Duyệt bằng cách chia đôi tập hợp

MỞ ĐẦU

Trong việc lập trình cho máy tính, phương pháp duyệt toàn bộ các cấu hình để tìm phương án tối ưu hay đếm số lượng các cấu hình thỏa mãn một điều kiện nào đó, là một trong những phương pháp quan trọng.

Duyệt toàn bộ là phương pháp liệt kê tất cả các phần tử của một tập hợp D hữu hạn nào đó, từ đó chỉ ra một phần tử thoả mãn tiêu chí tối ưu hoặc là đếm số lượng các phần tử thoả mãn yêu cầu nào đó. Cách tư duy này xuất phát từ tập hợp D là hữu hạn. Có thể nói đây là cách tư duy đơn giản dễ viết chương trình, là phương án lập trình đầu tiên mà mọi học sinh khi bắt đầu học lập trình đều làm quen. Các phương pháp duyệt toàn bộ thường gặp: duyệt toàn bộ bằng cách sử dụng các vòng lặp lồng nhau, duyệt quay lui.

Tuy nhiên có thể thấy rằng phương pháp này còn hạn chế khi số lượng các phần tử của tập D lớn. Nó thể hiện ở chỗ thời gian tính toán để cho ra kết quả thường không chấp nhận được. Do đó trong phương pháp duyệt toàn bộ cần phải bổ sung các phương pháp cho phép bỏ qua hoặc gộp một số phần tử. Điều này cải thiện đáng kể thời gian thực hiện chương trình. Một số phương pháp duyệt cải tiến được đưa ra: duyệt ưu tiên, duyệt nhánh cận, duyệt bằng cách chia đôi tập hợp..

Để xây dựng một chương trình đầy đủ cho tất cả các vấn đề đòi hỏi nhiều công sức của các nhà khoa học giáo dục. Tuy nhiên dựa trên những yêu cầu tối thiểu cho mỗi vấn đề và với vốn kiến thức, kinh nghiệm của bản thân, mỗi giáo viên có thể đưa ra cho

mình một hay nhiều bài giảng, chuyên đề giúp cho học sinh tiếp cận kiến thức một cách phù hợp. Qua quá trình giảng dạy, tôi cũng đã tự xây dựng cho mình một số nội dung đáp ứng nhu cầu giảng dạy cho đối tượng học sinh chuyên.

Trong bài viết này tôi xin trình bày phương pháp ***"Duyệt bằng cách chia đôi tập hợp"***

Duyệt bằng cách chia đôi tập hợp

I. Lý thuyết

1. Biểu diễn các tập con của một tập hợp

- Cho X = {x_1, x_2, . . . , x_n} là một tập hợp gồm n phần tử.
- Mỗi tập con Y của tập X có thể được biểu diễn bằng một dãy nhị phân (b_1, b_2, . . . , b_n) xác định như sau: b_i = 1, nếu xi ∈ Y , ngược lại b_i = 0.
- Nói riêng, tập Y là tập rỗng tương ứng với dãy (0, 0, . . . , 0) và tập Y ≡ X tương ứng với dãy (1, 1, . . . , 1).
- Ta thấy b_i, i = 1, 2, . . . , n nhận giá trị nhị phân nên số tập con của tập X là 2^n.
- Nếu ta coi mỗi dãy nhị phân là biểu diễn nhị phân của một số nguyên không âm thì mỗi tập con của tập X ứng với một số nguyên trong đoạn [0, 2^n – 1].

Bài toán: Hãy liệt kê mọi tập con của một tập hợp gồm n phần tử.

Ví dụ, các tập con của tập gồm 3 phần tử {1, 2, 3 } là:

{},

{1}, {2}, {3},

{1, 2}, {1, 3}, {2, 3},

{1, 2, 3}.

Chú ý:

Số tập con của một tập gồm n phần tử là 2^n, là rất lớn nếu n lớn.

Vì vậy, bài toán này chỉ có thể giải được nếu n nhỏ (n ≤ 20).

2. Một số thuật toán sinh các tập con của một tập hợp

- Thuật toán cộng một
- Thuật toán đệ quy
- Sử dụng BITMASKS
- Thuật toán mã Gray (phương pháp đệ quy, phương pháp tính nhanh bằng xor, phương pháp đảo bít)

2.1. Thuật toán cộng một

Biểu diễn dãy nhị phân của các tập hợp gợi ý cho ta một phương pháp đơn giản để sinh mọi tập hợp của n phần tử như sau:

1. Xuất phát từ tập rỗng, ứng với số k = 0 hay dãy nhị phân a^0 = (0, 0, . . . , 0);

2. Trong mỗi bước, số k được cộng thêm 1 và tìm các biểu diễn nhị phân tương ứng của nó. Ví dụ, 5 dãy nhị phân tiếp theo là:

a^1 = (0, 0, . . . , 0, 0, 1)

a^2 = (0, 0, . . . , 0, 1, 0)

a^3 = (0, 0, . . . , 0, 1, 1)

a^4 = (0, 0, . . . , 1, 0, 0)

a^5 = (0, 0, . . . , 1, 0, 1)

3. Dừng thuật toán khi k = 2^n – 1 hay khi dãy nhị phân là (1, 1, . . . , 1, 1).

Ta có thể tăng tốc độ của thuật toán dựa trên quan sát đơn giản: dãy nhị phân đứng sau có thể được sinh từ dãy nhị phân đứng trước bằng cách quy nạp.

