýt di chuyển từ điểm hẹn thứ i đến điểm hẹn thứ i+1 (điểm hẹn thứ n+1 sẽ là trụ sở làm việc của công ty) và số nguyên k là số lượng nhân viên đến điểm hẹn i, tiếp theo k số nguyên là các thời điểm đến điểm hẹn của k nhân viên.

Kết quả

Gồm một dòng duy nhất, là thời gian ngắn nhất tìm được.

Giới hạn: $1 \le n \le 200000$, $1 \le m \le 20000$. Tổng số nhân viên không vượt quá 200000. Kết quả không vượt quá $2^{31}-1$.

Ví dụ

INPUT	OUTPUT
3 2 3 2 4 3 1 3 6 3 7 5 1 5	10

Giải thích:

Trên đường đến công ty có 3 trạm xe buýt. Từ trạm 1 đến trạm 2, trạm 2 đến trạm 3, và từ trạm 3 đến công ty lần lượt mất 3, 1 và 5 đơn vị thời gian. Xe buýt có thể đi như sau: đến thẳng trạm 2, đón người thứ 2, đến trạm 3, chờ 1 đơn vị thời gian để đón người duy nhất ở trạm này, và cuối cùng đến công ty. Tổng cộng xe buýt đi mất 3 + 1 + 1 + 5 = 10 đơn vị thời gian.

Thuật toán:

Nếu xe buýt đến 1 điểm càng muộn thì càng có cơ hội đón được nhiều người (do mọi người sẽ luôn đứng đợi tại bến). Như vậy, cách tốt nhất để đón đủ số người là cho xe buýt đứng chờ tại điểm xuất phát 1 khoảng thời gian, sau đó chạy một mạch đến từng bến, đón khách lên và không cần trở lại bến trước, cũng như không cần chờ ở bến đó.

Việc còn lại là xác định khoảng thời gian nhỏ nhất để đón đủ ***m*** người theo cách đưa đón này. Một lần nữa, ta chú ý rằng, đến công ti càng muộn thì càng đón được nhiều người. Do vậy, ta chặt nhị phân theo khoảng thời gian đến công ti, và với mỗi mốc thời gian ta kiểm tra xem có đón đủ ***m*** người hay không.

Kiểm tra khá đơn giản. Tại mỗi bến ta xem có bao nhiêu người đến bến trước mốc thời gian ***d***. Việc này có thể làm được nếu ta sắp xếp thứ tự thời gian của mỗi người đến bến, và tìm kiếm nhị phân 1 lần nữa.

III- Chặt nhị phân trên dãy số:

1) Xét bài toán: DÃY CON TĂNG DÀI NHẤT

Cho một dãy gồm N số nguyên ($1 \le N \le 10^6$). Hãy tìm dãy con tăng dài nhất trong dãy đó. In ra số lượng phần tử của dãy con. Các số trong phạm vi longint.

Dữ liệu: Tệp văn bản LIS.INP

+ Dòng đầu tiên gồm số nguyên N.

+ Dòng thứ hai gồm N số mô tả dãy.

Kết quả: Ghi ra tệp văn bản LIS.OUT

Gồm một số nguyên duy nhất là đáp số của bài toán

Ví dụ

INPUT	OUTPUT
5 2 1 4 3 5	3

2) Giải quyết bài toán

Đây là một bài toán cơ bản và thường được giải bằng thuật toán quy hoạch động như sau:

+ Dùng mảng A gồm N phần tử lưu dãy số đã cho.

+ Thêm 2 phần tử $a_0 = -\infty$; $a_{n+1} = +\infty$

+ Dãy con đơn điệu tăng dài nhất chắc chắn bắt đầu ở a_0 và kết thúc ở a_{n+1} Dãy con tăng kết thúc tại a_i được thành lập bằng cách lấy a_i ghép vào sau một dãy con tăng kết thúc tại a_j nào đó đứng trước a_i

+ Gọi L[i] là độ dài của dãy con tăng dài nhất kết thúc tại a[i]

+ Cơ sở quy hoạch động: L[0] = 1;

+ Công thức quy hoạch động L[i] = Max{L[j]} với $0 \le j < i$ và a[j] < a[i]

```
Procedure   Optimize;
Var   i, j, jmax: longint;
Begin
  a[0]:= low(longint); a[n + 1]:= high(longint);
  L[0] := 1;
  for i := 1 to n+1 do
   begin
     jmax := 0;
     for j := 1 to i-1 do
      if (a[j] < a[i]) and (L[j] > L[jmax]) then jmax:= j;
     L[i] := L[jmax] + 1;
   end;
   Writeln('Do dai day con tang la : ', L[n+1] - 2);
end;
```

Kết quả bài toán L[n+1]-2.

Độ phức tạp của thuật toán là $O(N^2)$.

Theo đề bài thì $N \leq 10^6$ nên thuật toán trên chưa đáp ứng được yêu cầu. Ta cần giảm độ phức tạp thuật toán trên.

Xét *i* phần tử đầu tiên (a_0, a_1, ...,a_{i-1}). Với mỗi độ dài k, tìm cách lưu trữ thông tin về dãy con tăng độ dài k, gọi CS[k] lưu chỉ số *x* của phần tử a_x thỏa mãn:

+ Dãy con tăng dài nhất kết thúc ở a_x có độ dài k

+ Phần tử kết thúc a_x nhỏ nhất có thể (khả năng nối thêm cao nhất).

+ Một cách dễ hiểu thì CS[k] là chỉ số của số hạng nhỏ nhất trong các số hạng cuối cùng của các dãy con tăng có độ dài k. Đương nhiên CS[1] < CS[2] <.. < CS[k].

