----------*@*----------

CHUYÊN ĐỀ
CÁC BÀI TOÁN VỀ CẤU HÌNH TẬP HỢP

(Chuyên đề tham gia Hội thảo Khoa học các trường THPT chuyên
Khu vực Duyên hải và Đồng bằng Bắc bộ lần thứ XI,
tại tỉnh Hưng Yên năm 2018)

PHẦN MỞ ĐẦU

Trong quá trình nghiên cứu giải quyết các bài toán trong tin học người ta đã nhận thấy rằng có nhiều bài toán có chung một dạng yêu cầu là tìm các đối tượng để thỏa mãn những điều kiện nhất định trong một tập của các đối tượng cho trước. Các bài toán như vậy được gọi chung là bài toán tìm cấu hình tổ hợp.

Trong các kì thi VMO; VNTST, hay IMO thì các bài toán về tổ hợp thường đóng vai trò quan trọng. Ta có thể thấy việc xây dựng cấu hình cho bài toán chính là mấu chốt trong các cuộc thi lớn ở khu vực và quốc tế, phương pháp này đã được sử dụng một cách rộng rãi.

Các bài toán tìm cấu hình tổ hợp thường gặp:

- **Bài toán đếm**: "Có bao nhiêu cấu hình thỏa mãn điều kiện đã nêu?"Dùng để đánh giá hiệu quả công việc như: tính xác xuất của 1 sự kiện, độ phức tạp của một thuật toán.
- **Bài toán liệt kê**: "Liệt kê tất cả các cấu hình tổ hợp có thể được" (các phương án) Bài toán liệt kê làm nền, làm cơ sở cho nhiều bài toán khác.
- **Bài toán tối ưu**: "Chỉ ra một hoặc một số cấu hình tốt nhất theo một nghĩa nào đó" (có giá lớn nhất hay có giá nhỏ nhất), là bài toán phổ biến trong cuộc sống
- **Bài toán tồn tại**: "Có tồn tại hay không tồn tại cấu hình thỏa mãn điều kiện nào đó?"

Trong chuyên đề này, tôi xin nêu một số bài toán tìm cấu hình tổ hợp thường gặp như đếm số lượng cấu hình, cho một cấu hình tìm thứ tự từ điển, cho thứ tự từ điển tìm cấu hình,....

PHẦN NỘI DUNG CHUYÊN ĐỀ

I. Khái niệm

Lý thuyết tổ hợp được áp dụng nhiều trong Tin học và thường có dạng: Tìm các đối tượng x thỏa mãn những điều kiện nhất định trong một tập S các đối tượng cho trước.

Bài toán tìm cấu hình tổ hợp là bài toán yêu cầu tìm các đối tượng x có dạng là một vector thỏa mãn điều kiện sau:

1. Đối tượng x gồm n phần tử: $x=(x_1,x_2,..x_n)$.
2. Mỗi phần tử x_i có thể nhận một trong các giá trị rời rạc $a_1, a_2,…a_m$.
3. x thỏa mãn các ràng buộc có thể cho bởi hàm logic G(x).

Tùy từng trường hợp mà bài toán có thể yêu cầu: tìm một nghiệm, tìm tất cả các nghiệm hoặc đếm các nghiệm.

II. Các phương pháp chung giải bài toán tổ hợp.

- Phương pháp duyệt toàn bộ
- Phương pháp quay lui
- Phương pháp sinh
- Phương pháp tham lam

…

III. Một số bài tập áp dụng

Bài toán 1.

Hàng năm, Đại học Z tổ chức một cuộc thi tin học sinh viên. Mỗi đội gồm ba sinh viên trong đó có một nam và hai nữ.

Để làm cho cuộc thi thú vị, hiệu trưởng của trường đã quyết định gửi K sinh viên thực tập tại một đất nước xa xôi. Những sinh viên này không thể tham dự cuộc thi

Với số lượng M sinh viên nữ, số lượng N sinh viên nam, và số lượng các sinh viên phải được gửi đi thực tập là K, Hiệu trưởng phải tạo ra số lượng tối đa các đội có thể tham dự cuộc thi.

Ví dụ, nếu M là 6, N là 3 và K là 2, Hiệu trưởng có thể gửi một sinh viên nữ và một sinh viên nam đi thực tập, còn lại có 5 sinh viên nữ và 2 sinh viên nam. Ông có thể tạo ra hai đội (một sinh viên nữ còn lại không được tham gia).

Dữ liệu vào: trong file **TH.INP**

Dòng đầu tiên và duy nhất của đầu vào có chứa ba số nguyên cách nhau bởi khoảng trắng: M ($0 \le M \le 100$), số lượng sinh viên nữ, N ($0 \le N \le 100$), số sinh viên nam, và K ($0 \le K \le M + N$), số lượng các sinh viên đi thực tập

Dữ liệu ra: trong file **TH.OUT**

Dòng đầu tiên và duy nhất của đầu ra chứa số lượng tối đa các đội có thể được hình thành.