Giả sử đã sinh được dãy nhị phân $a^i = (b_0, b_1, . . . , b_n)$, dãy a^{i+1} được tìm bằng cách:

1. Xét các bít b_j với j giảm dần, bắt đầu từ n.

2. Lặp, trong khi $j \geq 1$:

- nếu $b_j = 1$ thì đặt $b_j = 0$ và tiếp tục xét b_{j-1};
- nếu $b_j = 0$ thì đặt $b_j = 1$ và dừng vòng lặp.

Với n = 4, các dãy nhị phân sinh bởi thuật toán là:

0	(0, 0, 0, 0)	8	(1, 0, 0, 0)
1	(0, 0, 0, 1)	9	(1, 0, 0, 1)
2	(0, 0, 1, 0)	10	(1, 0, 1, 0)
3	(0, 0, 1, 1)	11	(1, 0, 1, 1)
4	(0, 1, 0, 0)	12	(1, 1, 0, 0)
5	(0, 1, 0, 1)	13	(1, 1, 0, 1)
6	(0, 1, 1, 0)	14	(1, 1, 1, 0)
7	(0, 1, 1, 1)	15	(1, 1, 1, 1)

2.2. Thuật *toán đệ quy*

Ta có thể liệt kê mọi dãy nhị phân độ dài n bằng thuật toán đệ quy:

```
procedure Attempt(i: Integer); {Thử các cách chọn b[i]}
var
  j: Integer;
begin
  for j := 0 to 1 do {Xét các giá trị có thể gán cho b[i], với mỗi giá trị đó}
    begin
      b[i] := j; {Thử đặt b[i]}
      if i = n then PrintResult {Nếu i = n thì in kết quả}
```

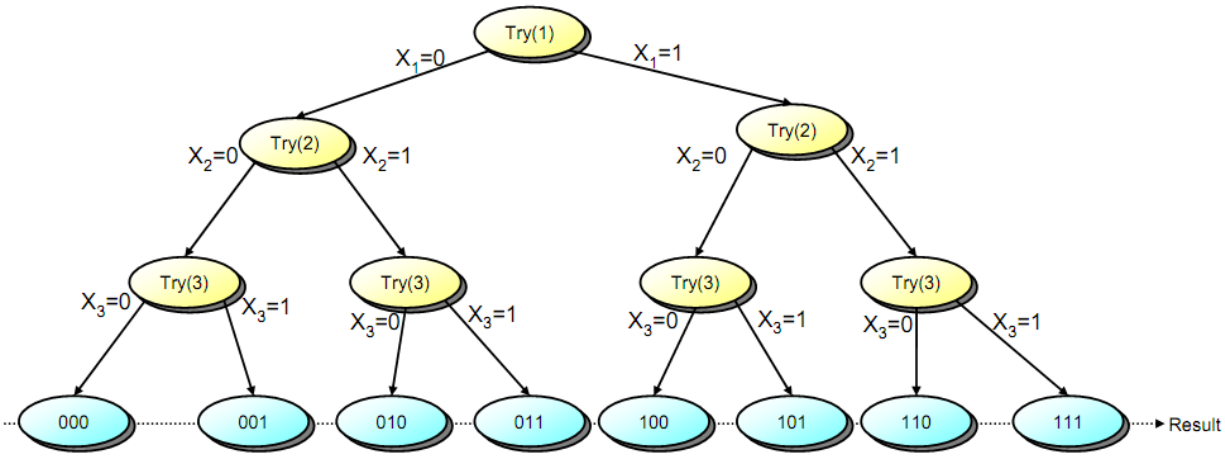
```
      else Attempt(i + 1);{Nếu i chưa phải là phần tử cuối thì tìm tiếp b[i+1]}

    end;

end;
```

Chương trình chính gọi Attempt(1)

Ví dụ, với n = 3, cây đệ quy tìm các dãy nhị phân được minh họa trong hình sau:



2.3. Sử dụng BITMASKS

Để biểu diễn trạng thái cho nhiều đối tượng, ta phải dùng nhiều biến để lưu lại trạng thái của chúng. Thay vào đó ta dùng duy nhất một biến để biểu trạng thái cho tất cả.

Ta bắt đầu làm rõ kĩ thuật này qua ví dụ sau:

Chẳng hạn ta có 3 bóng đèn. Mỗi bóng đèn có 2 trạng thái là bật hay tắt.