Nhận xét: Nếu *m* là độ dài dãy con tăng dài nhất: $a_{CS}[1] < a_{CS}[2] < ... < a_{CS}[m]$

Nếu có thêm một phần tử a_i

Nối a_i vào một dãy con tăng độ dài k để được dãy con tăng độ dài k+1 Sử dụng kỹ thuật chặt nhị phân và cập nhật lại CS[k+1].

Các bài toán áp dụng kỹ thuật chặt nhị phân trên dãy số thường có yêu cầu là tìm dãy con tăng, tìm dãy con giảm, tìm dãy con không tăng, tìm dãy con không giảm

Các yêu cầu thì khác nhau nhưng về cơ bản ta có cấu trúc đoạn chương trình chặt trên dãy số chỉ khác nhau ở các dấu so sánh.

❖ ***Trường hợp tìm dãy con tăng:***

```
Res:=1;
  CS[1]:=1;
For i:=2 to n do
Begin
        If A[i] < A[CS[1]] then  CS[1]:=i
        else
           if A[i] > A[CS[Res]] then Begin
                    Inc(Res);  CS[Res]:=i;
             End
              else
             Begin //Chặt nhị phân cập nhật lại mảng  CS
                 Dau:=1; Cuoi:=Res;
                 While Dau<Cuoi  do
                   Begin
                           Giua:=(Dau+Cuoi) div 2; J:=CS[Giua];
                           If A[i] > A[j]  then
                                  Dau:=Giua+1
                           else   Cuoi:=Giua;
                   End;
                   CS[Dau]:=i;
              End;
         End;
       // Res là biến lưu độ dài của dãy con tăng dài  nhất
```

❖ ***Trường hợp tìm dãy con giảm:***

```
Res:=1;
 CS[1]:=1;
For i:=2 to n do
Begin
       If A[i] > A[CS[1]] then  CS[1]:=i
       else
          if A[i] < A[CS[Res]] then
            Begin
                  Inc(Res); CS[Res]:=i;
            End
             else
            Begin //Chặt nhị phân cập nhật lại mảng  CS
                 Dau:=1; Cuoi:=Res;
                 While Dau<Cuoi do
                  Begin
                         Giua:=(Dau+Cuoi) div 2;
                         J:=CS[Giua];
                         If A[i] < A[j]  then
                               Dau:=Giua+1
                         else  Cuoi:=Giua;
                  End;
                   CS[Dau]:=i;
             End;
        End;
      // Res là biến lưu độ dài của dãy con giảm dài  nhất
```

Độ phức tạp thuật toán là ***O(NlogN)*** đáp ứng yêu cầu bài toán

Cấu trúc chặt nhị phân trên thường được áp dụng kết hợp với thuật toán sắp xếp Quick Sort.

*** Chương trình tham khảo:**

```
const fi= 'LIS.INP';
      fo= 'LIS.INP';
      maxd= 100001;
var   a: array[0..maxd] of longint;
      l: array[0..maxd] of integer;
      n: longint;
      d: longint;

procedure nhap;
var i:longint;
    f:text;
begin
     assign(f,fi); reset(f);
     readln(f,n);
     for i:=1 to n do read(f,a[i]);
     close(f);
end;

procedure xuli;
var i,k,dau,cuoi:longint;
begin
        l[1]:=1;d:=1;
        for i:=2 to n do
        begin
            if a[i]>a[l[d]] then
            begin
```

```
                    inc(d);
                    l[d]:=i;
                  end
                    else
                     begin
                         dau:=1;
                         cuoi:=d;
                         while dau<cuoi do
                         begin
                               k:=(dau+cuoi) div 2;
                               if a[l[k]]<a[i] then dau:=k+1
                                 else cuoi:=k;
                         end;
                         l[dau]:=i;

                     end;
          end;
end;

procedure xuat;
begin
     assign(f,fo); rewrite(f);
     write(d);
     close(f);
end;

BEGIN
     nhap;
     xuli;
     xuat;
END.
```

3) Bài tập áp dụng:

Bài 1: CÁC ĐOẠN NGUYÊN (PBCSEQ)

Mirko có một tập hợp các đoạn nguyên. Đầu tiên, anh ấy lấy ra 1 đoạn bất kì. Sau đó thực hiện lấy các đoạn khác, sao cho: đoạn lấy ra nằm trong đoạn vừa được lấy trước nó. Mirko tiếp tục cho đến khi không tìm được đoạn thoả mãn nữa.

Yêu cầu: Tìm số đoạn lớn nhất có thể lấy ra.

Dữ liệu: + Dòng đầu tiên chứa số nguyên N, là số đoạn nguyên trong tập hợp.

+ Dòng thứ i trong số N dòng sau, chứa 2 số nguyên A,B biểu thị cho đoạn i.

Kết quả: Một số duy nhất là kết quả của bài toán.

Giới hạn : $1 \le N \le 10^6$, $1 \le A < B \le 10^6$

Ví dụ

INPUT1	OUTPUT1	INPUT2	OUTPUT2
3 1 6 2 5 3 4	3	6 1 4 1 5 1 6 1 7 2 5 3 5	5

Thuật toán:

Bước 1: Sắp xếp giảm theo chỉ số đầu A, trong trường hợp A[i]=A[j] bằng ta sắp xếp tăng theo chỉ số cuối B

Bước 2: Chặt nhị phân tìm độ dài dãy con không giảm dài nhất của dãy B, đây chính là kết quả bài toán

