

I. ĐẶT VẤN ĐỀ

➤ Từ năm 2006 đến nay cùng với việc đổi mới cách ra đề của các kỳ thi IOI thì các kỳ thi học sinh giỏi môn Tin học cũng đã có những cách tiếp cận mới. Những năm trước đây đề thi thường tập trung vào các thuật toán phức tạp, nhiều khi học sinh phải thuộc lòng thuật toán và áp dụng một cách máy móc. Hiện nay các đề thi luôn yêu cầu học sinh vận dụng thuật toán một cách linh hoạt, với mỗi bài toán học sinh không chỉ phải đưa ra được thuật toán đúng mà thuật toán đó còn phải “nhanh” đáp ứng yêu cầu về thời gian. Việc bồi dưỡng học sinh giỏi môn Tin học phần lớn vẫn chỉ tập trung ở mức độ tìm thuật toán đúng cho bài toán. Ví dụ: cho một dãy số nguyên gồm N phần tử, hãy sắp xếp để dãy số đã cho trở thành dãy không giảm? Trước tiên ta có thuật toán sắp xếp tráo đổi (thuật toán sắp xếp “nồi bột”) có độ phức tạp $O(N^2)$. Đây là thuật toán đúng nhưng chỉ khả thi 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 $O(N\log N)$.

➤ 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 $O(\log K)$ và đây cũng là lý do tác giả chọn đề tài này.

➤ Trong quá trình giảng dạy tôi thường chia kỹ thuật này thành hai dạng:

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

➤ Nội dung của đề tài sẽ thể hiện hai kỹ thuật thông qua ví dụ cụ thể, với mỗi kỹ thuật sẽ được trình bày theo cấu trúc:

- Xét bài toán điển hình
- Giải quyết bài toán
- Bài tập áp dụng

➤ Các đoạn code chương trình minh họa thể hiện trên phần mềm Free Pascal

II. NỘI DUNG

✓ CƠ SỞ LÝ LUẬN

➤ Bản chất của kỹ thuật “chặt nhị phân” là thuật toán tìm kiếm nhị phân. Thuật toán này học sinh đã được học trong chương trình lớp 10 THPT (trang 42 – SGK Tin học 10), chương trình Tin học lớp 11 lại một lần nữa được đưa vào giảng dạy điều này cho thấy tầm quan trọng của thuật toán tìm kiếm nhị phân.

➤ Trước khi trình bài kỹ thuật “chặt nhị phân” ta tìm hiểu bài toán:

Cho dãy A là dãy tăng gồm N số nguyên khác nhau $a_1, a_2 \dots, a_n$ 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 $O(N)$, còn có thuật toán tìm kiếm nhị phân có độ phức tạp $O(\log N)$ với ý tưởng như sau:

- 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 A_{giua} 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 $a_1, a_2 \dots, a_{giua-1}$
- Nếu $A_{giua} < k$ thì việc tìm kiếm tiếp theo chỉ xét trên dãy $a_{giua+1}, a_{giua+2} \dots, a_N$

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.

Trên đây là ý tưởng của thuật toán tìm kiếm nhị phân, đề tài sẽ dựa trên ý tưởng này để xây dựng kỹ thuật “**chặt nhị phân**”.

✓ *THỦC TRANG*

➤ Phần lớn giáo viên sau khi giảng dạy thuật toán tìm kiếm nhị phân thì gần như rất ít khi cho học sinh vận dụng thuật toán này vào việc giải các bài toán. Theo tôi nghĩ do đây là một thuật toán khó với những học sinh bình thường mà chủ yếu dành cho những học sinh khá, giỏi. Còn trong quá trình bồi dưỡng học sinh giỏi thì nguồn tài liệu về kỹ thuật này là rất hạn chế. Theo kinh nghiệm các năm giảng dạy học sinh giỏi, nhất là học sinh giỏi quốc gia thì tài liệu về kỹ thuật này chủ yếu tích lũy trong quá trình giải các bài tập, tham gia các kỳ thi Online...

