Mục lục

Phần 1: Quy hoạch động (qhđ)

I. Dãy con đơn điệu của dãy số và ứng dụng.

1. Dãy con đơn điệu tăng dài nhất:

Bản dễ : http://vn.spoj.com/problems/LIQ

Bản khó : http://vn.spoj.com/problems/LIS

Cách 1:

+Mảng qhđ F[i] với i = (1 → n):độ dài dãy con tăng dài nhất có phần tử cuối cùng là phần tử thứ i.

+Cơ sở qhđ F[0] = 0 với a[0] = –oo;

+Công thức qhđ F[i] = max(F[j] + 1)

với j = (0 → i – 1) và a[j] > a[i];

+Kết quả: res = max(F[i]) với i = (1 → n);

+ĐPT: $O(n^2)$.

Cách 2:

+Mảng qhđ h[i] với i = (1 → res): -F[h[i]] = i;

-a[h[i]] nhỏ nhất có thể.

Ví dụ: n = 5;

a: 2 1 3 4 1

F: 1 1 2 3 1

mảng h sau khi tính xong: 5 3 4

+Cơ sở qhđ h[1] = 1; res = 1;

+Xét i chạy từ 2 → n ta có:

- Mảng h đã được xây dựng cho dãy số từ 0 → i – 1;
- Để tính F[i] ta phải tìm 1 chỉ số j từ 1 → i – 1 sao cho a[i] > a[j] và F[j] lớn nhất, do tính chất của dãy h nên ta sẽ chặt nhị phân trên dãy h để tìm 1 chỉ số d sao cho a[h[d]] < a[i] ≤ a[h[d + 1]]; Lúc đó F[i] = d + 1;

Mô tả thuật toán:

```
h[1] := 1;
res := 1;
for i := 2 to n do
  begin
    d := 0;
```

```
            c := res + 1;
            while d + 1 <> c do
              begin
                    m := (d + c) div 2;
                    if a[i] <= a[H[m]] then c := m
                    else d := m;
              end;
            h[d + 1] := i;
            if d = res then inc(res);
          end;
```

NOTE: Câu lệnh : if a[i] <= a[H[m]] then c := m else d := m;

dấu <= sẽ tùy trường hợp và thay đổi, cụ thể:

tìm dãy con tăng: <=

tìm dãy con giảm: >=

tìm dãy con ko giảm: <

tìm dãy con ko tăng: >

<dễ dàng cm đc>

2. Sequences:

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

+Mảng qhđ:

- T[i] độ dãy con tăng dài nhất từ 1 → i nhận a[i] làm phần tử cuối cùng
- S[i] độ dãy con tăng dài nhất từ n → i nhận a[i] làm phần tử cuối cùng
- 2 mảng trên tính bằng bài LIS

+Kết quả : res = max{2 * min(F[i], S[i]) – 1} với i = 1 → n

3. Chia dãy

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

+Kết quả bài toán: res = độ dài dãy con giảm dài nhất

+Chứng minh:

- Không có cách chia với số nhóm < res:
  - Giả sử dãy con giảm dài nhất là a[i.1];a[i.2]...a[i.res]. Do số nhóm < res nên theo nguyên lí dirichle phải có ít 2 số trong res số trên thuộc cùng 1 nhóm. Giả sử 2 số đó là a[i.h] và a[i.k] với i.h < i.k. Do 2 số trên thuộc 1 nhóm nên

a[i.h] ≤ a[i.k];mặt khác 2 số trên thuộc dãy giảm dần nên a[i.h] > a[i.k] → Vô lý

- Tồn tại cách chia với số nhóm = res:
  - Gọi F[i] là độ dài dãy con giảm dài nhất từ 1 → i với a[i] là phần tử cuối cùng (1 ≤ F[i] ≤ res). Tất cả những phần tử có cùng giá trị F[i] sẽ cùng thuộc 1 nhóm → có res nhóm với mỗi nhóm đều lập thành 1 dãy ko giảm (dễ dàng chứng minh được).

NOTE: Các trường hợp chia nhóm sao cho các phần tử trong nhóm lập thành: (sẽ có kết quả tương ứng là:)

- dãy tăng: độ dài dãy con không tăng dài nhất
- dãy giảm: độ dài dãy con ko giảm dài nhất
- dãy ko giảm: độ dài dãy con giảm dài nhất
- dãy ko tăng: độ dài dãy con tăng dài nhất

4. Búp bê nga.

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

+Áp dụng bài toán 4: Ta sắp xếp các doll theo chiều tăng dần của w và áp dụng bài toán 4 lên dãy h

+Do 1 con đặt vào trong 1 con khác khi cả w và h đều nhỏ hơn nên nếu có 2 con có w bằng nhau,ta phải sắp xếp sao cho con có h lớn hơn đứng trước để tránh trường hợp 2 con này đặt vào nhau

+Để sắp xếp được như trên và không TLE ta dùng thuật toán Quicksort lồng

+Mô tả thuật toán Quicksort lồng:

```
Operator   <(a, b: doll)c: boolean;
begin
        exit((a.w < b.w) or ((a.w = b.w) and (a.h > b.h)));
end;
Procedure qs(l, h: longint);
var       i, j: longint;
          tmp, m: doll;
begin
       if l >= h then exit;
       i := l;
       j := h;
```

```
        tmp := a[i + random(j – i + 1)];
        repeat
            while a[i] < tmp do inc(i);
            while tmp < a[j] do dec(j);
             if i <= j then
            begin
                m := a[i];
                a[i] := a[j];
                a[j] := m;
                inc(i); dec(j);
            end;
        until i > j;
        qs(l, j);
        qs(i, h);
end;
```

5. Wooden Sticks.

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

+Tương tự thuật toán MDOLLS.

6. Card Sorting.

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

+Nhận xét: Khi số màu bài bằng 1 thì có cách chuyển tối ưu là giữ nguyên 1 dãy con ko giảm dài nhất và chuyển những con bài còn lại. Hay kết quả trong trường hợp này:n*c-lmax với lmax = độ dài dãy con ko giảm dài nhất.

+Thuật toán: Ta lấy các hoán vị số màu.Với mỗi hoán vị ta sẽ tạo 1 dãy số mới b[i] = a[x[i]] * 1000 + y[i] với a[j] là thứ tự của màu j trong hoán vị đang xét; x[i], y[i] là màu và giá trị của con bài thứ i và áp dụng nhận xét trên để update kết quả.

7. Số lượng dãy con tăng.

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

+Áp dụng bản chất bài LIS.

8. Con voi.

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

+Áp dụng bài NTSEQ

II. Quy hoạch động cấu hình.

Những 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.

1. Số hiệu hoán vị.

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

+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 thõa mãn.
Ở đây có giá trị 1 và 2 thõa 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

2. Số hiệu chỉnh hợp.

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

+Mảng qhđ: F[i]:số chỉnh hợp chập i của n – k + i

```
F[0] := 1;
F[i] := F[i – 1] * (n – k + i);
```

+Làm tương tự như bài SHHV

+Dùng số lớn

3. Số hiệu tổ hợp.

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

+Mảng qhđ:C[i,j] là tổ hợp chập i của j

```
C[0, 0] := 1;
C[i, j] := C[i, j – 1] + C[i – 1, j – 1];
```

+Làm tương tự bài SHHV;

+Dùng số lớn.

4. Thứ tự từ điển.

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

+Mảng qhđ: F[i, j] (i, j ≤ 26):chỉnh hợp chập i của j

```
F[0, j] := 1;
F[i, j] := F[i – 1, j – 1] * j;
```

Do dữ liệu cho kết quả và số nhập vào ≤ 2.10^9 nên những F[i,j] > maxlongint ta có thể gán = +∞ để tránh dùng số lớn

+Biến đổi của bài SHCH

5. Chuỗi hạt.

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

+Mảng qhđ: F[i, j] (1 ≤ j ≤ n, j ≤ i ≤ 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.

6. Xâu nhị phân.

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

+Mảng qhđ: F[i, t](1 ≤ i ≤ n, 0 ≤ t ≤ 1): số xâu nhị phân độ dài i ko có quá k số 0 or 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 ≤ i ≤ 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 ≤ k thì mọi xâu nhị phân độ dài i đều là 1 xâu thõa 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].

7. SPBINARY2.

http://vn.spoj.com/problems/BINARY2
+Tương tự SPBINARY.

8. Tung đồng xu.

http://vn.spoj.com/problems/NKTOSS
+Mảng qhđ:F[i, j] ($k \le i \le n, 0 \le j \le 1$):số xâu nhị phân độ dài i có tối thiểu k chữ số 0 liên tiếp kết thúc bằng bit j
+F[k, 0] := 1; F[k, 1] := 0;
+F[i, 1] := F[i – 1, 0] + F[i – 1, 1];
F[i, 0] := F[i – 1, 0] + F[i – 1, 1] – F[i – k, 1] + $2^{(i-k-1)}$;
Giải thích:Tính F[i, 0]
Ta xét lần lượt các xâu độ dài i kết thúc bằng t chữ số 0 liên tiếp
Với $1 \le t \le k-1$ ta có số xâu thõa mãn là F[i – t, 1]
Với $i > t \ge k$ ta có số xâu thõa mãn là $2^{(i-t-1)}$
Với t = i số xâu thõa mãn là 1
⇨ F[i, 0] = $\sum_{t=1}^{k-1} F[i-t,1] + 2^0 + .. + 2^{i-k-1} + 1$
=$\sum_{t=1}^{k-1} F[i-t,1] + 2^{i-k}$
Vậy F[i – 1, 0] = $\sum_{t=1}^{k-1} F[i-t-1,1] + 2^{i-k-1}$
⇨ F[i, 0] = F[i – 1, 0] + F[i – 1, 1] – F[i – k, 1] + 2^{i-k-1};

9. Dãy số Catalan.

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

+Xét lưới vuông (n + 1) x (n + 1). Một quy tắc đi thõa mãn:

- Xuất phát từ ô (1, 1) đến ô (n + 1, n + 1)
- Mỗi bước đi chỉ được di chuyển đến ô kề cạnh bên phải hoặc bên dưới.
- Không được di chuyển qua ranh giới là đường chéo nối đỉnh trái trên và phải dưới của lưới
- Nếu quy định a[1, 1] = 0 và

  a[i + 1, j] = a[i, j] + 1, a[i, j + 1] = a[i, j] – 1.

- Dễ thấy mỗi cách đi từ ô (1, 1) đến ô (n + 1, n + 1) là 1 dãy catalan tương ứng.

