NGUYỄN XUÂN HUY

BRUTE FORCE

HÀ NỘI – 2021

Mục lục

LỜI NÓI ĐẦU

Vét cạn hay brute Force (nỗ lực thuần túy) là phương pháp thiết kế thuật toán trên cơ sở xét mọi tình huống của dữ liệu rồi chọn ra những tình huống đáp ứng được điều kiện của đề bài.

Hầu hết các bài trong bản này được minh họa cho phương pháp vét cạn, mặc dù chúng có thể được giải bằng một thuật toán tốt.

Bạn chỉ nên chọn cách giải vét cạn (brute force) khi bạn bị rơi vào một trong hai tình huống sau đây:

- *Bạn chưa tìm được thuật toán tốt.*
- *Bản thân bài đó không có thuật toán tốt. Ví dụ, đoán mật khẩu.*

Đề bài sẽ có thêm chú thích đặt trong ngoặc (thuật toán tốt).

Ví dụ Bống (quy hoạch động)

cho biết lời giải bài Bống được trình bày theo phương pháp vét cạn, nhưng thuật toán tốt cho bài này là quy hoạch động.

Để có thể vét cạn bạn cần tự trang bị các kiến thức của toán rời rạc, cụ thể là hoán vị, tổ hợp, chỉnh hợp,…

Permutations

Sáu hoán vị của 3 phần tử 1, 2, 3 được liệt kê theo trật tự từ điển như sau:

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

Hãy liệt kê các hoán vị của n phần tử 1:n theo trật tự từ điển ?

Algorithm

n phần tử 1:n sẽ sinh ra n! hoán vị. Hoán vị nhỏ nhất là

1 2 … n

được gọi là hoán vị đơn vị.

Ta xuất phát từ hoán vị đơn vị và gọi hàm Next để sinh ra hoán vị kế tiếp.

```
Algorithm Next

Input: Hoán vị h[1:n]
Output: Hoán vị sát sau hoán vị h (h → h)
begin
   Duyệt ngược từ cuối h tìm điểm gãy i: h[i] < h[i+1]
   Nếu không có điểm gãy:
      h là hoán vị lớn nhất
      return false
   // có điểm gãy i
   Duyệt ngược từ cuối h tìm điểm vượt j: h[j] > h[i]
   Đổi chỗ h[i], h[j]
   Lật h[i+1:n]
   return true
end Next
```

Example

index	*1*	*2*	*3*	*4*	*5*	*6*	
h	1	2	**3**	6	**4**	2	
			i			←	Duyệt ngược tìm điểm gãy i
					j	←	Duyệt ngược tìm điểm vượt j
	1	2	**4**	6	**3**	2	Đổi chỗ h[i], h[j]
kết qủa h				2	3	6	Lật h[i+1:n]

```
bool Next() {
  int i, j;
  for (i = n-1; i >= 0; --i)
     if (h[i] < h[i+1]) break;
   if (i == 0) return false;
   for (j = n; j > i; --j)
     if (h[j] > h[i]) break;
   int t = h[i]; h[i] = h[j]; h[j] = t;
   ++i; j = n;
```

```
   while(i < j) {
      t = h[i]; h[i] = h[j]; h[j] = t;
      ++i; --j;
   }
   return true;
}
```

Program

```
// Permutation.cpp
#include <iostream>
using namespace std;

const int MN = 10;
int h[MN];
int n;

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i) cout << " " << x[i];
}

bool Next() {
  int i, j;
  for (i = n-1; i > 0; --i)
     if (h[i] < h[i+1]) break;
   if (i == 0) return false;
   for (j = n; j > i; --j)
     if (h[j] > h[i]) break;
   int t = h[i]; h[i] = h[j]; h[j] = t;
   ++i; j = n;
   while(i < j) {
      t = h[i]; h[i] = h[j]; h[j] = t;
      ++i; --j;
   }
   return true;
}

void Test(int nn) {
   n = nn;
   int m = 0;
   // init
   for (int i = 1; i <= n; ++i) h[i] = i;
   do {
     ++m;
     cout << "\n Permutation " << m << ". ";
     Print(h, 1, n, "");
   } while(Next());
}

main() {
   Test(3);
   cout << "\n T h e   E n d";

}
```

Result

```
 Permutation 1.  1 2 3
 Permutation 2.  1 3 2
 Permutation 3.  2 1 3
```

```
Permutation 4.  2 3 1
Permutation 5.  3 1 2
Permutation 6.  3 2 1
T h e   E n d
```

Độ phức tạp tính toán

O(n!)

Queens *(Back Tracking)*

Đặt n Hậu trên bàn cờ n× n sao cho các Hậu không chiếu nhau.

Algorithm

Duyệt các hoán vị h[1:n], h[i] là dòng đặt Hậu thứ i, chọn ra các hoán vị thỏa điều kiện đầu bài.

```
Algorithm Queens(n)

Input: n Hậu
Output: các phương án xếp Hậu
begin
   Init h[i] = i, i = 1:n
   do
      if Accept(h):
         Hiển thị h
       end if
    while (Next())
 end Queens
```

Program

```
// Queens.cpp
#include <iostream>
#include <cmath>

using namespace std;

const int MN = 30;
int h[MN];
int n;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
    cout << msg;
     for (int i = d; i <= c; ++i) cout << " " << x[i];
}

bool Next() {
   int i, j;
   for (i = n-1; i > 0; --i)
        if (h[i] < h[i+1]) break;
   if (i < 1) return false;
   for (j = n; j > i; --j)
      if (h[j] > h[i]) break;
   int t = h[i]; h[i] = h[j]; h[j] = t;
   ++i; j = n;
   while(i < j) {
      t = h[i]; h[i] = h[j]; h[j] = t;
      ++i; --j;
```

```
   }
  return true;
}

bool Accept() {
   for (int i = 2; i <= n; ++i) {
        for (int j = 1; j < i; ++j) {
           if (i-j == abs(h[i]-h[j]))
            return false;
        }
   }
   return true;
}

int Queens(int nn) {
   n = nn;
   cout << "\n\n " << n << " Queen: ";
   if (n < 1 || n == 2 || n == 3) {
      cout << "\n No solutions.";
        return 0;
   }
   // init
   int sn = 0;
   for (int i = 1; i <= n; ++i) h[i] = i;
   do {
     if (Accept()) {
          ++sn;
        cout << "\n Solution No " << sn << ". ";
        Print(h, 1, n, "");
        }
   } while(Next());
   cout << "\n Total " << sn << " solution(s).";
   Go();
   return sn;
}

void Test() {
   for (int n = -2; n <= 8; ++n) {
      Queens(n);
   }
}

main() {
   Test();
   cout << "\n T h e   E n d";
}
```

