CẶP GHÉP TRONG ĐỒ THỊ KHÔNG CÓ TRỌNG SỐ

A, Định nghĩa cặp ghép

Xét hai tập hữu hạn X, Y gồm n phần tử:

$X = \{x_1, x_2,...,x_n\}$

$Y = \{y_1, y_2,...,y_n\}$

Cặp phần tử (x,y) với x thuộc X, y thuộc Y được gọi là một cặp ghép. Hai cặp ghép (x,y) và (x', y') được gọi là rời nhau nếu x # x' và y # y'. Tập M gồm các cặp ghép rời nhau được gọi là một tập cặp ghép.

{ Thông thường bài toán xây dựng các cặp ghép được tiếp cận theo 2 hướng hoặc thỏa mãn một điều kiện ghép cặp nào đấy, khi đó người ta quan tâm đến khả năng ghép cặp tối đa, hoặc lượng hoá việc ghép cặp, khi đó người ta quan tâm đến phương án ghép cặp tối ưu theo các giá trị đã lượng hoá. }

Vì số tập cặp ghép là hữu hạn, nên một phương pháp xây dựng đơn giản là thử tất cả các khả năng. Tuy nhiên, số khả năng như vậy là rất lớn. Vì thế, phải tìm kiếm những thuật giải thật hữu hiệu, dễ cài đặt chương trình và có tính khả thi cao.

Bài toán tìm cặp ghép đầy đủ tối ưu

a. <u>Giới thiệu bài toán</u>: Bài toán tìm cặp ghép đầy đủ tối ưu có nhiều mô hình ứng dụng thực tế. Một trong những mô hình này là người ta quan tâm đến việc ghép cặp sao cho có hiệu quả nhất.

Để lượng hoá việc ghép mỗi phần tử x thuộc X với một phần tử y thuộc Y, người ta đưa vào ma trận số C_{ij} (i,j = 1, 2,..., n) với ý nghĩa C_{ij} mô tả hiệu quả của việc ghép x_i với y_j. Bài toán được đặt ra là: Xây dựng một tập cặp ghép đầy đủ có tổng hiệu quả lớn nhất.

Bài toán vừa nêu thường được phát biểu dưới dạng một mô hình thực tế là bài toán phân công:

Bài toán phân công: *Có n người và n công việc. Biết C_{ij} là số tiền làm ra nếu giao công việc j cho người i thực hiện. Hãy tìm cách phân công mỗi người mỗi việc để tổng số tiền làm ra là lớn nhất*.

b. <u>Định lý về cặp ghép:</u> Việc xây dựng tập cặp ghép đầy đủ tối ưu dựa vào dấu hiệu nhận biết một tập cặp ghép đầy đủ khi nào là tối ưu. Dĩ nhiên việc thử dấu hiệu này không phải là việc so sánh với tất cả các cặp ghép, mà phải được xây dựng mang tính khả thi. Để làm điều này, người ta xây dựng hàm số F, xác định trên tập các phần tử x_i thuộc X, y_j thuộc Y, mà ta sẽ gọi là nhãn của các phần tử. Nhãn F được gọi là chấp nhận được nếu thoả mãn bất đẳng thức

$F(x_i)+F(y_j)>=C_{ij}$ với mọi x_i thuộc X, y_j thuộc Y. Tập cặp ghép M và nhãn F được gọi là tương thích với nhau nếu thoả mãn đẳng thức $F(x_i)+F(y_j)=C_{ij}$ với mọi x_i thuộc X, y_j thuộc Y. Nói riêng, tập cặp ghép rỗng được xem như tương thích với mọi nhãn.

Định lý: *Tập cặp ghép đầy đủ M là tối ưu khi tồn tại nhãn F chấp nhận được là tương thích với nó.*

c. Thuật toán Kuhn-Munkres: Nội dung chủ yếu của phương pháp là xuất phát từ một tập cặp ghép nào đó chưa đầy đủ (có thể là rỗng), ta tăng dần số cặp ghép sao cho khi trở thành đầy đủ, các cặp ghép thu được cũng đồng thời thoả mãn tính tối ưu. Có nhiều hình thức trình bày phương pháp này. Dưới đây là cách trình bày trên ngôn ngữ đồ thị kèm với việc dùng thuật toán tìm đường đi.

Giả sử F là một nhãn chấp nhận được và M là tập cặp ghép tương thích với F. Xem các phần tử của X và Y như những đỉnh của một đồ thị có hướng hai phía. Các cạnh của đồ thị này được xác định tuỳ thuộc nội dung của nhãn F và tập cặp ghép M như sau:

- Mỗi cặp phần tử x_i thuộc X, y_j thuộc Y thoả mãn đẳng thức $F(x_i)+F(y_j) = C_{ij}$ sẽ xác định một cạnh của đồ thị

- Cạnh này có hướng từ X sang Y nếu cặp (x_i,y_j) không thuộc M và ngược lại.

Đồ thị xây dựng theo quy tắc vừa nêu được gọi là đồ thị cân bằng tương ứng với F, M và được ký hiệu là G(F, M).

Bước 1. Khởi tạo:

Xây dựng nhãn F chấp nhận được như sau:

$F(x_i):=Max\{C_{ij}, y_j$ thuộc Y$\}$ x_i thuộc X

$F(y_j):=0$ y_j thuộc Y

M là tập cặp ghép rỗng.

Bước 2. Tìm đỉnh tự do thuộc X:

Tìm đỉnh u thuộc X chưa được ghép cặp. Nếu không còn đỉnh nào của X chưa ghép cặp thì kết thúc: tập cặp ghép M hiện hành là tập cặp ghép đầy đủ tối ưu. Trái lại sang bước kế tiếp.

Bước 3. Tìm đường tăng cặp ghép:

Xuất phát từ u, thực hiện việc tìm kiếm trên đồ thị G(F, M). Kết quả tìm kiếm có hai trường hợp:

- Nếu đến được một đỉnh z thuộc Y chưa ghép cặp thì ghi nhận đường đi từ u đến z và chuyển sang bước tăng cặp ghép trên đường đi này.

- Nếu không tồn tại một đường đi như vậy thì chuyển sang bước sửa nhãn F.

Bước 4. Tăng cặp ghép: điều chỉnh M như sau:

- Giữ nguyên những cặp ghép của M nằm ngoài đường tăng cặp ghép.

- Trên đường tăng cặp ghép, bỏ đi những cặp ghép của M là cạnh ngược và thêm vào M những cặp ghép là cạnh thuận.