+Mảng qhđ:

- F[i, j]: số cách để đi từ ô (i, j) đến ô (n + 1, n + 1)

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

+Giả sử tại ô (u,v) ta di chuyển xuống ô phía dưới thì cách đi này sẽ có số thứ tự lớn hơn cách đi từ ô (u,v) di chuyển sang ô bên phải (với u > v). Do đó ta chỉ quan tâm đến những ô (u, v) mà tại đó thự c hiện di chuyển xuống ô phía dưới, ta cộng vào s thêm F[u, v + 1].

+Kết quả số thứ tự của dãy catalan tương ứng với cách đi là s + 1.

+Cho số tìm dãy thì làm ngược lại.

10. Pilots.

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

+Ta nhận thấy cách chọn lái chính và thư kí là 1 xâu ngoặc đúng.s[i] = '(' thì người i đc chọn làm trợ lý,ngược lại người i đc chọn làm lái chính.

+sử dụng bảng số như trên ta gọi f[i,j] là chi phí ít nhất để tạo 1 xâu ngoặc độ dài j + i – 1

+f[i, j] := min(f[i – 1, j] + b[j + i – 1], f[i, j – 1] + a[j + i – 1]) với a,b là chi phí trả cho lái chính và trợ lý

11. Dãy ngoặc.

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

+Tương tự bài catalan.

12. 0 0 Pairs.

http://vn.spoj.com/problems/M00PAIR

+Mảng qhđ F[i,1]: số cặp 01 có trong xâu sau i lần biến đổi

F[i,0]: số cặp 00 có trong xâu sau i lần biến đổi

+F[1, 1] := 1;

F[1, 0] := 0;

+F[i, 1] := F[i – 1, 0] + 2^{i-2};

F[i, 0] := F[i – 1, 1]

với i ≥ 2

13. Suft Permutation

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

+Mảng qhđ:F[i, j]:số hoán vị độ dài j bắt đầu = i và là sóng tăng

G[i, j]:số hoán vị độ dài j bắt đầu =i và là sóng giảm

+F[i, j]=$\sum_{t=i}^{j-1} G[t, j-1]$ với 1 ≤ i ≤ j ≤ n;

⇨ F[1,j] := $\sum_{t=1}^{j-1} G[t, j-1]$

F[i, j] := F[i – 1, j] – G[i – 1, j – 1] với 2 ≤ i ≤ j

G[i, j] := $\sum_{t=1}^{i-1} F[t, j-1]$ với 1 ≤ i ≤ j ≤ n;

⇨ G[j, j] := $\sum_{t=1}^{j-1} F[t, j-1]$

G[i, j] := G[i + 1, j] – F[i, j – 1] với 1 ≤ i < j.

+Áp dụng quy tắc ta sẽ tìm hoán vị độ dài n có thứ tự từ điển là k như sau:

-Tìm a[1]: Xét i từ 1 đến n nếu có k > f[i, n] + g[i, n] thì k := k – f [i, n] – g[i, n] còn không thì a[1] := i (break)

-Tìm a[2]: Xét i từ 1 đến n

nếu i < a[1] thì dãy này là sóng giảm,ta sẽ xét cho f[i, n – 1] k > f[i, n – 1] thì k := k – f[i, n – 1] còn không thì a[2] := i;

nếu i > a[1] thì sóng này là sóng tăng,ta sẽ xét cho g[i, n – 1] k > g[i, n – 1] thì k := k – g[i, n – 1] còn không thì a[2] := i;

-Tìm a[i]: i từ 3 → n – 1: do tính đơn điệu của sóng nên ta sẽ biết tiếp theo là sóng tăng hay sóng giảm từ đó xét các giá trị phù hợp với mảng f hay mảng g

+Mô tả thuật toán

fillchar(fr, sizeof(fr), true);

```
 for i := 1 to n do d[i] := i;
 for i := 1 to n do
 begin
        tam := f[i, n] + g[i, n];
       if k > tam then k := k – tam
       else break;
 end;
a[1] := i;
for i := 1 to n do
if i < a[1] then
begin
       if k > f[i, n – 1] then k := k – f[i, n – 1]
       else break;
end
else if i > a[1] then
begin
       if k > g[i – 1, n – 1] then k := k – g[i – 1, n – 1]
       else break;
end;
a[2] := i;
if a[2] > a[1] then h := 1 else h := 0;
fr[a[1]]  := false;
fr[a[2]]  := false;
for i := 1 to n do
 begin
       if i > a[1] then dec(d[i]);
       if i > a[2] then dec(d[i]);
 end;
for i := 3 to n – 1 do
 begin
       case h of
        0:begin
             for j := a[i – 1] + 1 to n do
               if fr[j] then
                begin
                     if k > g[d[j], n – i + 1] then
                           k := k – g[d[j], n – i + 1]
                    else break;
                end;
```

```
                a[i] := j;
                fr[j] := false;
            end;
        1:begin
            for j := 1 to a[i – 1] – 1 do
            if fr[j] then
            begin
                if k > f[d[j], n – i + 1] then
                    k := k – f[d[j], n – i  + 1]
                else break;
            end;
            a[i] := j;
            fr[j] := false;
        end;
      end;
    h := 1 – h;
    for j := 1 to n do
    if j > a[i] then dec(d[j]);
  end;
if n > 2 then
begin
  for i := 1 to n do
   if fr[i] then break;
  a[n] := i;
end;
```

14. Số nhị phân có nghĩa.

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

+Mảng qhđ:c[i, j] tổ hợp chập j của i.

c[0, 0] := 1

c[i, j] := c[i – 1, j] + c[i – 1, j – 1]

+Gọi x là xâu nhị phân của n và l là độ dài xâu x

Những xâu nhị phân có k số 0 có nghĩa thứ tự từ điển bé hơn xâu x(tức giá trị nhỏ hơn n) thuộc 1 trong 2 TH sau:

TH1:độ dài xâu bé hơn l:có $\sum_{t=1}^{l-2} c[t,\ k]$ xâu

TH2:độ dài xâu bằng l và có i – 1 bit đầu tiên giống xâu x và bit i = 0 ($2 \le i \le l - 1$; x[i] = 1):c[l – i, k – t – 1] với t là số bit 0 từ 1 → i – 1(lưu ý trong trường hợp $t \ge k$)

+kết quả bài toán là tổng các TH trên

+chú ý những trường hợp k = 1; k > l hay n = 0;

15. Closest number.
http://vn.spoj.com/problems/MCLONUM
+Ở cả 2 công việc ta đều tìm 1 kết quả sao cho có phần đầu xâu dài nhất giống với xâu a.

16. Whirligig number.
http://vn.spoj.com/problems/MZVRK
+Để tính tổng các số whirligig của các số trong đoạn [a, b] ta đi tìm tổng các số whirligig của các số trong đoạn [1, a – 1] và [1, b] rồi trừ cho nhau.
+Để tính tổng các số whirligig của các số trong đoạn [1,x] phân tích x thành hệ nhị phân được xâu s, n = length(s).
Số lượng số trong đoạn [1, x] có số whirligig dạng 10..0(i số 0):nếu s[n – i] = 1: x shr (i + 1) + 1
nếu s[n – i] = 0: x shr (i + 1)
dễ dàng c/m đc điều trên.
+Từ nhận xét trên ta suy ra kết quả bài toán.

17. MZVRK:
http://vn.spoj.com/problems/TPMZVRK
+Tương tự MVZRK.

18. Số không (I).
http://vn.spoj.com/problems/VN_ZR_I
+Với i chạy từ 1 → 32:Tính từ 1 đến n có bao nhiêu số có i chữ số 0 có nghĩa,áp dụng bài BINARY rồi update vào kết quả.

19. Số không (II).
http://vn.spoj.com/problems/VN_ZR_II
+Mảng qhđ:F[i]:số lượng chữ số 0 được tô màu đỏ và có i bit(bit đầu tiên =1)
F[i] =

$$F[i-1] + \sum_{t=1}^{i-1} 2^{i-t-2} * ((t-1) \operatorname{div} k + 1) + F[i-t-1]$$

mặc định $2^{-1} = 1$.
+Phân tích n thành nhị phân x, xét từ đầu đến cuối và update kết quả.

+Dùng qword để tránh tle.

20. Mật mã.
http://vn.spoj.com/problems/KPASS
+Dựa vào bài SPBINARY.

21. Số lượng bậc.
http://vn.spoj.com/problems/DEGREE
+Mảng qhđ: c[i, j] tổ hợp.

22. Mã số thuế.
http://vn.spoj.com/problems/TAXID
+nh := (p + 1) div 2;
+Mảng qhđ: F[i]: Số lượng mã số thuế độ dài i và được phân chia cho nhóm thuế nh
(tức là tồn tại 1 bit có c[nh – 1] ≤ gtri <c[nh])
chọn j từ 0 → c[nh] – 1 làm bit đầu tiên
Nếu j < c[nh – 1] thì F[i] := F[i] + F[i – 1];
Nếu j ≥ c[nh – 1] thì F[i] := F[i] + $c[nh]^{i-1}$;
+Cách tìm cấu hình cũng làm tương tự như trên.

23. Thứ tự dãy con.
http://vn.spoj.com/problems/C11SEQ2
+ Tương tự SHTH

24. Trật tự.
http://vn.spoj.com/problems/NK05ORDR
+Chặt nhị phân giá trị n,với mỗi giá trị n tìm vị trí của m.

25. Đếm số.
http://vn.spoj.com/problems/DEMSO
+Mảng qhđ:F[i, j, t] ($0 \le i \le 16, 0 \le j \le k, 0 \le t \le 9$):số lượng số có độ dài i có j vị trí xấu và bắt đầu = t

26. Counting Digits.
http://vn.spoj.com/problems/MDIGITS

27. Sums.
http://vn.spoj.com/problems/SPSUM

28. LQDNUMBERS
http://vn.spoj.com/problems/LQDNUMS

29. Tresnja.
http://vn.spoj.com/problems/TRES

30. Clear Numbers.
http://vn.spoj.com/problems/CLEAR
+Nhận xét:1 số là số rõ ràng khi và chỉ khi tổng chính phương của 1 số là rõ ràng.
+Xây dựng 1 mảng d[0..16*9*9]:nhận giá trị 0 hoặc 1,bằng 1 nếu số đó là 1 số rõ ràng.
+F[i, j] ($1 \le i \le 16$, $0 \le j \le 16*9*9$):số lượng số độ dài i (có thể bắt đầu bằng 0) và có tổng chính phương = j.
+Qhđ tiếp để tìm kết quả.