Ví dụ

TH.INP	TH.OUT
6 3 2	2

***Phân tích thuật toán:**

Quan sát rằng một đội bóng có thể được hình thành nếu ba điều kiện sau thực hiện: số lượng các cô gái ít nhất là 2, số chàng trai là ít nhất là 1, và $M + N \ge K + 3$ (từ một nhóm gồm ba sinh viên, và sinh viên K cần phải đi thực tập). Chúng ta có thuật toán tham lam như sau - hình thành đội miễn là các điều kiện được đáp ứng.

while (M ≥ 2 and N ≥ 1 and M+N ≥ K+3) do
{
result := result+1; (tăng số đội)
M := M-2; (giảm số sinh viên nữ)
N := N-1; (giảm số sinh viên nam)
}

Code

```
#include<iostream>
using namespace std;
int main ()
{
   int m, n, k;
   cin >> m >> n >> k;
   int res = 0;
   while(m >= 2 && n >= 1 && m+n >= k+3)
   {
      res++;
      m -= 2;
      n -= 1;
   }
   cout << res << endl;
```

```
    return 0;
}
```

Test + code : https://www.mediafire.com/file/n27czsbjdzjumdl/TH.rar

Bài toán 2: Lập lịch ưu tiên đúng hạn.

Có n công việc được đánh số từ 1 đến n và có một máy để thực hiện chúng. Biết:

+ p_i là thời gian cần thiết để hoàn thành công việc i.

+ d_i là thời hạn hoàn thành công việc i.

Mỗi công việc cần được thực hiện liên tục từ lúc bắt đầu cho tới khi kết thúc, không cho phép ngắt quãng. Khoảng thời gian thực hiện hai công việc bất kì chỉ được có nhiều nhất 1 điểm chung. Thời điểm bắt đầu thực hiện n công việc tính từ 0. Giả sử c_i là thời điểm hoàn thành công việc i. Khi đó, nếu $c_i > d_i$ ta nói công việc i bị hoàn thành trễ hạn, nếu nếu $c_i < d_i$ ta nói công việc i bị hoàn thành đúng hạn.

Yêu cầu*:* Tìm trình tự thực hiện các công việc sao cho số công việc được hoàn thành đúng hạn là lớn nhất.

Dữ liệu: Vào từ file văn bản LICHDH.INP:

+ Dòng đầu tiên chứa số nguyên dương n ($0 < n < 100$).

+ Dòng thứ 2 chứa n số nguyên dương $p_1, p_2, \ldots, p_n$.

+ Dòng thứ 3 chứa n số nguyên dương $d_1, d_2, \ldots, d_n$.

Kết quả*:* Ghi ra file LICHDH.OUT gồm:

+ Dòng đầu tiên ghi số lượng công việc được hoàn thành đúng hạn theo trình tự tìm được.

+ Dòng tiếp theo ghi trình tự thực hiện các công việc đã cho.

Ví dụ:

LICHDH.INP	LICHDH.OUT
6 4 1 2 3 1 3 5 6 6 7 8 7	4 2 3 4 5 1 6

****Phân tích thuật toán:***

- Một trình tự thực hiện các công việc (một phương án của bài toán) là một hoán vị của n phần tử {1, 2, …, n}.

Cụ thể: mỗi phương án là 1 bộ x = (x1, x2, …,xn); trong đó xi thuộc {1, 2, …, n}, xi <> xj nếu i<>j;

- Giá của phương án x (số công việc đúng hạn) bằng: f(x) = k

Với điều kiện g(x) = $\sum p_i \le d_i$ với mọi i = 1,2, …, k

Bài toán lập lịch đúng hạn đưa về bài toán: "Tìm Max{f(x)| $x \in D$},

với D={x|x=(x_1,x_2,…,x_n), $x_i \in$ {1, 2, …, n}, xi <> xj nếu i<>j; và g(x) $\le d_i$}

Code:

```
program LapLichUuTienDungHan;
uses crt;
const fin = 'LICHD.INP';
      fout = 'LICHD.OUT';
      maxn = 100;

var p,d,x,kqx        : array[1..maxn] of integer;
    cx,kqcx          : array[1..maxn] of boolean;
    f                : text;
    n                : integer;
    tg,count,max: integer;
{-------------------------------------------------}
procedure Init;
var i: integer;
begin
    assign(f,fin);reset(f);        readln(f,n);
    for i:=1 to n do read(f,p[i]);
    readln(f);
    for i:=1 to n do read(f,d[i]);
    close(f);

    fillchar(cx,sizeof(cx),true);
    tg := 0;             count:=0;   max:=0;
end;
procedure Try(i: integer);
var j: integer;
begin
    for j:=1 to n do
    begin
        if cx[j] then
```

```
        begin
            x[i]:=j;
            cx[j]:=false;
            tg := tg + p[j];
            if (tg <= d[j]) then inc(count);
            if (i = n) then
            begin
                if count>max then

                begin
                    max:=count;
                    kqx:=x;
                end;
            end
            else if count+n-i>max then Try(i+1);

            if tg <= d[j] then dec(count);
            tg := tg - p[j];
            cx[j]:=true;
        end;
    end;
end;
{-----------------------------------------------------}
procedure Result;
var i,j: integer;
begin
    assign(f,fout);rewrite(f);
    writeln(f,max);
    for i:=1 to n do write(f,kqx[i],' '); close(f);
end;
{-----------------------------------------------------}
BEGIN
    Init;
    Try(1);
    Result;
```

END.