Result

```
 -2 Queen:
 No solutions.

 -1 Queen:
 No solutions.

 0 Queen:
 No solutions.

 1 Queen:
 Solution No 1.  1
 Total 1 solution(s). ?
```

```
2 Queen:
No solutions.

3 Queen:
No solutions.

4 Queen:
Solution No 1.  2 4 1 3
Solution No 2.  3 1 4 2
Total 2 solution(s). ?


5 Queen:
Solution No 1.  1 3 5 2 4
Solution No 2.  1 4 2 5 3
Solution No 3.  2 4 1 3 5
Solution No 4.  2 5 3 1 4
Solution No 5.  3 1 4 2 5
Solution No 6.  3 5 2 4 1
Solution No 7.  4 1 3 5 2
Solution No 8.  4 2 5 3 1
Solution No 9.  5 2 4 1 3
Solution No 10.  5 3 1 4 2
Total 10 solution(s). ?


6 Queen:
Solution No 1.  2 4 6 1 3 5
Solution No 2.  3 6 2 5 1 4
Solution No 3.  4 1 5 2 6 3
Solution No 4.  5 3 1 6 4 2
Total 4 solution(s). ?


7 Queen:
Solution No 1.  1 3 5 7 2 4 6
Solution No 2.  1 4 7 3 6 2 5
Solution No 3.  1 5 2 6 3 7 4
Solution No 4.  1 6 4 2 7 5 3
Solution No 5.  2 4 1 7 5 3 6
Solution No 6.  2 4 6 1 3 5 7
Solution No 7.  2 5 1 4 7 3 6
Solution No 8.  2 5 3 1 7 4 6
Solution No 9.  2 5 7 4 1 3 6
Solution No 10.  2 6 3 7 4 1 5
Solution No 11.  2 7 5 3 1 6 4
Solution No 12.  3 1 6 2 5 7 4
Solution No 13.  3 1 6 4 2 7 5
Solution No 14.  3 5 7 2 4 6 1
Solution No 15.  3 6 2 5 1 4 7
Solution No 16.  3 7 2 4 6 1 5
Solution No 17.  3 7 4 1 5 2 6
Solution No 18.  4 1 3 6 2 7 5
Solution No 19.  4 1 5 2 6 3 7
Solution No 20.  4 2 7 5 3 1 6
Solution No 21.  4 6 1 3 5 7 2
Solution No 22.  4 7 3 6 2 5 1
Solution No 23.  4 7 5 2 6 1 3
```

```
Solution No 24.  5 1 4 7 3 6 2
Solution No 25.  5 1 6 4 2 7 3
Solution No 26.  5 2 6 3 7 4 1
Solution No 27.  5 3 1 6 4 2 7
Solution No 28.  5 7 2 4 6 1 3
Solution No 29.  5 7 2 6 3 1 4
Solution No 30.  6 1 3 5 7 2 4
Solution No 31.  6 2 5 1 4 7 3
Solution No 32.  6 3 1 4 7 5 2
Solution No 33.  6 3 5 7 1 4 2
Solution No 34.  6 3 7 4 1 5 2
Solution No 35.  6 4 2 7 5 3 1
Solution No 36.  6 4 7 1 3 5 2
Solution No 37.  7 2 4 6 1 3 5
Solution No 38.  7 3 6 2 5 1 4
Solution No 39.  7 4 1 5 2 6 3
Solution No 40.  7 5 3 1 6 4 2
Total 40 solution(s). ?
```

```
8 Queen:
Solution No 1.  1 5 8 6 3 7 2 4
Solution No 2.  1 6 8 3 7 4 2 5
Solution No 3.  1 7 4 6 8 2 5 3
Solution No 4.  1 7 5 8 2 4 6 3
Solution No 5.  2 4 6 8 3 1 7 5
Solution No 6.  2 5 7 1 3 8 6 4
Solution No 7.  2 5 7 4 1 8 6 3
Solution No 8.  2 6 1 7 4 8 3 5
Solution No 9.  2 6 8 3 1 4 7 5
Solution No 10.  2 7 3 6 8 5 1 4
Solution No 11.  2 7 5 8 1 4 6 3
Solution No 12.  2 8 6 1 3 5 7 4
Solution No 13.  3 1 7 5 8 2 4 6
Solution No 14.  3 5 2 8 1 7 4 6
Solution No 15.  3 5 2 8 6 4 7 1
Solution No 16.  3 5 7 1 4 2 8 6
Solution No 17.  3 5 8 4 1 7 2 6
Solution No 18.  3 6 2 5 8 1 7 4
Solution No 19.  3 6 2 7 1 4 8 5
Solution No 20.  3 6 2 7 5 1 8 4
Solution No 21.  3 6 4 1 8 5 7 2
Solution No 22.  3 6 4 2 8 5 7 1
Solution No 23.  3 6 8 1 4 7 5 2
Solution No 24.  3 6 8 1 5 7 2 4
Solution No 25.  3 6 8 2 4 1 7 5
Solution No 26.  3 7 2 8 5 1 4 6
Solution No 27.  3 7 2 8 6 4 1 5
Solution No 28.  3 8 4 7 1 6 2 5
Solution No 29.  4 1 5 8 2 7 3 6
Solution No 30.  4 1 5 8 6 3 7 2
Solution No 31.  4 2 5 8 6 1 3 7
Solution No 32.  4 2 7 3 6 8 1 5
Solution No 33.  4 2 7 3 6 8 5 1
Solution No 34.  4 2 7 5 1 8 6 3
Solution No 35.  4 2 8 5 7 1 3 6
Solution No 36.  4 2 8 6 1 3 5 7
Solution No 37.  4 6 1 5 2 8 3 7
Solution No 38.  4 6 8 2 7 1 3 5
Solution No 39.  4 6 8 3 1 7 5 2
```

```
Solution No 40.  4 7 1 8 5 2 6 3
Solution No 41.  4 7 3 8 2 5 1 6
Solution No 42.  4 7 5 2 6 1 3 8
Solution No 43.  4 7 5 3 1 6 8 2
Solution No 44.  4 8 1 3 6 2 7 5
Solution No 45.  4 8 1 5 7 2 6 3
Solution No 46.  4 8 5 3 1 7 2 6
Solution No 47.  5 1 4 6 8 2 7 3
Solution No 48.  5 1 8 4 2 7 3 6
Solution No 49.  5 1 8 6 3 7 2 4
Solution No 50.  5 2 4 6 8 3 1 7
Solution No 51.  5 2 4 7 3 8 6 1
Solution No 52.  5 2 6 1 7 4 8 3
Solution No 53.  5 2 8 1 4 7 3 6
Solution No 54.  5 3 1 6 8 2 4 7
Solution No 55.  5 3 1 7 2 8 6 4
Solution No 56.  5 3 8 4 7 1 6 2
Solution No 57.  5 7 1 3 8 6 4 2
Solution No 58.  5 7 1 4 2 8 6 3
Solution No 59.  5 7 2 4 8 1 3 6
Solution No 60.  5 7 2 6 3 1 4 8
Solution No 61.  5 7 2 6 3 1 8 4
Solution No 62.  5 7 4 1 3 8 6 2
Solution No 63.  5 8 4 1 3 6 2 7
Solution No 64.  5 8 4 1 7 2 6 3
Solution No 65.  6 1 5 2 8 3 7 4
Solution No 66.  6 2 7 1 3 5 8 4
Solution No 67.  6 2 7 1 4 8 5 3
Solution No 68.  6 3 1 7 5 8 2 4
Solution No 69.  6 3 1 8 4 2 7 5
Solution No 70.  6 3 1 8 5 2 4 7
Solution No 71.  6 3 5 7 1 4 2 8
Solution No 72.  6 3 5 8 1 4 2 7
Solution No 73.  6 3 7 2 4 8 1 5
Solution No 74.  6 3 7 2 8 5 1 4
Solution No 75.  6 3 7 4 1 8 2 5
Solution No 76.  6 4 1 5 8 2 7 3
Solution No 77.  6 4 2 8 5 7 1 3
Solution No 78.  6 4 7 1 3 5 2 8
Solution No 79.  6 4 7 1 8 2 5 3
Solution No 80.  6 8 2 4 1 7 5 3
Solution No 81.  7 1 3 8 6 4 2 5
Solution No 82.  7 2 4 1 8 5 3 6
Solution No 83.  7 2 6 3 1 4 8 5
Solution No 84.  7 3 1 6 8 5 2 4
Solution No 85.  7 3 8 2 5 1 6 4
Solution No 86.  7 4 2 5 8 1 3 6
Solution No 87.  7 4 2 8 6 1 3 5
Solution No 88.  7 5 3 1 6 8 2 4
Solution No 89.  8 2 4 1 7 5 3 6
Solution No 90.  8 2 5 3 1 7 4 6
Solution No 91.  8 3 1 6 2 5 7 4
Solution No 92.  8 4 1 3 6 2 7 5
Total 92 solution(s). ?

T h e   E n d
```

Độ phức tạp tính toán

O(n!)

Next Number

Cho số nguyên dương s dài tối đa 1000 chữ số. Hãy đổi chỗ một vài chữ số trong s để thu được số x sát sau s và giữa s và x không có số nào nhận được bằng cahs trên.

Example

s = 74952

x = 75249

Algorithm

Gọi hàm Next

Program

```
// NextNumber.cpp
#include <iostream>
#include <cmath>

using namespace std;

const int MN = 30;
string s;
int n;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

bool Next() {
   int i, j;
   n = s.length();
   for (i = n-2; i >= 0; --i)
        if (s[i] < s[i+1]) break;
   if (i < 0) return false;
   for (j = n-1; j > i; --j)
      if (s[j] > s[i]) break;
   int t = s[i]; s[i] = s[j]; s[j] = t;
   ++i; j = n-1;
   while(i < j) {
      t = s[i]; s[i] = s[j]; s[j] = t;
      ++i; --j;
   }
  return true;
}

void Test() {
   s = "5744953";
   cout << "\n Given:  " << s;
   Next();
   cout << "\n Result: " << s;
}

main() {
```

```
    Test();
    cout << "\n T h e   E n d";
}
```

Result

```
Given:  5744953
Result: 5745349
T h e   E n d
```

Độ phức tạp tính toán

O(n)

Combinations

Tổ hợp chặp k của n phần tử, C(n, k) là số lượng các tập gồm k phần tử được lấy từ tập nền n phần tử. Vì là tập nên ta không xét đến thứ tự liệt kê của các phần tử. Như vậy, hai tập {1,2,3} và {3, 1, 2} là bằng nhau:

Formula

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!\,(n-k)!} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1}$$

0! = 1.

Ta có

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} = \frac{n!}{(k-1)!\,(n-k+1)!} \times \frac{n-k+1}{k} = \binom{n}{k-1} \times \frac{n-k+1}{k}$$

Nếu sử dụng cách ghi tuyến tính *C*(*n, k*) cho hàm tính tổ hợp chập *k* của *n,* ta có:

$$C(n, k) = C(n, k-1)(n-k+1) \text{ div } k \quad (1)$$

$$C(n, 0) = 1.$$

$$C(n, 1) = n.$$

Một phương án đệ quy cho hàm *C*(*n, k*) là:

C(n, k) = (k == 1) ? n : C(n, k – 1) * (n – k + 1) div k.

Thay *k* trong (1) bằng *i* + 1 ta thu được

$$C(n, i+1) = C(n, i)(n-i) \text{ div } (i+1) \quad (2)$$

Trong dạng không đệ quy, vận dụng (2), hàm *C*(*n, k*) sẽ được tính như sau:

```
// non recursive of C(n, k), n > 0, k > 0
Long C(int n, int k) {
  Long r = 1;
  for (int i = 0; i < k; ++i) {
    r = r * (n-i) / (i+1);
  }
  return r;
}

// Recursive of C(n, k), n > 0, k > 0
Long RC(int n, int k) {
  return (k == 1)? n : RC(n,k-1)*(n-k+1)/k;
}
```

Commented [WU1]: Việt hóa dòng này

Commented [WU2]: Việt hóa dòng này

Example

Có 6 tổ hợp chặp 2 của bốn số 0, 1, 2, 3 là

```
01
02
03
12
13
```

```
23

C(4, 2) = RC(4,2) = 6
```

Program

```
// Combination.cpp
#include <iostream>
#include <cmath>

using namespace std;

typedef long long Long;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

// non recursive of C(n, k), n > 0, k > 0
Long C(int n, int k) {
  Long r = 1;
  for (int i = 0; i < k; ++i) {
    r = r * (n-i) / (i+1);
  }
  return r;
}

// Recursive of C(n, k), n > 0, k > 0
Long RC(int n, int k) {
  return (k == 1)? n : RC(n,k-1)*(n-k+1)/k;
}

void Test() {
   cout << "\n " << C(4, 2) << " " << RC(4,2);
}

main() {
   Test();
   cout << "\n T h e   E n d";
}
```

Result

```
 6 6
 T h e   E n d
```

Độ phức tạp tính toán

O(k)

List Combinations

Ta cần liệt kê các tổ hợp chặp k của n phần tử 1:n.

Example

```
Combination(4,3)
2 3 4
1 3 4
1 2 4
1 2 3
```

Algorithm

Ta có thể vận dụng lại hàm Next để liệt kê C(n,k) như sau:

```
Khởi trị mảng h với n-k số 0 tại nửa trái và k số 1 tại dầu phải.
Gọi hàm Next đến khi false:
   Hiển thị các giá trị i nếu h[i] = 1.
```

Program

```
// Combinations.CPP
#include <iostream>
#include <windows.h>

using namespace std;

typedef long long Long;

const int MN = 30;
int h[MN];
int n, k;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i) cout << " " << x[i];
}

void PrintVal(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i)
        if (x[i])
         cout << " " << i;
}

// non recursive of C(n, k), n > 0, k > 0
Long C(int n, int k) {
  Long r = 1;
  for (int i = 0; i < k; ++i) {
    r = r * (n-i) / (i+1);
  }
  return r;
}
```

```
// Recursive of C(n, k), n > 0, k > 0
Long RC(int n, int k) {
  return (k == 1)? n : RC(n,k-1)*(n-k+1)/k;
}

bool Next() { // 0111
  int i, j;
  for (i = n-1; i > 0; --i)
     if (h[i] < h[i+1]) break;
   if (i < 1) return false;
   for (j = n; j > i; --j)
     if (h[j] > h[i]) break;
   int t = h[i]; h[i] = h[j]; h[j] = t;
   ++i; j = n;
   while(i < j) {
      t = h[i]; h[i] = h[j]; h[j] = t;
      ++i; --j;
   }
   return true;
}

int Combination(int nn, int kk) {
   // init
   n = nn; k = kk;
   cout << "\n Combination " << k << " of " << n << ":";
   // h = 0...01...1
   memset(h, 0, (n+1)*sizeof(int));
   int cb = 0;
   for (int i = n-k+1; i <= n; ++i) h[i] = 1;
   do {
     ++cb;
     PrintVal(h, 1, n, "\n ");  // Go();
     // Print(h, 1, n, "\n Init h: ");  Go();
   } while(Next());
   cout << "\n Total: " << cb << " combination(s).";
   return cb;
}

void Test() {
   Combination(4, 2); Go();
   Combination(4, 3);
}

main() {
   Test();
   cout << "\n T h e   E n d";
}
```

Result

```
 Combination 2 of 4:
  3 4
  2 4
  2 3
  1 4
  1 3
  1 2
 Total: 6 combination(s). ?

 Combination 3 of 4:
```

```
 2 3 4
 1 3 4
 1 2 4
 1 2 3
Total: 4 combination(s).
T h e   E n d
```

Độ phức tạp tính toán

O(C(n,k))

Baggage key

Các va ly du lịch thường có khóa gồm 4 chữ số.