31. Bulls and cows.
http://vn.spoj.com/problems/CTNBULLS

32. K-DIGITS.
http://vn.spoj.com/problems/DIGIT

III. Đệ quy có nhớ.

1. Xâu gần nhất

Đề bài:Cho 3 xâu X,Y,Z có độ dài cùng bằng n chứa các kí tự từ 'A' đến 'Z'.Ta định nghĩa khoảng cách D(X,Y) giữa 2 xâu X,Y là tổng số cặp kí tự khác nhau trong 2 xâu,cụ thể:

D(X,Y)=$\sum_{i=1}^{n} Di(Xi,\ Yi)$

trong đó Di(Xi, Yi) = $\begin{cases} 0\ Nếu\ Xi = Yi \\ 1\ Nếu\ Xi \neq Yi \end{cases}$

Ví dụ:X='ABAB';Y='AAAB';Z='BBBB' khoảng cách 2 xâu X,Y bằng 1,khoảng cách Y và Z bằng 3.

Yêu cầu:Tìm xâu T sao cho khoảng cách lớn nhất của xâu T với X,Y,Z là nhỏ nhất.

Dữ liệu CSTR.INP gồm 3 dòng ,mỗi dòng 1 xâu độ dài ko vượt quá 100

Kết quả:CSTR.OUT:xâu cần tìm,có nhiều kết quả in ra thứ tự từ điển nhỏ nhất.

Thuật toán:

+Nhận thấy xâu kết quả có kí tự thứ i giống với x[i], y[i], z[i] hoặc bằng 'A';

+Đệ quy try(i, i1, i2, i3: longint):xét đến kí tự i và độ chênh lệch giữa xâu đang xét vs 3 xâu lần lượt là i1, i2, i3.

+Đánh dấu trạng thái F[i, i1, i2, i3],nếu xét rồi,sau đó ta ko xét nữa.

Mô tả thuật toán

```
procedure  try(i, i1, i2, i3: longint);
var        ch: char;
           x1, x2, x3: longint;
begin
        if f[i, i1, i2, i3] then exit;
        if i = n then
        begin
                tam := max(i1, max(i2, i3));
                if tam < kq then
                begin
                        kq := tam;
                        res := xau;
```

```
            end;
            exit;
        end;
        if max(i1, max(i2, i3)) >= kq then exit;
        for ch := 'A' to 'Z' do
         if (ch='A') or (ch=x[i+1]) or (ch=y[i+1]) or (ch=z[i+1])
         then
         begin
                x1 := i1+ord(ch<>x[i+1]);
                x2 := i2+ord(ch<>y[i+1]);
                x3 := i3+ord(ch<>z[i+1]);
                 xau := xau+ch;
                 try(i+1,x1,x2,x3);
                 delete(xau,i+1,1);
         end;
         f[i,i1,i2,i3] := true;
end;
```

2. Rebuss

Đề bài:Khi một số phần chữ số trong đẳng thức đúng của tổng hai số nguyên bị mất (được thay bởi các dấu sao "*"). Có một câu đố là: Hãy thay các dấu sao bởi các chữ số để cho đẳng thức vẫn đúng.

Ví dụ bắt đầu từ đẳng thức sau:

```
 9334
  789
--------
10123       (9334+789=10123)
```

Các ví dụ các chữ số bị mất được thay bằng các dấu sao như sau:

```
 *3*4          hay          ****
  78*                        ***
10123                       *****
```

Nhiệm vụ của bạn là viết chương trình thay các dấu sao thành các chữ số để được một đẳng thức đúng. Nếu có nhiều lời giải thì đưa ra một trong số đó. Nếu không có thì đưa ra thông báo: "No Solution".

Chú ý các chữ số ở đầu mỗi số phải khác 0.
Dữ liệu vào trong file "REBUSS.INP": gồm 3 dòng, mỗi dòng là một xâu ký tự gồm các chữ số hoặc ký tư "*" . Độ dài mỗi xâu không quá 50 ký tự. Dòng 1, dong 2 thể hiện là hai số được cộng, dòng 3 thể hiện là tổng hai số.
Kết quả ra file "REBUSS.OUT": Nếu có lời giải thì file kết quả gồm 3 dòng tương ứng với file dữ liệu vào, nếu không thì thông báo "No Solution"

REBUSS.INP	REBUSS.OUT
*3*4	9394
78*	789
10123	10123

+Đệ quy có nhớ try(i, nho:longint):xét đến vị trí thứ i có số nhớ là nho.

3. Tập số

Đề bài:Cho số nguyên không âm n ở hệ cơ số 10, không chứa các số 0 không có nghĩa ở đầu. Bằng cách xóa một hoặc một vài chữ số của n (nhưng không xóa hết tất cả các chữ số của n) ta nhận được những số mới. Số mới được chuẩn hóa bằng cách xóa các chữ số 0 vô nghĩa nếu có. Tập số nguyên D được xây dựng bằng cách đưa vào nó số n, các số mới khá nhau đã chuẩn hóa và khác n. Ví dụ, với n = 101 ta có thể nhận được các số mới như sau:

- Bằng cách xóa một chữ số ta có các số: 1 (từ 01), 11, 10;
- Bằng cách xóa hai chữ số ta có các số: 1, 1, 0;

Tập D nhận được từ n chứa các số {0, 1, 10, 11, 101}.
Yêu cầu: Cho số nguyên n và hai số nguyên không âm A, B. Hãy xác định số lượng số nằm trong [A, B] có mặt trong tập D được tạo thành từ n.

Dữ liệu: Vào từ file văn bản NUMSET.INP có dạng:

- Dòng 1: chứa số nguyên n;
- Dòng 2: chứa số nguyên A;
- Dòng 3: chứa số nguyên B.

Kết quả: Đưa ra file văn bản NUMSET.OUT một số nguyên – số lượng số tìm được.

+Đệ quy có nhớ try(i, p: longint;c1, c2: boolean):xét đến vị trí thứ i,vị trí thứ i-1 là ở vị trí p của xâu n;c1 đúng khi xâu đang xây dựng đã lớn hơn a và sai khi xâu đó đang bằng xâu a.c2 cũng tương tự.

4. BIỂU THỨC NGOẶC BẬC K

Đề bài:

Biểu thức ngoặc là xâu chỉ gồm các ký tự '(' hoặc ')'. Biểu thức ngoặc đúng và bậc của biểu thức ngoặc được định nghĩa một cách đệ qui như sau:

- Biểu thức rỗng là biểu thức ngoặc đúng và có bậc bằng 0.
- Nếu A là biểu thức ngoặc đúng có bậc bằng k thì (A) cũng là một biểu thức ngoặc đúng có bậc bằng k+1
- Nếu A và B là hai biểu thức ngoặc đúng và có bậc tương ứng là k1,k2 thì AB cũng là một biểu thức ngoặc đúng có bậc bằng max(k1,k2);

Ví dụ, '()(())' là một biểu thức ngoặc đúng có bậc bằng 2 còn '((()(()))' là một biểu thức ngoặc đúng và có bậc bằng 3.

Yêu cầu: Cho S là một xâu chỉ gồm các ký tự '(', ')' và số nguyên dương k, hãy đếm số xâu con khác nhau của S (xâu nhận được từ S bằng cách giữ nguyên xóa đi một số ký tự) là biểu thức ngoặc bậc k .

Input

-Dòng 1: xâu S có độ dài không vượt quá 1000 chỉ gồm các ký tự '(', ')'

-Dòng 2: số nguyên dương .

Output

- Ghi số lượng xâu con khác nhau của S là biểu thức ngoặc bậc k chia dư cho 111539786

+Đệ quy có nhớ try(i, p, c: longint;ok: boolean) xây dựng đc xâu độ dài i,vị trí thứ i – 1 là ở xâu p của S, c là chênh lệch giữa số ngoặc mở và đóng, ok=true nếu xâu đc xây dựng đã có bậc bằng k.

IV. Quy hoạch động trạng thái.

1. VOI06 – Chọn ô.

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

+Mỗi cột có bốn ô nên ta có thể sử dụng dãy bit độ dài 4 để lưu trạng thái ở cột i, với một bit k trong dãy bit, nếu k bằng 1 thì ta chọn vật thứ k tại cột i.

+Mảng qhđ:F[i, j](i ≤ n, 0 ≤ j ≤ 15):cách chọn tối ưu khi xét từ cột 1 → i và cột i đc chọn theo xâu nhị phân 4 bit trong cơ số 10 là j(ko có2 bit 1 gần nhau);

+F[i, j] = max(F[i – 1, t]) + c

Trong đó: j and t = 0;

c:tổng trọng số của các ô đc chọn trong cột i theo xâu nhị phân j.

2. Trò chơi trên ma trận.

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

+Làm tương tự bài QBSELECT.

3. TRIP.

http://vn.spoj.com/problems/LEM3

+Mảng qhđ:F[i, j](0 ≤ i ≤ 2^n – 1; 1 ≤ j ≤ n):chi phí ít nhất để đi được trạng thái i (i là cơ số 10 của xâu nhị phân mà bit thứ t bằng 1 biểu diễn thành phố t đã được đi) và đỉnh đến cuối cùng là j (bit thứ j=1);

+F[i, j] = min(F[u, t] + c[t, j]) với u=offbit(i, j); t là những thành phố được đi trong trạng thái u

+Kết quả min(F[1 shl n – 1, i]) với 1 ≤ i ≤ n.

4. Đàn bò hỗn loạn.

http://vn.spoj.com/problems/MIXUP2

+Làm tương tự LEM3

5. Cô gái chăn bò.

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

+Nhận xét: Do m*n ≤ 30 nên trong 2 số m,n luôn tồn tại 1 số ≤ 5,lấy số nhỏ hơn làm cột.

+Mảng qhđ: F[i, j] $1 \le i \le m$; $0 \le j \le 2^n - 1$:số cách xếp thỏa mản các hàng 1 → i và hàng i có trạng thái sắp xếp được mã hóa thành xâu nhị phân vs cơ số 10 bằng j.

+F[1, j] = 1 với $0 \le j \le 2^n - 1$;

+F[i, j] = $\sum F[i - 1, t]$ sao cho

u =j and t thì u ko có 2 bit 1 nào kề nhau

v = j or t thì v ko có 2 bit 0 nào kề nhau(trong n bit đầu tiên).

Với $i \ge 2$.