Chương trình trên chạy chậm với n >=20 vì vậy ta có thể xét thuật toán tham lam như sau

Giả sử có 5 công việc với thời gian thực hiện và thời gian hoàn thành như sau:

i	1	2	3	4	5
Pi	6	3	5	7	2
di	8	4	15	20	3

Nếu thực hiện theo thứ tự 1, 2, 3, 4, 5 thì sẽ có 3 công việc bị trễ hạn là công việc 2, 4, 5.

Còn nếu thực hiện theo thứ tự 5, 1, 3, 4, 2 thì chỉ có 1 công việc bị trễ hạn là công việc 2.

Đây là thứ tự thực hiện mà một số công việc bị trễ hạn ít nhất (tối ưu)

Ta có hai nhận xét:

- Nếu thứ tự thực hiện các công việc mà có công việc bị trễ hạn được xếp trước một công việc đúng hạn thì ta sẽ nhận được trình tự tốt hơn bằng cách chuyển công việc trễ hạn xuốn cuối cùng (vì đằng nào công việc này cũng bị trễ hạn) . như vậy ta chỉ quan tâm đến việc xếp lịch cho các công việc hoàn thành đúng hạn còn các công việc bị trễ hạn có thể thực hiện theo trình tự bất kì.
- Giả sử Q là tập gồm k công việc mà cả k công việc trong Q có thể thực hiện đúng hạn và T = (i1, i2, …, ik) là một hoán vị của các công việc trong Q sao cho $d_{i1} \le d_{i2} \le \ldots \le d_{ik}$ thì thứ tự trong tập T là thứ tự để hoàn thành đúng hạn được cả k công việc

Sử dụng chiến lược tham lam ta xây dựng tập công việc Q theo từng bước, ban đầu khởi tạo Q rỗng. Hàm chọn được xây dựng như sau: tại mỗi bước ta sẽ chọn công việc x mà có thời gian thực hiện nhỏ nhất trong số các công việc còn lại cho vào tập Q. nếu khi kết nạp x các công việc trong tập Q đều có thể thực hiện đúng hạn thì cố định việc kết nạp x vào tập Q, nếu không thì không kết nạp x.

Code dùng thuật toán tham lam

```
const MAX =100;
fi ='LICHD.inp';
fo ='LICHD.out';
type TJob =record
p, d :longint;
name :longint;
end;
TArrJobs =array[1..MAX]of TJob;
var jobs,Js :TArrJobs;
d :array[1..MAX]of longint; f :text;
n,m :longint;
```

```
procedure input;
var g :text;
i :longint;
begin
        assign(g,fi);
        reset(g);
        readln(g,n);
        for i:=1 to n do read(g,jobs[i].p);
        for i:=1 to n do read(g,jobs[i].d);
        close(g);
        for i:=1 to n do jobs[i].name:=i;
end;
procedure swap(var j1,j2:TJob);
var tmp :TJob;
        begin
        tmp:=j1;
        j1:=j2;
        j2:=tmp;
end;
function check(var Js:TArrJobs; nJob:longint):boolean;
var i,j :longint;
t :longint;
begin
         for i:=1 to nJob-1 do
         for j:=i+1 to nJob do
         if Js[i].d>Js[j].d then swap(Js[i],Js[j]);
         t:=0;
         for i:=1 to nJob do begin
         if t+Js[i].p>Js[i].d then exit(false);
         t:=t+Js[i].p;
         end;
         exit(true);
end;
procedure Greedy;
var i,j :longint;
```

```
Js2 :TArrJobs;
begin
        for i:=1 to n-1 do
        for j:=i+1 to n do
        if jobs[i].p > jobs[j].p then
        swap(jobs[i],jobs[j]);
        fillchar(d,sizeof(d),0);
        m:=0;
        for i:=1 to n do begin
        Js2:=Js;
        Js2[m+1]:=jobs[i];
        if check(Js2,m+1) then begin
        m:=m+1;
        Js:=Js2;
        d[i]:=1;
        end;
        end;
        writeln(f,m);
        for i:=1 to n do
        if d[i]=0 then begin
        m:=m+1;
        Js[m]:=jobs[i];
        end;
end;
procedure printResult;
var
i :longint;
begin
          for i:=1 to n do write(f,Js[i].name,' ');
end;
BEGIN
          assign(f,fo); rewrite(f);
          input;
          Greedy;
          printResult;
```

close(f);

END.

Test + code: https://www.mediafire.com/file/y11ni36h9xjk631/LICHD.rar

4.3 Bài toán về quy hoạch động cấu hình

Bài toán về quy hoạch động cấu hình hầu như là những bài toán về thứ tự từ điển với 2 công việc:

- Cho dãy và tìm thứ tự từ điển:chạy i từ 1 đến độ dài của dãy,cộng vào kết quả số cấu hình nhận đoạn từ 1 →i – 1 của dãy làm đầu và đoạn còn lại thì nhỏ hơn đoạn sau của dãy.
- Cho số thứ tự và tìm dãy:với mỗi vị trí thì xét các phần tử chưa được chọn theo thứ tự từ nhỏ đến lớn cho đến khi tìm được phần tử phù hợp.

