*Chuyên đề:*

# *VẬN DỤNG THUẬT TOÁN*

# TÌM KIẾM NHỊ PHÂN

## *GIẢI QUYẾT MỘT SỐ BÀI TOÁN*

## I. ĐẶT VẤN ĐỀ

Tìm kiếm là một việc thường xảy ra trong cuộc sống. Tìm kiếm luôn là thao tác nền móng cho rất nhiều tác vụ tính toán. Thuật toán tìm kiếm nhị phân là một trong những thuật toán tìm kiếm quan trọng nhất của tin học. Thuật toán này còn được gọi là thuật toán chặt nhị phân hay thuật toán chia đôi được áp dụng rất nhiều trong giải toán, nó làm giảm được nhiều thời gian tìm kiếm, giúp chương trình chạy nhanh hơn.

## IV. NỘI DUNG

### 1.Phương pháp tìm kiếm:

Thuật toán tìm kiếm nhị phân liên quan đến bài toán sau:

“ Cho mảng n phần tử đã được sắp tăng dần và một phần tử x. Tìm xem x có trong mảng hay không”

**Yêu cầu:** Thuật toán này chỉ có thể được dùng khi dãy số được sắp xếp đơn điệu theo thứ tự tăng hoặc giảm dần.

**Tư tưởng của thuật toán:** chọn phần tử ở vị trí giữa làm chốt, chia dãy thành 2 phần có kích thước nhỏ hơn. Sau đó so sánh phần tử cần tìm x với chốt, nếu x lớn hơn chốt tìm ở nửa sau của dãy, nếu x nhỏ hơn chốt tìm ở nửa trước của dãy (áp dụng với dãy tăng), quá trình trên tiếp tục cho tới khi tìm được x hoặc dãy chia không còn phần tử nào.

**Ví dụ:**

Cho dãy số: A={-6,1,3,5,8,10,14,16,19,21 }; x=5; dãy gồm 10 phần tử

Gọi phần tử chốt là k, ban đầu k=8

Bước 1: k=8, so sánh x với k, x<k ta tìm kiếm x ở nửa trước {6, 1,3,5,8}

Bước 2: k=3, so sánh x với k, x>k ta tìm kiếm x ở nửa sau {3,5,8}

Bước 3: k=5, so sánh x với k, x=k ta tìm được x kết thúc.

```
Procedure TKNP (x: Item, a1,a2,...,an: Item);
Begin
  d := 1;  {d là điểm đầu của đoạn tìm kiếm}
  c := n;  {c là điểm cuối của đoạn tìm kiếm}
  tim_thay:=false;
while (d <=c) and not tim_thay do
    begin
     g:= (d+c) div 2
       if  x = a[g]  then  tim_thay :=true
        else if  x<a[g] then c := g-1
```

```
        else d:=g+1
      end
   If tim_thay  then kq:=g
    else  kq := 0 {kq là vị trí của số hạng bằng x hoặc
0 nếu không tìm thấy x}
End;
```

## 2.Độ phức tạp :

Để tìm kiếm một phần tử trong mảng. Với cách thông thường, ta phải duyệt tất cả các số từ a[1] đến a[n], tức là mất độ phức tạp O(n).

Tuy nhiên với mảng đơn điệu, ta dùng chặt nhị phân để tìm kiếm thì :

$T_{tốt}$= O(1) ( x nằm ở vị trí giữa mảng)

$T_{xấu}$= O(logn)

Logarit là một hàm tăng chậm. Trong trường hợp ta còn băn khoăn về tính hiệu quả khi tìm kiếm nhị phân, hãy xét việc tìm kiếm một tên trong một cuốn danh bạ điện thoại có chứa một triệu tên. Tìm kiếm nhị phân cho phép ta tìm thấy bất kỳ tên nào chỉ sau nhiều nhất 21 lượt so sánh. Nếu ta có thể quản lý một danh sách có chứa tất cả mọi người trên thế giới được sắp xếp theo tên, ta có thể tìm thấy bất kỳ người nào trong vòng chưa đầy 35 bước.

## 3.Bài tập vận dụng

### A.Phần bài tập cơ bản

## Bài 1. ĐOÁN SỐ

Hai người chơi như sau: Người thứ nhất sẽ nghĩ ra một số nguyên dương trong khoảng từ 1 đến N (N được cho biết trước). Người thứ hai sẽ lần lượt đưa ra các số dự đoán. Với mỗi số dự đoán này, người thứ hai sẽ nhận được câu trả lời cho biết số mình vừa nêu ra lớn hơn, nhỏ hơn, hay bằng với số mà người thứ nhất đã nghĩ. Em hãy giúp người thứ hai chọn đúng số cần tìm với số lần đoán càng ít càng tốt.

**Thuật toán:**

Người thứ hai muốn chọn đúng số mà người thứ nhất nghĩ với số lần đoán ít nhất thì người thứ hai chắc chắn phải sử dụng đến thuật toán tìm kiếm nhị phân. Các bước sẽ lần lượt như sau:

*Bước 1.X:=1; y:=n; a:=0;*

*Bước 2.A:=(x+y) div 2*

*Bước 3.Lần đoán thứ i: a*

*Bước 4.Nếu a lớn hơn số cần tìm thì gán y:=a*

*Nếu a nhỏ hơn số cần tìm thì gán x:=a*

*Bước 5.Nếu số lần đoán vượt quá $log_2N$ thì chấm dứt*

*Ngược lại thì trở lại bước 2*

## Bài 2. BÀI TOÁN CỔ

*"Vừa gà vừa chó*

*Bó lại cho tròn*

*Ba mươi sáu con*

*Một trăm chân chẵn*

*Hỏi có bao nhiêu gà bao nhiêu chó?"*

Bài toán này các em đều đã rất quen thuộc từ hồi học cấp I. Phương pháp để giải bài toán này là phương pháp ***giả thiết tạm***.

Tất cả có 36 con vật. Chúng không thể đều là gà vì như vậy sẽ chỉ có 72 chân. Cũng không thể nào là chó cả vì như vậy sẽ có cả thảy 144 chân (Số chân của chúng là 100 chân).

Áp dụng tư tưởng chặt nhị phân cho bài toán này như sau:

- Sắp xếp số gà theo thứ tự tăng dần (theo chiều từ dưới lên)

- Chia đôi tổng số gà, nếu tại điểm giữa đó tổng số chân gà và chân gà và chân chó lớn hơn 100 thì lấy nửa trên, và ngược lại lấy nửa dưới.

***Thuật toán:***

```
1. d:=0;  c:=36;  x:=100;
2.Trong khi d<c thì
   2.1.g:= (d+c) div 2;
   2.2.Nếu x=2*g+4*(36-g) thì
        y:=c-g; Số gà là g; Số chó là y
   2.3.Nếu x>2*g+4*(36-g) thì c:=g
   2.4Nếu x<2*g+4*(36-g) thì d:=g
```

3.Quay lại bước 2

4.Kết thúc.

## Bài 3. SỐ 0 CUỐI CÙNG

Cho một dãy số khoảng 1000000 kí tự số toàn 0 và 1. Biết rằng các số 0 đứng trước các chữ số 1: 000....0011...11. Hãy cho biết vị trí của số 0 cuối cùng trong dãy.

***Thuật toán:***

Ta tiến hành tìm kiếm nhị phân trên xâu kí tự để tìm ra vị trí số 0 cuối cùng như sau:

- Tìm phần tử giữa xâu đang xét
- So sánh kí tự ở vị trí giữa xâu với kí tự 0.
- Nếu kí tự giữa xâu là kí tự 0 thì ta tìm ở nửa sau của xâu, nếu không phải kí tự 0 (mà là 1) thì ta tìm ở nửa trước của xâu.