✓ *GIAI QUYẾT VẤN ĐỀ*

1. Chặt nhị phân theo kết quả

a. Xét bài toán

Cho n đoạn dây điện ($1 \leq n \leq 10^5$). Đoạn dây thứ i có độ dài a_i ($0 < a_i \leq 10^9$). 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 a_i

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	200
802	
743	
547	
539	

b. 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 $n \leq 10^5$ và $a_i \leq 10^9$, 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 \Rightarrow 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 $O(n\log L)$ 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 chia đôi 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 chia đôi 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:

c. **Bài tập áp dụng**

Bài 1:XẾP TRÚNG

Cho N quả trứng được đưa vào dây chuyền theo thứ tự (quả trứng thứ i có thể tích là a_i). 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

- ➊ Dòng đầu: Ghi 2 số nguyên n, m ($0 < n, m \leq 10^9$)
- ➋ Các dòng tiếp theo: dãy a_i ($0 < a_i \leq 10^6$).

Kết quả

- ➌ Một số duy nhất là số k tìm được.

Ví dụ:

INPUT	OUTPUT	GIẢI THÍCH
5 3	12	Thùng 1: a_1, a_2
6		Thùng 2: a_3, a_4
5		Thùng 3: a_5
4		
8		
9		

Gợi ý:

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,Giuua: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 $O(N \log(\text{sum}))$

Bài 2: BẢO TỒN ĐỘNG VẬT HOANG DÃ (Nguồn bài: Lê Minh Hoàng)

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 ($2 \leq n \leq 200$; $A \neq B$)
- ✿ Dòng 2: Chứa n số tự nhiên t_1, t_2, \dots, t_n ($\forall i: 0 \leq t_i \leq 20000$)
- ✿ 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 2 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	<pre> graph LR 1((1)) --- 20((20)) 1 --- 4((4)) 2((2)) --- 1 2 --- 5((5)) 3((3)) --- 29((29)) 3 --- 6((6)) 4 --- 1 4 --- 30((30)) 4 --- 6 5 --- 2 5 --- 24((24)) 6 --- 3 6 --- 27((27)) 7((7)) --- 26((26)) 8((24)) --- 5 </pre>

Gợi ý:

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 lớn nhất, vậy **Res** nằm trong $[0..MaxT]$. Ứng với mỗi giá trị chẵn nhị phân ta dùng thuật toán tìm kiếm theo chiều rộng kiểm tra xem có đường đi từ A tới B không.

```

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);
        end;
  end;
end;
  
```

```

        Push(v);
    end;
until First > Last;
BFS := False;
end;

```

Đoạn chương trình chặt nhị phân tìm Res

```

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 $O(N^2 \log T)$ với $N \leq 200$ thì chi phí thuật toán vẫn đảm bảo. ($T = \max(t_i), i: 1..N$)

Bài 3: MOUNTAINWALKING (Nguồn bài: USACO 2003 US Open)

Cho một bản đồ kích thước $N \times N$ ($2 \leq N \leq 100$), mỗi ô mang giá trị là độ cao của ô đó ($0 \leq \text{độ cao} \leq 110$). 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, mỗi dòng 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	2
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	

Gợi ý:

Bài này tương tự như “Bảo tồn động vật hoan dã”, điểm khác biệt là độ chênh lệch bài này là giữa điểm có độ cao lớn nhất với điểm thấp nhất trên đường đi. Dùng thuật toán Loang để kiểm tra nhưng thêm 2 giá trị min, delta. với min: là giá trị nhỏ nhất trên đường đi, còn delta là độ lệch giữa min và giá trị lớn nhất trên đường đi. Xét lần lượt từng giá trị min, ứng với giá trị min đó chia nhị phân giá trị delta và Loang(min,delta).

Hàm kiểm tra đường đi

```
Function Loang(min,delta:longint):Boolean;
Var u,v,u1,v1,k:integer;
Begin
  init;
  {Kiem tra dinh (1,1) co thoa man khong}
  if(a[1,1]<min)or(a[1,1]-min> delta) then exit(false);
  while first<=last do
    begin
```

```

pop(u,v);
for k:=1 to 4 do
begin
  u1:=u+hx[k];
  v1:=v+hy[k];
  if (0<u1)and(u1<=n)and(v1>0)and(v1<=n)then
if (a[u1,v1]>=min)and(a[u1,v1]-min<= delta)and(free[u1,v1])then
begin
  if (v1=n)and(u1=n) then exit(true);
  push(u1,v1);
  free[u1,v1]:=false;
end;
end;
end;
Loang:=false;
End;

```

Gọi **Dmin** là độ cao thấp nhất, **Dmax** là độ cao lớn nhất, độ chênh lệch độ cao nằm trong đoạn $[0..D_{max}-D_{min}]$. Thủ tục chính áp dụng chặt nhị phân theo kết quả như sau:

```

Procedure Solve;
Var min,Dau,Cuoi,Giuua:longint;
Begin
  for min:=dmin to dmax do
  begin
    delta:=maxlongint;
    Dau:=-1; Cuoi:=(dmax-dmin)+1;
    while Dau<Cuoi do
    begin
      Giua:=(Dau+Cuoi) div 2;
      if Loang(min,Giuua) then
      begin
        Delta:=Giuua;
        Cuoi:=Giuua-1;
      end
      else Dau:=Giuua+1;
    end;
    if res>Delta then
      res:=Delta;
  end;
end;

```

Với cách cài đặt như trên thì độ phức tạp thuật toán là $O(N^2 M \log D)$ trong đó $N \leq 100$, $M = (dmax-dmin) \leq 110$, $D \leq 110$ là chênh lệch độ cao.

Bài 4:ICEFROG

(*Nguồn bài:CROATIAN OPEN 2009*)

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 ($0 < N, M \leq 500$) là kích thước của khu rừng.
- ✿ N dòng sau mỗi dòng gồm N 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	0	4 4	3
.....		+...	
.++.		
.+.+		
V+.J+		V..J	

Gợi ý:

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

Bài 5:BUS

(*Nguồn bài:* Adapted from Ukrainian OI 2000)

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

Ví dụ

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

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.

Gợi ý:

Lưu ý: 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 chia 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.

2. Chặt nhị phân trên dãy số

a. Xét bài toán

Cho một dãy gồm N số nguyên ($1 \leq N \leq 10^6$). 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

- ✿ Dòng đầu tiên gồm số nguyên N.
- ✿ Dòng thứ hai gồm N số mô tả dãy.

Kết quả

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

Ví dụ

INPUT	OUTPUT
5	3
2 1 4 3 5	

b. 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 $L[i] = \text{Max}\{L[j]\}$ với $0 \leq j < i$ và $a[j] < a[i]$

```

Procedure Optimize;
Vari, 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 $(a_0, a_1, \dots, a_{i-1})$. 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 $CS[1] < CS[2] < \dots < CS[k]$.
 - Nhận xét: Nếu m là độ dài dãy con tăng dài nhất: $a_{cs[1]} < a_{cs[2]} < \dots < a_{cs[m]}$
- ✿ 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 chia đôi và cập nhật lại $CS[k+1]$.
- ✿ Các bài toán áp dụng kỹ thuật chia đôi 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 $O(N \log N)$ đáp ứng yêu cầu bài toán

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

c. Bài tập áp dụng

Bài 1: CÁC ĐOẠN NGUYÊN(Nguồn bài:CROATIAN OPEN 2007)

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 thỏa 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

- ✿ $1 \leq N \leq 10^6$, $1 \leq A < B \leq 10^6$

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

Gợi ý:

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 được thể hiện như sau:

Bước 1: Sắp xếp

```
Program CAC_DOAN_NGUYEN;
Const maxn = 100000;
Type doan=record
    a,b:longint;
    end;
Var T : array[1..maxn] of doan;
    n : longint;
//--
Procedure Enter;
Var i : longint;
Begin
    readln(n);
    For i := 1 to n do readln(T[i].a, T[i].b);
End;
//--
Procedure Swap(Var x, y : doan);
Var t : doan;
Begin
    t := x;
    x := y;
    y := t;
End;
// Định nghĩa một phép bé hơn
Operator < (x, y : doan) c : boolean;
```

```

Begin
    c := (x.a < y.a) or ((x.a = y.a) and (x.b > y.b));
End;
// Sắp xếp theo chiều giảm của chỉ số đầu a
Procedure QuickSort(l, r : longint);
Var i, j : longint;
    x : doan;
Begin
    If l >= r then exit;
    i := l;
    j := r;
    x := T[l + random(r - l + 1)];
    Repeat
        While x < T[i] do inc(i);
        While T[j] < x do dec(j);
        If i <= j then
            Begin
                Swap(T[i], T[j]);
                inc(i);
                dec(j);
            End;
        Until i > j;
        QuickSort(l, j);
        QuickSort(i, r);
    End;