*** Chương trình:**

```
program PBCSEQ;
const maxN =100000;
type TSegment = record
                  x,y :longInt
                end;
var n,res :LongInt;
  A :Array[1..maxN] of TSegment;
  F :Array[1..maxN] of longInt;

procedure Enter;
var i :LongInt;
begin
    read(n);
    for i:=1 to n do with (A[i]) do read(x,y);
end;

procedure QSort(l,r :longInt);
var i,j :LongInt;
    Key,Tmp :TSegment;
begin
    if (l>=r) then exit;
    i:=l; j:=r; Key:=A[(l+r) div 2];
    repeat
      while (A[i].x>Key.x)or((A[i].x=Key.x)and(A[i].y<Key.y)) do
inc(i);
      while (A[j].x<Key.x)or((A[j].x=Key.x)and A[j].y>Key.y)) do
dec(j);
      if (i<=j) then
        begin
          if (i<j) then
            begin
              Tmp:=A[i]; A[i]:=A[j]; A[j]:=Tmp;
            end;
          Inc(i); Dec(j);
        end;
    until (i>j);
    QSort(l,j); QSort(i,r);
end;

function BSearch(i :LongInt) :LongInt;
var left,right,mid :LongInt;
begin
    left:=1; right:=res;
    while (left<right) do
      begin
        mid:=(left+right) div 2;
        if (A[i].y>=A[F[mid]].y) then left:=mid+1 else right:=mid;
      end;
    exit(left);
end;

procedure Optimize;
var  i :LongInt;
begin
    F[1]:=1; res:=1;
    for i:=2 to n do
      if (A[i].y>=A[F[res]].y) then
        begin
```

```
            Inc(res); F[res]:=i;
          end
        else
          F[BSearch(i)]:=i;
end;

BEGIN
  assign(Input,'PBCSEQ.INP'); reset(Input);
  assign(Output,'PBCSEQ.OUT'); rewrite(Output);
  Enter;
  QSort(1,n);
  Optimize;
  write(res);
  close(Input); close(Output);
END.
```

Bài toán này ta còn một thuật toán khác là cài đặt cây chỉ số nhị phân BIT để tìm độ dài dãy con, độ phức tạp thuật toán vẫn là ***O(NlogN)***.

Bài 2: SEQUENCES (SPSEQ).

W là 1 dãy các số nguyên dương. Nó có các đặc điểm sau:

+ Độ dài của dãy là 1 số lẻ: **L = 2*N + 1**

+ N + 1 số nguyên đầu tiên của dãy tạo thành 1 dãy tăng

+ N + 1 số nguyên cuối của dãy tạo thành 1 dãy giảm

+ Không có 2 số nguyên nào cạnh nhau trong dãy có giá trị bằng nhau

Ví dụ: **1, 2, 3, 4, 5, 4, 3, 2, 1** là 1 dãy W. độ dài **9**. Tuy nhiên, dãy **1, 2, 3, 4, 5, 4, 3, 2, 2** không là 1 dãy W.

Yêu cầu: Trong các dãy con của dãy số cho trước, tìm dãy W có độ dài dài nhất.

Dữ liệu: Vào từ file văn bản SEQ.INP

Dòng 1: số nguyên dương N ($N <= 10^5$), độ dài dãy số.

Dòng 2: N số nguyên dương a_i ($a_i <= 10^9$).

Kết quả: Ghi ra file văn bản SEQ.OUT

Một số nguyên dương duy nhất là độ dài dãy W. dài nhất.

Ví dụ

INPUT	OUTPUT
19 1 2 3 2 1 2 3 4 3 2 1 5 4 1 2 3 2 2 1	9

INPUT	OUTPUT
10 1 2 3 4 5 4 3 2 1 10	9

Thuật toán:

Gọi F1[i] là độ dài dãy con tăng dài nhất kết thúc tại i, F2[i] là độ dài của dãy con giảm dài nhất bắt đầu tại i. Với thuật toán quy hoạch động ta có độ phức tạp thuật toán là $O(L^2)$ không đáp ứng được yêu cầu.

Để xây dựng F1 và F2 ta thực hiện chặt nhị phân hai lần: Lần 1: Tìm dãy con tăng bắt đầu từ đầu dãy để xây dựng F1.

Lần 2: Tìm dãy con tăng bắt đầu từ cuối dãy trở ngược về đầu dãy để xây dựng F2, về cơ bản hai đoạn chặt nhị phân này là giống nhau

*** Chương trình:**

```
program SPSEQ;
uses Math;
const  maxN =100000;
type Matrix =Array[1..maxN] of LongInt;
var n :LongInt;
    A,F1,F2,L1,L2 :Matrix;

rocedure Enter;
var i :longInt;
begin
    read(n);
    for i:=1 to n do Read(A[i]);
end;

function Search(i,j :LongInt; var F :Matrix) :LongInt;
var l,r,k :LongInt;
begin
    l:=1; r:=j;
    while (l<r) do
      begin
        k:=(l+r) div 2;
        if (A[F[k]]<A[i]) then l:=k+1 else r:=k;
      end;
    Search:=l;
end;

procedure Optimize;
var i,j,res :LongInt;
begin
    F1[1]:=1; L1[1]:=1; res:=1;
    for i:=2 to n do
      if (A[i]>A[F1[res]]) then
        begin
          inc(res); F1[res]:=i; L1[i]:=res;
        end
      else
        begin
          j:=Search(i,res,F1); F1[j]:=i; L1[i]:=j;
        end;
    F2[1]:=n; L2[n]:=1; res:=1;
    for i:=n-1 downto 1 do
      if (A[i]>A[F2[res]]) then
        begin
          Inc(res); F2[res]:=i; L2[i]:=res;
        end
      else
        begin
          j:=Search(i,res,F2); F2[j]:=i; L2[i]:=j;
```

```
            end;
    end;

    procedure Escape;
    var i,res :LongInt;
    begin
        res:=0;
        for i:=1 to n do res:=Max(res,2*Min(L1[i],L2[i])-1);
        Write(res);
    end;

    Begin
      assign(Input,'SPSEQ.INP'); reset(Input);
      assign(Output,'SPSEQ.OUT'); rewrite(Output);
      Enter;
      Optimize;
      Escape;
      close(Input); close(Output);
    End.
```

Bài 3*:* CHIA DÃY (QBDIVSEQ)

Dãy số M phần tử B được gọi là dãy con của dãy số A gồm N phần tử nếu tồn tại một mã chuyển C gồm M phần tử thoả mãn B[i]=A[C[i]] với mọi i = 1…M và $1 \le C[1] < C[2] < ... < C[m] \le N$.