*1.d:=1; c:=length(s);*

*2. trong khi d<c thì*

*2.1 g:=(d+c+1) div 2;*

*2.2 Nếu s[g]='0' thì d:=g*

*2.3 Nếu s[g]='1' thì c:=g-1;*

*3.Quay lại bước 3*

*4.Kết thúc*

Vậy vị trí của số 0 cuối cùng trong dãy là *d*

## Bài 4. TÌM N

Tìm số nguyên dương n thỏa mãn hệ thức : $C^1_{2n}+ C^3_{2n} +….+C^{2n-1}_{2n} =2048$

($C^k_n$ là tổ hợp chập k của n phần tử)

Ta biết rằng $C^1_{2n}+ C^3_{2n} +….+C^{2n-1}_{2n} =2048$ thì n≤ 8 vì nếu n >8 thì

$C^5_{2.8} =C^5_{16} = (12.13.14.15.16)/(1.2.3.4.5)= 13.7.3.16=4368 > 2048$

***Thuật toán:***

1. *d=0;c= 8; x=2048;*
2. *trong khi d<c thì*
   - *2.1.g:=(d+c) div 2;*
   - *2.2. nếu x=Tong(g) ( tong(g)= $C^1_{2g}+ C^3_{2g} +….+C^{g-1}_{2g}$) thì*

     *Xuất ra số cần tìm là g*
   - *2.3.Nếu (x>tong(g)) thì l:=g*
   - *2.4.Nếu (x<tong(g)) thì r:=g*
3. *Quay lại bước 2*
4. *Kết thúc*

Chắc chắn là trước khi áp dụng thuật toán tìm kiếm nhị phân để tìm N ta phải xây dựng một hàm tính giai thừa của một số nguyên, một hàm tính tổ hợp theo công thức $C^k_n$=n!/(k!*(n-k)!)

## Bài 5. HÀNG CÂY

Trong khu vườn, người ta trồng một hàng cây chạy dài gồm có N cây, mỗi cây có độ cao là $a_1$, $a_2$,…$a_N$.

Người ta cần lấy M mét gỗ bằng cách đặt cưa máy sao cho lưỡi cưa ở độ cao H (mét) để cưa tất cả các cây có độ cao lớn hơn H (dĩ nhiên những cây có độ cao không lớn hơn H thì không bị cưa).

Ví dụ: Nếu hàng cây có các cây với độ cao tương ứng là 20; 15; 10 và 18 mét, cần lấy 7 mét gỗ. Lưỡi cưa đặt tại độ cao hợp lí là 15 mét thì độ cao của các cây còn lại sau khi bị cưa tương ứng là 15; 15; 10 và 15 mét. Tổng số mét gỗ lấy được là 8 mét (dư 1 mét).

***Yêu cầu:*** Hãy tìm vị trí đặt lưỡi cưa hợp lí (số nguyên H lớn nhất) sao cho lấy được M mét gỗ và số mét gỗ dư ra là ít nhất.

***Dữ liệu***:

- Dòng thứ nhất chứa 2 số nguyên dương N và M ($1 \le N \le 10^6$; $1 \le M \le 2x10^9$) cách nhau một dấu cách.
- Dòng thứ hai chứa N số nguyên dương $a_i$ là độ cao của mỗi cây trong hàng ($1 \le a_i \le 10^9$; i=1…N), mỗi số cách nhau ít nhất một dấu cách.

***Kết quả***: Đưa ra màn hình một số nguyên cho biết giá trị cần tìm.

***Ví dụ:***

| **WOOD.INP** | **WOOD.OUT** |
|---|---|
| 4 7 | 15 |

| | |
|---|---|
| 20 15 10 18 | |

***Thuật toán :***

+ Đầu=0, cuối= chiều cao cây cao nhất.

+ Kiểm tra số mét gỗ S lấy được khi chặt ở độ cao h=(đầu+cuối) div 2:

- Nếu S=M: Thì in ra h và dừng chương trình.
- Nếu S<M thì đầu =h
- Nếu S>M thì cuối =h
- Tiếp tục kiểm tra h cho đến khi chênh lệch của M,S lặp lại.

```
writeln('Chieu cao cua cac cay la:');
  for i:=1 to n do
   begin
     write('cay ',i,'=');readln(a[i]);
     if a[i]>c then c:=a[i];
   end;
  d:=0;kt:=true;
  while kt and (d<=c) do
    begin
      s:=0;
      g:=(d+c) div 2;
      for i:=1 to n do
```

```
            begin
              t:=a[i]-g;
              if t>=0 then s:=s+t;
            end;
          t:=s-m;
          if (t=0 )or (cl=t)then kt:=false
           else begin
                  cl:=t;
                  if t<0 then c:=g else d:=g;
                end;
        end;
      if t>=0 then write('h=',g) else write('ko tim
    dc');
```

## B. PHẦN NÂNG CAO

Trong thực tế, ta thường gặp dạng bài toán tìm thời điểm kết thúc sớm nhất (hay muộn nhất) của một công việc, tìm chi phí bé nhất (hay lớn nhất),… với các yêu cầu ràng buộc trong đề bài. Khi đó ta nghĩ đến một thuật toán rất hiệu quả - thuật toán tìm kiếm nhị phân.

Sau đây là một số bài tập trong các đề thi học sinh giỏi

## Bài 6. CHỈNH DÃY SỐ

Cho một dãy N (N≤2x105) số nguyên dương không quá 109. Có thể giữ nguyên hoặc bỏ không quá một đoạn liên tiếp các

số trong dãy. Hãy thực hiện một trong hai cách trên sao cho dãy số thu được có độ dài đoạn liên tiếp tăng dần là lớn nhất.

Ví dụ : với dãy 1 2 3 1 4 5 sẽ chỉ có cách bỏ số 1 thứ 2 trong dãy đi để thu được dãy mới có độ dài đoạn con liên tiếp tăng dần là 5

Với dãy 1 3 4 6 ta không cần bỏ số nào đi

Dữ liệu vào từ file DEFENSE.INP

- Dòng 1 : số nguyên T≤25 là số bộ test
- T nhóm dòng sau: mỗi nhóm gồm 2 dòng. Dòng đầu ghi số nguyên N, dòng sau ghi N số nguyên là các số trong dãy theo thứ tự từ trái qua phải

Kết quả ghi ra flie DEFENSE.OUT chứa độ dài đoạn con liên tiếp tăng dần lớn nhất có thể thu được.

Ví dụ :

| DEFENSE.INP | DEFENSE.OUT |
|---|---|
| 2<br>9<br>5 3 4 9 2 8 6 7 1<br>7<br>1 2 3 10 4 5 6 | 4<br>6 |

**Thuật toán :**

Gọi L[i] là độ dài dãy dài nhất các số liên tiếp tăng dần kết thúc tại A[i]

R[i] là độ dài dãy dài nhất các số liên tiếp tăng dần bắt đầu tại A[i]

R và L tính được nhờ thuật toán quy hoạch động. Ta phải tìm giá trị cực đại của tổng L[i]+R[j] sao cho i ≤ j và A[i]<A[j].

Xét hàm Find(L,R) để tìm giá trị Max của L[i]+R[j] với A[i]<A[j] và L≤ i <j≤ R.

Ta chia đoạn [L,R] thành hai đoạn con [L,m] và [m+1,R] với m=(L+R) div 2.