Bài toán 3 SHHV - Số hiệu hoán vị

http://vn.spoj.com/problems/SHHV

Xét tất cả các hoán vị của dãy số tự nhiên (1, 2,..., n) (1 <= n <= 12)

Giả sử rằng các hoán vị được sắp xếp theo thứ tự từ điển.

Yêu cầu:

- 1: Cho trước 1 hoán vị. Tìm số hiệu của hoán vị đó trong dãy đã sắp xếp
- 2: Cho trước số hiệu của 1 hoán vị trong dãy hoán vị đã sắp xếp. Tìm hoán vị đó

Input: trong file **HV.inp**

- Dòng 1: Chứa n số a1, a2, …, an (dãy hoán vị n phần tử)
- Dòng 2: Chứa số p (số hiệu của hoán vị trong dãy hoán vị n phần tử)

Output: trong file **HV.out**

- Dòng 1: Ghi số q (số hiệu của dãy hoán vị ai)
- Dòng 2: Ghi n số b1, b2, …, bn (dãy hoán vị có số hiệu p)

Ví dụ

Hv.inp	Hv.out
2 1 3 4	3 2 3 1

Phân tích thuật toán

+ Bước 1: tạo mảng qhđ F[i]:số hoán vị độ dài i F[0] = 1; F[i] = F[i – 1] * i;

+ Bước 2: xét ví dụ N = 4 a: 3 4 1 2 Ta đi tìm số hoán vị có thứ tự từ điển nhỏ hơn dãy a là res.

i = 1:tìm số hoán vị có phần tử đầu tiên < a[1]các số nhỏ hơn a[1] gồm 1 và 2,cứ mỗi phần tử x nhỏ hơn này sẽ có F[n – 1] dãy hoán vị có phần tử đầu tiên bằng x và thứ tự từ điển của chúng luôn nhỏ hơn dãy a

chẳng hạn với x = 1 có :

1 2 3 4

1 2 4 3

1 3 2 4

1 3 4 2

1 4 2 3

1 4 3 2

Res := res + 2 * F[n – 1] = 12;

i = 2: Ta đi tìm số hoán vị có phần tử đầu tiên = 3 mà có thứ tự từ điển nhỏ hơn a. Những phần tử này phải có phần tử thứ 2 nhỏ hơn a[2] và cứ mỗi phần tử sẽ có F[n – 2] hoán vị thỏa mãn. Ở đây có giá trị 1 và 2 thỏa mãn res:=res + 2 * F[n – 2] = 16;

Tương tự i=3:Không có phần tử nào nhỏ hơn nữa

i=4: Cũng không có phần tử nào Vậy kết quả sẽ là (res + 1)=17;

Bước 3:Cho p, đi tìm dãy b có thứ tự từ điển p Làm ngược lại bước 2 ta có kết quả

Code

```
const  maxN = 15;
      fi='hv.inp';
      fo='hv.out';
var n:integer;
    dd:array[0..15] of boolean;
     a:array[1..15] of integer;
     //a:string[13];
     x:longint;
     F:array[0..15] of int64;
procedure solve1;
var ans,cnt,i,j:longint;
begin
     F[0]:= 1; FOR i:=1 to n do F[i]:= F[i-1] * i;
    ans:=1;
    for i:=1 to n do
    begin
       cnt:=0;
```

```
        FOR j:=1 to n do
        if not dd[j] and (j<a[i]) then cnt:=cnt+j;
        ans:=ans+ F[n-i] * cnt;
        dd[a[i]]:= true;
    end;
    writeln(ans);
end;
procedure  solve2;
var cnt,ord,i,j:longint;
begin
    FOR i:=1 to n do
      begin
       dd[i]:=false;
        a[i]:= 0;
      end;
    x:=x-1;
    FOR i:=1 to n do
    begin
        ord:= 0; cnt:= 0;
        while (x >= F[n-i]) do
        begin
            x :=x- F[n-i]; inc(ord);
        end;
        FOR j:=1 to n do
        begin
            if not dd[j] and (cnt = ord) then
            begin
                a[i] := j; dd[j]:= true; break;
            end;
            if (not dd[j]) then cnt:= cnt+1;
        end;
    end;
    FOR i:=1 to n do write(a[i]);
end;
begin
```

```
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    n:=1;
    while not eoln(input) do begin read(x); a[n]:=x; inc(n);  end;
    n:=n-1;  readln(x);
   solve1;
   solve2;
   close(input); close(output);
end.
```

Test + code : https://www.mediafire.com/file/1op3dhpbu7m0kdw/SHHV.rar

Bài toán 4 SPBINARY - Xâu Nhị Phân

http://vn.spoj.com/problems/SPBINARY

Cho 2 số n và k (2<=k <= n <= 600)

Hãy đếm xem có bao nhiêu xâu nhị phân độ dài n mà không có quá k số 0 hoặc k số 1 nào liên tiếp nhau.

Input: trong file **xaunp.inp**

- Gồm 1 dòng duy nhất là 2 số n và k

Output: trong file **xaunp.out**

- Gồm 1 dòng duy nhất là số dãy nhị phân thoả mãn

Example

Xaunp.inp	**Xaunp.out**
3 2	6

Giải thích : Đó là các xâu 001 , 010 , 011 , 100 , 101 , 110.