Một cách chia dãy A thành các dãy con "được chấp nhận" nếu các dãy con này là các dãy không giảm và mỗi phần tử của dãy A thuộc đúng một dãy con.

Yêu cầu: ***Bạn hãy chia dãy con ban đầu thành ít dãy con nhất mà vẫn "được chấp nhận"***.

Dữ liệu: Vào từ file văn bản QBDIVSEQ.INP

+ Dòng đầu tiên ghi số N là số phần tử của dãy A. ($N \le 10^5$)

+ N dòng tiếp theo ghi N số tự nhiên là các phần tử của dãy A. ($A_i \le 10^9$)

Kết quả: Ghi ra file văn bản QBDIVSEQ.OUT một duy nhất là số lượng dãy con ít nhất thỏa mãn.

Ví dụ

INPUT	OUTPUT
4 1 5 4 6	2

Thuật toán:

Vì đề bài yêu cầu chia thành các dãy con sao cho các dãy con này là các "Dãy con không giảm" và tất nhiên các dãy con này liên tiếp nhau. Ta nhận thấy là số dãy con ít nhất chính bằng độ dài của "Dãy con giảm cực đại". Áp dụng tương tự bài LIS

Độ dài của dãy con giảm dài nhất = Số lượng ít nhất dãy con không giảm mà mỗi phần tử của dãy thuộc đúng 1 dãy con.

Như vậy bài toán đưa về tìm độ dài dãy con giảm dài nhất của dãy A. Cách cài đặt chặt nhị phân như tìm độ dài của dãy con tăng dài nhất chỉ đổi chiều các dấu so sánh.

*** Chương trình:**

```
program QBDIVSEQ;
const maxN =100000;
var n,res: longInt;
  A,F:Array[1..maxN] of longInt;

procedure Enter;
var i :longInt;
begin
    readln(n);
    for i:=1 to n do read(A[i]);
end;

function Search(i :longInt): longInt;
var l,r,k :LongInt;
begin
    l:=1; r:=res;
    while (l<r) do
      begin
        k:=(l+r) div 2;
        if (A[F[k]]>A[i]) then l:=k+1 else r:=k;
      end;
    Search:=l;
end;

procedure Optimize;
var i,j,jMax :longInt;
begin
    F[1]:=1; res:=1;
    for i:=2 to n do
      if (A[i]<A[F[res]]) then
        begin
          inc(res); F[res]:=i;
        end
      else F[Search(i)]:=i;
end;

Begin
  assign(Input,'QBDIVSEQ.INP'); reset(Input);
  assign(Output,'QBDIVSEQ.OUT'); rewrite(Output);
  Enter;
  Optimize;
  erite(res);
  close(Input); close(Output);
End.
```

Bài 4: Yugi-Oh (YUGI)

Các bạn đã đọc bộ truyện tranh Nhật Bản Yugi-oh chắc hẳn ai cũng cực kì yêu thích trò chơi bài Magic. Bộ bài và chiến thuật chơi quyết định đến sự thắng thua của đối thủ *(mà sự thắng thua thì còn liên quan đến cả tính mạng !!!)*. Vì thế tầm quan trọng của bộ bài là rất lớn. Một bộ bài tốt không chỉ bao gồm các quân bài mạnh mà còn phụ thuộc vào sự hỗ trợ tương tác giữa các quân bài. Bộ bài của Yugi là một bộ bài có sự bổ sung, hỗ trợ cho nhau rất tốt, điều này là 1 trong các nguyên nhân khiến Kaiba luôn là kẻ chiến bại.

Tình cờ Kaiba đã tìm được 1 quân bài ma thuật mà chức năng của nó là chia bộ bài hiện có của đối thủ ra làm K phần, mỗi phần có ít nhất 1 quân bài (điều này làm giảm sức mạnh của đối thủ). Kaiba quyết định áp dụng chiến thuật này với Yugi. Hiện tại Yugi có trong tay N quân bài, 2 quân bài i, j có sức mạnh tương tác a[i,j] (a[i,j] = a[j,i]). Kaiba muốn chia các quân bài thành K phần theo quy tắc sau:

+ Giả sử K phần là P1, P2, …, Pk thì độ giảm sức mạnh giữa 2 phần u,v là b[u,v]= min(a[i,j] với i thuộc Pu, j thuộc Pv).

+ Độ giảm sức mạnh của bộ bài là S = min(b(u,v) với $1 \le u, v \le K$).

Kaiba muốn chia K phần sao cho S lớn nhất

Dữ liệu:

+ Dòng đầu là 2 số N,K ($2 \le K \le N \le 200$)

+ N dòng tiếp theo mỗi dòng là N số a[i,j] (a[i,j] ≤ 32767; nếu i = j thì a[i,j] = 0)

Kết quả:

Gồm 1 dòng duy nhất là S lớn nhất

Ví dụ

INPUT	OUTPUT
4 3 0 1 2 3 1 0 2 3 2 2 0 3 3 3 3 0	2

Thuật toán:

Chặt nhị phân giá trị sức mạnh cần tìm. Với mỗi giá trị sức mạnh ta kiểm tra xem có thể chia bộ bài ra đúng k phần mà thỏa giá trị sức mạnh đó hay không. Khâu kiểm tra khá đơn giản.