Sau bước này, số cặp ghép thuộc M được tăng thêm 1 và đỉnh u được bảo toàn. Sau đó quay về bước 2 để lặp lại với đỉnh tự do khác.

Bước 5. Sửa nhãn:

Gọi S là tập các đỉnh thuộc X và T là tập các đỉnh thuộc Y đã được đi đến trong quá trình tìm kiếm ở bước 3.

Việc sửa nhãn F được tiến hành như sau:

- Tìm lượng sửa nhãn:

$d:= \min\{F(x_i) + F(y_j) - C_{ij}, x_i \text{ thuộc } S, y_j \text{ thuộc } T\}$

- Gán lại nhãn:

$F(x_i):=F(x_i) - d$ với x_i thuộc S

$F(y_j):=f(y_j) + d$ với y_j thuộc T

Sau đó, quay về bước 3 để lặp lại việc tìm đường tăng cặp ghép.

Chú ý rằng, sau khi thay đổi, nhãn F vẫn giữ nguyên tính chấp nhận được và tính tương thích với M. Ngoài ra có thêm ít nhất một cặp (x_i, y_j) thoả mãn $F(x_i)+ F(y_j) = C_{ij}$, vì thế, sau một số lần sửa nhãn chắc chắn sẽ tăng được cặp ghép. Độ phức tạp của thuật toán là $O(n^3)$

Code này của Nguyễn Tiến Trung Kiên

```
#include <stdio.h>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
```

```
const int N = 102;
int n, m, Assigned[N];
int Visited[N], t=0;
vector<int> a[N];

bool visit(int u) {
   if (Visited[u]!=t)
      Visited[u]=t;
   else
      return false;

   for (int i=0; int v=a[u][i]; i++)
   if (!Assigned[v] || visit(Assigned[v])) {
      Assigned[v]=u;
      return true;
   }
   return false;
}

main() {
   scanf("%d%d", &m, &n);
   int x, y;
   while (scanf("%d%d", &x, &y) > 0)
      a[x].push_back(y);
   for (int i=1; i<=m; i++)
      a[i].push_back(0);

   int Count = 0;
   for (int i=1; i<=m; i++) {
      t++;
      Count += visit(i);
   }
   printf("%d\n", Count);
   for (int i=1; i<=n; i++)
   if (int j=Assigned[i])
   printf("%d %d\n", j, i);
}
```

Cách thứ 2: Của thầy Đỗ Đức Đông tư tưởng là tham trước ghép sau thì có thể giảm được số lượng đỉnh cần ghép chương trình chạy nhanh hơn.

Cách thứ 3: Thuật toán ghép nhanh Hopkroft Karp độ phức tạp O(nx$\sqrt{m}$)

Code này của thầy Lê Thanh Bình - THPT chuyên Nguyễn Trãi Hải Dương

```
// Program: Hopkroft Karp
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <algorithm>
#include <vector>
 #define MAXN 50005
#define INF 1000000007
#define tr(i,c) for(typeof((c).begin()) i=(c).begin();i!=(c).end();i++)
 using namespace std;
 int m,n,res=0;
vector<int> g[MAXN];
int x[MAXN],y[MAXN],d[MAXN], q[MAXN];
 bool FindPath() {
   int L=1, R=0;
   for(int u=1;u<=n;u++)
   if (x[u]==0) {
      d[u]=0;
      q[++R]=u;
   } else d[u]=INF;
   d[0]=INF;
   while (L<=R) {
```

```
        int u=q[L++];
        tr(i,g[u]) {
            int v=*i;
            if (d[y[v]]==INF) {
                d[y[v]]=d[u]+1;
                if (y[v]) q[++R]=y[v];
            }
        }
    }
    return (d[0]!=INF);
}
bool dfs(int u) {
    //if (u==0) return(true);
    tr(i,g[u]) {
        int v=*i;
        if (y[v]==0) {
            x[u]=v; y[v]=u;
            d[u]=INF;
            return true;
        }
        if (d[y[v]]==d[u]+1)
        if (dfs(y[v])) {
            x[u]=v; y[v]=u;
```

```
            d[u]=INF;
            return true;
        }
    }
    d[u]=INF;
    return false;
}

void GhepMax() {
    while (FindPath()) {
        for(int u=1;u<=n;u++)
            if (x[u]==0)
                if (dfs(u)) res++;
    }
    printf("%d",res);
}
int main() {
    #ifndef ONLINE_JUDGE
        freopen("inp.txt","r",stdin);
        freopen("out.txt","w",stdout);
    #endif // ONLINE_JUDGE
    int p;
    scanf("%d%d%d",&n,&m,&p);
```

```
  for(int i=1;i<=p;i++) {
     int u,v; scanf("%d%d",&u,&v);
     g[u].push_back(v);
  }
  GhepMax();
}
```

B, MỘT SỐ BÀI TẬP ÁP DỤNG

Bài 1: Hát giao duyên - Đề HSG Duyên Hải 2016

Lễ hội "Hát giao duyên" được tổ chức hàng năm ở nhiều vùng quê. Năm nay, Hiếu được tham gia tổ chức lễ hội ở quê hương mình. Có m chàng trai và n cô gái đăng ký tham gia lễ hội, mỗi người đăng ký hát một bài hát. Chàng trai thứ i đăng ký hát bài có mã số a_i (i = 1, 2,…, m), cô gái thứ j đăng ký hát bài có mã số b_j (j = 1, 2, …, n). Sau khi thu thập đầy đủ thôngtin đăng ký, Hiếu cần giúp Ban tổ chức sắp xếp các chàng trai và các cô gái thành các cặp biểu diễn, mỗi cặp gồm một chàng trai và một cô gái, mỗi người đăng ký một bài hát khác nhau. Mỗi chàng trai và mỗi cô gái chỉ thuộc không quá một cặp biểu diễn. Lễ hội sẽ càng vui và hấp dẫn nếu có được càng nhiều cặp biểu diễn.

Yêu cầu: Cho $a1$, $a2$,…, am và $b1$, $b2$,…, bn, hãy giúp Hiếu sắp xếp để có nhiều cặp biểu diễn nhất thỏa mãn điều kiện đặt ra.

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

- Dòng thứ nhất chứa hai số nguyên dương m, n;
- Dòng thứ hai chứa m số nguyên dương $a1$, $a2$,…, am ($1 \le a1, a2,\ldots, am \le 10000$);
- Dòng thứ ba chứa n số nguyên dương $b1$, $b2$,…, bn ($1 \le b1, b2,\ldots, bn \le$ 10000). Hai số liên tiếp trên cùng dòng được ghi cách nhau bởi dấu cách.