Để biểu diễn trạng thái của 3 bóng đèn, ta có thể dùng một dãy có 3 phần tử để biểu diễn, ví dụ *bool a[3]*. Nếu số bóng đèn ít, ta có thể dùng một cách khác để biểu diễn: ta dùng duy nhất một số nguyên để biểu diễn chúng. Giả sử hai bóng đầu bật và bóng

cuối tắt, ta có thể biểu diễn $110_{(cơ\ số\ 2)} = 6_{(cơ\ số\ 10)}$. Như vậy với một số 6 ta có thể biết được trạng thái hiện tại của ba bóng đèn, tương tự $7 = 111_2$ biểu diễn cả 3 bóng đều bật.

Trở lại với bài toán ban đầu: Hãy liệt kê tất cả các tập con (kể cả rỗng) của tập hợp đã cho. Xem như N phần tử là dãy N bit. Ta có thể biểu diễn tất cả các tập con bằng dãy N bit, giá trị 1 (hoặc 0) biểu diễn sự tồn tại (hoặc không tồn tại) của mỗi phần tử. Giá trị các dãy bit tương ứng từ 0 .. 2^n - 1. Để kiểm tra sự tồn tại của 1 phần tử trong dãy bit có giá trị X, ta sử dụng hàm GetBit như sau:

```
Function  GetBit(X,i:Word):Byte; //Hàm trả về giá trị 0 hoặc 1
Begin
     GetBit:=(X Shr i)AND 1;
End;
```

2.4. Thuật toán mã Gray

- Mã Gray n-bít là một danh sách gồm 2^n dãy nhị phân độ dài n trong đó dãy tiếp theo chỉ khác dãy đứng trước ở một bít. Mã Gray được phát minh năm 1947 bởi Frank Gray, một nghiên cứu viên ở Bell Labs, sau đó được công bố năm 1953 .
- Mã Gray còn được gọi là "mã nhị phân phản xạ" vì mã Gray n bít được xây dựng đệ quy từ mã Gray n – 1 bít bằng cách phản xạ mã này.
- Phản xạ: liệt kê các phần tử của danh sách dãy nhị phân theo thứ tự ngược lại.

Các bước cụ thể để sinh mã Gray n bít như sau:

1. Xuất phát từ mã Gray n – 1 bít là danh sách gồm $k = 2^{n-1}$ dãy nhị phân: $\alpha^1, \alpha^2, \ldots, \alpha^{k-1}, \alpha^k$
2. Phản xạ mã Gray này, tức liệt kê các dãy nhị phân của nó theo thứ tự ngược lại: $\alpha^k, \alpha^{k-1}, \ldots, \alpha^2, \alpha^1$

3. Đặt bít 0 lên trước các dãy trong danh sách ban đầu: $0\alpha^1$, $0\alpha^2$, . . . , $0\alpha^{k-1}$, $0\alpha^k$
4. Đặt bít 1 lên trước các dãy trong danh sách phản xạ: $1\alpha^k$, $1\alpha^{k-1}$, . . . , $1\alpha^2$, $1\alpha^1$
5. Ghép hai danh sách này lại sẽ thu được mã Gray n bít: $0\alpha^1$, $0\alpha^2$, . . . , $0\alpha^k$, $1\alpha^k$, $1\alpha^{k-1}$, . . . , $1\alpha^2$, $1\alpha^1$.

Ví dụ, mã Gray với n = 3 được sinh từ mã Gray với n = 2 như sau:

Mã 2-bít 00, 01, 11, 10

Mã phản xạ 10, 11, 01, 00

Đặt 0 lên trước mã ban đầu 000, 001, 011, 010

Đặt 1 lên trước mã phản xạ 110, 111, 101, 100

Ghép hai mã 000, 001, 011, 010, 110, 111, 101, 100

Mã Gray 1 bít là G_1 = (0, 1). Mã này có thể được sinh từ mã Gray 0 bít G_0 = ( ) chỉ gồm một dãy rỗng theo cách trên.

Trong quá trình sinh mã G_{n+1} từ G_n ta thấy một số tính chất sau:

- Nếu coi mỗi dãy nhị phân trong mã G_n là một số nguyên (trong cơ số 10) thì G_n là một hoán vị của dãy số (0, 1, . . . , $2^n - 1$).
- G_n được "nhúng" vào nửa đầu của G_{n+1}.
- Mỗi dãy của G_n chỉ khác với dãy đứng trước nó một bít.
- Dãy cuối cùng của G_n chỉ khác dãy đầu tiên một bít.

Ta có thể tìm mã G_{n+1} từ mã G_n bằng một thủ tục đệ quy.

Tuy nhiên, từ các tính chất của mã Gray ở trên, ta có thể tìm mã Gray bằng một thuật toán nhanh hơn dựa trên quan sát sau:

Chuỗi thứ i trong G_n là biểu diễn nhị phân của số (i/2) $\oplus$ i

trong đó i/2 là phép chia nguyên của i cho 2 và $\oplus$ là phép toán xor.