Dùng kỹ thuật DFS (hay BFS), đưa những cặp quân bài có A[i, j] < "giá trị sức mạnh đang xét" vào cùng một nhóm. Tương ứng những quân bài còn lại ta xem mỗi quân bài là 1 nhóm. Hàm kiểm tra sẽ trả về giá trị true nếu số nhóm chia được >= k. Giải thích điều kiện đó là do những cặp quân bài có giá trị < "giá trị sức mạnh đang xét" đã được đưa vào những nhóm "xác định". Những quân còn lại xếp sao tùy ý. Thì dù cho số nhóm có > k đi chăng nữa thì chỉ cần gộp những quân bài tự do đó theo một cách nào đó ta vẫn có số nhóm là k. Luôn tồn tại cách gộp nhóm như vậy nếu số nhóm >k. Nếu số nhóm = k thì khỏi cần làm gì cho mệt. Số nhóm < k thì bó tay thôi.

Lưu ý : Nếu hàm kiểm tra trả về giá trị true thì lưu giá trị sức mạnh hiện thời lại và xét trên đoạn có giá trị sức mạnh lớn hơn xem có còn giá trị sức mạnh nào lớn hơn thỏa mãn hay không. Nếu hàm kiểm tra trả về giá trị false thì ta buộc lòng phải xét ở khoảng có giá trị sức mạnh nhỏ hơn.

*** Chương trình tham khảo:**

```
program YUGI;
const maxN =200;
      maxV =32767;
var   n,k :Byte;
      A :Array[1..maxN,1..maxN] of SmallInt;
      Free :Array[1..maxN] of Boolean;
```

```
      ans :SmallInt;

procedure Enter;
var i,j :Byte;
begin
    read(n,k);
    for i:=1 to n do
      for j:=1 to n do Read(A[i,j]);
end;

procedure DFSVisit(u :Byte; m :SmallInt);
var  v :Byte;
begin
    Free[u]:=false;
    for v:=1 to n do
      if (A[u,v]<m) and (Free[v]) then DFSVisit(v,m);
end;

function Check(m :SmallInt) :Boolean;
var d,u :Byte;
begin
    d:=0;
    fillChar(Free,SizeOf(Free),true);
    for u:=1 to n do
      if (Free[u]) then
        begin
          Inc(d); DFSVisit(u,m);
        end;
    Exit(d>=k);
end;

procedure BS;
var left,right,mid :SmallInt;
begin
    ans:=0; left:=0; right:=maxV;
    while (left<=right) do
      begin
        mid:=(left+right) div 2;
        if (Check(mid)) then
          begin
            ans:=mid; left:=mid+1;
          end
        else right:=mid-1;
      end;
end;

Begin
    assign(Input,'YUGI.INP'); reset(Input);
    assign(Output,'YUGI.INP'); rewrite(Output);
    Enter;
    BS;
    write(ans);
    close(Input); close(Output);
End.
```

Bài 5: MINCUT – Đề HSG Quốc gia 2015, ngày 2 - Cắt hình

Cho A là lưới ô vuông gồm m dòng và n cột. Các dòng của lưới được đánh số từ 1 đến m, từ trên xuống dưới. Các cột của lưới được đánh số từ 1 đến n, từ trái sang phải. Ô nằm trên giao của dòng i và cột j của lưới, được gọi là ô (i, j), chứa số nguyên không âm $a_{i,j}$ có giá trị không vượt quá 10^6.

Các lưới ô vuông như vậy luôn là đối tượng cho nhiều nghiên cứu thú vị. Vừa qua, trong giờ học ôn luyện cho kỳ thi học sinh giỏi Tin học, Hùng được cô giáo giao cho giải quyết bài toán trả lời truy vấn sau đây đối với bảng đã cho:

Cho một hình chữ nhật con có ô trái trên là ô (x,y) và ô phải dưới là ô (u,v), cần đưa ra chênh lệch nhỏ nhất trong số các chênh lệch giữa hai tổng các số trong hai hình chữ nhật thu được bằng việc cắt ngang hoặc cắt dọc hình chữ nhật đã cho dọc theo đường kẻ của lưới. Giả thiết (x,y) và (u,v) là hai ô khác nhau trên lưới.

Bạn hãy giúp Hùng giải quyết bài toán đặt ra.

Yêu cầu: Cho lưới A và k bộ x_q, y_q, u_q, v_q ($q = 1, 2, ..., k$) tương ứng với k truy vấn, hãy đưa ra các câu trả lời cho k truy vấn.

Dữ liệu vào:

+ Dòng đầu tiên chứa ba số nguyên m, n, k ($k \le m \times n$);

+ m dòng tiếp theo, dòng thứ i chứa n số nguyên không âm $a_{i1}, a_{i2}, ..., a_{in}$;

+ Dòng thứ q trong số k dòng tiếp theo chứa 4 số nguyên x_q, y_q, u_q, v_q ($q = 1, 2, ..., k$).

Dữ liệu ra:

Ghi ra file văn bản MINCUT.OUT gồm k dòng, mỗi dòng chứa một số là câu trả lời cho một truy vấn theo thứ tự xuất hiện trong file dữ liệu vào.

Ràng buộc:

+ Có 30% số test ứng với 30% số điểm của bài có $m, n \le 10$.

+ Có 30% số test khác ứng với 30% số điểm của bài có $m, n \le 100$.

+ Có 40% số test ứng với 40% số điểm còn lại của bài có $m, n \le 1000$.

Ví dụ:

INPUT	OUTPUT
3 2 3 1 1 1 1 1 1 1 1 1 1 1 3 3 1 1 3 2	3 0

Thuật toán:

Sử dụng công thức tính tổng trên ma trận như bài BONUS, với mỗi truy vấn ta tiến hành làm như sau :

+ Chặt nhị phân 4 lần, từ trái sang phải và ngược lại, từ trên xuống dưới và ngược lại.

+ Trong quá trình ấy lưu lại độ chênh lệch nhỏ nhất của mỗi lần chặt.