Kết quả: Ghi ra file văn bản LOVESONG.OUT:

- Dòng đầu tiên ghi một số nguyên k là số lượng cặp biểu diễn nhiều nhất xếp được thỏa mãn điều kiện đặt ra.
- Dòng thứ r trong k dòng tiếp theo mô tả cặp biểu diễn thứ r, dòng chứa hai số nguyên dương i_r và j_r có nghĩa là chàng trai i_r ghép cặp với cô gái j_r.

Ví dụ:

LOVESONG.INP	LOVESONG.OUT
3 3 1 1 2 2 1 1	2 1 1 3 2

Ràng buộc:

- Có 30% số test ứng với 30% số điểm có *m, n* ≤ 10;
- Có 30% số test khác ứng với 30% số điểm có *m, n* ≤ 100;
- Có 40% số test còn lại ứng với 40% số điểm có *m, n* ≤ 10000.

*** Thuật toán:**

- Sub 1 :

Vì m, n ở sub này rất bé thế nên ta sẽ dùng cách duyệt với độ phức tạp là O(2^n * 2^m * (m+n)).

Với mỗi trạng thái của số người nam và số người nữ, ta xem ở trạng thái này số bit 1 của số người nào nhỏ hơn thì kiểm tra trên số người đó, nếu có người nào không thể ghép thì trạng thái đấy không hợp lệ, còn nếu hợp lệ thì sẽ so sánh với kết quả lấy max của kết quả trước so với số bit 1 nhỏ hơn.

- Sub 2 :

Vì m, n <= 100 cho nên ta sẽ dùng thuật toán cặp ghép với độ phức tạp m^3, với mỗi cô gái sẽ được ghép với 1 chàng trai khác số bài hát, khi đó thuật toán sẽ cho ta số cặp nhiều nhất mà có thể ghép được.

*** Nhận xét:** bộ test bài này yếu nên trong kì thi năm 2016 nhiều học sinh dùng thuật toán tham đã AC bài này tuy nhiên với thuật toán tham vẫn có thể chỉ ra các test chết ví dụ: 3 3 (1 2 4, 2 1 4) đầu ra đúng là 3 còn cách tham ra 2. Trong kỳ thi DH2016 có 10 thí sinh AC bài này tuy nhiên đều làm bằng cách tham.

Ta sẽ tham như sau: for 2 vòng, 1 vòng chạy số nữ và 1 vòng chạy số nam, nếu số hiệu người nữ đang xét, khác số hiệu người nam đang xét và cả 2 đều chưa ghép cặp với ai thì ta sẽ đánh dấu lại cả 2 người và tăng kết quả lên 1 , đồng thời sẽ chuyển luôn qua người nữ tiếp theo.

```
for (int i = 1 ; i < = n ; ++i)
for (int j = 1; j <= m ; ++j) if (a[i] != b[j] && !dd[j])
        {
        ans++;
        dd[j] = 1;
        break;
        }
```

Chương trình code theo tư tưởng cặp ghép của thầy Đông:

```
#include<bits/stdc++.h>
using namespace std;
#define Fr(i,a,b) for(int i=a;i<=b;i++)
#define For(i,a,b) for(int i=a;i<b;i++)
#define Frd(i,a,b) for(int i=a;i>=b;i--)
#define Ford(i,a,b) for(int i=a;i>b;i--)
#define fi first
#define se second
#define bit1(x) __builtin_popcount(x)   ///bit1 cua 32bit
#define bit1ll(x) __builtin_popcountll(x)  ///bit1 cua 64bit
#define endl '\n'
#define maxn 10005
typedef long long ll;
typedef pair<int,int> ii;
const int oo=INT_MAX;
const int mod=1e9+7;
int m,n,b[maxn],c[maxn],num[maxn],dem,ans,ghep[maxn];
vector<int> a[maxn];
bool dd[maxn];

bool visit(int u)
{
   if(num[u]!=dem) num[u]=dem;
   else return 0;
   For(i,0,a[u].size())
   {
      int v=a[u][i];
      if(!ghep[v] || visit(ghep[v]))
      {
         ghep[v]=u;
         return 1;
      }
   }
   return 0;
}

void trau()
{
   Fr(i,1,m) Fr(j,1,n) if(b[i]!=c[j])
   {
      a[i].push_back(j);
      if(!dd[i]&&!ghep[j])
      {
```

```
            ++ans;
            dd[i]=1,ghep[j]=i;
        }
    }
    Fr(i,1,m)
    {
        ++dem;
        if(!dd[i])ans+=visit(i);
    }
    printf("%d\n",ans);
    Fr(i,1,n) if(ghep[i]) printf("%d %d\n",ghep[i],i);
}

int main()
{
    freopen("lovesong.inp","r",stdin);
    freopen("lovesong.out","w",stdout);
    scanf("%d %d",&m,&n);
    Fr(i,1,m) scanf("%d",&b[i]);
    Fr(i,1,n) scanf("%d",&c[i]);
    trau();
            return 0;
}
```

Chương trình code theo tư tưởng Hopkroft Karp

```
#include <bits/stdc++.h>
#define f first
#define s second
#define ii pair <int, int>
#define HAN "lovesong"
#define pb push_back
#define mp make_pair
#define MOD 1000000009
#define ll long long
#define lli pair <long long , int>

using namespace std;
const int N = 1e4 + 5;
int m , n , res;
vector<int> a[N];
int dd[N], y[N], d[N], q[N], b[N] , c[N];

bool find_path() {
```

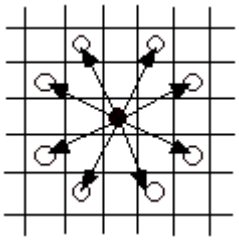
```
    int L = 1, R = 0;
    for(int u = 1; u <= n ; u++)
    if (!dd[u]) {
       d[u] = 0;
       q[++R] = u;
    } else d[u] = MOD;
    d[0] = MOD;
    while (L <= R) {
       int u=q[L++];
       for (int i = 0 ; i < a[u].size() ; ++i)
       {
          int v = a[u][i];
          if (d[y[v]]== MOD) {
             d[y[v]] = d[u]+1;
             if (y[v]) q[++R] = y[v];
          }
       }
    }
    return (d[0]!=MOD);
}

bool dfs(int u)
{
  for (int i = 0 ; i < a[u].size() ; ++i)
  {
     int v = a[u][i];
      if (!y[v]) {
         dd[u] = v;
         y[v] = u;
         d[u] = MOD;
         return 1;
      }
      if (d[y[v]] == d[u]+1)
      if (dfs(y[v])) {
         dd[u] = v;
         y[v] = u;
         d[u] = MOD;
         return 1;
      }
    }
    d[u] = MOD;
    return 0;
}
```

```
void setup()
{
   while (find_path()) {
      for(int i = 1; i <= n ; ++i)
         if (!dd[i])
            if (dfs(i)) res++;
   }
   printf("%d\n",res);
   for (int i = 1 ; i <= m ; ++i) if (y[i]) printf("%d %d\n" , y[i] , i);
}

int main()
{
   ios_base::sync_with_stdio(0);
   freopen(HAN".inp","r",stdin);
   freopen(HAN".out","w",stdout);
   scanf("%d%d", &n , &m);

   for (int i = 1 ; i <= n; ++i) scanf("%d" , &b[i] );
   for (int i = 1 ; i <= m; ++i) scanf("%d" , &c[i] );

   for (int i = 1; i <= n; ++i)
   for (int j = 1; j <= m ; ++j) if (b[i] != c[j]) a[i].pb(j);
   setup();
}
```