Nhắc lại: phép xor giữa hai bít a và b cho giá trị 1 nếu chỉ a hoặc b là 1.

Ta cũng có thể tìm chuỗi mã Gray thứ i + 1 từ chuỗi mã Gray thứ i dựa trên quan sát sau:

1. Nếu chuỗi thứ i có một số chẵn bít 1 thì ta đảo bít cuối cùng của nó sẽ có chuỗi thứ i + 1;

2. Nếu chuỗi thứ i có một số lẻ bít 1 thì ta tìm bít 1 ở bên phải nhất của chuỗi và đảo bít ở bên trái bít đó.

Sử dụng quy tắc đảo bít này, ta sinh được mã Gray G4 như sau:

$$000 \overset{(1)}{\Rightarrow} 001 \overset{(2)}{\Rightarrow} 011 \overset{(1)}{\Rightarrow} 010 \overset{(2)}{\Rightarrow} 110 \overset{(1)}{\Rightarrow} 111 \overset{(2)}{\Rightarrow} 101 \overset{(1)}{\Rightarrow} 100.$$

3. Duyệt bằng cách chia đôi tập hợp

Trong thực tế ta thường gặp dạng bài phải xét tất cả các cấu hình để có thể xác định cấu hình tối ưu. Độ phức tạp nếu như duyệt tổ hợp thông thường thường là $O(2^N)$

Với n cỡ 32 thì chi phí sẽ là 2^{32} ~ 4 tỷ (Máy tính loại trung bình có thể phải chạy mất vài phút), không thể đảm bảo chạy trong thời gian cho phép.

Có một phương pháp tối ưu hơn, tư tưởng của phương pháp này là ta sẽ chia tập hợp ban đầu {x1, x2,…xn} thành hai phần, mỗi phần có n/2 giá trị. Khi đó với mỗi phần ta có thể tiến hành duyệt toàn bộ riêng rẽ, độ phức tạp giảm xuống còn $O(2^{2/N})$, có thể đảm bảo chạy trong thời gian cho phép. Sau đó tìm cách tổ hợp kết quả của hai phần duyệt này với nhau thông qua mảng nhớ, hoặc có thể tiếp tục duyệt, QHĐ để tìm ra kết quả của bài toán ban đầu.

II. Bài tập

Bài 1: Tập con có tổng bằng K

Cho một tập S có N phần tử $\{a_1, a_2, a_3, \ldots a_N\}$ (N<=32) . Hỏi rằng có bao nhiêu tập con của S thỏa mãn tổng các phần tử của tập con đó là K.

Input

- Dòng thứ nhất ghi số N, K ($0 \le N \le 32$).
- Dòng thứ 2 ghi các số nguyên $a_1, a_2, a_3, \ldots a_N$ ($|a_i| \le 10^9$).

Output

- Gồm một số duy nhất là số tập con của S có tổng bằng K.

Ví dụ

Input	Output
6 10 2 2 3 3 3 1	4

Hướng dẫn:

Đây là bài toán NP, vì thế ta phải duyệt qua tất cả các tập con để tìm ra kết quả. Tập S có 2^N tập con, nếu duyệt qua bình thường thì ta tốn thời gian 2^N, con số này cũng khá cao, không thể chạy trong thời gian cho phép được.

Cải tiến:

- Ta chia tập S thành 2 tập S1 = $\{a_1, a_2, \ldots, a_{N/2}\}$ và S2=$\{a_{N/2+1}, a_{N/2+2}, \ldots, a_N\}$ hai tập này có số phần tử bằng nhau, nếu N lẻ thì S2 nhiều hơn 1 phần tử.
- Sinh tất cả các tập con của S1và S2.
- Sắp xếp các tập con của S2 theo thứ tự tăng dần theo giá trị tổng các phần tử của tập con đó.
- Ứng với mỗi tập con của S1 (giả sử tập con này là A có tổng là K1) ta dùng binary search để tìm ra tập con tương ứng của S2 có tổng là K-K1 (gọi tập con này là B). Lúc này ta có một tập con C = A U B của S thỏa tổng các phần tử là K. Cứ duyệt qua hết tập con S1 là ta có đáp án.

Độ phức tạp: Sinh S1, S2 mất $2*2^{(N/2)}$, mỗi tập con của S1, ta tìm một tập con của S2, thao tác này mất chi phí : $2^{(N/2)} *\log_2(2^{(N/2)}) = (N/2)*2^{(N/2)}$

Nhận xét: Với cách làm này thì ta có thể giải cho các trường hợp N<=32, nếu N lớn hơn cũng không khả thi vì bản chất bài này vẫn là NP.