Nếu bạn quên khóa thì phải thử những số nào ? Bao nhiêu lần ?

Algorithm

Ta cần liệt kê các tổ hợp chặp 4 của 10 phần tử 0:9. Trước hết ta chuyển thành bài toán liệt kê các tổ hợp chặp 4 của 10 phần tử 1:10. Khi hiển thị ta chỉ cần giảm mỗi số 1 đơn vị.

Program

```
// BaggageKey.cpp
#include <iostream>
//#include <cmath>
#include <windows.h>

using namespace std;

typedef long long Long;

const int MN = 30;
int h[MN];
int n, k;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i) cout << " " << x[i];
}

void PrintVal(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i)
        if (x[i] > 0)
         cout << " " << i-1;
}

// non recursive of C(n, k), n > 0, k > 0
Long C(int n, int k) {
  Long r = 1;
  for (int i = 0; i < k; ++i) {
    r = r * (n-i) / (i+1);
  }
  return r;
}

// Recursive of C(n, k), n > 0, k > 0
Long RC(int n, int k) {
  return (k == 1)? n : RC(n,k-1)*(n-k+1)/k;
}
```

```
bool Next() { // 0111
  int i, j;
  for (i = n-1; i > 0; --i)
     if (h[i] < h[i+1]) break;
   if (i < 1) return false;
   for (j = n; j > i; --j)
     if (h[j] > h[i]) break;
   int t = h[i]; h[i] = h[j]; h[j] = t;
   ++i; j = n;
   while(i < j) {
      t = h[i]; h[i] = h[j]; h[j] = t;
      ++i; --j;
   }
   return true;
}

int Combination(int nn, int kk) {
   // init
   n = nn; k = kk;
   cout << "\n Combination " << k << " of " << n << ": "
        << C(10,4) << " times.";  Go();
   memset(h, 0, (n+1)*sizeof(int));
   for (int i = n-k+1; i <= n; ++i) h[i] = 1;
   int cb = 0;
   do {
     ++cb;
     PrintVal(h, 1, n, "\n ");
   } while(Next());
   cout << "\n Total: " << cb << " combination(s).";
}

void Test() {
   Combination(10, 4); Go();
}

main() {
   Test();
   cout << "\n T h e   E n d";
}
```

Result

```
 Combination 4 of 10: 210 times. ?

  6 7 8 9
  5 7 8 9
  5 6 8 9
  5 6 7 9
  5 6 7 8
  4 7 8 9
  4 6 8 9
  . . .
Total: 210 combination(s).
```

Độ phức tạp tính toán

O(C(n,k))

Normal Form Of Combinations

Chúng ta muốn liệt kê các tổ hợp theo chiều tăng dần, ví dụ C(4,3) sẽ được liệt kê một cách tự nhiên như sau:

```
1 2 3
1 2 4
```

Dạng viết này được gọi là *dạng tự nhiên* hay *dạng chuẩn*.

Algorithm

Version 1

Ta khởi trị cho mảng h[1:k] là tổ hợp nhỏ nhất 1:k. Sau đó ta gọi hàm Next để sinh ra tổ hợp sát sau của h. Hàm Next hoạt động theo 2 pha như sau:

Pha 1. Dỡ. Duyệt ngược từ k qua trái bỏ qua những phần tử mang giá trị liên tiếp từ n trở xuống. Nếu sau khi dỡ h không còn phần tử nào thì kết thúc với Next = false với ý nghĩa là sát sau tổ hợp h không còn tổ hợp nào.

Ví dụ

n = 7, k = 5, h[1:5] = (2,3,5,6,7) thì sau khi dỡ ba phần tử cuối của h ta thu được i = 2, h[1:2] = (2,3). Điều này cho biết sẽ còn tổ hợp sát sau.

Pha 2. Xếp.

Xếp tiếp vào h[i:k] theo trật tự tăng dần liên tục từ h[i]+1. Tiếp tục với thí dụ trên ta thu được h[1:5] = (2,4,5,6,7).

Ta sử dụng phần tử h[0] = n làm lính canh.

Program

```
// NormalForm.cpp
#include <iostream>
#include <windows.h>

using namespace std;

typedef long long Long;

const int MN = 30;
int h[MN];
int n, k;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i)
        cout << " " << x[i];
}

// non recursive of C(n, k), n > 0, k > 0
Long C(int n, int k) {
```

```
  Long r = 1;
  for (int i = 0; i < k; ++i) {
    r = r * (n-i) / (i+1);
  }
  return r;
}

// Recursive of C(n, k), n > 0, k > 0
Long RC(int n, int k) {
  return (k == 1)? n : RC(n,k-1)*(n-k+1)/k;
}

bool Next() {
   int i, b;
   h[0] = n;
   // Phase 1. Do
   for (i = k, b = n; i > 0; --i, --b) {
      if (h[i] != b) break;
   }
   if (i == 0) return false;
   // Phase 2. Xep
   ++h[i];
   for (int j = i + 1; j <= k; ++j)
     h[j] = h[j-1] + 1;
   return true;
}

int Combination(int nn, int kk) {
   // init
   n = nn; k = kk;
   cout << "\n Combination " << k << " of " << n
        << " : " << RC(n, k);
   for (int i = 0; i <= k; ++i) h[i] = i;
   int cb = 0;
   do {
     ++cb;
     cout << "\n " << cb << ". ";
     Print(h, 1, k, " ");
   } while(Next());
   cout << "\n Total: " << cb << " combination(s).";
}


void Test() {
   Combination(4, 2); Go();
   Combination(5, 3);
}

main() {
   Test();
   cout << "\n T h e   E n d";
}
```

result

```
 Combination 2 of 4 : 6
 1.   1 2
 2.   1 3
 3.   1 4
 4.   2 3
 5.   2 4
```

```
6.   3 4
Total: 6 combination(s). ?

Combination 3 of 5 : 10
1.   1 2 3
2.   1 2 4
3.   1 2 5
4.   1 3 4
5.   1 3 5
6.   1 4 5
7.   2 3 4
8.   2 3 5
9.   2 4 5
10.   3 4 5
Total: 10 combination(s).
T h e   E n d
```

Version 2

Ta cải tiến hàm Next như sau. Giả sử sau pha 1 ta thu được vị trí i thỏa h[i] ≠ n–k+i. Ta gọi vị trí này là vị trí cập nhật và sẽ điều khiển nó thông qua một biến v. Ta khởi trị cho h và v như sau

```
h[i] = i, i = 1:k
v = (h[k] = n) ? 0 : k;
```

Sau đó mỗi lần gọi hàm Next ta kiểm tra

Nếu v = 0 thì dừng hàm Next.

Nếu v ≠ 0 ta thực hiện pha 2 sau đó chỉnh lại giá trị của v như sau:

Nếu h[k] = n thì tức là h[v:k] = (n–k–v, ..., n–1, n) thì lần gọi Next tiếp theo sẽ cập nhật tại vị trí v-1, ngược lại, nếu h[k] ≠ n thì lần gọi Next tiếp theo sẽ cập nhật tại vị trí k.

Program

```
// NormalForm.cpp, Ver 2
#include <iostream>
#include <windows.h>

using namespace std;

typedef long long Long;

const int MN = 30;
int h[MN];
int n, k, v;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i)
        cout << " " << x[i];
}

// non recursive of C(n, k), n > 0, k > 0
```

```
Long C(int n, int k) {
  Long r = 1;
  for (int i = 0; i < k; ++i) {
    r = r * (n-i) / (i+1);
  }
  return r;
}

// Recursive of C(n, k), n > 0, k > 0
Long RC(int n, int k) {
  return (k == 1)? n : RC(n,k-1)*(n-k+1)/k;
}

bool Next() {
   if (v == 0) return false;
   // Pha 2. Xep
   ++h[v];
   for (int i = v + 1; i <= k; ++i) h[i] = h[i-1] + 1;
   v = (h[k] == n) ? v-1 : k;
   return true;
}

int Combination(int nn, int kk) {
   // init
   n = nn; k = kk;
   cout << "\n Combination " << k << " of " << n
        << " : " << RC(n, k);
   for (int i = 0; i <= k; ++i) h[i] = i;
   v = (h[k] == n) ? 0 : k;
   int cb = 0;
   do {
     ++cb;
     cout << "\n " << cb << ". ";
     Print(h, 1, k, " ");
   } while(Next());
   cout << "\n Total: " << cb << " combination(s).";
}


void Test() {
   Combination(4, 2); Go();
   Combination(5, 3);
}

main() {
   Test();
   cout << "\n T h e   E n d";
}
```

Độ phức tạp tính toán

O(C(n,k))

Beauty (sliding window)

Cho một hoán vị của N số 0,…, N–1. Một đoạn gồm k số liên tiếp nhau được gọi là đẹp nếu trong đoạn đó có đủ k số từ 0 đến k–1.

Hãy cho biết có bao nhiêu đoạn đẹp trong hoán vị.

Input: n, hoán vị h[0:n-1]

Output: Hiển thị trên màn hình số lượng đoạn đẹp.

Ví dụ

```
n = 10
h = {6, 9, 5, 7, 4, 0, 3, 2, 1, 8};
Result: 6
```

Hoán vị này có 4 đoạn đẹp là:

(0)

(0,3,2,1)

(4,0,3,2,1)

(6,9,5,7,4,0,3,2,1,8)

Vị trí	0	1	2	3	4	5	6	7	8	9
Hoán vị	6	9	5	7	4	0	3	2	1	8
Đoạn đẹp 1 [5;5]						0				
Đoạn đẹp 2 [5;8]						0	3	2	1	
Đoạn đẹp 3 [4;8]					4	0	3	2	1	
Đoạn đẹp 4 [0;9]	6	9	5	7	4	0	3	2	1	8

Thuật toán

Xét mọi tổ hợp chặp 2 của n phần tử (a:b), a < b. Kiểm tra h[a:b] có đẹp không?