Bài 2: Xếp hàng 2 - Đỗ Đức Đông

Cho bàn cờ có kích thước *m*×*n*, các hàng được đánh số từ 1 đến *m*, các cột được đánh số từ 1 đến

n. Trên bàn cờ có một số ô cấm những ô còn lại là những ô tự do – ô có thể di chuyển vào được. Có một số quân mã đang đứng ở các ô tự do trên bàn cờ.



Quy tắc di chuyển của
quân mã

Tại mỗi bước, một quân mã có thể di chuyển (theo luật di chuyển của quân mã) vào ô

tự do không có quân mã khác đứng. Hãy tìm cách di chuyển tất cả các quân mã để chúng xếp thành một hàng ngang liên tiếp trên bàn cờ sao cho quân di phải chuyển xa nhất là nhỏ nhất.

Input

- Dòng đầu là hai số *m*, *n*;
- *m* dòng tiếp theo, mỗi dòng 1 xâu *n* ký tự, gồm các ký tự "." thể hiện ô trống, "#" thể hiện ô cấm không được phép đi vào, "M" thể hiện vị trí quân mã đang đứng.

Output

- Ghi số *k* là số bước ít nhất cần di chuyển.

Line2.INP	Line2.OUT
2 5 M..M.	1

Subtask 1: *m, n* ≤ 20 và có 2 quân mã;

Subtask 2: *m, n* ≤ 50 và có 3 quân mã;

Subtask 3: *m, n* ≤ 50 và có không quá 5 quân mã;

Subtask 4: *m, n* ≤ 100 và có không quá 50 quân mã;

- **Thuật toán:**
  - Đầu tiên, ta sẽ đánh dấu tất cả các quân mã lại. Từ mỗi quân mã, ta sẽ tìm đến từng vị trí không bị cấm trên quân cờ sao cho số lần đi là nhỏ nhất.
  - Tiếp theo, ta sẽ tạo lại 1 đồ thị mà có cạnh là quân mã và ô cờ trên bàn cờ.
  - Từ số quân mã trên bàn cờ, ta sẽ tìm 1 hàng ngang liên tiếp cho đủ số quân mã. Với một hàng này, ta sẽ thử tìm kiếm nhị phân theo kết quả và tạo thêm một đồ thị nữa để thực hiện cặp ghép trên đồ thị để xem kết quả tìm kiếm nhị phân có thỏa mãn hay không. Từ đó ta có kết quả nhỏ nhất khi cho những quân mã vào một hàng ngang, ta chỉ việc lấy kết quả nhỏ nhất của những hàng ngang này là sẽ có kết quả.

Chương trình:

```
#include <bits/stdc++.h>
#define ii pair<int, int>
```

```
#define x first
#define y second
#define oo 1000000005
using namespace std;

const int N = 102;
const int K = 51;

int dx[] = { 1, 2, 2, 1, -1, -2, -2, -1};
int dy[] = { -2, -1, 1, 2, 2, 1, -1, -2};

int n, m, a[N][N], pre[K], l[K], r[K], ok, b[N][N], d[K][N][N], k;
ii h[K], f[K];
vector<ii> OLD, NEW;

void change(int j)
{
   int i, tmp;
   while(j) {
      i = pre[j];
      tmp = r[j];
      l[i] = j;
      r[j] = i;
      j = tmp;
   }
}

void ghep(int i)
{
   if (ok) return;
   for(int j = 1; j <= k; j ++)
   if (a[i][j] && !pre[j]) {
      pre[j] = i;
      if (r[j] == 0) {
         change(j);
         ok = 1;
         return;
      }
      else ghep(r[j]);
      r[j] = i;
      l[i] = j;
      if (ok) return;
   }
}
```

```
int Check(int x,int y)
{
   if (x < 1 || m < x || y < 1 || n < y) return 0;
   if (b[x][y] == -1) return 0;
   return 1;
}

void bfs(int id, int x, int y)
{
   OLD.push_back( ii( x, y));
   int len = 0;
   d[id][x][y] = -1;
   while(OLD.size()) {
      len ++;
      for(int i = 0; i < OLD.size(); i ++) {
         int ux = OLD[i].x;
         int uy = OLD[i].y;
         for(int j = 0; j < 8; j ++) {
            int vx = ux + dx[j];
            int vy = uy + dy[j];
            if (!Check(vx, vy) || d[id][vx][vy]) continue;
            d[id][vx][vy] = len;
            NEW.push_back( ii( vx, vy));
         }
      }
      OLD = NEW;
      NEW.clear();
   }
   for(int i = 1; i <= m; i ++)
   for(int j = 1; j <= n; j ++) if (!d[id][i][j]) d[id][i][j] = oo;
   d[id][x][y] = 0;
}

int check(int val)
{
   for(int i = 1; i <= m; i ++)
   for(int j = 1; j <= n; j ++) {
      int mok = 1;
      for(int t = 1; t <= k; t ++) {
         f[t] = ii( i, j + t - 1);
         if (!Check(f[t].x, f[t].y)) mok = 0;
      }
      if (!mok) continue;
```

```
        ///
        memset(a, 0, sizeof(a));
        for(int ti = 1; ti <= k; ti ++)
        for(int tj = 1; tj <= k; tj ++) {
            ii u = h[ti];
            ii v = f[tj];
            if (d[ti][v.x][v.y] == oo) continue;
            if (d[ti][v.x][v.y] <= val) a[ti][tj] = 1;
        }
        ///
        memset(l, 0, sizeof(l));
        memset(r, 0, sizeof(r));
        for(int t = 1; t <= k; t ++) {
            memset(pre, 0, sizeof(pre));
            ok = 0;
            ghep(t);
        }
        mok = 1;
        for(int t = 1; t <= k; t ++) if (!l[t] || !r[t]) mok = 0;
        if (mok) return 1;
    }
    return 0;
}

int main()
{
    ios_base::sync_with_stdio(0);

    freopen("line2.inp","r",stdin);
    freopen("line2.out","w",stdout);

    cin >> m >> n;
    for(int i = 1; i <= m; i ++)
    for(int j = 1; j <= n; j ++) {
        char ch;
        cin >> ch;
        b[i][j] = -1;
        if (ch == '#') continue;
        b[i][j] = 0;
        if (ch == '.') continue;
        b[i][j] = ++ k;
        h[k] = ii( i, j);
    }
```

```
    for(int i = 1; i <= k; i ++) bfs(i, h[i].x, h[i].y);

    int left = 0;
    int right = m * n;
    int mid, ans;
    while(left <= right)
    {
        mid = (left + right) /2;
        if (check(mid)) ans = mid, right = mid - 1;
        else left = mid + 1;
    }

    cout << ans;
    return 0;
}
```