Phân tích thuật toán

- Mảng qhđ: F[i, t] ($1 \le i \le n, 0 \le t \le 1$): số xâu nhị phân độ dài i không có quá k số 0 hoặc số 1 liên tiếp kết thúc bằng bit t

+ F[0, 0] := 1; F[0, 1] := 1; F[1, 0] := 1; F[1, 1] := 1;

+ F[i ,0] := F[i – 1,0] + F[i – 1,1] F[i,1] := F[i – 1,0] + F[i – 1,1]

Với $2 \le i \le k$

+ F[i, t] := F[i – 1,0] + F[i – 1, 1] – F[i – k – 1, t – 1] với i > k

+ Dùng số lớn

Giải thích:

+ Với độ dài $i \le k$ thì mọi xâu nhị phân độ dài i đều là một xâu thỏa mãn nên

F[i, t] := F[i – 1, 0] + F[i – 1, 1];

+ Với độ dài i > k thì ta xét các xâu lần lượt kết thúc bằng 1, 2, , …, k bit 0 hoặc 1.
Chẳng hạn tính F[i, 0]

- Xâu kết thúc bằng 1 bit 0(xâu có dạng ab...xy10): F[i – 1, 1]
- Xâu kết thúc bằng 2 bit 0(xâu có dạng ab...xy100): F[i – 2, 1]
- .....
- Xâu kết thúc bằng k bit 0: F[i – k, 1]

→ F[i, 0] := F[i – 1, 1] + F[i – 2, 1] +....+ F[i – k, 1]

Vậy $F[i-1,0]=\sum_{t=1}^{k} F[i-t-1]$

→ F[i, 0] := F[i – 1, 0] + F[i – 1, 1] – F[i – k – 1, 1].

Code

```
const max = 600;
 type int69 = string;
 function add(a,b : int69) : int69;
var i,tmp : longint;
c : int69;
begin
        while length(a) < length(b) do a := '0' + a;
        while length(b) < length(a) do b := '0' + b;
         c := '';
        tmp := 0;
        for i := length(a) downto 1 do
        begin
        tmp := tmp + ord(a[i]) + ord(b[i]) - 2*48;
        c := char(tmp mod 10 + 48) + c;
        tmp := tmp div 10;
        end;
        if tmp > 0 then c := '1' + c;
        exit(c);
end;
 function sub(a,b : int69) : int69;
var i,tmp,sum : longint;
c : int69;
begin
      while length(a) < length(b) do a := '0' + a;
```

```
        while length(b) < length(a) do b := '0' + b;
         c := '';
        tmp := 0;
         for i := length(a) downto 1 do
        begin
        sum := ord(a[i]) - ord(b[i]) - tmp;
        if sum < 0 then
        begin
        sum := sum + 10;
        tmp := 1;
        end
        else tmp := 0;
        c := char(sum + 48) + c;
        end;
        while (c[1] = '0') and (length(c) > 1) do delete(c,1,1);
        exit(c);
end;
 var f : array[0..max,0..1] of int69;
n,k : longint;
 procedure Enter;
begin
          assign(input,'xaunp.inp'); reset(input);
          readln(n,k);
          close(input);
end;
 procedure Optimize;
var i,j : longint;
begin
          f[0,0] := '1';
          f[0,1] := '1';
          f[1,0] := '1';
          f[1,1] := '1';
           for i := 2 to k do
          for j := 0 to 1 do
          f[i,j] := add(f[i-1,0],f[i-1,1]);
```

```
        for i := k+1 to n do
        for j := 0 to 1 do
        f[i,j] := sub(add(f[i-1,0],f[i-1,1]),f[i-k-1,1-j]);
        assign(output,'xaunp.out'); rewrite(output);
        writeln(add(f[n,1],f[n,0]));
        close(output);
end;
 BEGIN
        Enter;
        Optimize;
END.
```

Test + code: https://www.mediafire.com/file/87lcyye8lr6o43h/XAUNP.rar

Bài toán 5. Tung đồng xu.

http://vn.spoj.com/problems/NKTOSS

Ngày xưa, cách đây đã lâu lắm rồi, ở vương quốc Byteland tươi đẹp có một nàng công chúa xinh đẹp tuyệt trần. Thật không may, chính vì sự xinh đẹp đó đã làm phù thủy Astral đã bắt làm về làm người hầu cho ông ta. Đức Vua vô cùng hoang mang khi chuyện này xảy ra, ông không biết phải làm cách nào để giải cứu con mình (ông không thể mang quân đến đánh vì điều đó là vô nghĩa). Tuy nhiên, tên phù thủy này lại rất sợ một câu thần chú được suy ra từ việc giải một bài toán cổ của Thần Sphinx. Bài toán đó có thể được mô tả một cách đơn giản như sau: "Khi ta tung một đồng xu, ta sẽ nhận được mặt sấp hoặc ngửa. Nếu ta tung lần lượt N đồng xu thì có bao nhiêu trường hợp mà có ít nhất K đồng xu liên tiếp cùng là ngửa ?" . Đức vua hứa sẽ thưởng rất hậu hĩnh và gả công chúa cho ai giải được bài toán này. Thực ra công chúa và anh chàng làm vườn trong hoàng cung đã yêu thương nhau từ lâu. Anh chàng giờ đây đang rất bối rối và cần sự giúp đỡ của bạn.