+ Lấy giá trị min của 4 lần chặt.

Lưu ý, do mỗi ô trên ma trận đều có giá trị dương nên chặt nhị phân mới hữu hiệu.

*** Chương trình tham khảo:**

```
program MINCUT;
uses Math;
const
  maxN =1000;
  maxM =1000;
  oo =10000000000000000;
var n,m,k,x1,y1,x2,y2 :LongInt;
    S :Array[0..maxN,0..maxM] of Int64;

procedure Enter;
var  i,j :LongInt;
begin
    read(n,m,k);
    fillChar(S,SizeOf(S),0);
    for i:=1 to n do
      for j:=1 to m do
        begin
          Read(S[i,j]);
          S[i,j]:=S[i,j]+S[i-1,j]+S[i,j-1]-S[i-1,j-1];
        end;
  end;

function Cal(u1,v1,u2,v2 :LongInt) :Int64;
begin
   exit(S[u2,v2]-S[u2,v1-1]-S[u1-1,v2]+S[u1-1,v1-1]);
end;

function BSearch1 :Int64;
varleft,right,mid :LongInt;
    v,res :Int64;
begin
    left:=x1; right:=x2; res:=oo;
    while (left<=right) do
      begin
        mid:=(left+right) div 2;
        v:=Cal(mid+1,y1,x2,y2)-Cal(x1,y1,mid,y2);
        if (v=0) then Exit(v);
        if (v>0) then
          begin
            res:=Min(res,v);
            left:=mid+1;
          end;
        if (v<0) then right:=mid-1;
      end;
    exit(res);
end;

function BSearch2 :Int64;
var left,right,mid :LongInt;
    v,res :Int64;
begin
    left:=x1; right:=x2; res:=oo;
    while (left<=right) do
```

```
            begin
              mid:=(left+right) div 2;
              v:=Cal(x1,y1,mid,y2)-Cal(mid+1,y1,x2,y2);
              if (v=0) then Exit(v);
              if (v>0) then
                begin
                  res:=Min(res,v);
                  right:=mid-1;
                end;;
              if (v<0) then left:=mid+1;
            end;
          exit(res);
      end;

      function BSearch3 :Int64;
      var  left,right,mid :LongInt;
           v,res :Int64;
      begin
          left:=y1; right:=y2; res:=oo;
          while (left<=right) do
            begin
              mid:=(left+right) div 2;
              v:=Cal(x1,mid+1,x2,y2)-Cal(x1,y1,x2,mid);
              if (v=0) then Exit(v);
              if (v>0) then
                begin
                  res:=Min(res,v);
                  left:=mid+1;
                end;;
              if (v<0) then right:=mid-1;
            end;
          exit(res);
      end;

      function BSearch4 :Int64;
      var  left,right,mid :LongInt;
           v,res :Int64;
      begin
          left:=y1; right:=y2; res:=oo;
          while (left<=right) do
            begin
              mid:=(left+right) div 2;
              v:=Cal(x1,y1,x2,mid)-Cal(x1,mid+1,x2,y2);
              if (v=0) then Exit(v);
              if (v>0) then
                begin
                  res:=Min(res,v);
                  right:=mid-1;
                end;;
              if (v<0) then left:=mid+1;
            end;
          Exit(res);
      end;

      procedure Solve;
      var i :LongInt;
      begin
          for i:=1 to k do
            begin
```

```
            Read(x1,y1,x2,y2);

WriteLn(Min(BSearch1,Min(BSearch2,Min(BSearch3,BSearch4))));
          end;
    end;

    Begin
        assign(Input,''); reset(Input);
        assign(Output,''); rewrite(Output);
        Enter;
        Solve;
        close(Input); close(Output);
    End.
```

Bài 6: Aladdin và cây đèn cầy (VMCANDLE)

Nến (còn gọi là đèn cầy) thường được thắp trong các buổi tiệc ngoài tời để tạo không khí huyền ảo, ấm cúng, lãng mạn. Hôm nay là sinh nhật Jasmine! Aladdin, Abu và thần đèn đã tổ chức một buổi tiệc thịnh soạn gồm cơ man nào là sơn hào hải vị. Trên bàn tiệc là một hàng N cây nến bằng đúng số tuổi của Jasmine. Điều đặc biệt là những cây nến này có phép (do của thần đèn).

Ban đầu N nến đều đang cháy.

Nếu thổi lần đầu thì cả N nến sẽ tắt.

Thổi lần hai thì các nến số chẵn cháy trở lại.

Thổi sang lần thứ 3 thì nến 3, 6, 9, 12, …. nếu đang cháy sẽ tắt, còn nếu đang tắt sẽ cháy.

Tương tự vậy với các lần 4, 5, 6, … , N.

Aladdin nhận thấy là sau khi thổi nến một số lần thì một số nến sẽ không bị tác động nữa, từ đó nghĩ ra một trò chơi. Aladdin đố Jasmine tìm ra cây nến còn sáng thứ K sau khi thổi hết cả N lần. Nếu Jasmine trả lời đúng sẽ nhận được một phần quà đặc biệt mà Aladdin bỏ ra cả mấy ngày để chuẩn bị.

Phải thổi hết N lần thì mất công quá L. Tuy nhiên Aladdin có một mẹo, không cần thổi mà cũng không cần biết có bao nhiêu nến tất cả vẫn tính được ngay số thứ tụ của cây nến đang cháy thứ K. Hãy giúp Jasmine giành được quà nào!

Yêu cầu : Cho K. Tìm số thứ tự của cây nến đang cháy thứ K sau N lần thổi nến

Dữ liệu vào: Một số nguyên dương duy nhất **K.**

Giới hạn

+ **K** <= 10^{18}

+ 33% số test có K <= 4000

Kết quả: Một số nguyên dương là số thứ tự của cây nến sáng thứ K.