```

Bước 2: Chặt nhị phân tìm độ dài của dãy con không giảm dài nhất theo chỉ số B

```

Procedure Solve;
Var CS : array[1..maxn] of longint;
    res,dau,cuoi,giua, i : longint;
Begin
    res := 1;
    CS[res] := 1;
    For i := 2 to n do
        If T[i].b < T[CS[1]].b then
            CS[1] := i
        Else
            If T[i].b >= T[CS[res]].b then
                Begin
                    inc(res);
                    CS[res] := i
                End
            Else
                Begin

```

```

dau := 1;
cuoi := res;
While dau < cuoi do
Begin
    giua := (dau + cuoi) div 2;
    If T[i].b >= T[CS[giua]].b then
dau := giua + 1
        else cuoi := giua;
    End;
    CS[dau] := i;
End;
writeln(res);
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 $O(N \log N)$.

Bài 2:SEQUENCES

(*Nguồn bài:* UVA)

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

- ✿ Dòng 1: số nguyên dương N ($N \leq 10^5$), độ dài dãy số.
- ✿ Dòng 2: N số nguyên dương a_i ($a_i \leq 10^9$).

Kết quả

- ✿ 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

Gợi ý:

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

```

Procedure Solve;
Var CS : array[1..maxn] of longint;
    i, d, Dau, Cuoi, Giua : longint;
Begin
//Chặt nhị phân xây dựng mảng F1
    d := 1; CS[d] := 1; f1[1] := 1;
    For i := 2 to n do
        If a[i] < a[CS[1]] then
            Begin CS[1] := i; f1[i] := 1; End
        Else
            If a[i] > a[CS[d]] then
                Begin inc(d); CS[d] := i; f1[i] := d; End
            Else
                Begin
                    Dau := 1; Cuoi := d;
                    While Dau < Cuoi do
                        Begin
                            Giua := (Dau + Cuoi) shr 1;
                            If a[i] > a[CS[Giua]] then Dau := Giua + 1
                            else Cuoi := Giua;
                        End;
                    CS[Dau] := i; f1[i] := Dau;
                End;
End;

```

```

        End;

//Chặt nhị phân xây dựng mảng F2
d := 1; CS[d] := n; f2[n] := 1;
For i := n - 1 downto 1 do
    If a[i] < a[CS[1]] then
        Begin CS[1] := i; f2[i] := 1; End
    Else
        If a[i] > a[CS[d]] then
            Begin inc(d); CS[d] := i; f2[i] := d; End
        Else
            Begin
                Dau := 1; Cuoi := d;
                While Dau < Cuoi do
                    Begin
                        Giua := (Dau + Cuoi) shr 1;
                        If a[i] > a[CS[Giua]] then Dau := Giua + 1
                        else Cuoi := Giua;
                    End;
                    CS[Dau] := i; f2[i] := Dau;
                End;
            End;
        End;
    End;
End;

```

Sau khi xây dựng xong 2 mảng F1, F2 độ dài của dãy W. dài nhất được tính như sau:

```

For i := 1 to n do
    Begin
        If F1[i] < F2[i] then t := F1[i] else t := F2[i];
        If Res < t then Res := t;
    End;
    writeln((res - 1) shl 1 + 1);

```

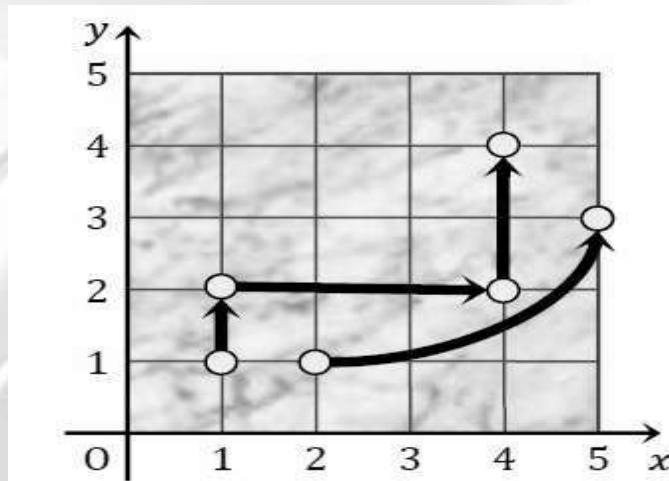
Bài 3:ROBOTCAM(*Nguồn bài*: Lê Minh Hoàng 2011)

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 $n \leq 10^5$ 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	2
1 1	
2 1	
1 2	
4 2	
5 3	
4 4	

Gợi ý:

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 được 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;
VarDau, 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+1else Cuoi:= Giua;
        end;
    h[Dau] := tungdo;
    if Dau> nRobots then Inc(nRobots);
    end;
end;//nRobots: là số lượng robot cần sử dụng