+Kết quả = $\sum_{j=0}^{2^n - 1} F[m, j]$

6. The problem for kid.

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

+Mảng qhđ: F[i] $0 \le i \le 2^n - 1$

giả sử i có k bit 1 ở các vị trí x1, x2 .. xk: thì F[i] là cách ghép tốt nhất các cô gái x1,x2..,xk với k chàng trai 1 → k.

+F[i]=max(F[j] + c[xt,k])

trong đó: j = offbit(i, xt) .

7. Quân mã.

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

+Làm tương tự QBSELECT.

V. Quy hoạch động trên cây

1. Tô màu nhỏ nhất.

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

+Ta chỉ tô màu các đỉnh với 3 màu.

+Mảng qhđ:F[i, j] ($1 \le i \le n$; $1 \le j \le 3$):chi phí ít nhất để tô các đỉnh thuộc gốc i và i đc tô bằng màu j.

+F[i, j] := j + ∑min(F[u, k]) với u là con của i;k <> j.

+Kết quả: min(F[goc, j]) với $1 \le j \le 3$.

Với goc la đỉnh đc chọn làm gốc của cây.

2. Cây P đỉnh.

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

+Xét cho cây gốc u:F[u, i]: Giá trị lớn nhất khi chọn i đỉnh của cây gốc u(có chọn cả u).

FF[u, ii]: Giá trị lớn nhất khi chọn ii đỉnh của cây gốc u (không chọn u).

+F[u, i] := FF[u, i – 1] + c[u].

3. Xây cầu.

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

+Xét thứ tự cha con trên cây.Cạnh i với 2 đỉnh u, v và u là cha của v.

+d[i]:lượng thời gian giảm được khi thay cạnh i bằng cầu

d[i] = con[v] * (n – con[v]) * l[i] * (vc – vp)/(vc * vp)

với con[v]:là số đỉnh của cây gốc v

+Sắp xếp mảng d giảm dần,chọn k cạnh đầu tiên.

4. Rải sỏi.

http://vn.spoj.com/problems/STONE1

+F[u]: số sỏi ít nhất để rải cây gốc u.

+Với mỗi đỉnh u có các đỉnh con là v1, v2, ..., vm.Ta sắp xếp các đỉnh trên theo thứ tự giảm dần của F và tính kết quả

5. Setnja

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

+F[i]:giá trị của tập hợp khi cuộc dạo chơi là xâu từ 1->i
C[i]:số cuộc dạo chơi phù hợp với xâu 1->i.
+F[0] := 1;
C[0] := 1;
+Bước thứ i là:
'L':F[i] = F[i – 1] * 2
C[i] = C[i – 1];
'R':F[i] = F[i – 1] * 2 + C[i – 1];
C[i] = C[i – 1];
'P':F[i] = F[i – 1];
C[i] = C[i – 1];
'*':F[i] = F[i – 1] * 5 + C[i];
C[i] = C[i – 1] * 3.
+Kết quả bài toán:F[n].
+Dùng số lớn.

6. Đường đi trên cây.
http://vn.spoj.com/problems/TREEPATH
+Tương tự bài SETNJA.

7. Tổng trọng số trên cây.
http://vn.spoj.com/problems/NTTREE
+Con[u]:số đỉnh của cây gốc u.
+Kết quả bài toán:
$\text{Res} = \sum_{i=1}^{m} \text{con[b[i]]} * (n - \text{con[b[i]]}) * \text{d[i]}$
m:số cạnh
a[i],b[i],d[i] lần lượt là 2 đỉnh và độ dài cạnh i,b[i] là con của a[i] theo thứ tự duyệt DFS.

8. Another Tree Problem.
http://vn.spoj.com/problems/MTREE
+F[u]:tổng trọng số đường đi của tất cả các đỉnh con của cây gốc u tới u.
+G[u]:tổng trọng số đường đi của các cặp đỉnh mà mỗi đỉnh thuộc 1 cây con khác nhau của u.
res=res + F[u] + G[u] với $1 \leq u \leq n$.

9. Đại diện hoàn hảo.

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

+F[u, 0]:số lượng đỉnh ít nhất của 1 tập thõa mãn 2 yêu cầu đầu của 1 đại diện hoàn hảo của cây gốc u mà u thuộc tập hợp đó.

+F[u, 1]:số lượng đỉnh ít nhất của 1 tập thõa mãn 2 yêu cầu đầu của 1 đại diện hoàn hảo của cây gốc u mà u không thuộc tập hợp đó và ít nhất 1 đỉnh là con trực tiếp của u thuộc tập hợp.

+F[u, 2]:số lượng đỉnh ít nhất để các cây con của u đều thõa mãn là 1 đại diện hoàn hảo và u ko nối trực tiếp với 1 đỉnh nào cả.

F[u, 0] := $\sum \min(F[v,0], F[v,1], F[v,2])$với v là con trực tiếp của u.

F[u,1] := $\sum \min(F[v,0], F[v,1])$+tam

tam=min(F[v,0]-min(F[v,0],F[v,1]));

F[u,2] := $\sum \min(F[v,0], F[v,1])$

Kết quả bài toán là min(F[gốc, 0], F[gốc, 1]).

10. Nhãn của cây.

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

+Tồn tại một cách đánh số đỉnh mà mỗi đỉnh chỉ có hai trọng số là 0 hoặc 1.

+Gọi:

- F[i] là trọng số nhỏ nhất khi đặt đỉnh i là số 1.
- F[i] = $\sum \min(F[j], w(i, j))$ Với j là con trực tiếp của i.

+Kết quả là F[1].

VI. Quy hoạch động với nhân ma trận

1. Lát gạch.
   http://vn.spoj.com/problems/LATGACH4

2. Xếp hình.
   http://vn.spoj.com/problems/FBRICK

3. Tính toán.
   http://vn.spoj.com/problems/C11CAL

4. Mực in.
   http://vn.spoj.com/problems/INKPRINT

5. Ant.
   http://vn.spoj.com/problems/PA06ANT

6. DK.
   http://vn.spoj.com/problems/C11DK2

7. Xâu Fibonacci 2.
   http://vn.spoj.com/problems/LQDFIBO2

8. Hoa hướng dương.
   http://vn.spoj.com/problems/QTLOVE2

VII. Một số bài toán quy hoạch động khác

1. Đường đi có tổng lớn nhất.
   http://vn.spoj.com/problems/QBMAX

2. Xâu con chung dài nhất.
   http://vn.spoj.com/problems/QBSTR

3. Chuỗi đối xứng.
   http://vn.spoj.com/problems/NKPALIN

4. Xếp hàng mua vé.
   http://vn.spoj.com/problems/NKTICK

5. Dãy con dài nhất có tổng chia hết cho K.
   http://vn.spoj.com/problems/QBSEQ

6. Dãy con chung không liền kề dài nhất.
   http://vn.spoj.com/problems/LNACS

7. Lát gạch.
   http://vn.spoj.com/problems/LATGACH

8. Phần thưởng.
   http://vn.spoj.com/problems/BONUS

9. Đi xem phim.
   http://vn.spoj.com/problems/VCOWFLIX

10. Steps.
    http://vn.spoj.com/problems/VSTEPS

11. Đếm chuỗi đối xứng.
    http://vn.spoj.com/problems/QBPAL

12. Lát gạch 3.
    http://vn.spoj.com/problems/M3TILE

13. Dãy số.
http://vn.spoj.com/problems/NKSEQ

14. Nối mạng.
http://vn.spoj.com/problems/NKCABLE

15. Hội trường.
http://vn.spoj.com/problems/NKREZ

16. BLAST.
http://vn.spoj.com/problems/MBLAST

17. Đoạn cao trào của bản nhạc.
http://vn.spoj.com/problems/THEME

18. Đổi tiền.
http://vn.spoj.com/problems/DTDOI

19. Chocolate Buying.
http://vn.spoj.com/problems/CBUYING

20. Đấu trường VM08.
http://vn.spoj.com/problems/LSFIGHT
+f[i, j] = true nếu người i và người j có thể thắng đc hết những người ở giữa theo chiều kim đồng hồ từ i → j.
+Một người i có khả năng chiến thắng nếu tồn tại 1 người J <> i mà (f[i, j]) and (f[j, i]) and (a[i, j] = 1);

21. Dãy con dài nhất.
http://vn.spoj.com/problems/NKMAXSEQ
+s[i] = $\sum_{i=1}^{n} a[i]$.
+d[i]($1 \le i \le n$):nhận 2 giá trị 0 và 1,bằng 1 nếu s[i] lớn hơn tất cả các số từ i + 1 → N
+Nếu kết quả là 2 chỉ số i, j thì j phải thõa mãn d[j] = 1
+chạy từ n về 1,với mỗi j tìm i xa nhất.

22. Dãy số.
http://vn.spoj.com/problems/AMSSEQ

23. Khối lập phương lớn nhất.
http://vn.spoj.com/problems/MAXCUB

24. Đếm dãy.
http://vn.spoj.com/problems/PBCDEM
+mảng qhđ:F[i,j]:số cách để phân tích số i thành j số x1, x2, ..., xj sao cho x1 + x2 +...+ xj = n và dãy x1,x2..,xj là 1 dãy số tăng dần.
Có 2 TH xảy ra:
Th1:x1≠1 hay x1, x2, ..., xj đều lớn hơn 1
x1 + x2 + ... + xj = i
⇔(x1 – 1) + (x2 – 1) + ... + (xj – 1) = i – j
⇨ số cách là F[i – j, j].
Th2:x1 = 1 tương tự ta có số cách là F[i – j, j – 1]
Vậy F[i, j] = F[i – j, j] + F[i – j, j – 1].

25. Đếm số Palindrome.
http://vn.spoj.com/problems/COUNTPL

26. Dãy con chung dài nhất.
http://vn.spoj.com/problems/LQDGONME
+F[i]:độ dài dãy con chung lớn nhất của m dãy mà số cuối cùng là i
F[0]=0;.
F[i]=max(F[j]+1) với số thứ tự của j nhỏ hơn số thứ tự của i trong cả m dãy.
Xét i theo thứ tự của dãy thứ nhất.