Dữ liệu: trong file NKTOSS.INP

- Một dòng duy nhất ghi hai số N và K.

Kết quả: trong file NKTOSS.OUT

- Một dòng duy nhất ghi số trường hợp đếm được.

Giới hạn

- $1 \leq K \leq N \leq 10000$

Ví dụ

NKTOSS.INP	NKTOSS.OUT
4 2	8

Phân tích thuật toán

Gọi 0 là sắp, 1 là ngửa

Gọi F[i] là số cách phù hợp cho đoạn gồm i đồng xu ngửa độ dài ít nhất là k, ở đây tự quy ước dãy mới tạo sẽ là có từ i-k+1 đến i đều là 1. Từ 1->k-1 f[i]=0

f[k]=1;

Giả sử ta muốn tính F[i], từ F[i-1], số dãy thỏa mãn độ dài i-1, ta có số trường hợp thỏa mãn độ dài i là F[i-1]*2 vì tiếp theo sẽ là 0 hoặc 1

Bắt đầu vào việc sinh dãy mới, ta sẽ coi dãy mới có từ i-k+1 đến i đều là 1, vậy số dãy mới tạo ra là 2^(i-k-1).

Tuy nhiên, một rắc rối là việc trùng dãy được tạo thành từ trước, nhưng nó cũng là dãy được tạo thành F[i-k-1]. Vậy ta chỉ việc lấy - F[i-k-1].

Vậy công thức tổng lại là:

F[i]=F[i-1]*2+2^(i-k-1)-F[i-k-1]

Vì kết quả là 1 số khá lớn nên cài thêm bignum.

Code

```
#include<bits/stdc++.h>
#define maxn 10005
#define base 1000000000
using namespace std;
typedef vector<int> bignum;
bignum f[maxn];
int n,k;
bignum to_big(int a)
{
	bignum c;
	c.clear();
	while(a)
{
	c.push_back(a%base);
	a/=base;
}
return c;
}
	bignum fix(bignum &a)
{
```

```
        a.push_back(0);
        for (int i=0; i<a.size(); ++i)
{
        a[i+1]+=a[i]/base;
        a[i]%=base;
        while(a[i]<0)
        {
        a[i]+=base;
        --a[i+1];
        }
}
while(a.size()>=2&&a.back()==0) a.pop_back();
return a;
}
ostream& operator << (ostream& cout, const bignum &a)
{
        printf("%d", a.back());
        for (int i=(int)a.size()-2; i>=0; i--)
        printf("%09d", a[i]);
        return cout;
}
 bignum operator +(bignum a,const bignum &b)
{
    a.resize(max(a.size(),b.size()));
    for (int i=0; i<b.size(); ++i)
    a[i]+=b[i];
    return fix(a);
}
bignum operator -(bignum a,const bignum &b)
{
     a.resize(max(a.size(),b.size()));
     for (int i=0; i<b.size(); ++i)
     a[i]-=b[i];
     return fix(a);
}
```

```
bignum operator *(const bignum &a,const bignum &b)
{
	bignum c(a.size()+b.size());
	for (int i=0; i<a.size(); ++i)
	for (int j=0; j<b.size(); ++j)
	c[i+j]+=a[i]*b[j];
	return fix(c);
}
int main()
{
	freopen("NKTOSS.inp","r",stdin);
	freopen("NKTOSS.out","w",stdout);
	scanf("%d %d",&n,&k);
	bignum one=to_big(1);
	bignum two=to_big(2);
	f[k]=one;
	bignum sum=one;
	for (int i=k+1; i<=n; ++i)
	{
	f[i]=two*f[i-1]+sum-f[i-k-1];
	sum=sum*two;
	}
	cout<<f[n];
	return 0;
}
```

Test + code:: https://www.mediafire.com/file/cg5dpug50b8589k/NKTOSS.rar

Một số bài tập luyện tập

Bài toán 6: Tìm tổng nhỏ nhất

Cho ma trận gồm m dòng, n cột có các phần tử là các số nguyên a[i,j], trong đó $|a[i,j]| \le 30000$ (i=1..m, j=1..n).

Yêu cầu: Hãy chọn trên mỗi dòng, mỗi cột của ma trận đúng một số sao cho không có hai số nào thuộc cùng một dòng hay cùng một cột để tổng các số được chọn là nhỏ nhất.

Dữ liệu: Vào từ file văn bản TONGMIN.INP có cấu trúc:

- Dòng đầu ghi 2 số nguyên m, n. ($1 \le m \le 100$, $1 \le n \le 30$);

- Trong m dòng tiếp theo, dòng thứ i ghi n số a[i,j] với i=1..m, j=1..n;

Kết quả: Ghi ra file văn bản TONGMIN.OUT có cấu trúc:

- Dòng đầu tiên ghi tổng nhỏ nhất tìm được

- Trong n dòng tiếp theo, mỗi dòng ghi 3 số: số thứ nhất là giá trị của số được chọn, hai số sau là toạ độ [dòng, cột] của số được chọn.