Program

```
// BEAUTY.CPP
#include <iostream>
#include <windows.h>

using namespace std;

typedef long long Long;

const int MN = 30;
int hv[MN] = {6, 9, 5, 7, 4, 0, 3, 2, 1, 8};
int h[MN];
int n = 10, k, v;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
```

```
   cout << msg;
    for (int i = d; i <= c; ++i) cout << " " << x[i];
}

bool Next() {
   if (v == 0) return false;
   // Pha 2. Xep
   ++h[v];
   for (int i = v + 1; i <= k; ++i) h[i] = h[i-1] + 1;
   v = (h[k] == n) ? v-1 : k;
   return true;
}

// hv[a:b] chua cac so 0:b-a
bool Beauty(int a, int b) {
   if (a == 0 && b == n-1) return true;
   int k = b-a;
   for (int i = a; i <= b; ++i)
      if (hv[i] > k) return false;
   return true;
}

int Combination(int nn, int kk) {
   // init
   n = nn; k = kk;
   cout << "\n Combination " << k << " of " << n;
   for (int i = 0; i <= k; ++i) h[i] = i;
   v = (h[k] == n) ? 0 : k;
   int b = 0;
   for (int i = 0; i < n; ++i)
        if (hv[i] == 0) {
        ++b;
        cout << "\n Beuty segment " << b << ". " << i << ":" << i;
        Print(hv,i,i, "\n ");
        break;
     }
   do {
     if (Beauty(h[1]-1, h[k]-1)) {
      ++b;
      cout << "\n Beuty segment " << h[1]-1 << " : " << h[k]-1;
      Print(hv, h[1]-1, h[k]-1, "\n ");
    }
   } while(Next());
   cout << "\n Total: " << b << " beauty segment(s).";
}


void Test() {
   Combination(10, 2); Go();
}

main() {
   Test();
   cout << "\n T h e   E n d";
}
```

Result

```
 Combination 2 of 10
```

```
Beuty segment 1. 5:5
 0
Beuty segment 0 : 9
 6 9 5 7 4 0 3 2 1 8
Beuty segment 4 : 8
 4 0 3 2 1
Beuty segment 5 : 8
 0 3 2 1
Total: 4 beauty segment(s). ?
```

Palindrome (math)

Số nguyên dương x được gọi là palindrome (đối xứng) nếu đọc xuôi hay ngược đều cho cùng một giá trị. Trong các số từ 10 đến 1000000 (một triệu, 1M) có bao nhiêu số đối xứng.

Thuật toán

Ví dụ, các số 515, 50705 là những số đối xứng, còn số 123, 37 không đối xứng.

Ký hiệu Rev(x) là hàm cho ra số lật của số x. Ví dụ, Rev(1234) = 4321. Khi đó x là đối xứng khi và chỉ khi Rev(x) = x.

Để cài đặt hàm Rev(x), ta lấy lần lượt từng chữ số đơn vị của x (từ trái qua phải), ghép với kết quả y.

x	y
1234	0
123	4
12	43
1	432
0	4321

Rev(1234)

Program

```
// PALINDROME.CPP
#include <iostream>
#include <fstream>

using namespace std;

const int MN = 10000000;

int rev(int x) {
    int y = 0;
    while (x){
        y = y * 10 + (x % 10);
        x /= 10;
    }
    return y;
}

void run(){
    int c = 0;
    for (int x = 10; x < MN; ++x)
        if (rev(x) == x)
            ++c;
   cout << c;
}

main() {
  run(); // 10989
  return 0;
  cout << "\n T h e   E n d ";
}
```

Độ phức tạp tính toán

Thuật toán duyệt n số, mỗi số duyệt k chữ số: O(kn), k là số chữ số thập phân tối đa trong máy tính.

Bình luận

Cho số x gồm k chữ số. Từ số x ta có thể tạo ra một số đối xứng 2k chữ số bằng cách ghép số lật của x vào sau chính nó tựa như ghép hai string: x|x'.

Cũng từ số x ta có thể tạo ra 10 số đối xứng 2k + 1 chữ số bằng cách ghép x|c|x', trong đó | là ký hiệu phép ghép, x' là số lật của x, x' = Rev(x), c là một trong mười chữ số 0, 1, ..., 9.

Ví dụ

Từ số x = 12 gồm 2 chữ số ta thu được 11 số đối xứng như sau:

- một số đối xứng 4 chữ số: 12|21 = 1221.
- mười số đối xứng 5 chữ số là:

  12021, 12121, 12221, 12321, 12421 , 12521, 12621, 12721, 12821, 12921.

Như vậy từ số x gồm k chữ số ta sinh ra được 11 số đối xứng khác nhau.

Với mỗi số x gồm 1 đến 3 chữ số, tức là $1 \le x < 999$ ta sinh ra 11 số đối xứng khác nhau có chiều dài từ hai đến 7 chữ số. Vậy tổng cộng ta thu được:

$$11 * 999 = 10989 \text{ số đối xứng.}$$

Bài này cũng cho thấy, đôi khi ta có thể bỏ qua khâu lập trình.

Necklace (Find-Union)

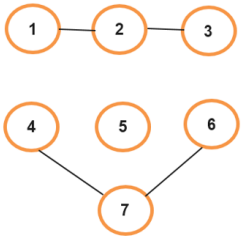
Hôm sinh nhật, Cam được các bạn tặng một hộp các hạt cườm, mỗi hạt đều có lỗ để xuyên dây kết nối với nhau.

Hộp đồ chơi và các chuỗi hạt của Cam

Cam chọn ra N hạt cườm rồi dùng các sợi chỉ kết nối từng cặp hai hạt khác nhau với nhau.

Giả sử các hạt cườm do Cam chọn được gán số từ 1 đến N. Mỗi khi Cam cầm một hạt nhấc lên khỏi mặt bàn thì có những hạt nào được nhấc lên theo?

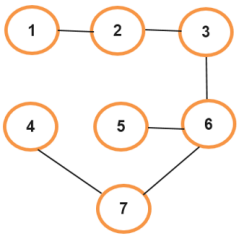
Ví dụ 1



N = 7 hạt
1 2
2 3
4 7
7 6

Nhấc hạt 3: 1, 2, 3
Nhấc hạt 5: 5
Nhấc hạt 7: 4, 6, 7

Ví dụ 2



N = 7 hạt
1 2
2 3
3 6
5 6
4 7
7 6

Nhấc hạt bất kỳ,

mọi hạt đều được nhấc lên theo.

Hiểu đề

Cho đồ thị G gồm N đỉnh mã số từ 1 đến N và M cạnh, mỗi cạnh (x,y) nối hai đỉnh khác nhau x và y. Các cạnh đều là hai chiều, nghĩa là ta có thể di chuyển từ đỉnh x đến đỉnh y hoặc ngược lại. G được gọi là đồ thị hữu hạn (vì có hữu hạn đỉnh) và vô hướng (vì có thể di chuyển xuôi, ngược trên mỗi cạnh). Đồ thị G là liên thông nếu từ một đỉnh bất kỳ có thể di chuyển qua các cạnh để đến một đỉnh bất kỳ khác. Ngược lại, nếu G không liên thông thì các đỉnh của G được tách thành k > 1 nhóm đỉnh, mỗi nhóm cùng với các cạnh trong nhóm đó tạo thành một đồ thị con liên thông được gọi là một *thành phần liên thông*.

Hãy xác định xem G có bao nhiêu thành phần liên thông, là những thành phần nào?

Đồ thị trong ví dụ 1 gồm N = 7 đỉnh 1, 2, 3, 4, 5, 6, 7 và 4 cạnh (1,2), (2,3), (6,7), (7,4) là đồ thị *không liên thông* vì có ba thành phần liên thông là {1, 2, 3}, {4, 6, 7} và {5}.

Đồ thị trong ví dụ 2 gồm N = 7 đỉnh 1, 2, 3, 4, 5, 6, 7 và 6 cạnh (1,2), (2,3), (6,3), (6,5), (6,7), (7,4) là đồ thị *liên thông*.

Gọi *k* là số thành phần liên thông của *G.* Ta cần xác định các thông số sau đây:

- Nếu k = 1 ta trả lời đồ thị G là liên thông.
- Nếu k ≥ 2 ta cần thông báo từng thành phần liên thông.

Với hai ví dụ trên ta có:

- k = 1: Đồ thị G liên thông (ví dụ 2).
- k = 3: Đồ thị Q không liên thông và gồm các thành phần liên thông sau (ví dụ 1):

  {1, 2, 3}, {4, 6, 7}, {5}.

Sau khi xác định được các thành phần liên thông thì ta có thể trả lời được các câu hỏi trong bài, cụ thể là:

- *Nếu bạn Cam cầm hạt cườm x nhấc lên thì toàn bộ các hạt cườm thuộc thành phần liên thông với x sẽ được nhấc lên theo.*

Input

Dữ liệu cho đồ thị được cho trong file text CUOM.INP như sau:

Dòng thứ nhất 2 số: số đỉnh N và số cạnh M được ghi cách nhau.

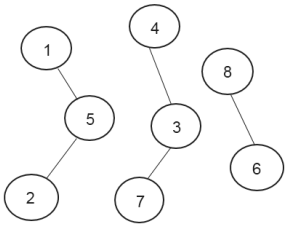
Tiếp đến là M dòng, mỗi dòng thể hiện một cạnh x y được ghi cách nhau.

Tiếp đến số test Q.

Tiếp sau Q là Q dòng, mỗi dòng là số hiệu hạt cườm được Cam nhấc lên.

Ví dụ

CUOM.INP	OUTPUT
8 5 2 5 3 7 8 6 5 1 4 3 3 5 7 8	1 2 5 3 4 7 6 8



Ý nghĩa:

- Đồ thị gồm *8 đỉnh và 5 cạnh:* (2,5), (3,7), (8,6), (5,1), (4,3). *Đồ thị có k = 3 thành phần liên thông* {1, 2, 5}, {3, 4, 7} và {6, 8}.
- 3 test
- Cam lần lượt nhấc các hạt cườm 5, 7 và 8.
  - Cam nhấc hạt cườm 5: các hạt cườm 1, 2, 5 được nhấc lên.
  - Cam nhấc hạt cườm 7: các hạt cườm 3, 4, 7 được nhấc lên.
  - Cam nhấc hạt cườm 8: các hạt cườm 6, 8 được nhấc lên.

Thuật toán

Gọi LT(v) là thành phần liên chứa hạt v, Ta phát triển dần v như sau.

Xuất phát LT = {v}

Lặp đến khi LT hết tăng trưởng

Duyệt m cạnh (x,y)

nếu x ∈ LT thì add y to LT

nếu y ∈ LT thì add x to LT

Nên đánh dấu các cạnh đã duyệt và đếm số lần thêm một hạt vào TPLT.