27. Xâu Fibonacci.
http://vn.spoj.com/problems/LQDFIBO

28. Đường đi trên lưới.
http://vn.spoj.com/problems/NKPATH

29. Bội số chung nhỏ nhất.
http://vn.spoj.com/problems/CTNOWN
+Ta xây dựng 1 mảng d[1..m] là các số nguyên tố nhỏ hơn n.

+F[i, j]: BCNN mà lớn nhất của các cách phân tích số i sao cho số cuối cùng có dạng $d[t]^k$ với t ≤ j.
+F[i, 0] = 1 với 1 ≤ i ≤ n.
F[0, i] = 1 với 1 ≤ i ≤ m.
+F[i, j] = max(F[i – h, j – 1]) với h = $d[i]^x$ và h ≤ i.

30. Bội số chung nhỏ nhất (Version 2).
http://vn.spoj.com/problems/OWN2
Bài CTNOWN nhưng dùng thêm số lớn.

31. Rectangles Perimeter.
http://vn.spoj.com/problems/MMAXPER

32. Help the PM!
http://vn.spoj.com/problems/HELPPM
Chặt từng cặp dòng và xử lý.

33. Bee walk.
http://vn.spoj.com/problems/MBEEWALK
+Đánh tọa độ cho các ô theo 1 trình tự phù hợp và lấy ô xuất phát có tọa độ (0,0);
+dx1:array[1..6] of longint = (0, –1, –1, –1, 0, 1)
dx2:array[1..6] of longint = (1, 0, –1, 0, 1, 1)
dy:array[1..6] of longint = (1, 1, 0, –1, –1, 0)
+Với mỗi ô tọa độ (x,y)
Nếu y mod 2=0 thì từ (x,y) đến đc 6 ô tọa độ (x + dx1[t], y + dy[t]); ngược lại (x + dx2[t], y + dy[t])
+Quy hoạch động: F[k, i, j] số cách để đi từ ô (0, 0) đến ô (i, j) bằng k bước.

34. Phân trang.
http://vn.spoj.com/problems/PTRANG

35. Mua vé tàu hỏa.
http://vn.spoj.com/problems/QBTICKET

36. BARICA
http://vn.spoj.com/problems/BARICAVN

+Nhận xét: 1 đường đi (x1, y1), (x2, y2), ..., (xm, ym) thõa mãn quy tắc thì luôn có (x1, x2, ..., xm) và (y1 , y2, ..., ym) là 1 dãy ko giảm.
+Sắp xếp các tọa độ từ theo tăng dần của x,những x bằng nhau thì sắp xếp theo y tăng dần.
+Duyệt dãy từ vị trí tọa độ thứ nhất đến n.
+Fx[i](0 ≤ i ≤ 10^5):giá trị max khi đỗ ở điểm hoành độ i
Fy[i]:giá trị max khi đỗ ở điểm tung độ i.

37. Bò Ba-ri.
http://vn.spoj.com/problems/BARIC
+F[i, j]: Sai số nhỏ nhất khi chọn ra j phần tử và phần tử thứ j là phép đo i.
+
F[i,1]=sum1(i) với sum1(i) = ∑2*abs(M[k] – M[i]) với k < i
với i < j F[i, j] = oo.
với i ≥ j F[i, j] = F[k, j – 1] + sum2(k, i) với 0 < k < i và
sum2(k, i) = abs(2 * M[t] – M[i] – M[k]) với k < t < i.
+Kết quả bài toán là k với k nhỏ nhất thõa mãn
tồn tại 1 số i để F[i, k] + sum3(i) ≤ E với
sum3(i)=∑2 * abs(M[k] – M[i]) vời k > i.

38. The country of heaven.
http://vn.spoj.com/problems/C11BC1
+Sắp xếp các thành phố theo chiều tăng dần của chỉ số ngôn ngữ.
+F[j]:tổng liên minh của tất cả các nhóm k thành phố.
S[j]:tổng liên minh của các nhóm k thành phố mà k thành phố này đều có chỉ số ngôn ngữ bằng nhau.
Chạy i từ 1→ n, j từ k về 1: F[j] := F[j]+F[j-1]*a[i].
Tương tự tính S[j] của từng nhóm.
Kết quả bài toán là F[k]-S[k].

39. Đồng hồ cát.
http://vn.spoj.com/problems/DHCAT

40. Fibonacci Sequence.
http://vn.spoj.com/problems/SPFIBO

41. Tháp Hà Nội.
http://vn.spoj.com/problems/CHNTOWER

42. Thang máy vũ trụ.
http://vn.spoj.com/problems/ELEVATOR

43. Bipalindrome.
http://vn.spoj.com/problems/MBIPALIN

44. Another Lucky Number.
http://vn.spoj.com/problems/MSE08G

45. Counting The Way of The Bracket Replacement.
http://vn.spoj.com/problems/MREPLBRC
+F[i, j]:số xâu ngoặc đúng khác nhau có thể tạo được từ xâu i → j.
+F[i, j] := get(i, j)*F[i + 1, j – 1] +
F[i + 1, k – 1]*get(i, k)*F[k + 1, j]
với i < k < j, j – i > 1.
F[i, j] = get(i, j) với j – i = 1.
Trong đó get(i, j) là số cặp ngoặc có thể tạo đc từ s[i], s[j] mà thuộc 3 cặp (), {}, [].

46. Minimum Rotation.
http://vn.spoj.com/problems/MINMOVE

47. Chuỗi hạt.
http://vn.spoj.com/problems/NKNL

48. Chuỗi hạt (hard version).
http://vn.spoj.com/problems/NKNL2

49. Cái túi 2.
http://vn.spoj.com/problems/DTTUI2
+Với mỗi vật ta phân tích số lượng A của vật đó thành dạng :A = $2^0 + 2^1 + 2^2 + \cdots + 2^d$+x
với x < 2^{d+1}
và tạo d vật mới có
w = 2^i*w0

v = 2^i*v0
với 0 ≤ i ≤ d
và 1 vật w = x*w0
v = x*v0
Sau đó qhđ như bài chiếc túi dạng đơn giản.
+Chứng minh:với các số $2^0, 2^1, \dots, 2^d$ và x ta luôn tạo ra được các số 1 → A.

50. Dãy con không giảm dài nhất.
http://vn.spoj.com/problems/VOLIS

51. ARITHMETIC PROGRESSION.
http://vn.spoj.com/problems/LEM5

52. Công viên Disneyland (version 1).
http://vn.spoj.com/problems/DISNEY1
+F[i, j](1 ≤ i ≤ j < n): chi phí ít nhất để 1 người ở i và 1 người ở j.
+j – i > 1: F[i, j] := F[i, j – 1] + a[j – 1, j];
J – i = 1 hoặc = 0: F[i, j] := F[k, i] + a[k, j] với k < i.

53. Công viên Disneyland (version 2).
http://vn.spoj.com/problems/DISNEY2
Tương tự DISNEY1

54. Chia đa giác.
http://vn.spoj.com/problems/NKPOLY

55. Trò chơi với bảng số.
http://vn.spoj.com/problems/C11GAME2

56. Trò chơi nhặt quà.
http://vn.spoj.com/problems/SCOLLECT

57. Nước đọng.
http://vn.spoj.com/problems/V11WATER

58. Dãy số vòng tròn.
http://vn.spoj.com/problems/PTQMSEQ

59. NERED.
http://vn.spoj.com/problems/MNERED

60. Nối điểm đen trắng.
http://vn.spoj.com/problems/BWPOINTS

61. Dãy số 1.
http://vn.spoj.com/problems/KSEQ1

62. Đổi chỗ.
http://vn.spoj.com/problems/VMSWAP

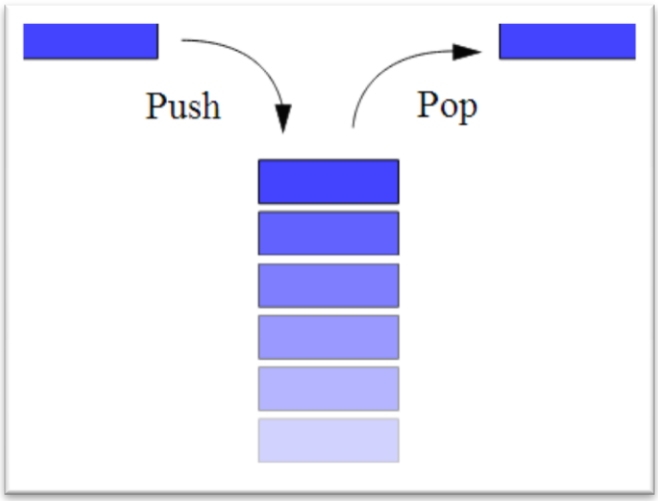
63. Khuyến mãi.
http://vn.spoj.com/problems/C11KM

Phần 2: Cấu trúc dữ liệu.

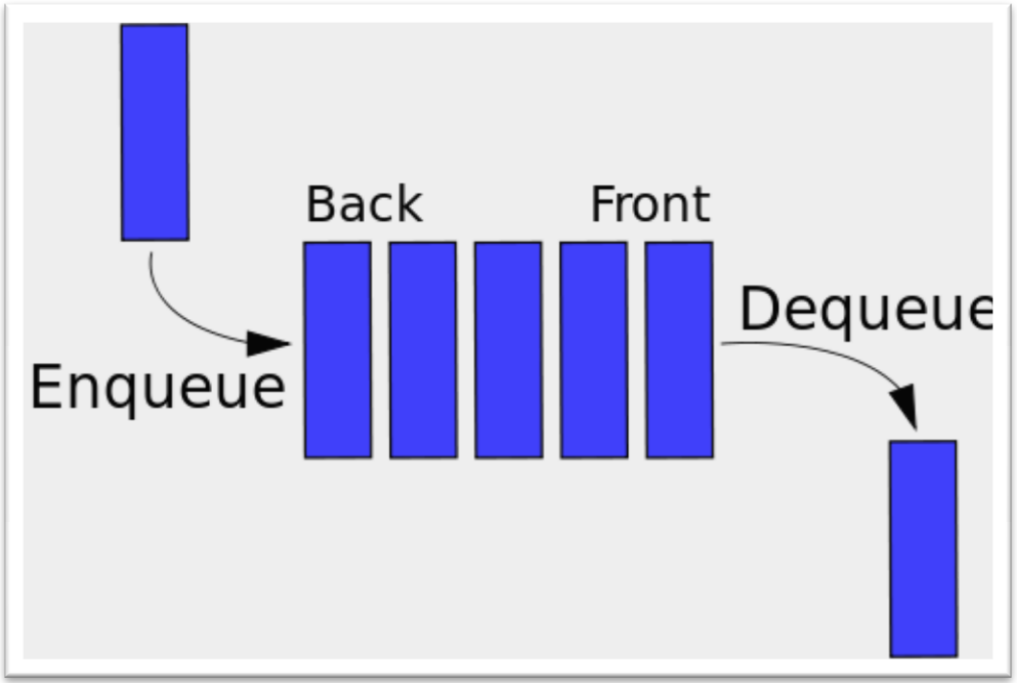
I. Ngăn xếp và Hàng đợi.