Ví dụ:

TONGMIN.INP	TONGMIN.OUT
7 4 8 6 -18 0 -15 1 15 0 -5 0 9 13 11 -14 -12 12 19 3 14 12 20 -4 2 17 10 -8 5 -20	-67 -18 1 3 -15 2 1 -14 4 2 -20 7 4

***Phân tích thuật toán:**

- Nếu m ≤ n thì : "duyệt các chỉnh hợp không lặp chập m của n phần tử {1,2,…,n}, tức là mỗi phương án cần tìm là một bộ x = (x1, x2, …, xm), xi = j nghĩa là chọn số ở hàng i cột j; i = 1, 2,.., m; j = 1,2, …,n; tính giá của phương án x bằng

công thức $f(x) = \sum_{i=1}^{m} a[i,x[i]]$ và giữ lại phương án có giá trị nhỏ nhất"

Nếu m>n thì "duyệt các chỉnh hợp không lặp chập n của m phần tử mỗi {1,2,.....,m}, tức là phương án cần tìm là một bộ x = (x1, x2, …, xn), xi = j nghĩa hàng j; i = 1, 2,.., n; j = 1,2, …,m; tính giá của phương án x bằng công thức $\sum_{i=1}^{n} a[x[i],i]$ và giữ lại giá trị nhỏ nhất"

Bài toán 7 Bài toán Rôbốt quét vôi

http://vn.spoj.pl/problems/NKROBOT

Có 9 căn phòng (đánh số từ 1 đến 9) đã được quét vôi với mầu trắng, xanh hoặc vàng. Có 9 rôbốt (đánh số từ 1 đến 9) phụ trách việc quét vôi các phòng. Mỗi rôbốt chỉ quét vôi một số phòng nhất định. Việc quét vôi được thực hiện nhờ một chương trình cài sẵn theo qui tắc:

- nếu phòng đang có màu trắng thì quét màu xanh,
- nếu phòng đang có màu xanh thì quét màu vàng,

- nếu phòng đang có màu vàng thì quét màu trắng,

Cần phải gọi lần lượt một số các rôbốt ra quét vôi (mỗi lần một rôbốt, một rôbốt có thể gọi nhiều lần và có thể có rôbốt không được gọi. Rôbốt được gọi sẽ quét vôi tất cả các phòng mà nó phụ trách) để cuối cùng các phòng đều có màu trắng.

Hãy tìn một phương án như vậy sao cho lượng vôi phải quét là ít nhất. Giả thiết rằng lượng vôi cho mỗi lượt quét đối với các phòng là như nhau.

Dữ liệu: vào từ fie văn bản ROBOTQV.INP gồm các dòng: 9 dòng đầu, mỗi dòng mô tả danh sách các phòng được quét vôi bởi một rôbốt theo thứ tự từ rôbốt 1 đến rôbốt 9. Mỗi dòng như vậy gồm các số hiệu phòng viết sát nhau. Chẳng hạn dòng thứ 2 có nội dung : 3578

Mô tả rôbốt 2 phụ trách việc quét vôi các phòng 3, 5, 7, 8.

Dòng cuối mô tả màu vôi ban đầu của các phòng. Dòng gồm 9 kí tự viết sát nhau, kí tự thứ i biểu diễn màu vôi của phòng i với qui ước: kí tự T chỉ màu trắng, kí tự X chỉ màu xanh, kí tự V chỉ màu vàng, .

Kết quả: ghi ra tệp văn bản ROBOTQV.OUT gồm một dòng dưới dạng:

- Nếu không có phương án thì ghi một chữ số 0.
- Trái lại ghi dãy thứ tự các rôbốt được gọi (các số hiệu rôbốt viết sát nhau)

Ví dụ:

ROBOTQV.INP	ROBOTQV.INP
159 123 357 147 5 369 456 789 258 XVXVXVTXT	2455688

Phân tích thuật toán

- Mỗi phương án gọi Rôbốt là một chỉnh hợp lặp chập 9 của 3 phần tử {0,1,2}. Kí hiệu $x = (x_1, x_2, ...,x_n)$; với x_i := j nghĩa là Rôbốt i được gọi j lần.

- Gọi r[i] là xâu kí tự cho biết danh sách các phòng do Rôbốt i phụ trách
- Khi đó giá của phương án X= $\sum_{i}^{9} Xi * length(r[i])$

Bài tập 8. CHUOIHAT - Chuỗi hạt

http://vn.spoj.com/problems/CHUOIHAT

Khi tiến hành khai quật khảo cổ ở một vương quốc xa xưa nọ, các nhà khoa học khai quật được rất nhiều chuỗi hạt lạ. Sau khi quan sát, các nhà khoa học thấy rằng các chuỗi hạt có một số đặc điểm chung.