Ví dụ:

INPUT	OUTPUT
1	2
6	8
21	26

Hướng dẫn:

Chặt nhị phân số lần thổi nến. Để ý nếu đến cây nến bất kì bị thổi thì hoặc là nó tắt

hoặc là nó cháy lại. Sau lần đó thì cây nến sẽ không thay đổi trạng thái nữa. Vì vậy nếu tìm được những vị trí mà từ đó trở về trước số nến đang cháy đúng bằng k thì 1 trong những vị trí đó có cây nến đang cháy thứ k mà ta cần tìm. Và đó chính là cây đầu tiên nếu ta duyệt tuần tự từ đầu đến cuối. Nghĩa là tìm vị trí đầu tiên nhất có số nến đang cháy bằng k. Những cây kia mặc dù từ đó trở về trước số nến cũng bằng k nhưng ngặc nỗi tại vị trí đó là 1 cây nến tắt.

Vấn đề còn lại là làm sao xác định số nến còn cháy. Để ý thấy cây nến có số thứ tự là 1 số chính phương thì sau khi thổi tại vị trí của nó, từ đó về sau cây nến đó luôn ở trạng thái tắt (vì số chính phương luôn có số ước lẻ, tính cả nó). Vì vậy tính từ vị trí i trở về trước số nến đang cháy là i – Trunc(Sqrt(i)). Giải thích cho công thức trên là do ứng với mỗi số từ 1 đến Trunc(Sqrt(i)) sẽ có đúng 1 số chính phương (lấy số đó bình phương).

Bài 7: ROBOTCAM

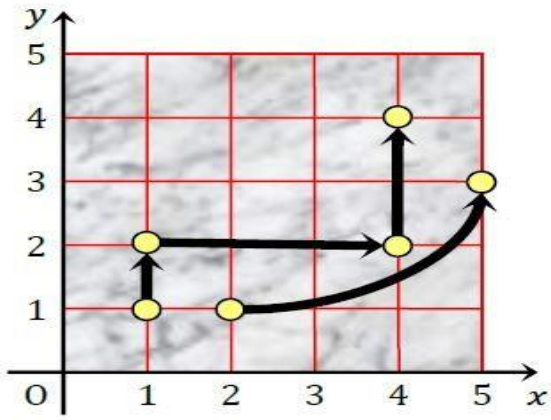
Cuộc thi RobotCam là một cuộc thi lớn về robot được tổ chức hàng năm ở hành tinh XYZ. Sân chơi có thể mô tả trên mặt phẳng với hệ toạ trực chuẩn. Luật chơi được mô tả như sau: Trên mặt phẳng đặt ***n*** phần quà tại các điểm hoàn toàn phân biệt. Các đội tham gia cuộc thi phải dùng các Robots của mình để thu nhặt tất cả các phần quà. Vấn đề trở nên khó khăn hơn đối với các đội chơi là các Robots tham gia thu nhặt quà không được di chuyển một cách tuỳ ý mà phải tuân thủ các điều kiện sau:

• Đường đi của mỗi robot phải bắt đầu và kết thúc tại các điểm trong số ***n*** điểm đã cho.

• Trong quá trình di chuyển, mỗi robot không được di chuyển tới điểm có hoành độ hay tung độ nhỏ hơn hoành độ hay tung độ điểm đang đứng.

• Hai đường đi của hai robots khác nhau không được có điểm chung Đường đi chỉ gồm đúng 1 điểm cũng được chấp nhận là hợp lệ

Dưới đây là hình mô tả vị trí của các điểm đánh dấu và một cách chơi hợp lệ:



Yêu cầu: ***Hãy xác định số lượng robot ít nhất cần sử dụng để thu nhặt tất cả các phần quà.***

Dữ liệu: Vào từ file văn bản ROBOTCAM.INP

+ Dòng đầu tiên chứa số nguyên dương $n \le 10^5$ là số lượng các phần quà.

+ N dòng tiếp theo, mỗi dòng chứa hoành độ và tung độ của một phần quà được ghi cách nhau một dấu cách. Các toạ độ là số nguyên có giá trị tuyệt đối không quá 10^9 .

Kết quả: Ghi ra file văn bản ROBOTCAM.OUT số lượng Robots ít nhất cần sử

dụng.

Ví dụ:

INPUT	OUTPUT
6 1 1 2 1 1 2 4 2 5 3 4 4	2

Thuật toán:

Khai báo mảng P gồm N phần tử mỗi phần tử gồm 2 trường x và y lưu tọa độ đặt các phần quà. Khai báo mảng H dùng để lưu tung độ các Robots được đặt.

Bước 1: Sắp xếp mảng P tăng theo tọa độ x, trong trường hợp bằng nhau ta sắp xếp tăng theo tọa độ y.

Bước 2: Xét lần lượt từng tọa độ (x,y) từ nhỏ tới lớn, với mỗi tọa độ chặt nhị phân xét xem trong các Robots đã đươc đặt có Robot nào có thể nhặt được quà tại tọa độ đang xét không, nếu không có ta thêm 1 Robot vào nhặt quà tọa độ (x,y) Cài đặt: bước 1 tương tự như bài 1 ***các đoạn nguyên***. Bước 2 ta có thể cài đặt như sau:

```
Procedure  Dem_so_Robots;
Var    Dau, Giua, Cuoi, i,  tungdo: Integer;
begin
 nRobots := 0;
 for i := 1 to n do
   begin
    tungdo := p[i].y;
    Dau:= 1;Cuoi:= nRobots; //h[1] > h[2] > ... > h[nRobots]
      while Dau < Cuoi do //h[dau - 1] > tungdo >=  h[cuoi+1]
        begin
         Giua:= (Dau+Cuoi) div 2;
         if h[Giua]>tungdo then Dau:= Giua+1 else Cuoi:= Giua;
        end;
      h[Dau] := tungdo;
      if Dau > nRobots then Inc(nRobots);
    end;
end;   //nRobots: là số lượng rôbốt cần sử dụng
```

Bài 7: SỐ LƯỢNG DÃY CON TĂNG

Cho dãy số $a_1,a_2,...,a_n$. Hãy đếm số lượng dãy con tăng dài nhất của dãy số trên.