Bài 3: Exam - VOI2016 - Tạo đề thi

Sơn đang chuẩn bị đề thi cho đợt tập huấn học sinh tham dự kỳ thi học sinh giỏi toán. Đề thi gồm n câu hỏi. Trong mỗi câu hỏi học sinh cần phải thực hiện việc tính giá trị của một biểu thức số học gồm không quá 4 toán hạng với các phép toàn cộng (+), trừ (-) hoặc nhân (*) giữa chúng. Sơn đã chuẩn bị n biểu thức như vậy với các toán hạng là các số nguyên không âm. Với cùng một biểu thức, có thể đặt thêm các cặp dấu ngoặc để chỉ ra trình tự thực hiện các phép toán và có thể dẫn đến các kết quả khác nhau.

Ví dụ:

- Với n=3 và các biểu thức được chuẩn bị là:

2*3-4; 2*3-4; 3*2-4;

Thì một đề thi đáp ứng yêu cầu đặt ra là:

2*3-4 = 2; 2*(3-4) = -2; 3*(2-4) = -6;

- Xét một ví dụ khác, với n=4 và các biểu thức là:

2*3-4; 2*3-4; 3*2-4; 3*2-4.

Với biểu thức 2*3-4 chỉ có hai trình tự tính toán dẫn đến kết quả là 2 và -2, còn đối với biểu thức 3*2-4 cũng chỉ có hai trình tự tính toán dẫn đến kết quả là 2 và -6. Như vậy, đối với các biểu thức đã cho chỉ có thể tạo tối đa 3 kết quả khác nhau, trong khi đó đề thi lại đòi hỏi đưa ra 4 câu hỏi với kết quả khác nhau. Vì vậy, đối với ví dụ này, câu trả lời là không thể tạo đề thi đáp ứng yêu cầu đặt ra.

Yêu cầu: Hãy giúp Sơn tạo đề thi với n biểu thức số học chọn trước đáp ứng yêu cầu đặt ra.

Dữ liệu:

- Dòng đầu tiên chứa số nguyên dương n là số lượng biểu thức số học;
- Dòng thứ i trong số n dòng tiếp theo chứa một biểu thức gồm ít nhất là hai và nhiều nhất là 4 toán hạng, mỗi toán hạng là số nguyên không âm không vượt quá 10^6, trong đó các toán hạng và phép toán được viết liên tiếp nhau không có dấu cách phân tách.

Kết quả: Ghi ra n dòng, mỗi dòng chứa một biểu thức (có thể có hoặc không có các dấu ngoặc) trong đề thi mà bạn tạo ra để đáp ứng yêu cầu đã nêu, trong đó các toán hạng, phép toán và dấu ngoặc được ghi liên tiếp nhau không có dấu phân tách. Nếu có nhiều cách tạo đề thi đáp ứng yêu cầu thi hãy đưa ra một cách tùy ý. Nếu câu trả lời là không thể tạo được đề thi đáp ứng yêu cầu thì ghi ra thông báo 'NO SOLUTION'.

Ví dụ:

EXAM.INP	EXAM.OUT	EXAM.INP	EXAM.OUT
3 2*3-4 2*3-4 3*2-4	2*3-4 2*(3-4) 3*(2-4)	4 2*3-4 2*3-4 3*2-4 3*2-4	NO SOLUTION

Ràng buộc:

- Có 50% số test ứng với 50% số điểm của bài có n<=20 và mỗi biểu thức gồm đúng 3 toán hạng.
- Có 50% số test khác ứng với 50% số điểm của bài có n<=2000.

- **Thuật toán:**

- Với sub đầu tiên, ta có thể xử lý rất dễ bởi có không quá 3 toán hạng và 20 biểu thức, vì vậy ta có thể sử dụng thuật toán duyệt đệ quy vì mỗi biểu thức có không quá 2 cách đặt dấu ngoặc.
- Với sub còn lại ta có thể xử lý như sau:
  - Đầu tiên từ mỗi biểu thức ta sẽ đặt thử xem sẽ có thể tạo ra những kết quả nào. Vì có thể có 4 toán hạng mà mỗi số có thể lên đến 10^6, vậy nên kết quả có thể sẽ vượt qua 2^{64}, thế nên ta có thể để nó tự tràn số hoặc nếu muốn có thể mod cho chắc.
  - Ta sẽ tạo một đồ thị mà có các cạnh là biểu thức nào sẽ tạo ra kết quả nào. Để ý rằng với mỗi biểu thức có 4 toán hạng ta có thể xử lý theo cách đặt ngoặc theo tất cả các trường hợp xảy ra, do đó nên nới giới hạn đặt mảng lên khoảng 20000 trở lên.

- Tiếp theo ta chỉ việc dùng thuật toán tìm số cặp ghép tối đa để tìm xem liệu có đủ n cặp ghép không, nếu có đủ thì thông báo “YES” và truy vết, nếu không thì in ra ”NO”.
- **Nhận xét:** bài này phải sử dụng thuật toán Hopcroft-Karp để AC hoặc ghép theo cách của thầy Đông tham trước ghép sau tuy nhiên chạy chậm hơn.