Mỗi chuỗi hạt là một sợi dây được đính các hạt ngọc làm bằng một chất liệu cổ xưa. Các chuỗi hạt đều có số lượng hạt ngọc bằng nhau. Hơn nữa, mỗi hạt ngọc là một hình cầu có đường kính là một số nguyên dương, và nếu lần từ trái sang phải trên chuỗi hạt, người ta thấy các hạt ngọc có đường kính tăng dần. Nếu đánh số vị trí các hạt ngọc bắt đầu từ 1, theo thứ tự từ trái sang phải, người ta nhận thấy rằng hạt ngọc thứ i có đường kính không vượt qúa 2i. Các nhà khoa học cho rằng, dân tộc cổ xưa này hẳn đã làm ra tất cả các chuỗi hạt có cùng những đặc điểm này, dù chúng hiện còn đang rải rác ở đâu đó trên trái đất.

Sau đó không lâu, các nhà khoa học tìm ra một mảnh da, trên đó có ghi một con số theo loại chữ số cổ xưa. Họ cho rằng mảnh da này có liên quan đến các chuỗi hạt kỳ lạ nọ. Sau nhiều cố gắng, các nhà khoa học đã đưa được con số trên mảnh da về hệ chữ số thập phân, và ký hiệu là X.

Manh mối đến đây thì dừng lại, vì các nhà khoa học không tìm thấy được vết tích nào khác nữa, và cũng không tìm ra được mối quan hệ giữa X và các chuỗi hạt.

Đến đây, một nhà khoa học người Việt đề nghị, hãy thử xác định chuỗi hạt có thứ tự từ điển là X, biết đâu đây sẽ là manh mối?

Yêu cầu

Bạn hãy viết chương trình giúp nhà khoa học xác định chuỗi hạt có thứ tự từ điển là X.

Dữ liệu

- Dòng 1: chứa số nguyên dương N, là số hạt ngọc trong mỗi chuỗi hạt
- Dòng 2: chứa số nguyên dương X

Kết qủa

Gồm 1 dòng duy nhất, chứa N số nguyên, cách nhau một khoảng trắng, xác định chuỗi hạt có thứ tự từ điển là X.

(để biểu diễn một chuỗi hạt, cần in ra N số nguyên tương ứng là đường kính của các hạt ngọc trong chuỗi hạt, theo thứ tự từ trái sang phải)

Giới hạn

- N là số nguyên dương trong phạm vi [1, 250]
- X là số nguyên dương trong phạm vi từ 1 đến số lượng tối đa các chuỗi hạt.

Ví dụ

Dữ liệu

2
4

Kết qủa

2 3

Giải thích

Các chuỗi hạt sắp theo thứ tự từ điển:
1 2, 1 3, 1 4, 2 3, 2 4
Chuỗi hạt thứ 4 là 2 3

Phân tích thuật toán

+Mảng qhđ: F[i, j] ($1 \le j \le n, j \le i \le 2*j$): số cấu hình a[j], a[j+1], .., a[n] với a[j] = i;

Ta có $F[i, j] = \sum_{k=i+1}^{2*(j+1)} F[k, j+1]$

→ $F[j, j] = F[i, j] := \sum_{k=j+1}^{2*(j+1)} F[k, j+1]$

F[i, j] = F[i – 1, j] – F[i, j + 1] với i > j;

+Ta tìm cấu hình theo cách chỉ trên.

+Mô tả thuật toán: Dãy a là kết quả

```
for i := 1 to n do
  begin
     for j := a[i – 1] + 1 to 2 * i do
       if F[j, i] < p then p:=p – F[j, i] else break;
     a[i] := j;
     write(j, #32);
  end;
```

+Dùng số lớn

KẾT LUẬN

Lý thuyết tổ hợp được áp dụng trong nhiều lĩnh vực khác nhau: lý thuyết số, hình học hữu hạn, biểu diễn nhóm, đại số không giao hoán, thống kê xác xuất,

Lý thuyết tổ hợp gắn liền với việc nghiên cứu sự phân bố các phần tử vào các tập hợp. Các phần tử là hữu hạn và việc phân bố chúng phải thỏa mãn những điều kiện nào đó tùy theo yêu cầu của công việc (bài toán). Mỗi cách phân bố như thế được gọi là một cấu hình tổ hợp (một bộ các phần tử nào đó). Có thể thấy rằng bài toán về tổ hợp thường đa dạng và đóng vai trò quan. Việc xây dựng cấu hình cho bài toán là mấu chốt và thường được sử dụng rộng rãi.

Trong quá trình nghiên cứu chuyên đề đã tập trung làm rõ các vấn đề:

- Tìm hiểu và trình bày lý thuyết tổ hợp
- Tìm hiểu và trình bày một số phương pháp chung giải quyết bài toán tìm cấu hình tổ hợp.
- Xây dựng chương trình cài đặt 5 bài toán điển hình tìm cầu hình tổ hợp

Mặc dù có nhiều thời gian tìm hiểu chuyên đề tuy nhiên do khả năng trình độ còn hạn chế việc thực hiện viết chuyên đề còn nhiều thiếu sót, rất mong được sự đóng góp của bạn bè và đồng nghiệp để chuyên đề được hoàn thiện hơn.

Xin trân trọng cảm ơn!

TÀI LIỆU THAM KHẢO

1. Cấu trúc dữ liệu và giải thuật – Lê Minh Hoàng (ĐHSP Hà Nội)
2. Tài Liệu sách giáo khoa chuyên tin tập 1,2 - Nhiều tác giả
3. http://vn.spoj.com
4. Website: http://google.com.vn