Program

```
 // NECLACE.CPP
#include <iostream>
#include <fstream>
#include <windows.h>

using namespace std;

int n, m; // n dinh, m canh
int *x, *y;
bool *s; // set
bool *used;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}
```

```
void Swap(int &a, int &b) {
   int t = a; a = b; b = t;
}

void LT(int v) {
   s = new bool[n+1];
   memset(s, false, (n+1)*sizeof(bool));
   used = new bool[m];
   memset(used, false, m*sizeof(bool));
   s[v] = true;
   int  c = 1;
   while (c) {
        c = 0;
        for (int i = 0; i < m; ++i) {
            if (used[i]) continue;
           if (s[x[i]] || s[y[i]]) {
           if (s[x[i]])
              s[y[i]] = true;
           else if (s[y[i]])
              s[x[i]] = true;
           ++c; used[i] = true;
        }
        }
   }
   for (int i = 1; i <= n; ++i)
        if (s[i]) cout << " " << i;

}

void Necklace() {
   ifstream f("CUOM.INP");
   f >> n >> m;
   cout << "\n " << n << " dinh " << m << " canh.";
   x = new int[m];
   y = new int[m];
   for (int i = 0; i < m; ++i) {
      f >> x[i] >> y[i];
      if (x[i] > y[i]) Swap(x[i], y[i]);
      cout << "\n Edge " << x[i] << "-" << y[i];
   }
   int q; // so test
   int v;
   f >> q;
   for (int i = 0; i < q; ++i) {
      f >> v;
      cout << "\n Test " << i << ". v = " << v;
      cout << " : LT = "; LT(v);
   }
   f.close();
}

main() {
   Necklace();
   cout << "\n T h e   E n d";
}
```

Result

```
 8 dinh 5 canh.
```

```
Edge 2-5
Edge 3-7
Edge 6-8
Edge 1-5
Edge 3-4
Test 0. v = 5 : LT =  1 2 5
Test 1. v = 7 : LT =  3 4 7
Test 2. v = 8 : LT =  6 8
T h e   E n d
```

Độ phức tạp tính toán

Hàm LT thực hiện tối đa N bước, mỗi lần duyệt M cạnh. Tổng cộng lại, với đồ thị có N đỉnh và M cạnh ta có độ phức tạp cỡ N×M – tuyến tính theo chiều dài input.

Magic Table

Điền các số 1:n^2 vào ma trận n×n sao cho tổng các số trên mỗi dòng, mỗi cột và mỗi đường chéo đều bằng nhau.

Example

8	1	6
3	5	7
4	9	2

sum = 15

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

sum = 65

Algorithm

Sinh các hoán vị 1:n^2 rồi kiểm tra.

Program

```
// MagicTable.cpp
#include <iostream>
#include <cmath>

using namespace std;

const int MN = 100;
int h[MN];
int n;
int n2;
int sum;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i) cout << " " << x[i];
}

void Print(int **x, int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i)
        Print(x[i], d, c, "\n ");
}

void PrintTable(string msg = "") {
   cout << "\n ";
   for (int i = 1; i <= n2; ++i) {
      cout << " " << h[i];
      if (i % n == 0) cout << "\n ";
   }
}
```

```
void Swap(int i, int j) {
   int t = h[i];
   h[i] = h[j];
   h[j] = t;
}

bool Next() {
   int i, j;
   for (i = n2-1; i > 0; --i)
        if (h[i] < h[i+1]) break;
   if (i < 1) return false;
   for (j = n2; j > i; --j)
      if (h[j] > h[i]) break;
   Swap(i, j);
   ++i; j = n2;
   while(i < j) {
      Swap(i++, j--);
   }
  return true;
}

// sum{h[i], i = d:c}
int GetSum(int start, int size, int d = 1) {
   int s = 0;
   for (int i = 1; i <= size; ++i) {
        s += h[start];
        start += d;
      }
   return s;
}

// -----*-----*-------*------
bool Accept() {
   int d = 1;
   for (int i = 1; i <= n; ++i) {
        if (GetSum(i,n,n) != sum) // column
           return false;
        if (GetSum(d,n,1) != sum) // row
           return false;
     d += n;
   }
   // cheo
    if (GetSum(1,n, n+1) != sum)
        return false;
     if (GetSum(n,n, n-1) != sum)
        return false;
   return true;
}

void MagicTable(int nn) {
   n = nn; n2 = n*n;
   cout << "\n\n " << " Magic Table of " << n << ":";
   if (n < 1 || n == 2) {
      cout << "\n No solutions.";
        return ;
   }
   if (n == 1) {
      cout << " 1 ";
      return;
   }
```

```
   // init
   for (int i = 1; i <= n2; ++i) h[i] = i;

   sum = (1+n2)*n/2;
   // Print("\n h: "); Go();
   do {
        //Print(h, 1, n2, "\n h:"); Go();
     if (Accept()) {
        PrintTable("\n Result: ");
        return;
        }
   } while(Next());
}


main() {
   MagicTable(3);
   cout << "\n T h e   E n d";
}
```

Độ phức tạp tính toán

$O(n^2!)$ rất cao nên chỉ chấp nhận khi n = 3.

Tom and Jerry (Divide and Conquer)

Giải mã văn bản bí mật sau:

Biết

A	C	E	H	M	O	T	U	V	*
									

Thuật toán

Ta thấy văn bản mật chỉ gồm hai ký hiệu là mèo  và chuột máy , do đó ta dễ dàng chuyển đổi thành hai trị: 0 cho mèo và 1 cho chuột máy . Đề bài khi đó sẽ được chuyển thành:

Giải mã

→ 01101001001111100001110010101101110010 10

Biết

A	C	E	H	M	O	T	U	V	*
000	001	0100	0101	011	100	1010	1011	110	111

Văn bản mật: 01101001001111100001110010101101110010 10

A	C	E	H	M	O	T	U	V	*
000	001	0100	0101	011	100	1010	1011	110	111

Input

```
10
A 000
C 001
E 0100
H 0101
M 011
O 100
T 1010
U 1011
V 110
* 111
01101001001111100001110010101101110010 10
```

Algorithm

So khớp từng mã vào đầu văn bản mật. Nếu khớp thì so tiếp.

Độ phức tạp tính toán

Nếu có m ký tự, mỗi ký tự dài trung bình k bit, văn bản mật chứa n ký tự thì để phát hiện một ký tự mật cần duyệt m lần, mỗi lần so khớp k bit. Vậy tổng cộng O(kmn).

Program

```
// TOMJERRY.CPP
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

int n;
string s;
string code[26];
string alpha;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i) cout << " " << x[i];
}

// so khop code[i] tai s[d:]
bool Match(int i, int d) {
   for (int j = 0; j < code[i].length(); ++j)
      if (s[d+j] != code[i][j])
        return false;
   return true;
}
void Decode() {
   cout << "\n Decoding |" << s << "|\n";
   int d = 0;
   while(d < s.length()) {
        for (int i = 0; i < n; ++i) {
           if (Match(i, d)) {
              cout << alpha[i];
              d += code[i].length();
              break;
           }
     }
    }
}

void TomJerry() {
   ifstream f("TOMJERRY.INP");
   f >> n; cout << "\n " << n;
   getline(f, s);
   for (int i = 0; i < n; ++i) {
      getline(f, s);
      //cout << "\n " << s;
      alpha += s[0];
      code[i] = "";
    for (int j = 0; j < s.length(); ++j)
      if (s[j] == '0' || s[j] == '1')
         code[i] += s[j];
   }
   for (int i = 0; i < n; ++i) {
```

```
            cout << "\n " << alpha[i] << " : |" << code[i] << "|";
        }
        f >> s;
        f.close();
        Decode();
    }


    main() {
        TomJerry();
        cout << "\n T h e   E n d";
    }
```

Result

```
 10
 A : |000|
 C : |001|
 E : |0100|
 H : |0101|
 M : |011|
 O : |100|
 T : |1010|
 U : |1011|
 V : |110|
 * : |111|
 Decoding |01101001001111100001110010101101110010 10|
MEO*VA*CHUOT
 T h e   E n d
```

Divisors

Cho các số nguyên a, b và c. Nếu a = b× c thì ta nói b là ước số của a, hoặc a chia hết cho b. Dĩ nhiên, c cũng là ước số của a và a chia hết cho c. Cho số nguyên k. Hãy cho biết trong các số nguyên dương đến 1M (một triệu, 1000000) có bao nhiêu số có đúng k ước.

Ví dụ

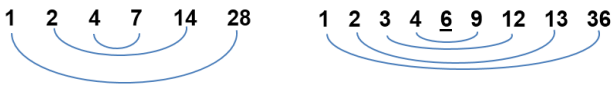
Hai mươi số nguyên dương đầu tiên có số ước như sau:

số	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*
số ước	1	2	2	3	2	4	2	4	3	4	2	6	2	4	4	5	2	6	2	6

Thuật toán

Ký hiệu τ(*n*) (đọc là *tau*) là hàm cho ra *số ước* của số nguyên dương *n.* Một phương pháp đơn giản để tính hàm τ(*n*) dựa trên tính chia hết là như sau. Nếu *a* là ước của *n*, tức là *n* = *ab* thì *b* cũng là ước khác của *n* nếu *b* ≠ *a*. Như vậy, các ước của số tự nhiên *n* có thể được viết dưới dạng các cặp đối xứng (*a, b*) trong đó *a* ≠ *b* và *ab* = *n*.

Ví dụ



Các ước của 28.
τ(28) = 6

Các ước của 36.
τ(36) = 9

Chương trình C++

Primes

Min Max in Range

Cho dãy N số nguyên a mã số từ 0 đến N–1. Với mỗi khoảng d, c, $0 \le d \le c \le N-1$ hãy hiển thị số lớn nhất trong khoảng đó.

Ví dụ

Với dãy a gồm 10 số

chỉ số	0	1	2	3	4	5	6	7	8	9
a	20	15	14	28	17	12	19	30	21	11

và 5 câu hỏi, chương trình sẽ hiển thị như sau:

d	c	Hiển thị
1	5	28
2	7	30
0	9	30
3	6	28
4	4	17

Giới hạn

$1 \le N \le 200000$

Số câu hỏi Q, $1 \le 400000$

Max Divisor

Trong bài này và bài tiếp theo sẽ trình bày thêm hai ứng dụng của cây Descartest.

Cho dãy N số nguyên dương a mã số từ 0 đến N–1. Với mỗi khoảng d, c, $0 \le d \le c \le N-1$ hãy hiển thị số có nhiều ước nhất trong khoảng đó.