Xét trường hợp :

+ Nếu i và j cùng thuộc một trong hai đoạn con này, giá trị max của L[i]+R[j] có thể tính được nhờ thủ tục find(L,m) và find(m+1,R).

+ Nếu i thuộc đoạn con thứ nhất và j thuộc đoạn con còn lại, ta áp dụng Merge sort để sắp xếp lại các A trong đoạn từ [L,R] tăng dần sau mỗi lần gọi find(L,R).

Từ đó với mỗi i∈[L,m] ta có thể tìm kiếm nhị phân ra j∈[m+1,R] tương ứng sao cho A[j] nhỏ nhất lớn hơn A[i] và tìm ra được giá trị lớn nhất của tổng L[i]+R[i]

## Cài đặt :

```
Program  defense ;

Uses math ;

Const  nfi='defense.inp' ;

           nfo='defense.out' ;
```

```
            maxn=100010 ;
var    a,r,l,c,f,d : array[0..maxn] of longint ;
n,res,ntest :longint ;
fi,fo :text ;

procedure enter ;
var i : longint ;
  begin
       read(fi,n)
       for i :=1 to n do read(fi,a[i]) ;
  end ;
procedure find(x,y:longint) ;
var i,u,v,m :longint ;
  begin
       if (x=y) then exit ;
        m:=(x+y) div 2 ;
        find(x,m); find(m+1,y);
        //f[i]=max(r[c[j]]) voi j>=i
       f[y]:=r[c[y]] ;
       for i:=y- 1 downto m+1 do
             f[i]:= max(r[c[i]], f[i+1]) ;
```

```
        // voi moi i ta se tim v nho nhat sao cho a[c[v]]>a[c[i]]
         v:= m+1 ;
         for i:=x to m do
            Begin
                While (v<y) and a[c[v]]<=a[c[i]] do inc (v) ;
                 If a[c[v]]> a[c[i]] then res:=max(res, l[c[i]]+f[v]) ;
             End ;
          //tron 2 doan da sap xep
          u:=x ; v:=m+1 ;
          for i :=x to y do
          if (v>y) or ((v<=y) and (u<=m) and (a[c[u]])< a[c[v]])) then
              begin
              d[i] :=c[u] ; inc(u) ; end
            else  begin
               d[i] :=c[v] ; inc(v) ; end ;
           for i := x to y do  c[i] :=d[i] ;
         end ;

        procedure solve ;
```

```
  var i:longint ;
    begin
//tinh l,r
L[1]:=1;
For i :=2 to n do
 begin
   L[i] :=1
          If (a[i]>a[i-1]) then l[i] :=l[i-1]+1 ;
         End ;
       R[n] :=1 ;
       For i :=n-1 downto 1 do
          begin
  R[i] :=1 ;
  If (a[i]<a[i+1]) then r[i]:=r[i+1]+1 ;
     End ;
       Res :=1 ;
       For i :=1 to n do c[i]=i
        //sap xep c theo chieu tang dan cua a[c[i]]
        //ket hop tinh gia tri cuc dai cua res
        Find(1,n) ;
        End ;
```

```
Begin
Assign(fi,nfi) ; reset(fi) ;
Assign(fo,nfo) ; rewrite(fo);
Readln(fi,ntest) ;
While ntest>0 do
 begin
    Dec(ntest);enter ;solve ;
    Writeln(fo, res);
   end ;
 Close(fo) ; close(fi);
 End.
```

## Bài 7. CĂN N

Biết rằng căn bậc N của một số S là một sốnguyên <$10^6$. Tìm căn bậc N của S

Dữ liệu vào: file **CANN.INP**
Dòng 1 là số N (N ≤ 100)
Dòng 2 là số S(0 ≤ S ≤ $10^{100}$ )
Kết quả ra: file **CANN.OUT**
Gồm 1 dòng duy nhất là căn bậc N của số S.

| CANN.INP | CANN.OUT |
|---|---|
| 4<br>81 | 3 |

***Thuật toán:***

- $C_{min}$ =0; $C_{max}$ = $10^6$.Kết quả sẽ nằm trong đoạn [$C_{min}$ ,$C_{max}$ ].
- Đặt $C_{tg}$ =($C_{min}$ +$C_{max}$ )div 2. Tính A= $C_{TG}$ $^N$. Để tính A ta dùng thuật toán nhân sốlớn.

Nếu A > S thì tìm kiếm trong đoạn [$C_{tg}$+1 ,$C_{max}$ ]

Nếu A < S thì tìm kiếm trong đoạn [ $C_{min}$ , $C_{tg}$ -1 ]

Nếu A=S thì căn bậc N của S chính là $C_{tg}$

- Tiếp tục tìm kiếm cho tới khi $C_{min}$ >$C_{max}$

***Cài đặt:***

```
type
    ds=array[1..1000] of 0..9;
var
 x,y,kqc:ds;
 a,b,s,skq,stg:string;
 ctg,n,m,j,k,p,spt,l,i,cmin,cmax:longint;
 f,g:text;
procedure htkq(x:ds;n:longint);
  var
   i:byte;
  begin
    for i:=1 to n do  write(x[i]);
  end;
procedure nhan(x,y:ds;vt:byte); {Nhan mot chu so o vi tri
vt cua so y voi so x}
 var kq:ds;
     nho,i,t,tg:byte;
 begin
   for k:=1 to m-vt  do kq[k]:=0;
   k:=m-vt;
   nho:=0;
   for i:=l downto 1 do
    begin
     k:=k+1;
     tg:=(nho+x[i]*y[vt]) div 10;
```

```
      kq[k]:=(nho + x[i]*y[vt]) mod 10;
      nho:=tg;
     end;
    if nho>0 then
     begin
      k:=k+1;
      kq[k]:=nho;
     end;
    {cong ket qua tinh duoc (kq) vao voi ket qua cuoi cung
(kqc)}
    nho:=0;
    for t:=1 to k do
      begin
       tg:=(nho+kq[t]+kqc[t]) div 10;
       kqc[t]:=(kq[t]+kqc[t]+nho) mod 10;
       nho:=tg;
      end;
    t:=k;
    while nho>0 do
     begin
      t:=t+1;
      tg:=(nho+kqc[t]) div 10;
      kqc[t]:=(nho+kqc[t])mod 10;
      nho:=tg;
     end;
    kqc[t]:=nho+kqc[t];
    spt:=t;
end;
procedure xlkq;
var i,j,tg:byte;
```

```
begin
     i:=1;
     j:=spt;
     while i<=j do
     begin
      tg:=kqc[i];kqc[i]:=kqc[j];kqc[j]:=tg;
      i:=i+1;
      j:=j-1;
     end;
end;
function ss(s1,s2:string):boolean;
var u,v,q:longint;
        kt:boolean;
begin
        u:=length(s1);
        v:=length(s2);
        if u>v then kt:=true
        else if u<v then kt:=false
        else
        begin
                q:=1;
                while (q<=u) and(s1[q]=s2[q]) do q:=q+1;
                if ord(s1[q])<ord(s2[q]) then kt:=false
                else kt:=true;
        end;
        ss:=kt;
end;

begin
 assign(f,'cann.inp');
```

```
reset(f);
readln(f,n);
readln(f,s);
assign(g,'cann.out');
rewrite(g);
cmin:=0;
cmax:=1000000;
skq:='';
while skq<>s do
begin
        ctg:=(cmin+cmax) div 2;
        skq:='';
        str(ctg,a);
        str(ctg,b);
        {tinh ctg mu n}
        for p:=2 to n do
        begin
                l:=length(a);
                m:=length(b);
                for i:=1 to l do x[i]:=ord(a[i])-48;
                for i:=1 to m do y[i]:=ord(b[i])-48;
                for k:=1 to l+m+1 do kqc[k]:=0;
                for j:=m downto 1 do  nhan(x,y,j);
                xlkq;
                a:='';
                for i:=1 to spt do
                begin
                        str(kqc[i],stg);
                        a:=a+stg;
               end;
```

```
        end;
        skq:=a;
        if ss(skq,s) then cmax:=ctg-1
        else cmin:=ctg+1;
 end;
 writeln(g,ctg);
 close(f);
 close(g);
end.
```

**Bài 8. Dãy con tăng dài nhất** (LIS (http://vn.spoj.pl/problems/LIS/))

Cho một dãy gồm N số nguyên (1 ≤ N ≤ 30000). 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.