- Ngăn xếp và hàng đợi là danh sách, tức là một tập hợp các phần tử cùng kiểu có tính thứ tự.

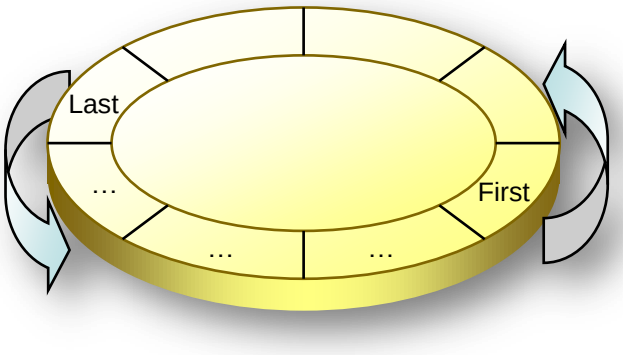
- Ngăn xếp hoạt động theo nguyên tắc "vào trước ra sau" nên nó có tên gọi là *danh sách kiểu LIFO(Last In First Out).*



- Hàng đợi hoạt động theo nguyên tắc "vào trước ra trước" nên nó có tên gọi là *danh sách kiểu FIFO(First In First Out).*



- Do hàng đợi thêm phần tử ở cuối danh sách và lấy phần tử ở đầu danh sách nên sẽ dễ bị tràn khi số lượng phần tử lớn. Vì vậy người ta mô tả hàng đợi bằng một danh sách vòng (cài đặt bằng mảng hoặc cấu trúc liên kết) coi như các phần tử được xếp quanh vòng theo một hướng nào đó.



Dùng danh sách vòng mô tả Queue

1. Mass of Molecule.

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

   +Bài này ta sử dụng ngăn xếp.

   +Ta xây dựng một ngăn xếp lưu các trọng lượng hóa chất.

   +Đi từ đầu về cuối xâu, nếu gặp:

   - Một nguyên tố hóa học thì chúng ta đẩy vào Stack khối lượng của nguyên tố đó.
   - Gặp một số thì chúng ta nhân giá trị ở đỉnh Stack với số đó.
   - Nếu gặp ngoặc mở thì chúng ta đẩy nó vào Stack.
   - Nếu gặp dấu ngoặc đóng thì chúng ta đẩy lần lượt các số ở đỉnh Stack và cộng chúng lại cho đến khi gặp dấu ngoặc mở. Sau đó đẩy đỉnh đó vào Stack.

   +Kết quả là tổng các phần tử còn lại trong Stack.

2. Những con đường quanh nông trang.

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

   +Bài này chúng ta hoàn toàn có thể giải quyết bằng phương pháp đệ quy. Nhưng phương pháp đó không nằm trong mục này, vì vậy chúng ta không đề cập nó ở đây. Ta sẽ tìm hiểu một cách khác là sử dụng hàng đợi.

   +Đầu tiên ta đẩy số lượng con bò vào hàng đợi.

   +Chúng ta sẽ lấy dần các phần tử ở đầu hàng đợi ra và thử chia chúng ra thành phần thỏa mãn yêu cầu bài toán, nếu chia được thì chúng ta đẩy chúng vào trong hàng đợi, còn không thì chúng ta sẽ chuyển sang phần tử tiếp theo.

3. Cho thuê xe.

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

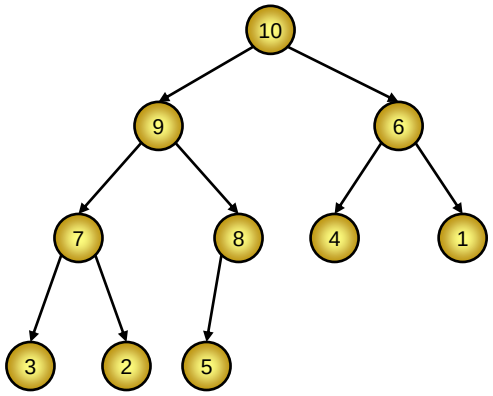
   +Kết hợp thuật toán Quy hoạch động và Ngăn xếp.

4. Đếm cặp.

   http://vn.spoj.com/problems/C11PAIRS

II. Hàng đợi ưu tiên.

- Hàng đợi ưu tiên (priority queue) là một cấu trúc dữ liệu quan trọng trong việc cài đặt nhiều thuật toán.
- Nếu còn chưa rõ hàng đợi ưu tiên xin đọc lại Tài liệu Giáo khoa Chuyên tin Tập 2.
- Có nhiều cách để cài đặt hàng đợi ưu tiên nhưng cách cài phổ biến nhất là Heap.
- Heap là một cây nhị phân gần hoàn chỉnh đặc biệt mà phần tử lưu tại mỗi nút luôn có độ ưu tiên lớn hơn hay bằng độ ưu tiên của các phần tử tại 2 nhánh con của nó.



1. Một chút về Huffman Tree.

http://vn.spoj.com/problems/HEAP1

+Xây dựng một heap min chứa tất cả các phần tử trong mảng ban đầu.

+Lặp:

- Lấy tổng 2 phần tử đầu heap và loại bỏ chúng khỏi heap.
- Đẩy tổng đó vào heap và chỉnh lại heap.

Cho đến khi heap chỉ còn một phần tử.

+Kết quả là phần tử cuối cùng còn lại trong heap.

2. Hàng đợi có độ ưu tiên.

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

3. KMIN.
   http://vn.spoj.com/problems/KMIN
   +Sắp xếp mảng a và mảng b tăng dần.
   +Chúng ta sử dụng một heap min chứa toàn bộ dãy tổng của các phần tử của dãy a với b[1].
   +Mỗi lần lấy một phần tử của heap ra, giả sử phần tử đó là a[i] + b[j], ta sẽ đếm cho đến khi nó bằng k, và ta sẽ thay phần tử đó bằng 1 phần tử mới là a[i] + b[j+1], chỉnh lại heap.

4. Phần tử trung vị.
   http://vn.spoj.com/problems/MEDIAN
   +Ta tạo hai heap, một heap min và một heap max, heap min lưu k div 2 phần tử cuối, heap max chứa k div 2 phần tử đầu.
   +Mỗi lần them một phần tử vào chúng ta chỉnh lại heap và cập nhật phần tử trung vị.

5. Ball game.
   http://vn.spoj.com/problems/BALLGAME
   +Đầu tiên sắp xếp lại theo thời gian của các lượt ném bóng.
   +Bài này ta dùng 1 heap min và 1 heap max lưu thời gian bóng ra khỏi lỗ, và 1 heap quản lý để lưu lại thời gian bong vào lỗ.
   +Mỗi lần đến lượt ném thì chúng ta thực hiện hai công việc:
   - Loại bỏ các phần tử ở heap quản lý có thời gian rời lỗ bé hơn thời gian ném, đồng thời them các phần tử đó vào heap min và heap max.
   - Thực hiện ném: Nếu ném từ bên trái thì dựa vào heap min để tìm ra ô có chỉ số nhỏ nhất, bên phải thì ngược lại. Sau đó loại bỏ ô này ra khỏi heap min và heap max, đồng thời them nó vào heap quản lý với 2 thông tin là chỉ số của ô và thời gian bóng rơi vào lỗ.

   +Nếu đến một lượt ném mà tất cả các ô đều có bóng thì trò chơi dừng lại.

6. Double Queue.
   http://vn.spoj.com/problems/MSE07B

7. Help Conan 13!
   http://vn.spoj.com/problems/MAXARR2

8. Help Conan 15!
   http://vn.spoj.com/problems/MAXARR3

Phần 3: Đồ thị.

I. Depth-First Search (DFS).

1. Quảng cáo.
http://vn.spoj.com/problems/ADS
+ Thực hiện thuật toán DFS để tính số vùng liên thông trên đồ thị (res).
+Kết quả là m + n – res.

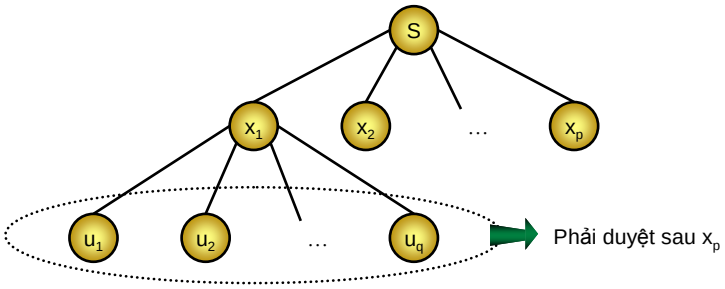
2. Dạo chơi đồng cỏ.
http://vn.spoj.com/problems/PWALK

3. Tổ chức đối lập.
http://vn.spoj.com/problems/V8ORG

4. Nút st – xung yếu.
http://vn.spoj.com/problems/STNODE
+DFS tìm 1 đường đi từ s → t.Chỉ có những đỉnh trên đường đi này mới có thể là 1 nút xung yếu:
dinh[count], dinh[count-1], ..., dinh[1].
với dinh[count] = s,dinh[1] = t.
u là số thứ tự của đỉnh dinh[u].
+Gán trọng số cho các đỉnh này f[u]:là đỉnh có số thứ tự nhỏ nhất mà u có thể đến được.
+Một đỉnh dinh[i] là nút xung yếu khi và chỉ khi
I <> count, i <> 1 và min(f[d[j]]) ≥ i với j = i + 1 → count.

II. Breadth-First Search (BFS).

Cơ sở của phương pháp cài đặt này là "lập lịch" duyệt các đỉnh. Việc thăm một đỉnh sẽ lên lịch duyệt các đỉnh kề nó sao cho thứ tự duyệt là ưu tiên chiều rộng (đỉnh nào gần S hơn sẽ được duyệt trước). Ví dụ: Bắt đầu ta thăm đỉnh S. Việc thăm đỉnh S sẽ phát sinh thứ tự duyệt những đỉnh (x_1, x_2, …, x_p) kề với S (những đỉnh gần S nhất). Khi thăm đỉnh x_1 sẽ lại phát sinh yêu cầu duyệt những đỉnh (u_1, u_2 …, u_q) kề với x_1. Nhưng rõ ràng các đỉnh u này "xa" S hơn những đỉnh x nên chúng chỉ được duyệt khi tất cả những đỉnh x đã duyệt xong. Tức là thứ tự duyệt đỉnh sau khi đã thăm x_1 sẽ là: (x_2, x_3…, x_p, u_1, u_2, …, u_q).