Ví dụ

Với dãy a gồm 10 số

chỉ số	0	1	2	3	4	5	6	7	8	9
a	9	15	14	28	47	12	51	30	21	11
Số ước	3	4	4	6	2	6	4	8	4	2

và 5 câu hỏi chương trình sẽ hiển thị như sau:

d c	Hiển thị
1 5	28 (6)
2 7	30 (8)
0 9	30 (8)
3 6	28 (6)
4 4	47 (2)

Các chương trình dưới đây minh họa tìm kiếm max theo khoảng với input data như sau:

Input data

MD.INP	Giải thích
10	10 số
9 15 14 28 47 12 51 30 21 11	giá trị các số
5	5 câu hỏi dạng
1 5	d c
2 7	
0 9	
3 6	
4 4	

Giới hạn

$1 \le N \le 200000$

Số câu hỏi Q, $1 \le 400000$

Thuật toán

Ví dụ

số	…	9	…	11	12	…	14	15	…	21	…	28	…	30	…	47	…	51
uoc[]		3		2	6		4	4		4		6		8		2		4

u[0:10] = (3, 4, 4, 6, 2, 6, 4, 8, 4, 2);

Meetings

IOI 2018, Japan

Dọc theo một đường thẳng nằm ngang có N ngọn núi được đánh số từ 0 đến N–1, từ trái qua phải. Chiều cao của ngọn núi i là H_i ($0 \le i \le N-1$). Có đúng một người sống trên đỉnh của mỗi ngọn núi.

Bạn cần tổ chức Q cuộc họp được đánh số từ 0 đến Q–1. Tham gia cuộc họp j ($0 \le j \le Q-1$) sẽ là tất cả những người sống trên đỉnh các ngọn núi từ Lj đến Rj, kể cả hai đầu mút ($0 \le Lj \le Rj \le N-1$). Đối với cuộc họp này bạn cần chọn ngọn núi x làm nơi diễn ra cuộc họp ($Lj \le x \le Rj$). Chi phí của cuộc họp này phụ thuộc vào lựa chọn của bạn và được tính như sau:

- Chi phí của thành viên đến từ ngọn núi y ($Lj \le y \le Rj$) là độ cao lớn nhất của các ngọn núi trong khoảng giữa hai ngọn núi Lj và Rj, kể cả hai đầu mút. Đặc biệt, chi phí của thành viên đến từ ngọn núi x là Hx, là chiều cao của ngọn núi đó.
- Chi phí của cuộc họp là tổng các chi phí của tất cả các thành viên.

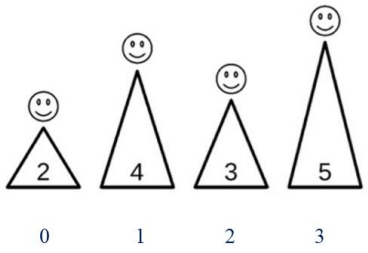
Đối với mỗi cuộc họp bạn phải tìm chi phí nhỏ nhất có thể để tổ chức nó.

Chú ý là tất cả các thành viên sẽ quay lại ngọn núi của mình sau mỗi cuộc họp, do đó chi phí của một cuộc họp là không bị ảnh hưởng từ các cuộc họp trước đó.

Ví dụ

Giả sử $N = 4, H = [2,4,3,5], Q = 2, L = [0,1]$ và $R = [2,3]$.

Trình chấm gọi minimum_costs $([2,4,3,5], [0,1], [2,3])$.



Cuộc họp j = 0 có Lj = 0 và Rj = 2, vì thế tham gia nó là những người sống ở các ngọn núi 0, 1 và 2. Nếu ngọn núi 0 được chọn làm nơi tổ chức cuộc họp thì chi phí của cuộc họp được tính như sau:

- Chi phí của thành viên đến từ ngọn núi 0 là $max\{H_0\} = 2$.
- Chi phí của thành viên đến từ ngọn núi 1 là $max\{H_0, H_1\} = 4$.
- Chi phí của thành viên đến từ ngọn núi 2 là $max\{H_0, H_1, H_2\} = 4$.
- Vậy, chi phí của cuộc họp 0 là 2 + 4 + 4 = 10.

Không thể tổ chức cuộc họp 0 với chi phí nhỏ hơn 10, do đó chi phí nhỏ nhất để tổ chức cuộc họp 0 là 10.

Cuộc họp j = 1 có Lj = 1 và Rj = 3, vì thế tham gia nó là những người sống ở các ngọn núi 1, 2 và 3. Nếu ngọn núi 2 được chọn làm nơi tổ chức cuộc họp, thì chi phí của cuộc họp được tính như sau:

- Chi phí của thành viên đến từ ngọn núi 1 là $max\{H_0, H_1\} = 4$.
- Chi phí của thành viên đến từ ngọn núi 2 là $max\{H_2\} = 3$.
- Chi phí của thành viên đến từ ngọn núi 3 là $max\{H_2, H_3\} = 5$.
- Vậy, chi phí của cuộc họp 1 là 4 + 3 + 5 = 12.

Không thể tổ chức cuộc họp 1 với chi phí nhỏ hơn 12, do đó chi phí nhỏ nhất để tổ chức cuộc họp 1 là 12.

Picnic

Một lớp có N học sinh mang bí danh lần lượt là A, B, C,…Các bạn học sinh kết thân với nhau theo từng nhóm. Nhóm này có thể rủ nhóm khác để tạo thành một nhóm mới. Thí dụ, AD > E cho biết hai bạn A và D tạo thành một nhóm và nhóm AD này có thể rủ được bạn E nhưng ngược lại thì không.

Một số bạn đứng ra tổ chức picnic. Hãy cho biết những ai có mặt trong buổi picnic.

Dữ liệu vào: Text file

Dòng đầu tiên hai số nguyên dương: số học sinh N và số quan hệ M

M dòng tiếp theo: mỗi dòng một quan hệ dạng X > Y, trong đó X và Y là hai dãy ký tự (string) chứa danh sách các bạn trong mỗi nhóm.

Dòng tiếp theo: số test Q ≥ 1

Q dòng tiếp theo, mỗi dòng một test, là một string chứa danh sách những bạn đứng ra tổ chức picnic.

Các học sinh có mã một chữ cái A, B, … trong bảng chữ tiếng Anh viết HOA từ ký tự đầu tiên đến ký tự thứ N.

Giới hạn

$1 \le N \le 26$

$1 \le M \le 200$

$1 \le Q \le 200$

Ví dụ

PICNIC.INP	Giải thích
6 3	6 bạn trẻ A, B, C, D, E, F, 3 quan hệ
AB > C	AB rủ được C
D > E	D rủ được E
E > BF	E rủ được BF
3	3 test
AD	AD tổ chức picnic
E	E tổ chức picnic
CD	CD tổ chức picnic

Thuật toán

Ta thấy quan hệ > có các tính chất sau đây:

Với các nhóm bạn trẻ *X*, *Y* và Z *ta* có:

R1. *Phản xạ*: $X > X$ (mỗi nhóm tự rủ mình)

R2. *Bắc cầu*: Nếu $X > Y$ và $Y > Z$ thì $X > Z$.

R3. *Gia tăng*: Nếu $X > Y$ thì $XZ > YZ$.

Trong đó ký hiệu *XZ* là hợp của hai nhóm *X* và *Z*: $X \cup Z$.

Ký hiệu:

U là danh sách toàn bộ các bạn trẻ,

F là tập các quan hệ trong đề bài,

X là nhóm bạn trẻ đứng ra tổ chức picnic,

Find(X) là tập các bạn trẻ đi cùng với *X*,

L > *R* là một quan hệ trong *F*.

Để tính hàm *Y* = Find(*X*) bạn hãy tưởng tượng một kịch bản như sau:

Do nhóm X đứng ra rủ mọi người đi picnic nên lúc đầu nhóm X có mặt tại địa điểm xuất phát.

Ta gọi Y là nhóm người tham gia picnic: Lúc đầu ta có Y = X.

Trong số những người Y này có thể có một nhóm con L có khả năng rủ được nhóm R: ta thêm nhóm R vào Y. Hiện tượng này được diễn tả như sau:

```
for each quan hệ h: L > R not used in F do
  if L ⊆ Y then
    add R to Y;
```

Mỗi lần ta duyệt các quan hệ *L* > *R* trong *F*. Nếu *L* ⊆ *Y* tức là trong đoàn picnic hiện có nhóm *L* có thể rủ thêm được nhóm *R* thì ta thêm nhóm *R* này vào *Y*. Sau lần duyệt nào đó mà không thêm được ai thì ta dừng thuật toán và thu được kết quả là *Y*.

Để xác định được điều kiện dừng của thuật toán ta dùng mảng used đánh dấu những quan hệ đã dùng và biến count đếm số lần dùng mỗi quan hệ trong mỗi lần duyệt.

Ví dụ

Lớp U = ABCDEF
Các quan hệ F = {AB > C, D > E, E > BF}
Nhóm tổ chức picnic X = AD

Lúc tập trung đầu tiên ta có:

```
Y = X = AD.
Do D rủ được E nên Y = ADE và quan hệ D > E được đánh dấu used.
Do E rủ được BF nên Y = ABDEF và quan hệ E > BF được đánh dấu used.
Do AB rủ được C nên Y = ABCDEF và quan hệ AB > C được đánh dấu used.
Kết quả: Find(AD) = ABCDEF.
```

	A	**B**	**C**	**D**	**E**	**F**	used
X	A			D			Lúc đầu
Y	A			D	**E**		D > E
Y	A	**B**		D	E	**F**	E > BF
Y	A	B	**C**	D	E	F	AB > C
Kết quả	Find(AD) = ABCDEF						

```
Algorithm Find
Input: Tập X
Output:  Tập Y những bạn đi cùng X, kể cả X
begin
    mark all F as not used;
    Y ← X;
    repeat
```

```
        count = 0; // đếm số lần vận dụng một
                   // quan hệ trong một lần duyệt F
        for each quan hệ h: L > R not used in F do
            if L ⊆ Y then
                add R to Y;
                mark h as used;
                count ← count + 1;
            endif
        endfor
    until count = 0
    return  Y
end Find
```

Thuật toán có vận dụng *các phép toán tập hợp.*

Nếu *dung lượng* tối đa của tập là N, tức là mỗi tập có thể chứa tối đa N phần tử, thì các phép toán tập hợp có độ phức tạp O(N).

Thuật toán Find duyệt M quan hệ, mỗi lần duyệt có thể thêm được ít nhất là một phần tử trong số N phần tử, do đó cần M×N phép toán tập hợp. Mỗi phép toán tập hợp có độ phức tạp O(N). Tổng hợp lại, thuật toán Find có độ phức tạp tính toán là $O(M \times N^2)$.

Trong C++ và Python có sẵn kiểu tập hợp set.

Nếu bạn không muốn gọi các hàm có sẵn để xử lý tập hợp thì bạn có thể tự cài đặt. Việc này cũng đơn giản.