**Input: File** Lis.Inp

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

**Output:** file Lis.Out Gồm một số nguyên duy nhất là đáp số của bài toán

**Ví dụ:**

| Lis.Inp | Lis.Out |
|---|---|
| 5<br>2 1 4 3 5 | **3** |

***Thuật toán:***

Gọi k là độ dài cực đại của dãy con tăng và ký hiệu H[1..k]

là dãy có ý nghĩa sau: H[i] là số hạng nhỏ nhất trong các số hạng cuối cùng của các dãy con tăng có độ dài i. Đương nhiên h[1] < h[2] <.. < h[k]. Mỗi khi xét thêm một giá trị mới trong dãy A thì các giá trị trong dãy H và giá trị k cũng tương ứng thay đổi.

```
Res:=1; H[1]:=1;
For i:=2 to n do
Begin
        If A[i] < a[h[1]] then h[1]:=i
        else if a[i] > a[h[res]] then
        Begin
                    Inc(Res); H[res]:=i;
        End
        else
        Begin
                     d:=1; c:=Res;
                   While d<>c do
                   begin
                            mid:=(d+c+1) div 2;
                            If A[i] > a[h[mid]] then d:=mid
else c:=mid-1;
                   End;
```

```
            Mid:=(d+c) div 2;
            If (A[h[mid]] < a[i]) and (a[i]<a[h[mid+1]]) then
            h[mid+1]:=i;
        End;
        End;
        Writeln(Res);
```

## Bài 9. YUGI(https://vn.spoj.pl/problems/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 ≤ u, v ≤ K).

Kaiba muốn chia K phần sao cho S lớn nhất

| yugi.Inp | yugi.Out |
|---|---|
| 4 3<br>0 1 2 3<br>1 0 2 3<br>2 2 0 3<br>3 3 3 0 | **2** |

**Input: file yugi.inp**

- Dòng đầu là 2 số N,K(2 ≤ K ≤ N ≤ 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)

**Output: file yugi.out** gồm 1 dòng duy nhất là S lớn nhất

***Thuật toán :***

Chặt nhị phân, với mỗi giá trị V(sức mạnh) đang xét, nhóm tất cả các quân bài (i,j) thành 1 nhóm nếu

+ i ≠ j.

+ a[i,j] <= V.

Để phân nhóm có thể sử dụng BFS.

Kiểm tra xem liệu số nhóm cần thiết để có giá trị V là bao nhiêu ? Nhỏ hơn K hay lớn hơn K , từ đó giảm dần khoảng cần xét .

***Cài đặt:***

```
Const   Finp='Yugi.inp';
        Fout='Yugi.out';
        Max=201;

VAR  Fi,Fo:Text;
     n,k,Maxv:Integer;
     a:array[1..Max,1..Max] Of Integer;
```

```
Procedure  Khoitao;
   Var i,j:integer;
  Begin
    Readln(Fi,n,k);
    Maxv:=0;
    For i:=1 to n do
      Begin
        For j:=1 to n do
          Begin Read(Fi,a[i,j]);
                If a[i,j]>Maxv then Maxv:=a[i,j];
          End;
        Readln(Fi);
      End;
  End;

Function  Ktra(Maxc:Integer):Boolean;{ktra xem co the chia
thanh k phan voi do giam suc manh va hay khong}
   Var i,j,dem,u,v,s,Nhom,d,c:integer;
       kt:boolean;
       dd:array[1..Max]Of  Boolean;
       q:array[1..Max] Of  Integer;
```

```
Begin
  Nhom:=0;dem:=0;
  Fillchar(dd,sizeof(dd),True);
  While (dem<n) do
    Begin
      Inc(nhom);
      s:=0;i:=1;kt:=False;
      While (i<=n-1)and(not(kt)) do
        Begin
          For j:=i+1 to n do
            If (a[i,j]<Maxc)and (dd[i])and (dd[j]) then
               Begin s:=i;kt:=True;break;End;
          Inc(i);
        End;
      If s<>0 then
        Begin
          Inc(dem);
          Fillchar(q,sizeof(q),0);
          d:=1;c:=1;q[1]:=s;dd[s]:=False;
          Repeat
```

```
            u:=q[d];inc(d);
            For v:=1 to n do
              If (u<>v)and(a[u,v]<Maxc )and(dd[v])then
                Begin
Inc(c);q[c]:=v;dd[v]:=False;Inc(dem);End;
          Until d>c;
        End
      Else Inc(dem);
    End;
  If Nhom>=k then Exit(True)  Else  Exit(False);
End;

Procedure  Chat;  {Chat nhi phan do giam suc manh}
 Var d,c,g,kq:Integer;
Begin
  d:=1; c:=maxv;
  While (d<c) do
    Begin
      g:=(d+c)div 2;
      If ktra(g) then
        Begin  kq:=g;
               d:=g+1;
```

```
          End
        Else  c:=g-1;
      End;
    If ktra(d) then kq:=d;
    Writeln(Fo,kq);
  End;

BEGIN
  Assign(Fi,Finp);Reset(Fi);
  Assign(Fo,Fout);Rewrite(Fo);
  Khoitao;
  Chat;
  Close(Fi);Close(Fo);
END.
```

## Bài 10. MTWALK (https://vn.spoj.pl/problems/MTWALK)

Cho một bản đồ kích thước NxN (2 <= N <= 100), mỗi ô mang giá trị là độ cao của ô đó (0 <= 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:** File MTWALK.INP

- Dòng 1: N
- Dòng 2..N+1: Mỗi dòng chứa N số nguyên, mỗi số cho biết cao độ của một ô.

| MTWALK.INP | MTWALK .OUT |
|---|---|
| 5<br>1 1 3 6 8<br>1 2 2 5 5<br>4 4 0 3 3<br>8 0 2 3 4<br>4 3 0 2 1 | **2** |

**Kết quả:** File MTWALK.OUT gồm một số nguyên là chênh lệch cao độ nhỏ nhất.

***Thuật toán :***

Ta có nhận xét các giá trị độ cao nằm trong khoảng [1,110], vì thế ta có thuật toán chặt nhị phân như sau :

Giả sử ta đang xét giá trị V ( Chênh lệch chiều cao)

- Vì chênh lệch tối ưu đang xét là V, suy ra nếu giá trị chiều cao nhỏ nhất trong đường đi tìm được là Min, thì giá trị chiều cao lớn nhất phải nhỏ hơn hoặc bằng Max = Min + V.

- Do chiều cao các ô <=110, nên việc thử lần lượt các giá trị dưới (giá trịMin) có thể chạy trong thời gian cho phép .