Chương trình:

```
#include <bits/stdc++.h>

#define pb push_back

using namespace std;

const int N=5e3+2;

const int M=1e5+2;

const int MOD=1e9+7;

int n,dem,t;

int d[M],dd[M],bd[M],kt[M],X[M],Y[M],dist[M],q[M];

int P[N][10],Q[N][10];

vector<int> vt[N];

string s;

string tv[N];

struct st

{

   int x,y;

   string z;

}f[M];



int tinh2(int x, int y, int z)
```

```
{
   if ( s[z] == '*' ) return x*y;
   if ( s[z] == '+' ) return x+y;
   return x-y;
}

int tinh3(int x ,int y, int z, int h, int k)
{
   if ( s[h] == '*' ) return ( tinh2(x*y,z,k) );
   if ( s[k] == '*' ) return ( tinh2(x,y*z,h) );
   return tinh2(tinh2(x,y,h),z,k);
}
int tinh4(int x, int y, int z, int t, int h, int k, int l)
{
   if ( s[h] == '*' ) return ( tinh3(x*y,z,t,k,l) );
   if ( s[k] == '*' ) return ( tinh3(x,y*z,t,h,l) );
   if ( s[l] == '*' ) return ( tinh3(x,y,z*t,h,k) );
   return tinh3(tinh2(x,y,h),z,t,k,l);
}
bool cmp(st h, st k)
{
   return ( h.x < k.x );
}
```

```
bool BFS()
{
   int l=1,r=0;
   for(int i=1;i<=n;i++)
      if ( X[i] == 0 ) dist[i]=0,q[++r]=i;
      else dist[i]=MOD;
   dist[0]=MOD;
   while(l<=r)
   {
      int u=q[l++];
      for(int i=0;i<vt[u].size();i++)
      {
         int v=vt[u][i];
         if ( dist[Y[v]] == MOD )
         {
            dist[Y[v]]=dist[u]+1;
            if ( Y[v] ) q[++r]=Y[v];
         }
      }
   }
   return ( dist[0] != MOD );
}
```

```
bool DFS(int u)
{
   for(int i=0;i<vt[u].size();i++)
   {
      int v=vt[u][i];
      if ( Y[v] == 0 )
      {
         X[u]=v,Y[v]=u,dist[u]=MOD;
         return 1;
      }
      if ( dist[Y[v]] == dist[u] + 1 && DFS(Y[v]) )
      {
         X[u]=v,Y[v]=u,dist[u]=MOD;
         return 1;
      }
   }
   dist[u]=MOD;
   return 0;
}
int Hopcroft_Karp()
{
   int match=0;
   while( BFS() )
```

```
    for(int i=1;i<=n;i++)
        if ( X[i] == 0 && DFS(i)  ) match++;
    return match;
}
int main()
{
    ios_base::sync_with_stdio(0);
    freopen("EXAM.inp","r",stdin);
    freopen("EXAM.out","w",stdout);
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>s;
        int m=s.size(),tong=1;
        int p[10]={0,0,0,0,0,0};
        int q[10]={0,0,0,0,0,0};
        s='0'+s;
        for(int j=1;j<=m;j++)
            if ( s[j] < '0' ) p[tong++]=j;
            else q[tong]=q[tong]*10+s[j]-'0';
        if ( tong == 1 ) f[++dem].x=q[1],f[dem].y=i,f[dem].z="0000";
        if ( tong == 2 ) f[++dem].x=tinh2(q[1],q[2],p[1]),f[dem].y=i,f[dem].z="0000";
        if ( tong == 3 )
```

```
      {
         f[++dem].x=tinh2(tinh2(q[1],q[2],p[1]),q[3],p[2]),f[dem].y=i,f[dem].z="1200";
         f[++dem].x=tinh2(q[1],tinh2(q[2],q[3],p[2]),p[1]),f[dem].y=i,f[dem].z="2300";
      }
      if ( tong == 4 )
      {
f[++dem].x=tinh3(tinh2(q[1],q[2],p[1]),q[3],q[4],p[2],p[3]),f[dem].y=i,f[dem].z="1200";
f[++dem].x=tinh2(tinh3(q[1],q[2],q[3],p[1],p[2]),q[4],p[3]),f[dem].y=i,f[dem].z="1300";
f[++dem].x=tinh3(q[1],tinh2(q[2],q[3],p[2]),q[4],p[1],p[3]),f[dem].y=i,f[dem].z="2300";
f[++dem].x=tinh2(q[1],tinh3(q[2],q[3],q[4],p[2],p[3]),p[1]),f[dem].y=i,f[dem].z="2400";
f[++dem].x=tinh3(q[1],q[2],tinh2(q[3],q[4],p[3]),p[1],p[2]),f[dem].y=i,f[dem].z="3400";
f[++dem].x=tinh2(tinh2(q[1],q[2],p[1]),tinh2(q[3],q[4],p[3]),p[2]),f[dem].y=i,f[dem].z="123
4";
         f[++dem].x=tinh4(q[1],q[2],q[3],q[4],p[1],p[2],p[3]),f[dem].y=i,f[dem].z="0000";
      }
      for(int j=1;j<=tong;j++)
         P[i][j]=s[p[j]],Q[i][j]=q[j];
      d[i]=tong;
   }
   int m=0,tong=0;
   sort(f+1,f+dem+1,cmp);
```

```
f[0].x=MOD;
for(int i=1;i<=dem;i++)
{
   if ( f[i].x != f[i-1].x ) kt[m++]=i-1,bd[m]=i;
   vt[f[i].y].pb(m);
}
kt[m]=dem;
if ( Hopcroft_Karp() != n )
{
   cout<<"NO SOLUTION";
   return 0;
}
for(int i=1;i<=m;i++)
   for(int j=bd[i];j<=kt[i];j++)
      if ( f[j].y == Y[i] ) tv[Y[i]]=f[j].z;
for(int i=1;i<=n;i++)
{
   t=0,tong=d[i];
   for(int j=1;j<tong;j++)
   {
      if ( tv[i][t] - '0' == j && t%2 == 0 ) cout<<"(",t++;
      cout<<Q[i][j];
      if ( tv[i][t] - '0' == j && t%2 != 0 ) cout<<")",t++;
```

```
            if ( tong > 1 ) cout<<char(P[i][j]);
        }
        cout<<Q[i][tong];
        if ( tv[i][t] - '0' == tong && t%2 ) cout<<")",t++;
        cout<<endl;
    }
}
```

Bài 4: FMATCH - Fast Maximum Matching (http://vn.spoj.com/problems/FMATCH)

FJ có N (1 ≤ N ≤ 50,000) cô bò và M (1 ≤ M ≤ 50,000) chú bò. Cho danh sách P (1 ≤ P ≤ 150,000) khả năng ghép đôi giữa các cô bò và chú bò, hãy tính số cặp lớn nhất có thể ghép được. Tất nhiên, một cô bò có thể ghép với tối đa là một chú bò và ngược lại.