Các phép toán tập hợp

Cho X và Y là hai tập hợp con của tập U. Ta định nghĩa các phép toán trên các tập X và Y như sau:

Phép hợp: $Z = X \cup Y$ cho kết quả là tập Z chứa mọi phần tử của X và Y

$$Z = \{a \mid a \in X \text{ hoặc } a \in Y\}$$

Phép giao: $Z = X \cap Y$ cho kết quả là tập Z chứa các phần tử chung của X và Y

$$Z = \{a \mid a \in X \text{ và } a \in Y\}$$

Phép trừ (hiệu): $Z = X - Y$ cho kết quả là tập Z chứa các phần tử của riêng X (chỉ có trong X, không xuất hiện trong Y)

$$Z = \{a \mid a \in X \text{ và } a \notin Y\}$$

Phép so sánh: $X \subseteq Y$ cho giá trị là true nếu X là tập con của Y. Cụ thể là:

$$X \subseteq Y \text{ khi và chỉ khi } \forall a \in X \Rightarrow a \in Y$$

Trong C++ bạn có thể dùng mảng bitset để quản lý các tập. Bit i = 1 cho biết i là phần tử của tập; ngược lại, nếu bit i = 0 thì i không có mặt trong tập. Ta cũng quy định mã số của 'A' là 0, 'B' là 1, 'C' là 2,… Như vậy C++ sẽ quản lý các tập số tự nhiên từ 0 đến 25 ứng với các chữ cái A đến Z. Khi đó các thao tác cơ bản sẽ như sau:

C++

Khai báo kiểu tập:

```
const int SETSIZE = 26;
```

```
typedef bitset<SETSIZE> set;
```

Khai báo biến `x` kiểu set:

```
set x;
```

Khởi trị tập rỗng `x`:

```
x.reset();
```

Nạp ký tự `c` vào tập `x`:

```
x[c-'A'] = 1; hoặc
x.set(c-'A');
```

Hợp hai tập `x` và `y`:

```
z = x | y;
```

Giao hai tập `x` và `y`:

```
z = x & y;
```

Hiệu hai tập `x` và `y`:

```
z = x & (~y);
```

trong đó `~y` là phép lật (đảo) mọi bit của `y` từ `0` thành `1` và ngược lại. Thí dụ, nếu

```
y =  11001 thì
~y = 00110
```

Bạn có thể tự cài đặt các phép toán tập hợp theo các thuật toán sau:

Hiển thị tập

```
Algorithm PrintSet(s, msg) // Hiển thị tập s kèm chú thích msg
Input: set s
       string msg
Output:  none
begin
   write(msg);
   for each element x in s do
      write(x);
   end for
end PrintSet
```

Hợp hai tập

```
Algorithm SetUnion(x, y) // Hợp hai tập
Input: set x
       set y
Output:  set z = {a: a ∈ x or a ∈ y}
begin
   set z;
   z ← x;
   for each element a in y do
      add a to z;
```

```
    end for
   return z;
end SetUnion
```

Giao hai tập

```
Algorithm SetIntersection(x, y) // Giao hai tập
Input: set x
       set y
Output:  set z = {a: a ∈ x and a ∈ y}
begin
   set z;
   reset(z);
   for each element a in x do
      if a in y then
        add a to z;
      end if
   end for
  return z;
end SetIntersection
```

Hiệu hai tập

```
Algorithm SetDiff(x, y) // Hiệu hai tập
Input: set x
       set y
Output:  set z = {a: a ∈ x and a ∉ y}
begin
   set z;
   reset(z);
   for each element a in x do
      if a not in y then
        add a to z;
      end if
   end for
  return z;
end SetDiff
```

So sánh hai tập x ⊆ y?

```
Algorithm SetLeq(x, y)  // x ⊆ y?
Input: set x
       set y
Output:   (x ⊆ y) ? true: false
begin
   for each element a in x
      if (a not in y) then return false end if;
   end for
```

```
    return true;
  end SetLeq
```

So sánh hai tập x = y?

```
Algorithm SetEqual(x, y) // x = y?
Input: set x
       set y
Output:  (x = y) ? true: false
begin
   return SetLeq(x, y)  and SetLeq(y, x);
end SetEqual
```

Đọc dữ liệu

Với các bài toán có dữ liệu vào ghi trong file text ta có thể chọn một trong hai phương thức đọc dữ liệu sau đây:

Phương thức 1:

- Mở file f.
- Đọc dần dữ liệu và xử lý theo tiến độ của bài toán.
- Đóng file f.

Phương thức 2:

- Mở file f.
- Đọc toàn bộ dữ liệu vào các biến chương trình.
- Đóng file f.
- Giải bài toán.

Chương trình C++

```
// Picnic.cpp
#include <iostream>
#include <fstream>
#include <bitset>
#include <windows.h>

using namespace std;
   const int MN = 200; // max n, m
   const int SETSIZE = 26; // 26 chu cai A : Z
   const char * fn = "PICNIC.INP";
   typedef bitset<SETSIZE> set;
   // x[e] = e: so e co trong tap x
   // x[e] = 0: so e khong co trong tap x

  int n; // so hoc sinh
  int m; // so quan he
  set L[MN]; // ve trai
  set R[MN]; // ve phai
  set HS;// danh sach lop
  char hsDau; // hs dau danh sach
  char hsCuoi; // hs cuoi danh sach
```

```
ifstream f(fn); // input file

 // Hien thi tap x kem chu thich msg
   void printSet(const set & x, const char * msg = "") {
      cout << msg;
      for(int i = 0; i < SETSIZE; ++i)
         if (x[i])
            cout << (char)(i+'A');
   } // printSet

   // Hien thi quan he thu i, kem chu thich msg
   // L[i] > R[i]
   void printQH(int i, const char * msg = "") {
      cout << msg;
      printSet(L[i]);
      printSet(R[i], " > ");
   }

   // nap phan tu e vao s
   void ins(set &s, int e) {
      if (e >= hsDau && e <= hsCuoi)
        s.set(e-'A');
   } // ins

   // doc dong line tu input file
   // gan tri cho tap s: s = line
   void readSet(set & s) {
      char line[MN];
      s.reset(); // all 0
      f.getline(line,MN,'\n');
      for (int i = 0; i < strlen(line); ++i)
         ins(s, line[i]);
   } // readSet

   // doc du lieu
  void read() {
    char s[MN];
    f >> n >> m; // so hs, so quan he
    cout << "\n So HS: " << n;
    HS.reset();
    hsDau = 'A';
    hsCuoi = 'A' + (n-1);
      // HS: danh sach lop
    for (int i = 0; i < n; ++i)
      HS.set(i);
    printSet(HS,"\n Hoc sinh: ");
    cout << "\n HS dau: " << hsDau;
    cout << "\n HS cuoi: " << hsCuoi;
    cout << "\n So Quan he: " << m;
    f.get(); // new line
      for (int i = 0; i < m; ++i) {
         f.getline(s,MN,'\n'); // doc 1 dong
         int j;
         // L[i].reset(); R[i].reset();
         // Get the left side
         for (j = 0; s[j] != '>'; ++j)
            ins(L[i], s[j]);
         // Get the right side
         for (++j; j < strlen(s); ++j)
           ins(R[i], s[j]);
```

```
         printQH(i,"\n   ");
      } // for i
    } // read

  // x <= y?
   bool leq(const set &x, const set &y) {
      for (int i = 0; i < SETSIZE; ++i) {
         if (x[i]) { // i in x
            if (!y[i]) return false;
         } // if
      } // for
      return true;
   } // leq

   set find(set & x) {
      set y = x;
      printSet(y,"\n * Input y = ");
      int count;
      bitset<MN> used;
      while(true) {
         count = 0;
         // m quan he
         for (int i = 0; i < m; ++i) {
            if (used[i]) // da used: bo qua
              continue;
            if (leq(L[i], y)) {
               // L[i] <= y
               printQH(i,"\n    used ");
                  y |= R[i]; // union
                  used[i] = 1; // danh dau quan he i is used
                  ++count; // dem so lan dung
                  printSet(y," to get ");
            }
         } // for
        if (count == 0) break;
      } // while
   return y;
} // find

  void run() {
     read(); // doc du lieu
     int q; // soTest;
     f >> q;
     f.get();
     set x;
     for (int t = 1; t <= q; ++t) {
        readSet(x);
        cout << "\n\n Test No " << t << ": ";
        printSet(find(x), "\n * Result: ");
     }
     f.close();
   }  // run

main() {
  run();
  cout << "\n T h e   E n d ";
  return 0;
} // main
```

Dữ liệu test

PICNIC.INP
6 3 AB > C D > E E > BF 3 AD E CD

Kết quả

```
So HS: 6
Hoc sinh: ABCDEF
HS dau: A
HS cuoi: F
So Quan he: 3
  AB > C
  D > E
  E > BF

Test No 1:
* Input x = AD
   used D > E to get ADE
   used E > BF to get ABDEF
   used AB > C to get ABCDEF
* Result: ABCDEF

Test No 2:
* Input x = E
   used E > BF to get BEF
* Result: BEF

Test No 3:
* Input x = CD
   used D > E to get CDE
   used E > BF to get BCDEF
* Result: BCDEF
T h e   E n d
```

Key Group

Một lớp U có N em học sinh mang bí danh lần lượt là A, B, C,…Các bạn học sinh kết thân với nhau theo từng nhóm. Nhóm này có thể rủ nhóm khác để tạo thành một nhóm mới. Thí dụ, AD > E cho biết nhóm gồm hai bạn A và D có thể rủ được bạn E nhưng ngược lại thì không.

Một nhóm K các bạn trong lớp tạo thành nhóm chính nếu K thỏa mãn hai tính chất B1 và B2 sau đây:

(B1) K có thể rủ được cả lớp: Find(K) = U.

(B2) Thiếu bất kỳ bạn nào trong K thì số còn lại không thể rủ được cả lớp:

$\forall x \in K$: *Find(K–{x}) ≠ U.*

Hãy tìm một nhóm chính của lớp.

Dữ liệu vào

KEY.INP	Giải thích
6 3	6 học sinh 3 quan hệ
AB > C	Danh sách lớp: ABCDEF
D > E	Kết quả:
E > BF	Nhóm chính K = AD.

Thuật toán

Ta dùng thuật toán loại trừ khá đơn giản như sau.

Xuất phát từ K = U (cả lớp). Ta thấy, vì K = U nên K có thể rủ được cả lớp.

Duyệt từng phần tử x của K, nếu thử loại x khỏi K mà phần còn lại là K–{x} vẫn rủ được cả lớp thì ta loại x khỏi K; ngược lại, nếu Find(K–{x}) ≠ U thì ta trả lại x cho K.