Một dãy con độ dài k của dãy a được xác định bởi một bộ chỉ số $(u_1 \le u_2 \le u_3 \le ... \le u_k)$ $(1 \le u_i \le n)$. Hai dãy con $(u_1,u_2,u_3,...,u_k)$ và $(v_1,v_2,v_3,...,v_t)$ được gọi là khác nhau nếu $k \ne t$ hoặc tồn tại một vị trí i sao cho $u_i \ne v_i$.

Dữ liệu: Vào từ file văn bản DEM.INP

+ Dòng đầu tiên ghi số nguyên dương n $(n \le 10^5)$

+ Dòng tiếp theo ghi n số nguyên mô tả dãy a ($a_i \le 10^9$)

Kết quả : Ghi ra file văn bản DEM.OUT

Một dòng duy nhất ghi kết quả theo module 1000000007

Ví dụ

INPUT	OUTPUT
6 1 1 2 2 3 3	8

Gợi ý: Khai báo mảng A, mỗi phần tử là một bản ghi gồm 3 trường: L, ID, Val. Trong đó trường Val lưu giá trị của dãy số, ID lưu số thứ tự ban đầu, L lưu độ dài của dãy con tăng dài nhất kết thúc tại A[i].val.

Bài này phải thực hiện chặt nhị phân 3 lần.

Lần 1: Chặt nhị phân để xây dựng giá trị cho trường L.

```
Res:=1;  cs[1]:=1;
a[1].l:=1;
For i:=2 to n do
  Begin
    If a[i].val<a[cs[1]].val then
     Begin
      cs[1]:=i;a[i].l:=1;
     End
    Else
    If a[i].val>a[cs[res]].val then
     Begin
      Inc(Res);
      cs[Res]:=i;
      a[i].l:=Res;
     End
    Else
     Begin
      Dau:=1; Cuoi:=Res;
      While Dau<Cuoi do
        Begin
         Giua:=(Dau+Cuoi) div 2;
         If a[i].val>a[cs[Giua]].val then
               Dau:=Giua+1
         else  Cuoi:=Giua;
        End;
      cs[Dau]:=i;
      a[i].l:=Dau;
     End;
  End;
```

Sắp xếp dãy A theo trường L, nếu L bằng nhau thì sắp xếp theo vị trí trong mảng A ban đầu. Sau khi sắp xếp xong, với mỗi đoạn có L bằng nhau thì vị trí sẽ tăng dần và hiển nhiên giá trị val sẽ giảm dần.

Gọi F[i] là số lượng dãy con tăng có độ dài lớn nhất tính đến a[i]. Với mỗi a[i] ta cần tìm các a[j] có a[j].L=a[i].L-1 và thỏa mãn a[j].val<a[i].val và a[j].id < a[i].id (id là vị trí ban đầu). Do đó cần chặt 2 lần theo Val và ID để tìm được đoạn chứa các a[j] thỏa mãn để cộng vào F[i].

KẾT LUẬN

Kỹ thuật chặt nhị phân về cơ bản hoàn toàn dựa trên ý tưởng thuật toán tìm kiếm nhị phân khá quen thuộc.

Chuyên đề này đã trình bày hai dạng bài của kỹ thuật "chặt nhị phân" thông qua các ví dụ minh họa. Những bài tập được đưa vào thường áp dụng được ngay cấu trúc chặt nhị phân đã nêu. Nhưng trong thực tế có nhiều bài chúng ta phải vận dụng linh hoạt kỹ thuật trên, việc thay đổi giá trị của hai biến ***dau*** và ***cuoi*** sau mỗi lần kiểm tra sẽ quyết định tính đúng đắn của thuật toán, nếu không kiểm soát được hai biến này chương trình có thể dẫn tới lặp vô hạn, đây là điều mà các học sinh rất ngại khi cài đặt bằng kỹ thuật trên. Do đó luyện tập với nhiều bài tập vẫn là cách tốt nhất để rút ra cấu trúc chặt nhị phân cho riêng mình.

Hy vọng chuyên đề sẽ giúp ích cho các em học sinh, bổ sung vào vốn kiến thức các em một kỹ thuật rất hữu dụng.

Mặc dù đã rất cố gắng nhưng chuyên đề chắc chắn vẫn còn nhiều thiếu sót. Rất mong nhận được sự góp ý của các bạn đồng nghiệp và các em học sinh để chuyên đề được hoàn thiện hơn.

Người viết: Nguyễn Chánh Tín – Giáo viên Trường THPT Chuyên Bạc Liêu.

Điện thoại: 0918.443.556.

Email: nctinbl@gmail.com

TÀI LIỆU THAM KHẢO

1. Sách giáo khoa Tin học 10, 11 – NXB Giáo dục.
2. Tài liệu tập huấn phát triển chuyên môn giáo viên trường THPT Chuyên môn Tin học năm 2011 – Vụ Giáo dục trung học, Bộ GD&ĐT.
3. Sách giáo khoa Chuyên Tin quyển 1 – NXB Giáo dục – Chủ biên: Hồ Sĩ Đàm
4. Thuật toán thông dụng (Dành cho bồi dưỡng HSG Tin học) –Trần Đỗ Hùng - NXB Khoa học kỹ thuật năm 2010.
5. Giải thuật và lập trình – Lê Minh Hoàng.
6. Một số tài liệu trên Internet.