-Với mỗi giá trị V, ta tiến hành loang để tìm một đường đi thỏa mãn 2 điều kiện trên. Độ phức tạp M log M * N^2 ( M là giá trị)

**Cài đặt :**

```
uses windows;

const
```

```
    finp='mtwalk.inp';
    fout='mtwalk.out';

var
    fi,fo : text;
    n,Hmax,Hmin,d,c,V,min,max,Vtrc : integer;
    a : array[0..101,0..101] of longint;
    t : array[0..101,0..101] of boolean;
    ok : boolean;
    time : longint;

procedure init;
var i,j : integer;
begin
    Hmin:=200;     Hmax:=-1;
    assign(fi,finp);    reset(fi);
    readln(fi,n);
    for i:=1 to n do
    begin
        for j:=1 to n do
        begin
```

```
            read(fi,a[i,j]);
            if a[i,j] < Hmin then Hmin:=a[i,j]; {tim o co
do cao min}
            if a[i,j] > Hmax then Hmax:=a[i,j]; {tim o co
do cao max}
        end;
    end;
    close(fi);
{tao cac o ria = gia tri lon de trong qua trinh di khong di
toi nhung o do}
    for i:=0 to n+1 do
    begin
        a[i,0]   := 200;
        a[i,n+1] := 200;
        a[0,i]   := 200;
        a[n+1,i] := 200;
    end;

    assign(fo,fout);    rewrite(fo);
end;

   {tim duong di bat dau tu o a[i,j] theo 4 phia}
```

```
procedure truy(i,j : byte);
begin
    if ok then exit;
    if (i=n) and (j=n)
      then ok:=true;  {da tim duoc gia tri V nho nhat thoa man}
    t[i,j]:=false; {danh dau o (i,j) la da di}

{kiem tra cac o thoa man de di}
    if (a[i+1,j] <= max) and (min <= a[i+1,j]) and t[i+1,j] then truy(i+1,j);
    if (a[i-1,j] <= max) and (min <= a[i-1,j]) and t[i-1,j] then truy(i-1,j);
    if (a[i,j+1] <= max) and (min <= a[i,j+1]) and t[i,j+1] then truy(i,j+1);
    if (a[i,j-1] <= max) and (min <= a[i,j-1]) and t[i,j-1] then truy(i,j-1);
end;

function kt(W : byte) : boolean;{ ham kt voi do chenh lech W co duong di hay ko}
begin
    min:=Hmin;
```

```
    while (min <= a[1,1])   and (min <= a[n,n])
      and (a[1,1] <= min+W) and (a[n,n] <= min+W) do
    begin
        max:=min+W;
        fillchar(t, sizeof(t), true);
        ok:=false;
        truy(1,1);
        if ok then begin kt:=true; exit; end;
        min:=min+1;
    end;
    kt:=false;
end;


procedure tim;  { chat nhi phan tim kiem}
begin
    {V la do chenh lech dang xet
     Vtrc la gia tri o lan chat truoc}
    V:=(d+c) div 2;
    if V=Vtrc then exit
                  else Vtrc:=V;
    if not kt(V) then begin d:=V; tim; end
```

```
                    else begin c:=V; tim; end;
end;


begin
    time:=gettickcount;
    init;
    d:=Hmin;
    c:=Hmax;
    Vtrc:=-1;
    tim;
    if kt(V) then writeln(fo,V)
             else writeln(fo,V+1);
    close(fo);
    writeln((gettickcount-time)/1000:0:8);
end.
```

## Bài 11. TẢI TRỌNG TUYẾN ĐƯỜNG

Một hệ thống giao thông liên thông gồm N thành phố với tên 1..N (N <= 100). Có một số đoạn đường hai chiều giữa một số cặp thành phố và mỗi đoạn đường có một tải trọng tối đa mà chỉ có các xe với tải trọng không lớn hơn mới đi qua được. Cần đi từ thành phố U tới V. Hãy tìm một hành trình sao cho tải trọng tối đa cho phép trên hành trình đó là lớn nhất có thể được.

**Dữ liệu** : Trong file văn bản TAITRONG.INP gồm

- Dòng đầu là 3 số N, U, V

- Tiếp theo là một số dòng, mỗi dòng ghi ba số nguy n d ng X Y Z với ý nghĩa có đường đi giữa X và Y với tải trọng tối đa cho phép là Z (0 < Z <= 10000).

**Kết quả** : Ra file văn bản TAITRONG.OUT gồm

- Dòng thứ nhất ghi tải trọng H tối đa của xe có thể.

- Trong các dòng tiếp, mỗi dòng ghi tên một thành phố trong hành trình từ U kết thúc tại V.

**Ví dụ** :

| **TAITRONG.INP** | **TAITRONG.OUT** |
|---|---|
| 4 1 4<br>1 2 10<br>2 4 1<br>1 3 5<br>3 4 3 | 3<br>1<br>3<br>4 |

**Thuật toán**

Đặt $H_{min}$ = Min(a[i, j]), $H_{max}$ = Max(a[i, j]). Với mỗi giá trị h ($H_{min} \le h \le H_{max}$) ta xây dựng đồ thị G(h) thỏa mãn:

- N đỉnh tương ứng với N thành phố.

- 2 đỉnh i, j có cạnh nối nếu a[i, j] ≥ h.

Như vậy, nếu ta tìm thấy được một đường đi từ U tới V

thì ta nói rằng : "Mạng giao thông có tải trọng tối thiểu h". Bài toán trở thành "Tìm giá trị h lớn nhất để tồn tại đường đi từ U tới V".

Ta sẽ sử dụng tìm kiếm nhị phân dựa theo nhận xét: Nếu mạng có tải trọng tối thiểu k, và h là giá trị lớn nhất để "mạng giao thông có tải trọng tối thiểu h" thì $k \le h \le H_{max}$. Ngược lại, nếu không có lộ trình với tải trọng tối thiểu là k thì $H_{min} \le h < k$.

Với mỗi giá trị h, có thể duyệt DFS hoặc BFS để kiểm tra tồn tại đường đi từ U tới V hay không.