1. Bàn cờ thế.

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

+Mã hóa mỗi trạng thái bàn cờ băng dãy bit gồm 16 bit.

+Sử dụng thuật toán BFS để tìm đường đi ngắn nhất từ trạng thái đầu đến trạng thái cuối.

2. Ổn định.

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

3. Laser Phone.

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

+Loang đường đi ngắn nhất trên ma trận.

4. Bảo vệ nông trang.

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

+Với mỗi đỉnh ta loang đến những đỉnh có cùng độ cao đến được từ đỉnh đó và cập nhật chiều cao lớn nhất của những đỉnh kề các đỉnh đó đỉnh đó có thể tới được.

+Nếu chiều cao đó ≤ a[i, j] thì tăng số đỉnh đổi.

5. Tính toán lượng nước.

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

+Tiến hành nâng dần mực nước.

+Với mỗi mức nước ta loang từ ngoài vào để xem có bao nhiêu ô có nước thoát ra ngoài. Cộng vào kết quả những ô còn lại mà mực nước lớn hơn hoặc bằng độ cao của ô đó.

6. Mountain Walking.

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

+Chặt nhị phân độ chênh lệch.

+Loang xem có đường đi thỏa mãn độ chênh lệch đó hay không.

7. Các đại lý.

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

+Gọi F[u, v, k] là thời gian nhỏ nhất để người một đến đỉnh u, người hai đến đỉnh v với k = 0..1, k = 0 là đến lượt người một đi, k = 1 là đến lượt người hai đi.

+Loang tìm đường đi ngắn nhất để hai người gặp nhau.

8. HAOI 6000.

http://vn.spoj.com/problems/HAOI6000

9. Robocon.

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

+Loang lần lượt hai con robot theo từng lượt đi cho đến khi chúng gặp nhau.

10. Mã và tốt.

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

11. Nối điểm.

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

12. Cleaning Robot.
http://vn.spoj.com/problems/MCLEAN
+Dùng thuật toán BFS để tính đường đi ngắn nhất giữa mọi ô bẩn trên sàn nhà và với vị trí của robot.
+Quy hoạch động trạng thái như bài LEM3 để tính kết quả.

13. NKTable.
http://vn.spoj.com/problems/NKTABLE

14. Di chuyển.
http://vn.spoj.com/problems/TPMOVE

15. Chỉnh sửa ảnh.
http://vn.spoj.com/problems/MAR

16. Bộ sưu tập.
http://vn.spoj.com/problems/COLLECT

III. Cầu và khớp.

1. Tìm khớp và cầu.
http://vn.spoj.com/problems/GRAPH_

2. Điều kiện thời tiết.
http://vn.spoj.com/problems/WEATHER
+Ta tính xem với mỗi cạnh có bao nhiêu cặp thành phố mà muốn đi đến nhau bắt buộc phải đi qua cạnh này.Vì thế ta chỉ xét các cạnh là cầu.
+Với mỗi cạnh (u, v) là cầu,u là cha của low[v] ≥ number[v] thì số cặp thành phố cần đến cạnh này là count*(n – count) với count là số đỉnh thuộc cây DFS gốc v.

3. Thành phố trọng yếu.
http://vn.spoj.com/problems/CRITICAL
+res = tổng độ quan trọng của n thành phố.
+sl[v]:số lượng đỉnh của cây DFS gốc v.
+Xét các cặp cạnh (u,v) với u = parent[v]
Nếu low[v] ≥ number[u] (v là khớp or gốc DFS) thì ta update kết quả dựa vào sl[v].

IV. Thành phần liên thông mạnh và Song liên thông.

1. Tìm thành phần liên thông mạnh.
http://vn.spoj.com/problems/TJALG

2. Truyền tin.
http://vn.spoj.com/problems/MESSAGE
+Ta tìm các thành phần liên thông mạnh,mã hóa thành đồ thị thành 1 đồ thị khác,đồ thị này có các đỉnh tượng trưng cho các thành phần liên thông mạnh,đồ thị mới có cung (x, y) khi và chỉ khi đồ thị cũ có cung (u, v) với u thuộc thành phần liên thông mạnh x, v thuộc thành phần liên thông mạnh y.
+Xét trên đồ thị mới, số đỉnh không có cung đến là kết quả bài toán.

3. Biến đổi số.
http://vn.spoj.com/problems/NUMBER
+Ta tìm các thành phần liên thông mạnh,mã hóa thành đồ thị thành 1 đồ thị khác,đồ thị này có các đỉnh tượng trưng cho các thành phần liên thông mạnh,đồ thị mới có cung (x, y) khi và chỉ khi đồ thị cũ có cung (u, v) với u thuộc thành phần liên thông mạnh x, v thuộc thành phần liên thông mạnh y.
+Kết quả bài toán là số đỉnh không có cung ra(trừ đỉnh là thành phần liên thông mạnh chứa t).

4. Police.
http://vn.spoj.com/problems/NKPOLICE

5. Cho kẹo hay bị phá nào.
http://vn.spoj.com/problems/TREAT

6. Mạng máy tính an toàn.
http://vn.spoj.com/problems/SAFENET2
+Tìm thành phần song liên thông lớn nhất trên đồ thị.

V. Chu trình.

1. Bộ ba cao thủ.

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

+Coi mỗi cao thủ là 1 đỉnh trên đồ thị.

+Mỗi trận đấu có kết quả 1 được coi là 1 cạnh có hướng trên đồ thị, nối từ người thắng đến người thua.

+Tìm 1 chu trình có độ dài 3 trên đồ thị.

2. Kênh xung yếu.

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

+Tìm một chu trình bất kì trên đồ thị.

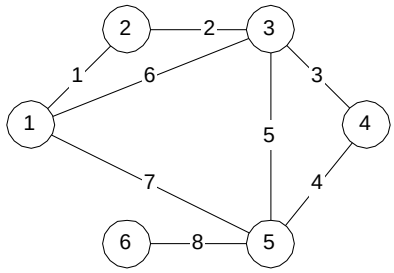
+Thử bỏ từng cạnh trên chu trình đó, kiểm tra đồ thị còn lại có còn chu trình hay không.

3. Biến đổi hoán vị.

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

VI. Chu trình Euler.

- Đường đi Euler là một đường đi xuất phát từ một đỉnh, đi qua tất cả các cạnh, mỗi cạnh đúng một lần.
- Chu trình Euler là một đường đi xuất phát từ một đỉnh, đi qua tất cả các cạnh, mỗi cạnh đúng một lần, và quay lại đỉnh xuất phát.



1. Người đưa thư.

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

+Tìm một chu trình Euler bắt đầu từ đỉnh 1.

2. Nối từ.

Đề bài :Để mở được cánh cửa bí mật ở một lâu đài cổ, các nhà khảo cổ học phải giải quyết vấn đề sau: Có một lượng lớn các mảnh nam châm ở trước cánh cửa, trên mỗi mảnh ghi một từ. Các mảnh cần được sắp xếp thành một dãy sao cho kí từ đầu tiên của từ trên mỗi mảnh (từ mảnh thứ 2 trở đi) phải giống kí tự cuối cùng của từ trên mảnh trước. Ví dụ mảnh có từ“acm” có thể xếp sau mảnh có từ “motorola”.

Cho biết các từ trên N mảnh nam châm, hãy giúp các nhà khảo cổ học kiểm tra xem có thể ghép các mảnh thành một dãy hay không để mở được cánh cửa bí mật.

Dữliệu:SECRET.INP

- Dòng đầu là số mảnh nam châm (N≤30000)
- N dòng tiếp theo, mỗi dòng một xâu kí tự mô tả một từ được viết trên một mảnh nam châm (các từ gồm các kí tự từ‘a’ đến ‘z’, các từ khác rỗng và có độ dài không quá 9).

Kết quả:SECRET.OUT

- Ghi trên dòng ðầu tiên, nếu có thể xếp được thì ghi ‘possible’, nếu không thì ghi ‘impossible’.

Hướng dẫn :Xây dựng đồ thị gồm 26 đỉnh, mỗi đỉnh đại diện cho một kí tự từ ‘a’ đến ‘z’. Mỗi từ ứng với một cạnh nối từ kí tự đầu tới kí tự cuối. Tìm chu trình Euler hoặc đường đi Euler trên đồ thị vừa tạo.

3. Exploring the maze.

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

4. Đầu và cuối của xâu.

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

+Coi mỗi đoạn đầu và cuối của mỗi xâu là một đỉnh.

+Sắp xếp chúng theo thứ tự và đánh số lại.

+Tìm đỉnh đầu là đỉnh cuối của một chu trình Euler hay một đường đi Euler.

VII. Sắp xếp Topo.

1. Đường đi dài nhất :

Đề bài :Cho đồ thị có hướng không chu trình, tìm đường đi dài nhất xuất phát từ 1 đỉnh và kết thúc tại 1 đỉnh khác (độ dài đường đi được tính bằng số cung trên đường đi.

Hướng dẫn :

+Dùng thuật toán sắp xếp Topo để đánh số lại đồ thị.

+Gọi F[i] là độ dài đường đi dài nhất kết thúc tại đỉnh có chỉ số i.

+Áp dụng quy hoạch động :

- F[n] = 0.
- F[i] = Max(F[j] + 1)

(i = n – 1 → 1) (j = i + 1 → n) (i, j) là cung.

2. Tổng số đường đi:

Đề bài:Cho đồ thị có hướng không chu trình, tính số đường đi từ đỉnh s đến đỉnh t.

Hướng dẫn:

+Dùng thuật toán sắp xếp Topo để đánh số lại đồ thị.

+Gọi num[u] là chỉ số của đỉnh u và q[i] là đỉnh có chỉ số i

Num[u] = i ⇔ q[i] = u.

+Gọi F[i] là số con đường xuất phát từ đỉnh i và kết thúc tại đỉnh t.