```

Bài 4: CHIA DÃY(*Nguồn bài: IOICAM 2006*)

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 $I = 1 \dots M$ và $1 \leq C[1] < C[2] < \dots < C[m] \leq N$.

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

- ✿ Dòng đầu tiên ghi số N là số phần tử của dãy A. ($N \leq 10^5$)
- ✿ N dòng tiếp theo ghi N số tự nhiên là các phần tử của dãy A. ($A_i \leq 10^9$)

Kết quả

- ✿ Ghi một duy nhất là số lượng dãy con ít nhất thỏa mãn.

Ví dụ

INPUT	OUTPUT
4	2
1	
5	
4	
6	

Gợi ý:

Từ cách giải của bài 3 **ROBOTCAM** đã chứng minh cho nhận xét:

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.

Bài 5:SỐ LUỢNG DÃY CON TĂNG

Cho dãy số a_1, a_2, \dots, 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 $(u_1 \leq u_2 \leq u_3 \leq \dots \leq u_k)$ ($1 \leq u_i \leq n$). Hai dãy con $(u_1, u_2, u_3, \dots, u_k)$ và $(v_1, v_2, v_3, \dots, v_t)$ được gọi là khác nhau nếu $k \neq t$ hoặc tồn tại một vị trí i sao cho $u_i \neq v_i$.

Dữ liệu

- ✿ Dòng đầu tiên ghi số nguyên dương n ($n \leq 10^5$)
- ✿ Dòng tiếp theo ghi n số nguyên mô tả dãy a ($a_i \leq 10^9$)

Kết quả

- ✿ 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 chia nhỏ thành 3 bước.

Lần 1: Chia nhỏ 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[1]].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].

✓ HIỆU QUẢ CỦA SÁNG KIẾN KINH NGHIỆM

Nội dung trong SKKN đã được áp dụng trong việc giảng dạy các lớp chuyên tin học, bồi dưỡng HSG Tỉnh, các đội dự tuyển và đội tuyển quốc gia môn tin học.

Khi giảng dạy nội dung của SKKN học sinh vận dụng tốt vào việc giải các bài tập đã cho vì các bài tập theo từng dạng đã được phân loại từ dễ tới khó, vận dụng bài tập trước sẽ làm được bài tập sau.

III. KẾT LUẬN

Trên đây tôi 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 tài liệu sẽ giúp ích cho các học sinh giỏi nhất là những học sinh trong đội tuyển quốc gia, bổ sung vào vốn kiến thức các em một kỹ thuật rất hữu dụng .

Kinh nghiệm giảng dạy chưa nhiều nên đề tài chắc còn có hạn chế, mong rằng sẽ nhận được sự góp ý của các đồng nghiệp.

Tài liệu tham khảo: Chủ yếu được tổng hợp trên các Website giải bài trực tuyến như:

1. www.vn.spoj.pl
2. <http://www.topcoder.com/>
3. www.codeforces.com
4. <http://hsin.hr/coci/>

MỤC LỤC

Thứ tự nội dung	Trang
I. ĐẶT VẤN ĐỀ	1
II. NỘI DUNG	2
✓ Cơ sở lý luận	2
✓ Thực trạng	3
✓ Giải quyết vấn đề	3
1. Chặt nhị phân theo kết quả	3
a. Xét bài toán	3
b. Giải quyết bài toán	4
c. Bài tập áp dụng	6
2. Chặt nhị phân trên dãy số	16
a. Xét bài toán	16
b. Giải quyết bài toán	16
c. Bài tập áp dụng	19
✓ Hiệu quả của SKKN	29
III. KẾT LUẬN	30