***Cài đặt:***

```
Const
        finp='taitrong.inp';
        fout='taitrong.out';
Var
        fi,fo:text;
        n,u,v,x,y:byte;
        z,i,j,hmin,hmax,dau,cuoi,giua:integer;
        a,b:array[1..100,1..100] of integer;
        h:array[1..4950] of integer;
        trc:array[1..100] of byte;

Procedure Sort( Left, Right: Integer);
```

```
Var
        i,j,k,tg: Integer;
Begin
        i:= Left;
        j:= Right;
        k:= h[(Left + Right) Div 2];
        Repeat
                While h[i] < k Do i:=i+1;;
                While k < h[j] Do j:=j-1;;
                If i <> j Then
                        Begin
                        tg:=h[i];
                        h[i]:=h[j];
                        h[j]:=tg;
                        End;
                i:=i+1;
                j:=j-1;
        Until i > j;
        If Left < j Then Sort(Left,j);
        If i < Right Then Sort(i,Right);
end;
```

```
{kiem tra neu mang co tai trong toi thieu k thi co duong
di tu u den v khong}
Function kt(k:integer):boolean;
Var
        dd:array[1..100] of boolean;
        i,j:byte;
Procedure dfs(t:byte);
Var i:byte;
Begin
        dd[t]:=true;
        for i:=1 to n do if (not dd[i]) and (b[t,i]=1)
then
                                                begin

trc[i]:=t;
                                                dfs(i);
                                                end;
End;
Begin
        for i:=1 to n do dd[i]:=false;
        for i:=1 to n do
```

```
                for j:=1 to n do
                        if a[i,j]>=k then b[i,j]:=1 else
b[i,j]:=0;
        dfs(u);
        if dd[v] then kt:=true else kt:=false;
End;

Procedure print;{in ket qua}
Var
        tam:array[1..100] of byte;
        i,j:byte;
Procedure nhap(t:byte);
Begin
        i:=i+1;
        tam[i]:=trc[t];
        if tam[i]<>u then
        nhap(trc[t]);
End;
Begin
        writeln(fo,h[giua]);
        tam[1]:=v;
        i:=1;
```

```
        nhap(v);
        for j:=i downto 1 do writeln(fo,tam[j]);
End;
{chat nhi phan mang h}
Procedure tim(d,c:integer);
Begin
        giua:=(d+c) div 2;
        if kt(h[giua]) then d:=giua else c:=giua-1;
        if c-d=1 then
                        begin
                        giua:=(d+c) div 2;
                        if kt(h[giua]) then print
                                else
                                begin
                                giua:=giua+1;
                                print;
                                end;
                        end
        else
        if d=c then print else if d<c then tim(d,c);
End;
```

```
Procedure doc;
Begin
        assign(fi,finp);reset(fi);
        assign(fo,fout);rewrite(fo);
        readln(fi,n,u,v);
        {xay dung do thi ban dau}
        for i:=1 to n do
                for j:=1 to n do a[i,j]:=0;
        i:=0;
        while not eof(fi) do
                begin
                readln(fi,x,y,z);
                a[x,y]:=z;
                a[y,x]:=z;
                i:=i+1;
                h[i]:=z;
                end;
        dau:=1;cuoi:=i;
End;
```

```
Begin
        doc;
        sort(1,cuoi);{sap xep mang h}
        tim(dau,cuoi);
        close(fi);close(fo);
    End.
```

## Bài 12. ĐIỀU ĐỘNG

Sau khi thực thi quy hoạch của Bộ Giao thông, sơ đồ giao thông của thành phố H gồm n tuyến đường ngang và n tuyến đường dọc cắt nhau tạo thành một lưới ô vuông với n x n nút giao thông. Các nút giao thông được gán tọa độ theo hàng từ 1 đến n, từ trên xuống dưới và theo cột từ 1 đến n, từ trái sang phải. Ban chỉ đạo an toàn giao thông quyết định điều n cảnh sát giao thông đến các nút giao thông làm nhiệm vụ. Ban đầu mỗi cảnh sát được phân công đứng trên một nút của một tuyến đường ngang khác nhau. Đến giờ cao điểm, xuất hiện ùn tắc tại các tuyến đường dọc không có cảnh sát giao thông. Để sớm giải quyết tình trạng này, Ban chỉ đạo an toàn giao thông quyết định điều động một số cảnh sát giao thông ở một số nút, từ nút hiện tại sang một nút khác cùng hàng ngang để đảm bảo mỗi tuyến đường dọc đều có mặt của cảnh sát giao thông.

**Yêu cầu**: Biết rằng cảnh sát ở hàng ngang thứ i cần $t_i$ đơn vị thời gian để di chuyển qua 1 cạnh của lưới ô vuông (i = 1, 2, ..., n), hãy giúp Ban chỉ đạo an toàn giao thông tìm cách điều động các cảnh sát thỏa mãn yêu cầu đặt ra sao cho việc điều động được hoàn thành tại thời điểm sớm

nhất. Giả thiết là các cảnh sát được điều động đồng thời thực hiện việc di chuyển đến vị trị mới tại thời điểm 0.

**Ràng buộc**: 50% số tests ứng với 50% số điểm của bài có n ≤ 100.

**Dữ liệu vào từ file MOVE.INP**

- Dòng thứ nhất chứa một số nguyên dương n (n ≤ 10000).
- Dòng thứ i trong số n dòng tiếp theo chứa hai số nguyên dương $c_i$, $t_i$ ($t_i$ ≤ 10000) tương ứng là tọa độ cột và thời gian để di chuyển qua 1 cạnh của lưới ô vuông của cảnh sát đứng trên tuyến đường ngang thứ i (i = 1, 2, ..., n).

Hai số trên cùng một dòng được ghi cách nhau ít nhất một dấu cách.

**Kết quả ghi ra file MOVE.OUT**

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

**Ví dụ:**

| MOVE.INP | MOVE.OUT |
|---|---|
| 5<br>5 10<br>3 10<br>3 20<br>2 9 | 10 |

| | |
|---|---|
| 2 15 | |

**Thuật toán:**

Với yêu cầu của bài toán, ta nhận thấy cần áp dụng thuật toán tìm kiếm nhị phân. Ta phải giải quyết một bài tập đơn giản hơn là kiểm tra xem trong thời gian t, các cảnh sát có thể di chuyển đến các vị trí thỏa mãn yêu cầu đề bài hay không.

Với khoảng thời gian t cho trước, mỗi cảnh sát i có thể di chuyển đến các cột trong khoảng $[l_i, r_i]$ nào đó. Ta có nhận xét rằng trong các cách chọn đúng, tồn tại một cách chọn mà ở đó cảnh sát đi đến cột 1 là cảnh sát có $r_i$ nhỏ nhất trong các cảnh sát có thể đến cột 1.

Thật vậy, xét một cách chọn đúng, cảnh sát đi đến cột 1 là j và có $r_j > r_i$ còn cảnh sát sẽ đi đến cột k, ta có $l_j = l_i = 1$; $k \leq r_i < r_j$. Do đó ta có thể đổi lại là cảnh sát i đến cột 1 còn cảnh sát j đến cột k. Việc chọn cảnh sát tiếp theo để đi đến các cột 2,3,…, n cũng tương tự như vậy.

Thuật toán sẽ là xét lần lượt các cột từ 1 đến n, khi đến cột i, trong các cảnh sát chưa được chọn và đến được cột i, cảnh sát có r tương ứng nhỏ nhất sẽ được chọn để đến cột i. Dùng heap để quản lý được các cảnh sát đến được cột hiện tại.

Thuật toán có độ phức tạp O(nlog(n)). Kết quả bài toán sẽ đạt giá trị max là $10^8$, do đó độ phức cho cả bài toán sẽ là O(nlog(n)log($10^8$))

**Cài đặt:**

```
Program move;
const    nfi='move.inp';
             nfo='move.out';
             maxn=10010;
var  c,t,l,r, heap:array[0..maxn] of longint;
        n, nheap:longint;
       fi,fo:text;
function min(x,y:longint):longint;
begin
if x<y then exit(x) else exit(y);
end;
function max(x,y:longint):longint;
begin
```

```
if x<y then exit(y) else exit(x);
end;
procedure enter;
var i: longint;
begin
     read(fi,n);
    for i:= 1 to n do read(fi,c[i],t[i]);
end;
function  pop:longint;
//lay phan tu dau cua heap ra
var u, pa,ch,key:longint;
begin
    u:=heap[1];dec(nheap);
    if (nheap=0) then exit(u);
    heap[1]:=heap[nheap+1];
    pa:=1;ch:=2;key:=heap[1];
    while(ch<=nheap) do begin
                if (ch+1<=nheap) and (r[heap[ch+1]])<
r[heap[ch]] )
                  then inc(ch);
                if r[key]< r[heap[ch]] then break;
```

```
                heap[pa]:=heap[ch];
                 pa:=ch;ch:=pa*2;
       end;
      heap[pa]:=key; exit(u);
end;
procedure push (u:longint);//them u vao heap
var ch,pa,key:longint;
begin
    inc(nheap);
    heap[nheap]:=u;
    ch:=nheap; pa:=ch div 2; key:=u;
    while(pa>0) and (r[heap[pa]]>r[key]) do
            begin  heap[ch]:=heap[pa];
                      ch:=pa;pa:=ch div 2;
           end;
          heap[ch]:=key;
  end;

procedure swap(var x, y:longint);
var tg: longint;
begin tg:=x;x:=y;y:=tg; end;
```

```
procedure qsort(a,b:longint);
var  i,j,k:longint;
begin
        if a>=b then exit;
        i:=a;j:=b;k:=l[a+random(b-a)];
        repeat
        while (l[i]<k) do inc(i);
        while (l[j]>k) do dec(j);
        if (i<=j) then begin
             swap(l[i],l[j]);swap(r[i],r[j]);
             inc(i);dec(j);
                   end;
      until i>j;
      qsort(a,j); qsort(i,b);
end;
procedure solve
var left,right,mid,d,u,ok,i,res:longint;
begin
left:=0; right:=100000000;
//tim kiem nhi phan
```