Cài đặt tham khảo:

```
program Tong_bang_K;
const fi='sums.inp';
      fo='sums.out';
      maxn=34;
var a:array[1..maxn] of longint;
    n:integer;
    T,P,d:array[0..1000000] of longint;

    K:longint;
    ans:int64;
    dem1,dem2,dem1a:longint;

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

```
   close(f);
end;

procedure ketqua;
var g:text;
begin
   assign(g,fo); rewrite(g);
   writeln(g,ans);
   close(g);
end;

function GetBit(X:longint; i:longint):byte;
begin
   GetBit:=(X shr i) and 1;
end;

procedure chuanbi;
var j:longint;
    m1,m2,k:longint;
begin
    m1:=n div 2; dem1:=1 shl m1 -1;
    for j:=0 to dem1 do
       begin
           T[j]:=0;
           for k:= m1-1 downto 0 do T[j]:=T[j]+GetBit(j,k)*a[m1-k];
       end;
```

```
    m2:=n-m1; dem2:= 1 shl m2 -1;
    for j:=0 to dem2 do
       begin
           P[j]:=0;
           for k:= m2-1 downto 0 do P[j]:=P[j]+GetBit(j,k)*a[m1+m2-k];
       end;
end;
procedure sort(l,r:longint);
var i,j:longint;
    tg,x:longint;
begin
   i:=l; j:=r;
   x:=T[(i+j)div 2];
   repeat
      while T[i]<x do inc(i);
      while T[j]>x do dec(j);
      if i<=j then
         begin
             tg:=T[i]; T[i]:=T[j]; T[j]:=tg;
             inc(i); dec(j);
         end;
   until i>j;
   if i<r then sort(i,r);
   if l<j then sort(l,j);
end;
procedure nen;
var i:longint;
begin
```

```
    dem1a:=0;
    d[dem1a]:=1;
    for i:=1 to dem1 do
       if T[i]=T[i-1] then inc(d[dem1a])
       else begin inc(dem1a); d[dem1a]:=1; T[dem1a]:=T[i]; end;
end;
function check(x:longint):longint;
var dau,cuoi,giua:longint;

begin
    dau:=0; cuoi:=dem1a;
    while dau<=cuoi do
       begin
          giua:=(dau+cuoi)div 2;
          if T[giua]=x then exit(d[giua])
          else if T[giua]>x then cuoi:=giua-1
               else dau:=giua+1;
       end;
    exit(0);
end;
procedure xuli1;
var i:longint;
begin
     sort(0,dem1);
     nen;
     ans:=0;
end;
```

```
procedure xuli2;
var i:longint;
begin
    for i:=0 to dem2 do
       ans:=ans+check(S-P[i]);
end;

BEGIN
   nhapdl;
   chuanbi;
   xuli1;
   xuli2;
   ketqua;
END.
```

Bài 2: Tổng vector (http://vn.spoj.pl/problems/VECTOR)

Trong mặt phẳng tọa độ có N véc tơ. Mỗi một véc tơ được cho bởi hai chỉ số x và y. Tổng của hai véc tơ (x_i, y_i) và (x_j, y_j) được định nghĩa là một véc tơ $(x_i + x_j, y_i + y_j)$. Bài toán đặt ra là cần chọn một số véc tơ trong N véc tơ đã cho sao cho tổng của các vec tơ đó là véc tơ (U, V).

Yêu cầu: Đếm số cách chọn thoả mãn yêu cầu bài toán đặt ra ở trên.

Input

- Dòng thứ nhất ghi số N ($0 \le N \le 30$).
- N dòng tiếp theo, dòng thứ i ghi các số nguyên xi, yi lần lượt là hai chỉ số của véc tơ thứ i. ($|x_i|, |y_i| \le 100$).
- Dòng cuối cùng ghi số hai số nguyên U, V ($|U|, |V| \le 10^9$).

Output

- Gồm một số duy nhất là số cách chọn thoả mãn.

Ví dụ

Input	Output
4 0 0 -1 2 2 5 3 3 2 5	4

Hướng dẫn:

Nếu duyệt tổ hợp của N vector thì độ phức tạp của thuật toán là 2^N và chương trình sẽ không cho kết quả trong thời gian cho phép với cỡ N ≤ 32. Cách giải quyết như sau:

- Chia tập N vector thành 2 phần A và B, mỗi phần có N/2 véc tơ
- Duyệt tất cả các tập con của A, được 2^(N/2) véc tơ tổng, lưu vào một mảng P. Sắp xếp mảng P tăng dần theo chỉ số x rồi y
- Duyệt phần B, với mỗi véc tơ tổng (x,y) duyệt được của phần B, tìm xem có bao nhiêu véc tơ tương ứng (U-x,V-y) trong phần A. Ta có thể tìm kiếm nhị phân (do mảng P đã được sắp xếp)
- Độ phức tạp $O(2^{(N/2)} * (N/2))$

***Nhận xét*:**

Do kích thước bài toán: N ≤ 30, tọa độ của các véc tơ ($|x_i|$, $|y_i| \le 100$) nên tọa độ các véc tơ tổng (x,y) của phần A đều có $|x|, |y| \le 1500$. Ta có thể dùng kĩ thuật đánh dấu.

- Mảng F[-1500..1500, -1500..1500] kiểu integer , trong đó F[x, y] là số cách chọn để có vector tổng (x, y) trên tập A. Khởi tạo mảng F toàn 0.
- Duyệt tất cả các tập con của tập A, sau khi tính được vector tổng của mỗi tập con là (x,y) ta chỉ việc inc(F[x, y]).

- Duyệt tất cả các tập con của B, sau khi tính được vector tổng (x1, y1) của mỗi tập con, nếu (|U-x1|, |V-y1|<=1500 thì ta sẽ inc(Res, F[U - x1, V - y1]) với Res là kết quả của bài toán.
- Độ phức tạp $O(2^{(N/2)})$

Cài đặt tham khảo:

```
Program Tong_vec_to;

const   fi='vecto.inp';

        fo='vecto.out';

var

   a,b,x1:array[0..31] of integer;

   f,g:array[0..1 shl 15] of integer;

   t:array[-3000..3000,-3000..3000] of integer;

   n,res,p,kl,gt,X,Y:longint;


procedure nhapdl;

var   i:longint;

      ff:text;

begin

   assign(ff,fi); reset(ff);

   readln(ff,n);

   for i:=1 to n do readln(ff,a[i],b[i]);

   readln(x,y);

   close(ff);

end;



procedure nap1;
```

```
begin
   inc(t[kl,gt]);
end;

procedure thu1(k:longint);
var   i:longint;
begin
   if k>n div 2 then
   begin
      nap1;
      exit;
   end;
   for i:=0 to 1 do
   begin
      x1[k]:=i;
      kl:=kl+a[k]*i;
      gt:=gt+b[k]*i;
      thu1(k+1);
      gt:=gt-b[k]*i;
      kl:=kl-a[k]*i;
   end;
end;

procedure nap2;
begin
   inc(p);
   f[p]:=gt; g[p]:=kl;
end;
```

```
procedure thu2(k:longint);
var   i:longint;
begin
   if k>n then
   begin
      nap2;
      exit;
   end;
   for i:=0 to 1 do
   begin
      x1[k]:=i;
      kl:=kl+a[k]*i;
      gt:=gt+b[k]*i;
      thu2(k+1);
      gt:=gt-b[k]*i;
      kl:=kl-a[k]*i;
   end;
end;

procedure xuli;
var   i,j:longint;
begin
   kl:=0; gt:=0;
   thu1(1);
   p:=0;
   kl:=0; gt:=0;
   thu2(n div 2+1);
```

```
   for i:=1 to p do
      if (abs(x-g[i])<=1500) and (abs(y-f[i])<=1500) then
         inc(res,t[x-g[i],y-f[i]]);
end;

procedure ketqua;
var  ff:text;
begin
   assign(ff,fo); rewrite(ff);
   writeln(ff,res);
   close(ff);
end;