+Áp dụng tư tưởng QHĐ:

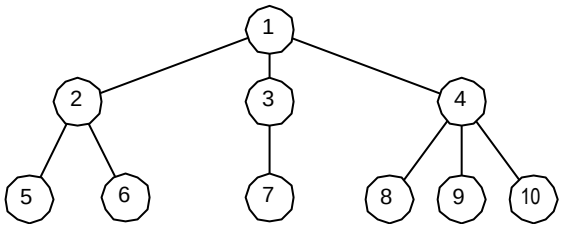
- F[q[t]] = 1.
- F[i] = ∑F[j] \ j = i + 1 → n và (q[i], q[j]) là cung.
  I = q[t] – 1 → q[s].

3. Vòng đua xe đạp.

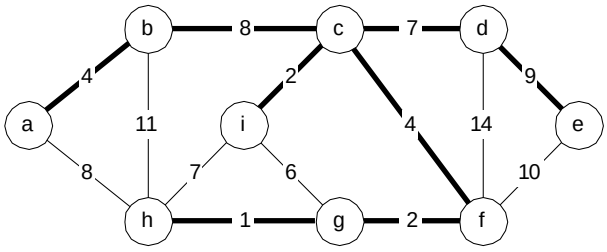
http://vn.spoj.com/problems/BIC

VIII. Cây khung nhỏ nhất.

- Cây là một đồ thị vô hướng, liên thông, không có chu trình đơn.



- Bài toán đặt ra là tìm một cây khung trên một đồ thị vô hướng liên thông có trọng số mỗi cạnh sao cho tổng trọng số trên mỗi cạnh trong cây khung là nhỏ nhất.



- Có hai thuật toán chính để tìm cây khung nhỏ nhất là:
  - Thuật toán Kruskal. O(m.log(m) + m.$\alpha(n)$)
  - Thuật toán Prim. O(n^2)

1. Xây dựng thành phố.
http://vn.spoj.com/problems/NKCITY
+Đây là một bài tập về cây khung cơ bản.
+Ta tìm cây khung nhỏ nhất của đồ thị.
+Thuật toán được sử dụng ở đây là Kruskal hoặc Prim.

2. Cây khung nhỏ nhất.
http://vn.spoj.com/problems/QBMST
+Ta sử dụng thuật toán Kruskal để tìm cây khung nhỏ nhất.

3. Các thùng nước.
http://vn.spoj.com/problems/IOIBIN

+Ta sử dụng thuật toán Dijoint Set Forest để hợp nhất các ống nước có đường nối lại.
+Lại sử dụng thuật toán trên để kiểm tra hai ống có lien thông hay không.

4. Tưới nước đồng cỏ.
http://vn.spoj.com/problems/FWATER
+Ta tạo thêm một đỉnh ảo để đại diện cho giếng nước, đỉnh này nối tới tất cả các cánh đồng với trọng số là chi phí xây dựng giếng tại cánh đồng đó.
+Vì đồ thị có số đỉnh nhỏ nên ta dùng thuật toán Prim để tìm cây khung nhỏ nhất trên đồ thị mới.

5. Robin.
http://vn.spoj.com/problems/C11BC2
+Đầu tiên ta bỏ hết tất cả các cạnh có trọng số là 2.
+Dùng thuật toán Dijoint Set Forest để hợp nhất các đỉnh thuộc các cạnh còn lại.
+Với mỗi truy vấn, nếu đường đi từ đỉnh x đến đỉnh y có một cạnh có trọng số 2 thì tương đương với việc hai đỉnh x và y phải thuộc hai vùng liên thông khác nhau (Dùng thuật toán Dijoint Set Forest để kiểm tra).

6. Vòng đua F1.
http://vn.spoj.com/problems/NKRACING
+Tìm cây khung lớn nhất của đồ thị.
+Do tính chất cứ thêm mỗi cạnh vào cây khung thì luôn tạo được một chu trình nên cạnh được thêm vào cây khung trên sẽ là cạnh nhỏ nhất trên chu trình đó.
+Vì vậy kết quả bài toán sẽ là tổng trọng số các cạnh còn lại không thuộc cây khung.

7. Các băng đảng.
http://vn.spoj.com/problems/PBCGANGS

8. Đế chế.
http://vn.spoj.com/problems/VNEMPIRE

9. Động viên đàn bò.

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

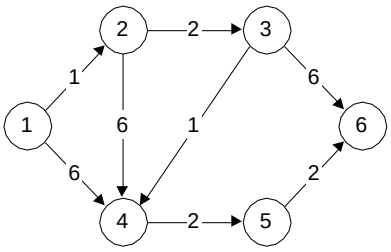
+Đổi trọng số các cạnh thành 2*w(u, v) + c(u) + c(v).

+Tìm cây khung nhỏ nhất trên đồ thị.

+Chú ý phải cộng thêm thời gian ở đỉnh xuất phát.

IX. Đường đi ngắn nhất.

- Cho đồ thị G = (V, E), mỗi cạnh e = (u, v) được gán cho một trọng số w(u, v).



- Bài toán đặt ra là :
  - Tìm đường đi ngắn nhất giữa hai cặp đỉnh (s, t).
  - Tìm đường đi giữa mọi cặp đỉnh (u, v).

- Có những thuật toán tìm đường đi ngắn nhất sau :
  - Trong trường hợp đồ thị không có chu trình âm, ta có thể sử dụng 1 trong 2 thuật toán sau:
    - Ford Bellman O(n^3) hay kết hợp Queue vòng để tăng tốc độ.
    - Dijkstra O(n^2).
    - Dijkstra + Heap O((m+n).log(n)).
  - Trong trường hợp đồ thị không có chu trình đơn, chúng ta có thể sử dụng thuật toán sắp xếp topo để tìm đường đi ngắn nhất. O(n+m).
  - Để tìm đường đi ngắn nhất giữa mọi cặp đỉnh, chúng ta sử dụng thuật toán Floyd để tìm. O(n^3).

1. Đến trường.
http://vn.spoj.com/problems/QBSCHOOL
+Thực hiện thuật toán Dijkstra kết hợp với kĩ thuật Quy hoạch động để tính số đường đi.
+Chú ý: Số đường đi có thể vượt 32-bit.

2. Vị trí tốt nhất.
http://vn.spoj.com/problems/BESTSPOT

+Sử dụng thuật toán Ford Bellman + Queue vòng để không bị TLE.

3. Nguy hiểm rõ rang trước mắt.

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

+Sử dụng thuật toán Floyd để tìm đường đi ngắn nhất giữa mọi cặp đỉnh.

+Làm theo yêu cầu của đề.

4. Tham quan thành cổ.

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

+Làm như bài VDANGER.

5. Mạng 3 đỉnh.

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

+Ta sẽ tìm một đỉnh trung gian nối với 3 đỉnh 1, 2, 3 sao cho khoảng cách giữa chúng là nhỏ nhất.

6. Xây dựng đường.

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

+Làm tương tự THREE.

7. CENTRE

http://vn.spoj.com/problems/CENTRE28

+Gọi:

- D1[i] = Độ dài đường đi ngắn nhất từ 1 → i.
- D2[i] = Độ dài đường đi ngắn nhất từ i → n.
- C1[i] = Số đường đi ngắn nhất từ 1 → i.
- C2[i] = Số đường đi ngắn nhất từ i → n.

+Dùng thuật toán Dijkstra + Heap để tìm các thông số trên.

+Một thành phố được nhận làm trung tâm kinh tế khi.

$$\begin{cases} D1[i] + D2[i] \neq D1[n] \\ D1[i] + D2[i] = D1[n] \\ C1[i] * C2[i] \neq C1[n] \end{cases}$$

8. Tăng tốc mạng máy tính.

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

+Gọi d[i, j] là độ dài nhỏ nhất khi đi từ 1 → i mà dùng j thiết bị.

+Dùng thuật toán Dijkstra để tính.

9. Traffic Network.
http://vn.spoj.com/problems/TRAFFICN
+Gọi d1[i] là đường đi ngắn nhất từ 1 → i, d2[i] là chi phí đường đi ngắn nhất từ i → n.
+Thử nâng cấp từng đường và cập nhật kết quả.

10. Revamping Trails
http://vn.spoj.com/problems/REVAMP
+Tương tự bài NETACCEL.

11. GONDOR.
http://vn.spoj.com/problems/GONDOR

12. VOI07 Robot cứu hỏa.
http://vn.spoj.com/problems/QBROBOT
+Ta tìm đường đi ngắn nhất từ 1 đến n (d).
+Chặt nhị phân w, sử dụng thuật toán BFS để xem khi nào thì độ dài từ 1 → n bằng d.

13. Mất điện.
http://vn.spoj.com/problems/PWRFAIL

14. Xúc xắc.
http://vn.spoj.com/problems/XUCXAC

15. Bin Laden.
http://vn.spoj.com/problems/BINLADEN

16. Số phụ thuộc.
http://vn.spoj.com/problems/SUMS
+Ta sẽ xây dựng đồ thị có a_1 đỉnh $\{0, 1, \ldots, a_1 - 1\}$.
+Gọi $d_i = a_i \bmod a_1$, thì ta sẽ tạo các cạnh một chiều nối các cặp đỉnh là $(0, d), (1, d_i + 1), \ldots, (a_1 - d_i - 1, a_1 - 1)$ với trọng số là a_i. Nếu có hai số a_i có cùng số dư khi chia cho a_1 thì ta chỉ giữ lại số nhỏ hơn.

+Gọi D[i] = độ dài đường đi ngắn nhất khi đi từ đỉnh 0 đến đỉnh i. Thì D[i] cũng là tổng nhỏ nhất thỏa mãn tổng đó có số dư khi chia cho a_1 là i.
+Từ đó ta suy ra nếu đặt j = b_i mod a_1 thì b_i là số phụ thuộc khi và chỉ khi $b_i \geq D[j]$.
+Chứng minh:

- Do mỗi cạnh sẽ có trọng số là số nhỏ nhất tạo được số dư là d. Theo tính chất mod thì:
  - a mod d + b mod d = (a + b) mod d.
- Từ đó suy ra tổng D[i] mod a_1 = i, và do thuật toán Dijkstra nên tổng này luôn nhỏ nhất.
- Nếu $b_i \geq D[j]$ thì $b_i = D[j] + a_1 * k$. ($k \in [0, +\infty]$).
  ➔ b_i là số phụ thuộc.

17. ELEVATOR II.
http://vn.spoj.com/problems/MELE2
+Làm tương tự SUMS.

18. Sắp xếp.
http://vn.spoj.com/problems/V8SORT
+Ta coi mỗi hoán vị là một đỉnh.
+Dùng thuật toán Dijkstra để tìm đường đi ngắn nhất từ trạng thái đầu về trạng thái cuối.