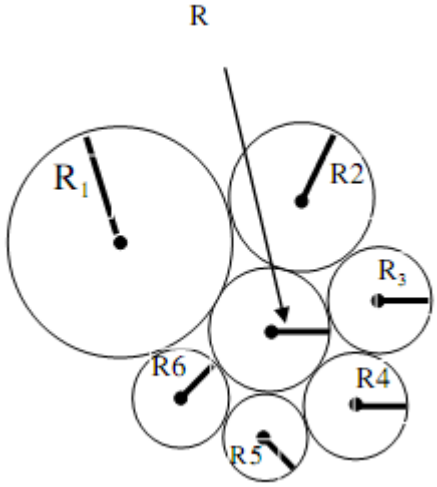
```
    while (left <=right) do begin
         mid:=(left+right) div 2;
        //tinh l[i], r[i] voi thoi gian mid
       for i:=1 to n do   begin
          l[i]:=max(1,c[i]-mid div t[i]);
          r[i]:=min(n,c[i]+mid div t[i]);
        end;
    //sap xep cac canh sat tang dan theo l[i];
      qsort(1,n); nheap:=0;
      d:=1;ok:=1;
      //lan luot tim canh sat den cot i
      for i:=1 to n do begin
      //day tat ca cac canh sat den duoc cot i vao heap
      while (d<=n) and (l[d]=i) do
            begin
          push(d); inc(d);end;
          if nheap=0 then begin
//neu khong co canh sat nao den duoc i ket luan la khong
thuc hien duoc
                    ok:=-1; break;end
                else
```

```
                    begin u:=pop;
                   if (r[u]<i) then begin
    //khong xep duoc cot cho canh sat u, ket luan la
    khong thuc hien duoc
                    ok:=-1; break;end;
                  end;
          end;
          if (ok=1) then begin
                  right:=mid-1;
                  res :=mid;end
          else left :=mid+1; end;
    writeln(fo,res);
    end;
    begin
          assign(fi,nfi); reset(fi);
          assign(fo,nfo); rewrite(fo);
          enter;solve;
          close(fi);close(fo);
    end.
```

## C. BÀI TẬP TỰ GIẢI

**Bài 1. Bộ sưu tập các đồng xu (http://vn.spoj.pl/problems/LEM1/)**

Cho N đồng xu có bán kính lần lượt là các số thực dương r1.. rN. Được đặt xung quanh một vòng tròn sao cho: Mỗi đồng xu tiếp xúc với 2 đồng xu đặt cạnh nó và tiếp xúc với vòng tròn. Biết được bán kính của từng đồng xu.




Yêu cầu: Tìm bán kính vòng tròn.

Input

- Dòng đầu ghi số nguyên dương N

- Dòng tiếp theo ghi N số ri ( 1 ≤ i ≤ N )

Output

- Gồm 1 dòng duy nhất ghi bán kính hình tròn ( độ chính xác đến 3 chữ số sau dấu phẩy )

Ví dụ

| Input: | Output: |
|---|---|
| 4<br><br>2 2 2 2 | 0.828 |

Giới hạn

1 ≤ N ≤ 10000

$1 \le ri \le 100000$

**Bài 2. Xâu con – COCI 2006-2007**

Steve chiến thắng trong một lần cá cược và John phải mời anh ấy đi xem phim.Trong khi chờ đợi Steve, John để ý một thông báo trên màn hình quảng cáo phía trên chỗ anh đang đứng. Steve đến muộn nên John đã để ý trên màn hình quảng cáo một thời gian và thấy rằng có một vài thông báo xuất hiện trên màn hình nhiều hơn một lần. Tự nhiên, anh ấy viết tất cả các thông báo ra giấy. John muốn biết chiều dài của xâu dài nhất xuất hiện ít nhất hai lần (xuất hiện trong hai vị trí khác nhau trên trang giấy)?

Yêu cầu: Hãy giúp John trả lời câu hỏi đó.

Dữ liệu vào cho trong tệp SUBSTR.INP

- Dòng 1 chứa số nguyên L ($1 \le L \le 200000$) là độ dài mà Jonh đã viết trên giấy.

- Dòng 2 chứa một xâu có độ dài L gồm các kí tự chữ thường trong bảng chữ cái tiếng anh.

Kết quả ghi ra tệp SUBSTR.OUT một số duy nhất là độ dài dài nhất của xâu xuất hiện ít nhất hai lần trong xâu đã cho. Nếu không tồn tại xâu con thỏa mãn đưa ra số 0.

| SUBSTR.INP | SUBSTR.OUT |
|---|---|
| 18 | 4 |

| | |
|---|---|
| trutrutiktiktappop | |

**Bài 3. Palindrome dài nhất – PALINY**

**(http://vn.spoj.com/problems/PALINY**

| PALINY.INP | PALINY.OUT |
|---|---|
| 5<br><br>Abacd | 3 |

Cho xâu S. Tìm xâu đối xứng dài nhất gồm các kí tự liên tiếp trong S

Dữ liệu vào: cho trong file PALINY.INP

Dòng 1: N (số ký tự của xâu S;

N<=50 000)

Dòng 2: Xâu ký tự độ dài N

Kết quả ghi ra file PALINY.OUT

1 dòng duy nhất gồm độ dài của xâu đối xứng dài nhất

## Bài 4. Đóng gói sản phẩm-Đề thi HSGQG2005

Ở đầu ra của một dây chuyền sản xuất trong nhà máy ZXY có một máy xếp tự động. Sau khi kết thúc việc gia công trên dây chuyền, các sản phẩm sẽ được xếp vào các hộp có cùng dung lượng *M*. Sản phẩm rời khỏi dây chuyền được xếp vào hộp đang mở (khi bắt đầu ca làm việc có một hộp rỗng được mở sẵn) nếu như dung lượng của hộp còn đủ để chứa sản

phẩm. Trong trường hợp ngược lại, máy sẽ tự động đóng nắp hộp hiện tại, cho xuất xưởng rồi mở một hộp rỗng mới để xếp sản phẩm vào. Trong một ca làm việc có $n$ sản phẩm đánh số từ 1 đến $n$ theo đúng thứ tự mà chúng rời khỏi dây chuyền. Sản phẩm thứ $i$ có trọng lượng là $a_i$, $i$ = 1, 2, …, $n$. Ban Giám đốc nhà máy qui định rằng sản phẩm xuất xưởng của mỗi ca làm việc phải được xếp vào trong không quá $k$ hộp.

