BÀI 1. DÃY CON KHÔNG GIẢM

*** Thuật toán đề xuất:**

Gọi $f[i]$ là độ dài của đoạn con liên tiếp không giảm kết thúc tại $a[i]$.

Suy ra $max(f[\,])$ - độ dài của đoạn con liên tiếp không giảm dài nhất, là kết quả bài toán.

Khởi tạo $f[1] = 1$;

Xét $a[i]$ $(2 \le i \le n)$:

Nếu $a[i - 1] \le a[i]$ thì độ dài đoạn con liên tiếp không giảm sẽ được nối dài thêm 1 đơn vị, khi đó $f[i] = f[i - 1] + 1$.

Nếu $a[i - 1] > a[i]$ thì $a[i]$ sẽ là phần tử đầu tiên của một đoạn con liên tiếp không giảm mới, khi đó $f[i] = 1$

Chủ đề: ***Quy hoạch động***

*** Chương trình minh họa bằng ngôn ngữ lập trình C++:**

```
#include <bits/stdc++.h>
#define N 10001
using namespace std;
int f[N], a[N];
int n, res = 0;
int main()
{
    freopen("cau1.inp","r",stdin);
    freopen("cau1.out","w",stdout);
    cin >> n;
    for (int i=1; i<=n; i++) cin >> a[i];
    f[1] = 1;
    for (int i=2; i<=n; i++)
    {
        if (a[i-1] <= a[i]) f[i] = f[i-1] + 1;
        else f[i] = 1;
        res = max(res,f[i]);
    }
    cout << res;
    return 0;
}
```

*** Chương trình minh họa bằng ngôn ngữ lập trình Python:**

```
finp = open("cau1.inp","r")
n = int(finp.readline())
a = [0] * (n+1)
f = [0] * (n+1)
for i in range(1,n+1):
    a[i] = int(finp.readline())
finp.close()

f[1] = 1; res = 0
for i in range(2,n+1):
    if (a[i-1] <= a[i]): f[i] = f[i-1]+1
    else: f[i] = 1
    res = max(res,f[i])

fout = open("cau1.out","w")
fout.write(str(res))
fout.close()
```

BÀI 2. TUYẾN ĐƯỜNG XUNG YẾU

*** Thuật toán đề xuất:**

Bước 1: tính k = số thành phần liên thông của đồ thị.

Bước 2: loại bỏ cạnh (u, v) của ra khỏi đồ thị:

+ *Bước 2.1*: tính p = số thành phần liên thông của đồ thị

+ *Bước 2.2.*: nếu ($p > k$ và (u,v) chỉ có 1 cạnh nối trực tiếp) thì (u,v) là một tuyến đường xung yếu.

Chủ đề: *Đồ thị + DFS*

*** Chương trình minh họa bằng ngôn ngữ lập trình C++:**

```
#include <bits/stdc++.h>
#define N 101
#define M 501
#define pii pair<int, int>
using namespace std;
int a[N][N];
int d[N];
pii lst[M];
int n,m,res;

void DFS(int u)
{
    d[u] = 1;
    for (int v=1; v<=n; v++)
    if (a[u][v] > 0 && d[v] == 0) DFS(v);
}
int CountConnected()
{
    memset(d,0,sizeof(d));
    int c = 0;
```

```
    for (int i=1; i<=n; i++)
    if (d[i] == 0)
    {
        c++;
        DFS(i);
    }
    return c;
}
int main()
{
    freopen("cau2.inp","r",stdin);
    freopen("cau2.out","w",stdout);
    cin >> n >> m;
    for (int u,v,i=1; i<=m; i++)
    {
        cin >> u >> v;
        lst[i] = pii(u,v);
        a[u][v] = ++a[v][u];
    }
    int k = CountConnected();
    for (int i=1; i<=m; i++)
    {
        int u = lst[i].first, v = lst[i].second;
        int tmp = a[u][v];
        a[u][v] = a[v][u] = 0;
        int p = CountConnected();
        if (p > k && tmp == 1) res++;
        a[u][v] = a[v][u] = tmp;
    }
    cout << res;
    return 0;
}
```