Input

Dòng đầu tiên chứa 3 số nguyên, N, M, và P. Mỗi dòng trong số P dòng tiếp theo chứa 2 số nguyên A (1 ≤ A ≤ N) và B (1 ≤ B ≤ M), thể hiện việc cô bò A có thể ghép được với chú bò B.

Output

In ra một số nguyên thể hiện số cặp lớn nhất có thể đạt được.

Input:	Output:
5 4 6 5 2 1 2 4 3 3 1 2 2 4 4	3

Cô bò 1 có thể được ghép với chú bò 2, cô bò 3 với chú bò 1, và cô bò 4 với chú bò 3.

* Thuật toán:

Để giải quyết bài toán này, ta sẽ sử dụng phương pháp cặp ghép với thuật toán của Hopcroft – Karp.
Coi số bò cái là tập X , số bò đực là tập Y. Khi đó với p cặp có khả năng ghép với nhau, ta sẽ lần lượt thử ghép số phần tử tập X với số phần tử của tập Y bằng các đường (là các cặp bò có thể ghép được với nhau) để sao cho số cặp ghép được là nhiều nhất. Vì dữ liệu rất to nên ta không thể sử dụng thuật toán Hungarian, vì vậy phải sử dụng thuật toán Hopcroft – Karp của cặp ghép để có thể giải quyết được bài toán này trong thời gian quy định. Thuật toán sẽ có ta kết quả tối ưu nhất trong khoảng thời gian mà chương trình được chạy

* Chương trình:

```
// Program: Hopkroft Karp

 #include <cstdio>

#include <cstdlib>

#include <cmath>

#include <iostream>

#include <algorithm>

#include <vector>

 #define MAXN 50005

#define INF 1000000007

#define tr(i,c) for(typeof((c).begin()) i=(c).begin();i!=(c).end();i++)

using namespace std;

 int m,n,res=0;

vector<int> g[MAXN];

int x[MAXN],y[MAXN],d[MAXN], q[MAXN];

 bool FindPath() {

    int L=1, R=0;

    for(int u=1;u<=n;u++)

    if (x[u]==0) {
```

```
        d[u]=0;
        q[++R]=u;
    } else d[u]=INF;
    d[0]=INF;
    while (L<=R) {
        int u=q[L++];
        tr(i,g[u]) {
            int v=*i;
            if (d[y[v]]==INF) {
                d[y[v]]=d[u]+1;
                if (y[v]) q[++R]=y[v];
            }
        }
    }
    return (d[0]!=INF);
}
 bool dfs(int u) {
    //if (u==0) return(true);
    tr(i,g[u]) {
        int v=*i;
        if (y[v]==0) {
            x[u]=v; y[v]=u;
            d[u]=INF;
```

```
            return true;
        }
        if (d[y[v]]==d[u]+1)
        if (dfs(y[v])) {
            x[u]=v; y[v]=u;
            d[u]=INF;
            return true;
        }
    }
    d[u]=INF;
    return false;
}
void GhepMax() {
    while (FindPath()) {
        for(int u=1;u<=n;u++)
            if (x[u]==0)
                if (dfs(u)) res++;
    }
    printf("%d",res);
}
 int main() {
    #ifndef ONLINE_JUDGE
        freopen("inp.txt","r",stdin);
```

```
    freopen("out.txt","w",stdout);
#endif // ONLINE_JUDGE
  int p;
  scanf("%d%d%d",&n,&m,&p);
  for(int i=1;i<=p;i++) {
    int u,v; scanf("%d%d",&u,&v);
    g[u].push_back(v);
  }
  GhepMax();
}
```

Bài 5: Năng suất dây chuyền (Sưu tầm không biết của ai)

Một dây chuyền sản xuất có N vị trí làm việc đánh số từ 1 đến N. Có N công nhân để xếp vào làm việc trên các vị trí này. Biết s_{ij} là năng suất để công nhân i trên vị trí làm việc j của dây chuyền (i,j=1,2,..,n). Cho trước một cách bố trí công nhân đứng làm việc trên các vị trí của dây chuyền, ta có thể tính năng suất của dây chuyền theo cách bố trí đã cho như là năng suất nhỏ nhất của công nhân trên dây chuyền.

Yêu cầu: Tìm cách bố trí N công nhân vào làm việc trên N vị trí của một dây chuyền sản xuất sao cho năng suất của dây chuyền là lớn nhất.

Dữ liệu: Vào từ file NANGSUAT.INP

- Dòng đầu tiên chứa số nguyên dương N ($N \leq 200$)
- Dòng thứ i trong số N dòng tiếp theo chứa n số nguyên dương $s_{i1}, s_{i2}, ..., s_{in}$ (i=1,2,...,n). Các số này có giá trị không vượt quá 20000 và cách nhau bởi dấu trống.

Kết quả: Ghi ra file văn bản NANGSUAT.OUT

- Dòng đầu tiên ghi năng suất của dây chuyền theo cách bố trí tìm được.
- Dòng thứ i trong số N dòng tiếp theo ghi vị trí làm việc của công nhân i trên dây chuyền theo cách bố trí tìm được.

Ví dụ:

NANGSUAT.INP	NANGSUAT.OUT
4	7
9 4 4 12	1
8 7 8 13	4
2 2 8 3	3
6 7 3 7	2

*** Thuật toán:**

- Ta sẽ tìm kiếm nhị phân theo kết quả, với mỗi kết quả ta sẽ tạo1 đồ thị với cạnh là ai sẽ được vào vị trí nào .
- Từ đồ thị này, ta sẽ dùng thuật toán cặp ghép để tìm thử xem liệu có thể xếp cả n người vào n vị trí hay không, nghĩa là sẽ có đủ n cặp ghép. Nếu thỏa mãn, ta sẽ tăng kết quả lên, nếu không thỏa mãn cần giảm kết quả xuống.