**Yêu cầu:** Hãy giúp người quản đốc của ca làm việc xác định giá trị $M$ nhỏ nhất sao cho số hộp mà máy tự động cần sử dụng để xếp dãy $n$ sản phẩm

xuất xưởng của ca không vượt quá số $k$ cho trước.

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

- Dòng đầu tiên chứa hai số nguyên $n$ và $k$, ($1 \le k \le n \le 15000$);

- Dòng thứ $i$ trong $n$ dòng tiếp theo chứa số nguyên dương $a_i$ ($a_i \le 30000$), $i$ =1, 2, …, $n$.

Các số trên một dòng cách nhau ít nhất một dấu cách.

**Kết quả:** Ghi ra file ZXY.OUT một số nguyên duy nhất là dung lượng của hộp.

**Ví dụ:**

| **ZXY.INP** | **ZXY.OUT** |
|---|---|
| 9 4 | 5 |

| | |
|---|---|
| 1<br>1<br>1<br>3<br>2<br>2<br>1<br>3<br>1 | |

## Bài 5. Mã số thuế - Đề thi HSGQG 2010

Để thực hiện luật Thuế thu nhập cá nhân, Tổng cục thuế phải cấp cho mỗi người có thu nhập một mã số thuế sao cho không có hai người nào có mã số thuế trùng nhau. Tổng cục thuế quyết định chọn mã số thuế từ tập S bao gồm các biểu diễn trong hệ đếm cơ số 36 của tất cả các số nguyên dương trong phạm vi từ 1 đến n ($36 \le n \le 10^{16}$). Để biểu diễn các chữ số trong hệ đếm cơ số 36, Tổng cục thế sử dụng các kí tự từ 0 đến 9 và 26 chữ cái latinh từ a đến z theo quy tắc chỉ ra trong bảng 1. Một số trong hệ đếm cơ số 36 có thể hiểu là số biểu diễn trong hệ đếm cơ số q ($2 \le q \le 36$) nếu nó chỉ chứa các các chữ số trong q chữ số đầu tiên trong hệ đếm cơ số 36.

| Số hệ 10 | Chữ số hệ 36 |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |

| Số hệ 10 | Chữ số hệ 36 |
|---|---|
| 10 | a |
| 11 | b |
| 12 | c |
| 13 | d |
| 14 | e |
| 15 | f |
| 16 | g |
| 17 | h |
| 18 | i |
| 19 | j |

| Số hệ 10 | Chữ số hệ 36 |
|---|---|
| 20 | k |
| 21 | l |
| 22 | m |
| 23 | n |
| 24 | o |
| 25 | p |
| 26 | q |
| 27 | r |
| 28 | s |
| 29 | t |

| Số hệ 10 | Chữ số hệ 36 |
|---|---|
| 30 | u |
| 31 | v |
| 32 | w |
| 33 | x |
| 34 | y |
| 35 | z |
| | |
| | |
| | |
| | |

Bảng 1

Có tất cả m Cục thuế được đánh số từ 1 đến m làm nhiệm vụ duyệt hồ sơ và cấp mã số thuế ($3 \le m \le 70$).

Để việc cấp mã số thuế có thể được tiến hành song song ở tất cả các Cục thuế, trước hết Tổng cục thuế chọn dãy số nguyên $c_1, c_2, \ldots, c_k$ thỏa mãn $1 < c_1 < c_2 < \ldots < c_k < 36$, trong đó $k = [(m-1)/2]$ (kí hiệu $[\alpha]$ là số nguyên lớn nhất bé hơn hoặc bằng $\alpha$) và sau đó tiến hành phân phối mã số thuế cho các Cục thuế như sau: đầu tiên, từ tập S lọc ra tất cả các số có thể hiểu như là số trong hệ đếm cơ số $c_1$, chuyển cho các Cục thuế thứ nhất và thứ hai sử dụng, sau đó loại bỏ tất cả các số này khỏi tập S; tiếp đến, lọc ra tất cả các số còn lại trong S có thể hiểu như số đếm ở hệ đếm cơ số $c_2$ chuyển cho các Cục thuế thứ 3 và thứ 4 sử dụng, sau đó loại bỏ tất cả các số này khỏi tập S; ... Cục thuế cuối cùng (nếu m lẻ) hoặc 2 Cục thuế cuối cùng (nếu m chẵn) được sử dụng các mã còn lại trong tập S.

Tại các Cục thuế, mã số thuế được cấp theo quy tắc sau: các Cục thuế với số hiệu lẻ cấp mã số thuế theo tứ tự từ nhỏ đến lớn, còn các Cục thuế với số hiệu chẵn cấp mã số

thuế theo thứ tự từ lớn đến nhỏ trong tập các mã số được phân phối.

Ví dụ, với n = 50, m = 3 và $c_1$ = 16. Ta có tập mã số thuế ban đầu S = {1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1a, 1b, 1c, 1d, 1e}. Khí đó các Cục thuế 1 và 2 được sử dụng các mã trong tập {1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1a, 1b, 1c, 1d, 1e}; Cục thuế 3 được sử dụng các mã {g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z}. Người thứ nhất đến Cục thuế 2 được cấp mã số thuế 1e, người thứ hai đến Cục thuế này được cấp mã số thuế 1d, ...

**Yêu cầu**

Cho ở dạng hệ đếm cơ số 10 các số nguyên dương n, m, $c_1$, $c_2$, ..., $c_k$, p và q. Hãy xác định mã số thuế của người thứ q ở Cục thuế p.

**Dữ liệu: Trong file TAXID.INP**

- Dòng thứ nhất chứa các số nguyên n, m, p, q.
- Dòng thứ hai chứ k số nguyên $c_1$, $c_2$, ..., $c_k$ (k = [(m-1)/2]).

Các số trên một dòng được ghi cách nhau một dấu cách. Dữ liệu đảm bảo tồn tại mã số thuế.

**Kết quả : Đưa ra file TAXID.OUT**

Đưa ra mã số thuế tìm được.

**Ví dụ**

| TAXID.INP | TAXID.OUT |
|---|---|
| 30 5 2 2 | 1d |

| | |
|---|---|
| 16 | |

## D. LỜI KẾT:

Trên đây là những kinh nghiệm mà tôi đã nghiên cứu vận dụng trong quá trình giảng dạy thực tế. Thông qua một số bài tập trên, học sinh được rèn luyện cách xây dựng các biến thể của thuật toán tìm kiếm nhị phân. Từ đó các em có những kinh nghiệm vận dụng linh hoạt một thuật toán rất cơ bản vào từng bài toán cụ thể. Để đề tài của tôi được hoàn thiện hơn, việc sử dụng đạt hiệu quả cao hơn, tôi rất mong các thầy cô và các đồng nghiệp đóng góp ý kiến để việc giảng dạy trong nhà trường ngày càng hiệu quả hơn, giúp các em học sinh học tốt hơn.

## E.TÀI LIỆU THAM KHẢO

1. TÀI LIỆU GIÁO KHOA CHUYÊN TIN QUYỂN 1-NHÀ XUẤT BẢN GIÁO DỤC

2. BÀI GIẢNG CỦA LÊ MINH HOÀNG - ĐHSP HÀ NỘI

3. TẠP CHÍ TIN HỌC - NHÀ TRƯỜNG

4. MỘT SỐ ĐỀ THI CHỌN HSG THPT, ĐỀ CHỌN HSG CÁC TỈNH DUYÊN HẢI ĐỒNG BẰNG BẮC BỘ, ĐỀ THI HSG QUỐC GIA 2005, 2010, 2012

5.BÀI TẬP TRÊN **https://vn.spoj.pl/problem**