BEGIN
   Nhapdl;
   Xuli;
   Ketqua;
END.
```

Bài 3: Phân tập (http://vn.spoj.pl/problems/LQDDIV/)

Cho N người(2≤N≤32), mỗi người có một số a_i ($1 \le a_i \le 10^9$) được gọi là độ tin cậy. Cần phân chia N người này vào 2 tập sao cho:

- Mỗi người thuộc đúng một tập
- Chênh lệch tổng độ tin cậy của 2 phần là bé nhất

Input

- Dòng đầu chứa số nguyên N
- Dòng tiếp theo chứa N số : số thứ i là độ tin cậy của người thứ i

Output

- Ghi ra hai số u và v với u là độ chênh lệch nhỏ nhất và v là số cách phân chia

Ví dụ

Input	Output
5 1 5 6 7 8	1 3

Chú thích : Độ chênh lệch ít nhất của 2 phần là 1
Có 3 cách phân chia .3 cách phân chia nhóm 1 là (3,5) ,(1,3,4) và (1,2,5)

Hướng dẫn:

Tư tưởng của bài toán vẫn là chia đôi để duyệt. Ta sẽ chọn một số người ở lần duyệt thứ nhất và một số người ở lần duyệt thứ hai để cho vào "nhóm 1".

Giả sử tổng độ tin cậy lần duyệt thứ nhất là x, lần duyệt thứ hai là y.

Gọi S là tổng độ tin cậy của N người.

Tổng độ tin cậy của "nhóm 1" trong trường hợp này là x + y, và của "nhóm 2" là S - (x + y). Độ chênh lệch tạo thành là Abs(S - (x + y) - (x + y)) = Abs(S - 2*(x+y)). Cách giải quyết cụ thể như sau :

- Duyệt N div 2 người, các tổng độ tin cậy thu được lưu vào một mảng C có tối đa là 2^{16} phần tử.
- Để tiện cho tính toán sau này, ta sẽ tối ưu mảng bằng cách loại bỏ những tổng bằng nhau và chỉ giữ lại một, đồng thời dùng thêm 1 mảng để đếm số lần xuất hiện của tổng đó (cần sắp xếp lại mảng C trước khi tối ưu nó). Ví dụ D[i] là số lần xuất hiện của tổng độ tin cậy C[i].

- Duyệt phần còn lại, mỗi tổng độ tin cậy X sinh ra, ta sẽ tìm C[i] sao cho Abs(S - 2*(X+C[i])) nhỏ nhất có thể. Sau đó cập nhật kết quả tối ưu. Tìm kiếm nhị phân giá trị C[i] sẽ là rất hợp lí bởi mảng C đã được sắp xếp và kích thước của nó cũng khá lớn.

Bài 4: Cái túi (http://vn.spoj.pl/problems/DTTUI1)

Cây khế nhà Khánh rất sai quả nên có một con chim to to đến ăn. Ăn xong, chim chở Khánh ra đảo để trả công bằng vàng. Đảo có N cục vàng. Anh ấy muốn chuyển hết cả N cục vàng của mình về nhà. Nhưng khổ nỗi các cục vàng này lại có trọng lượng và kích thước khổng lồ. Khánh đem theo một cái túi ba trăm gang to đùng nhưng vẫn chưa chắc chứa hết đống vàng này. Khổ quá đi! Lấy cục nào, bỏ cục nào bây giờ! Các bạn giúp anh ấy tìm ra một cách chọn vàng để thu được giá trị lớn nhất mà vẫn không làm rách túi đi.

Input

- Dòng 1: Chứa 2 số nguyên: số cục vàng N ($1 \le N \le 40$) và tải trọng tối đa của túi M ($1 \le M \le 10^9$).
- N dòng sau: Mỗi dòng chứa 2 số nguyên: trọng lượng W_i và giá trị V_i của cục vàng thứ i ($1 \le W_i, V_i \le 10^8$).

Output

- Một số nguyên duy nhất là giá trị lớn nhất thu được.

Ví dụ

Input	Output
3 4 1 4 2 5 3 6	10

Hướng dẫn:

- Gọi S là tập N cục vàng đã cho; Ans là đáp số cần tìm.
- Chia tập S thành 2 tập S1, S2; tập S1 có N div 2 phần tử, tập S2 có (N- N div 2) phần tử
- Sinh ra mọi tập con của S1, S2 thỏa mãn điều kiện tổng khối lượng của tập con đó nhỏ hơn hoặc bằng M, lưu vào hai mảng tương ứng là A và B.
- Sắp xếp mảng B tăng dần theo w (khối lượng), nếu cùng w thì tăng dần theo v (giá trị). Sau đó để tiện tính toán ta có thể tối ưu mảng B như sau: trong số những phần tử có w

bằng nhau ta loại bỏ bớt, chỉ giữ lại một phần tử có v lớn nhất.

- Dựa vào B ta xây dựng một mảng maxV với ý nghĩa: maxV[i] là giá trị lớn nhất có thể thu được khi ta chọn các phần tử từ tập S2, với giới hạn trọng lượng tối đa cho phép là B[i].w
- Duyệt từng tập con của S1 được lưu trong mảng A: với mỗi tập con có trọng lượng w1 và giá trị v1, ta tìm trên mảng B tập con có trọng lượng w2 nằm trong đoạn [0..M-w1] mà có giá trị lớn nhất, giả sử là v2. Nếu v1+v2 > Ans thì cập nhật lại Ans.

Bài 5: 34 đồng xu (http://vn.spoj.pl/problems/COIN34)

Bạn có 34 đồng xu có giá trị như sau:

```
xu(1) có giá trị 2

xu(2) có giá trị 3

xu(3) có giá trị 5

for n = 4 to 34

    xu(n) có giá trị (xu(n-1) + xu(n-2) + xu(n-3))
```

Yêu cầu: Bạn hãy dùng nhiều đồng xu nhất để mua một món hàng có giá là X

Input

- Dòng đầu tiên là số test (không quá 10^3).
- Mỗi dòng tiếp theo chứa một số nguyên X ($1 \le X \le 2.10^9$).

Output

- Với mỗi test, in ra "Case #" + số hiệu test + ": " + số lượng lớn nhất đồng xu cần dùng. Nếu không có cách nào để đạt giá trị X thì in ra -1.

Ví dụ

Input	Output
4 1	Case #1: -1 Case #2: 2

5 8 9	Case #3: 2 Case #4: -1

Hướng dẫn:

Bài tập này có nhiều điểm giống với bài VECTO, nhưng do dữ liệu đề bài nên ta sẽ chia các đồng xu đã cho thành 2 phần:

Phần 1: các đồng xu thứ 1, 2, …, 20

Phần 2: các đồng xu thứ 21, 22, …, 34

- Duyệt phần 1 được 2^20 "tổng", với mỗi tổng ta lưu số đồng xu nhiều nhất để đạt được tổng đó vào mảng F.
- Duyệt phần 2: duyệt tất cả các tổng, với mỗi tổng S được sinh ra, với mỗi test ta tìm số đồng xu nhiều nhất ở phần 1 mà tổng giá trị của chúng là (X-S), từ đó tìm được kết quả.

Bài 6: Nhà hàng Trung Quốc (http://vn.spoj.pl/problems/CHNREST)

Hàng năm vì muốn có không khí ấm cúng và cũng để tiết kiệm nên bạn thường tổ chức sinh nhật ở nhà. Tuy nhiên trước sinh nhật năm nay vài hôm bạn đã thi đậu vào đội tuyển tin học quốc gia. Đây là một sự kiện đặc biệt có ý nghĩa nên bạn quyết định mừng ngày sinh nhật của mình tại một nhà hàng Trung Quốc sang trọng và bạn tự nhủ lần này nhất định phải tiêu xài rộng tay hơn. Mọi việc chuẩn bị đã gần xong nhưng còn một vấn đề làm bạn khá nhức đầu, đó là làm sao chọn được những món ăn mà mọi người cùng thích.

Nhà hàng có M món ăn khác nhau và thú vị ở chỗ là mỗi món ăn rất nhiều nên có thể đủ cho bao nhiêu người cũng được, vì thế vấn đề là gọi món nào chứ không phải mỗi món gọi bao nhiêu. Có tất cả N người đến dự tiệc sinh nhật (bao gồm cả bạn trong đó). Bạn đã tìm hiểu được danh sách những món ăn yêu thích của từng người và bạn muốn rằng đối với mỗi người phải có ít nhất 2 món mà họ thích. Tuy nhiên sau khi ăn xong còn nhiều tiết mục hấp

dẫn khác nên bạn cũng muốn rằng bất kỳ ai cũng không có quá 2 món ăn yêu thích trong danh sách được đặt trước. Và vấn đề cuối cùng, đây là tiền của bố mẹ nên cũng không nên tiêu xài quá đáng.

Yêu cầu: Hãy cho biết số tiền ít nhất phải trả để gọi một thực đơn thỏa mãn các yêu cầu trên.

Input

- Dòng đầu tiên chứa hai số M, N (M ≤ 30, N ≤ 10)
- Dòng thứ hai chứa M số P_i là giá của món thứ i.
- Trong N dòng cuối cùng, dòng thứ k ghi danh sách các món yêu thích của người thứ k.

Output

- Gồm một số duy nhất là kết quả của bài toán, hoặc in ra -1 nếu không có cách gọi món nào thỏa mãn.

Ví dụ

Input	Output
5 3 100 150 300 425 200 1 2 4 1 3 4 5 1 4 5	450

Hướng dẫn:

Theo đề ra thì mỗi người thích đúng 2 món trong thực đơn được chọn. Duyệt với m div 2 món, với mỗi tổ hợp ta có dãy A1, A2, A3, ..., An là số các món yêu thích của N người. Tương tự với lần thứ 2 là B1, B2, B3, ..., Bn. (Với Ai, Bi <= 2 và Ai + Bi = 2). Chúng ta tìm cách mã hóa dãy A, B thành số tự nhiên để tiện cho việc lưu trữ và tính toán.

Lời kết

Với kinh nghiệm còn hạn chế, bài viết mới chỉ dừng ở mức độ tập hợp một số bài toán có thể giải quyết bằng phương pháp duyệt chia đôi tập. Một số bài chưa đi sâu phân tích cụ thể để đưa ra thuật toán chi tiết, cũng như chưa tập hợp đầy đủ các bài toán có thể giải bằng phương pháp này. Bên cạnh đó còn vấn đề cần phải có một phương pháp truyền đạt như thế nào để học sinh tích cực, chủ động lĩnh hội được các kiến thức chuyên sâu. Trong bài viết cũng chưa có điều kiện thảo luận về vấn đề này. Tôi hy vọng nhận được sự giúp đỡ, góp ý của đồng nghiệp để bài viết được hoàn thiện hơn, góp phần nhỏ cho công tác bồi dưỡng học sinh giỏi.

Tôi xin chân thành cảm ơn!

Tài liệu tham khảo

1. Cấu trúc dữ liệu và giải thuật – Lê Minh Hoàng (ĐHSP Hà Nội)

2. Tài liệu tập huấn phát triển chuyên môn giáo viên Tin học - Nhiều tác giả

3. Thuật toán quay lui – Lê Sỹ Hùng (Hương Sơn – Hà Tĩnh)

4. Tài Liệu sách giáo khoa chuyên tin tập 1,2 - Nhiều tác giả

5. VNOI - Olympic tin học Việt Nam

6. Website: http://vn.spoj.pl