***Chương trình:**

```
#include <bits/stdc++.h>

#define pb push_back

using namespace std;

const int N=5e3+2;

int n,d[N],dd[N];

int c[N][N];

vector<int> a[N];



void sinh()

{

   freopen("NANGSUAT.inp","w",stdout);

   srand(time(0));

   n=200;

   cout<<n<<endl;
```

```
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
            cout<<rand()%10000+10000<<" ";
        cout<<endl;
    }
}

bool visit(int u,int t)
{
    if ( dd[u] != t ) dd[u]=t;
    else return 0;
    for(int i=0;i<a[u].size();i++)
    {
        int v=a[u][i];
        if ( !d[v] || visit(d[v],t) )
        {
            d[v]=u;
            return 1;
        }
    }
    return 0;
}
```

```
bool kt()
{
   for(int i=1;i<=n*2;i++)
      d[i]=dd[i]=0;
   for(int i=1;i<=n;i++)
      if ( !d[i] && !visit(i,i) ) return 0;
   return 1;
}
int main()
{
   ios_base::sync_with_stdio(0);
   //sinh();
   freopen("NANGSUAT.inp","r",stdin);
   freopen("NANGSUAT.out","w",stdout);
   cin>>n;
   for(int i=1;i<=n;i++)
      for(int j=1;j<=n;j++)
         cin>>c[i][j];
   int l=0,r=2e4+2;
   while( l < r - 1 )
   {
      int mid=(l+r)/2;
      for(int i=1;i<=n;i++)
```

```
            a[i].clear(),a[i+n].clear();
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
                if ( c[i][j] >= mid ) a[i].pb(j+n),a[j+n].pb(i);
        if ( kt() ) l=mid;
        else r=mid;
    }
    for(int i=1;i<=n;i++)
        a[i].clear(),a[i+n].clear();
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            if ( c[i][j] >= l ) a[i].pb(j+n),a[j+n].pb(i);
    kt();
    cout<<l<<endl;
    for(int i=n+1;i<=2*n;i++)
        d[d[i]]=i-n;
    for(int i=1;i<=n;i++)
        cout<<d[i]<<endl;
}
```

Bài 6: Bố trí khách sạn (Sưu tầm không biết của ai)

Tại SEAGAMES22, các môn thi đấu được tổ chức tại N địa điểm khác nhau (đánh số từ 1 đến N), mỗi địa điểm tổ chức thi đấu cho ít nhất một môn. Ban tổ chức SEAGAMES22 đã bố bố trí N khách sạn phục vụ vận động viên các nước (các khách sạn cũng đánh số từ 1 đến N). Có tất cả K môn thi đấu khác nhau(đánh số từ 1 đến K). Khoảng cách giữa địa điểm thi đấu i và khách sạn j là a_{ij}.

Yêu cầu: Viết chương trình thực hiện cách bố trí khách sạn sao cho hai khách sạn khác nhau sẽ phục vụ các vận động viên thi đấu tại hai địa điểm khác nhau và thời gian đi từ chỗ ở đến nơi thi đấu của tất cả các vận động viên là nhanh nhất (giả thiết rằng các ô tô chở vận động viên đều chạy với tốc độ như nhau).

Dữ liệu: Vào từ file SEAGAMES.INP:

- Dòng đầu tiên ghi ba số nguyên dương N, K ($1 \le N \le 100, N \le K \le 200$)
- Dòng thứ hai ghi K số nguyên $c_1, c_2, ..., c_k$ thể hiện môn thi đấu i tổ chức tại địa điểm c_i ($1 \le i \le K$)
- Tiếp theo là N dòng, dòng thứ i ghi các số $a_{i1}, a_{i2}, ..., a_{in}$ trong đó a_{ij} là khoảng cách từ địa điểm thi đấu i đến khách sạn j. $0 < a_{ij} < 30000$

Kết quả: Ghi ra file SEAGAMES.OUT gồm một dòng ghi các số $x_1, x_2, ..., x_k$ trong đó x_i là số hiệu khách sạn mà vận động viên thi đấu môn thi i ở
Các số trên một dòng trong các file dữ liệu vào/ra ghi cách nhau ít nhất một khoảng trắng
Ví dụ:

SEAGAMES.INP	SEAGAMES.OUT
5 7 2 1 1 3 4 5 5 7 8 6 5 4 2 9 5 4 6 3 6 8 7 5 6 8 9 2 4 6 5 4 5 7	3 4 4 1 5 2 2

***Thuật toán:**

- Ta sẽ tìm kiếm nhị phân theo kết quả, mỗi lần có 1 kết quả ta sẽ tạo 1 đồ thị với số cạnh là từ địa điểm thi đấu nào tới khách sạn đi không quá kết quả trên.
- Tiếp theo ta chỉ việc dùng thuật toán cặp ghép, tìm xem liệu có đủ n cặp ghép hay không. Nếu có đủ n cặp ghép, ta chỉ việc giảm kết quả xuống, nếu không ta sẽ tăng kết quả lên.

***Chương trình:**

```
#include <bits/stdc++.h>

#define pb push_back

using namespace std;

const int N=1e3;

int n,m;
```

```
int c[N],assigned[N],visited[N];
int a[N][N];
vector<int> d[N];
void sinh()
{
   srand(time(0));
   freopen("SEAGAMES.inp","w",stdout);
   int n=100,m=200;
   cout<<n<<" "<<m<<endl;
   for(int i=1;i<=n;i++)
   {
      int x=rand()%m+1;
      while( c[x] ) x=rand()%n+1;
      c[x]=i;
   }
   for(int i=1;i<=m;i++)
   {
      int x=rand()%n+1;
      if ( !c[i] ) c[i]=x;
      cout<<c[i]<<" ";
   }
   for(int i=1;i<=n;i++)
   {
```

```
        for(int j=1;j<=n;j++)
            cout<<(rand()%10000+20000)<<" ";
        cout<<endl;
    }
}
bool visit(int u, int t)
{
    if (visited[u]!=t) visited[u]=t;
    else return 0;
    for (int i=0;i<d[u].size();i++)
    {
        int v=d[u][i];
        if ( ( !assigned[v] || visit(assigned[v],t) ) )
        {
            assigned[v]=u;
            return 1;
        }
    }
    return 0;
}
bool kt(int k)
{
    int t=0;
```

```
    for(int i=1;i<=n*2;i++)
        d[i].clear(),assigned[i]=visited[i]=0;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            if ( a[j][i] <= k ) d[i].pb(j+n),d[j+n].pb(i);
    for(int i=1;i<=n;i++)
        if ( !visit(i,++t) ) return 0;
    return 1;
}

int main()
{
    ios_base::sync_with_stdio(0);
    //sinh();
    freopen("SEAGAMES.inp","r",stdin);
    freopen("SEAGAMES.out","w",stdout);
    cin>>n>>m;
    for(int i=1;i<=m;i++)
        cin>>c[i];
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            cin>>a[i][j];
    int l=0,r=3e4+4;
```

```
    while( l < r - 1 )
    {
        int mid=(l+r)/2;
        if ( kt(mid) ) r=mid;
        else l=mid;
    }
    kt(r);
    for(int i=1;i<=m;i++)
        cout<<assigned[c[i]+n]<<" ";
}
```