Hệ thức: Find(K–{x}) = U được gọi là *bất biến* của phép duyệt.

```
Algorithm Key
Input: Tập U: các HS trong lớp.
       Tập F các quan hệ dạng L > R, L, R ⊆ U.
Output: Tập K ⊆ U thỏa
          Find(K) = U
          ∀ x ∈ K: Find(K-{x}) ≠ U
begin
   K ← U;
   for each x in U do
      if Find(K-{x}) = U then
         delete x from K
      end if
   end for
   return K
end Key
```

Ví dụ

Dữ liệu		Key
U = ABCDEF	Khởi trị	ABCDEF
AB > C	Xét A: Find(BCDEF) = BDEF ≠ U	ABCDEF (không loại A)
D > E	Xét B: Find(ACDEF) = U	ACDEF (Loại B)
E > BF	Xét C: Find(ADEF) = U	ADEF (loại C)
	Xét D: Find(AEF) = ABCEF ≠ U	ADEF (không loại D)
	Xét E: Find(ADF) = ABCDEF = U	ADF (loại E)
	Xét F: Find(AD) = ABCDEF = U	AD (loại F)
	Kết quả Key:	Key = AD

Dưới đây là một vài nhận xét quan trọng giúp bạn đẩy nhanh tốc độ xử lý.

Nhận xét

- Nếu một học sinh xuất hiện trong cả hai vế trái và phải của một quan hệ thì ta có thể bỏ học sinh đó khỏi về phải. Tức là nếu có quan hệ xL > xR, trong đó x là một học sinh xuất hiện trong cả hai vế, L và R là các nhóm học sinh, thì có thể thay quan hệ đó bằng quan hệ xL > R, vì hiển nhiên ta có xL > x (mỗi người tự rủ mình). Nhận xét này dẫn đến quy định là hai vế trái và phải của mọi quan hệ là rời nhau: $L \cap R = \varnothing$. Theo quy định này, khi đọc các quan hệ từ input file, ta cần thay L > R bằng L > R–L.
- Nếu học sinh nào không xuất hiện ở mọi vế phải thì không ai rủ được HS đó, do đó bạn ấy phải có mặt trong mọi key group.
- Công thức tìm các học sinh không xuất hiện ở mọi vế phải:
  - G = U – các vế phải.
  - Với ví dụ đã cho, ta tính được:
  - G = ABCDEF – {C} – {E} – {B, F} = AD
- Nếu G là tập các HS không xuất hiện ở mọi vế phải thì G là key group khi và chỉ khi Find(G) = U.
- Khi đã biết G phải có mặt trong mọi key thì lúc duyệt ta chỉ thử bỏ các phần tử không thuộc G.
- Nếu ta đã có bất biến K > U thì sau khi thử bỏ phần tử x khỏi K, thay vì kiểm tra Find(K–{x}) = U? ta chỉ cần kiểm tra $x \in$ Find(K–{x})?

```
Algorithm Key (phương án cải tiến)
Input: Tập U: các HS trong lớp.
       Tập F các quan hệ dạng L > R, L, R ⊆ U.
Output: Tập K ⊆ U  thỏa
          Find(K) = U
          ∀ x ∈ K: Find(K-{x}) ≠ U
begin
   G ← U - các vế phải;
   // xét trường hợp đặc biệt
   if Find(G) = U then
      return G;
   end if
```

```
    K ← U;
    for each x in U-G do
      if x ∈ Find(K-{x}) then
         delete x from K
       end if
    end for
    return K
end Key
```

Cài đặt

Ta cài đặt hàm bool TryDel(K,e) thực hiện thử bỏ phần tử e khỏi key K xem có thu lại được e hay không. Nếu được, hàm cho ra giá trị true, ngược lại, hàm cho ra giá trị false. Thuật toán này khá giống với thuật toán Find. Trước hết ta bỏ e khỏi K sau đó, trong quá trình duyệt các quan hệ L > R thỏa điều kiện L ⊆ K kéo theo việc hợp R vào K ta kiểm tra trước xem e ∈ R?

```
Algorithm tryDel(K, e)
Input: Tập K
Output:  e ∈ Find(K-{e})
begin
    mark all F as not used;
    delete e from K;
    repeat
        count = 0; // đếm số lần vận dụng một
                   // quan hệ trong một lần duyệt F
       for each quan hệ h: L > R not used in F do
          if L ⊆ Y then
             if  e ∈ R then
                 return true;
             end if
             add R to Y;
             mark h as used;
             count ← count + 1;
          endif
       endfor
    until count = 0
    return  false;
end tryDel
```

Khi đó, dạng cải tiến lần thứ hai cho Key sẽ như sau:

```
Algorithm Key (cải tiến lần 2)
Input: Tập U: các HS trong lớp.
       Tập F các quan hệ dạng L > R, L, R ⊆ U.
Output: Tập K ⊆ U  thỏa
          Find(K) = U
          ∀ x ∈ K: Find(K-{x}) ≠ U
begin
    G ← U - các vế phải;
```

```
    // xet truong hop dac biet
    if Find(G) = U then
        return G;
    end if
    K ← U;
    for each e in U-G do
      if tryDel(K, e) then
         delete e from K
       end if
    end for
    return K
end Key
```

Thuật toán Key gọi thuật toán Find hoặc tryDel tối đa N lần mỗi lần tốn thời gian cỡ MN^2, do đó thuật toán Key có độ phức tạp tính toán là $O(MN^3)$.

Chương trình C++

```
// Key.cpp
#include <iostream>
#include <fstream>
#include <bitset>
#include <windows.h>

using namespace std;
   const int MN = 200; // max n, m
   const int SETSIZE = 26; // 26 chu cai A : Z
   const char * fn = "KEY.INP";
   typedef bitset<SETSIZE> set;
   // x[e] = e: so e co trong tap x
   // x[e] = 0: so e khong co trong tap x

  int n; // so hoc sinh
  int m; // so quan he
  set L[MN]; // ve trai
  set R[MN]; // ve phai
  set HS;// danh sach lop
  char hsDau; // hs dau danh sach
  char hsCuoi; // hs cuoi danh sach
  ifstream f(fn); // input file

   // Hien thi tap x kem chu thich msg
      void printSet(const set & x, const char * msg = "") {
         cout << msg;
         for(int i = 0; i < SETSIZE; ++i)
            if (x[i])
               cout << (char)(i+'A');
      } // printSet

      // Hien thi quan he thu i, kem chu thich msg
      // L[i] > R[i]
      void printQH(int i, const char * msg = "") {
         cout << msg;
         printSet(L[i]);
         printSet(R[i], " > ");
       }

      // nap phan tu e vao s
```

```
  void ins(set &s, int e) {
    if (e >= hsDau && e <= hsCuoi)
      s.set(e-'A');
  } // ins

  // doc dong line tu input file
  // gan tri cho tap s: s = line
  void readSet(set & s) {
    char line[MN];
    s.reset(); // all 0
    f.getline(line,MN,'\n');
    for (int i = 0; i < strlen(line); ++i)
      ins(s, line[i]);
  } // readSet

  // doc du lieu
 void read() {
   char s[MN];
   f >> n >> m; // so hs, so quan he
   cout << "\n So HS: " << n;
   HS.reset();
   hsDau = 'A';
   hsCuoi = 'A' + (n-1);
     // HS: danh sach lop
   for (int i = 0; i < n; ++i)
     HS.set(i);
   printSet(HS,"\n Hoc sinh: ");
   cout << "\n HS dau: " << hsDau;
   cout << "\n HS cuoi: " << hsCuoi;
   cout << "\n So Quan he: " << m;
   f.get(); // new line
     for (int i = 0; i < m; ++i) {
       f.getline(s,MN,'\n'); // doc 1 dong
       int j;
       // L[i].reset(); R[i].reset();
       // Get the left side
       for (j = 0; s[j] != '>'; ++j)
          ins(L[i], s[j]);
       // Get the right side
       for (++j; j < strlen(s); ++j)
         ins(R[i], s[j]);

       R[i] &= ~L[i]; // x-y = x & (~y)

       printQH(i,"\n   ");
     } // for i
 } // read

// x <= y?
bool leq(const set &x, const set &y) {
  for (int i = 0; i < SETSIZE; ++i) {
    if (x[i]) { // i in x
      if (!y[i]) return false;
    }
  }
  return true;
} // leq

set find(set & x) {
  set y = x;
```

```
   int count;
   bitset<MN> used;
   while(true) {
      count = 0;
      // m quan he
      for (int i = 0; i < m; ++i) {
         if (used[i]) // da used: bo qua
           continue;
         if (leq(L[i], y)) {
            // L[i] <= y
               y |= R[i]; // union
               used[i] = 1; // danh dau quan he i is used
               ++count; // dem so lan dung
         }
      } // for
     if (count == 0) break;
   } // while
   return y;
} // find

// thu bo e khoi k
 // co timl lai duoc e?
bool tryDel(set k, int e) {
  cout << "\n Try to delete " << (char)(e+'A');
   if (0 <= e && e < n)
      k[e] = 0; // delete e
 int count;
 bitset<MN> used;
 while(true) {
   count = 0;
   // m quan he
   for (int i = 0; i < m; ++i) {
     if (used[i]) // da used: bo qua
          continue;
      if (leq(L[i], k)) {
         // L[i] <= x
         if (R[i][e]) return 1; // true: k chua e
            k |= R[i]; // them ve phai
            used[i] = 1; // danh dau quan he i is used
            ++count; // dem so lan dung
         }
     } // for
     if (count == 0) break;
   } // while
   return 0; // false
} // try(k,e)

set Key() {
 // g = ko xuat hien trong cac ve phai
 set g = HS;
 for (int i = 0; i < m; ++i)
   g &= ~R[i]; // g = g - R[i]
 if (find(g) == HS) {
   printSet(g, "\n Truong hop dac biet: ");
      return g;
   }
 set k = HS; // Xuat phat tu ca lop
   for (int i = 0; i < n; ++i) {
      if (!g[i]) {
        if (tryDel(k,i))
```

```
            k[i] = 0; // delete i
         }
      }
      return k;
   }

   void run() {
      read();
      f.close();
      printSet(Key(), "\n Key: ");
    }  // run

main() {
  run();
  cout << "\n T h e   E n d ";
  return 0;
} // main
```

Dữ liệu test 1

KEY.INP
6 3
AB > C
D > E
E > BF

Kết quả Test 1

```
So HS: 6
Hoc sinh: ABCDEF
HS dau: A
HS cuoi: F
So Quan he: 3
  AB > C
  D > E
  E > BF
Truong hop dac biet: AD
Key: AD
T h e   E n d
```

Dữ liệu test 2

KEY.INP
6 4
AD > F
B > C
AF > D
E > B

Kết quả Test 2

```
So HS: 6
Hoc sinh: ABCDEF
HS dau: A
HS cuoi: F
So Quan he: 4
  AD > F
  B > C
  AF > D
  E > B
```

```
Try to delete B
Try to delete C
Try to delete D
Try to delete F
Key: AEF
T h e   E n d
```