顺序排列 题解

Subtask 1

DFS 搜索即可。

本 Subtask 的标程为 `permutation_sol1.cpp`。

Subtask 2

不同限制之间完全不相同，一个数最多在一个限制中出现一次。

因此，我们可以根据每个限制分配每个位置的数，不会产生冲突。

本 Subtask 的标程为 `permutation_sol2.cpp`。

Subtask 3

y_i 互不相同且只有 $n-1$ 个，并且 $[1,n]$ 在限制中至少出现一次。

那么，我们可以从 x_i 到 y_i 连一条边，就能得到一棵树。

之后，输出这棵树各个点的 DFS 序即可。

本 Subtask 的标程为 `permutation_sol3.cpp`。

Subtask 4

n^2 可过，给不会拓扑排序的同学准备的。

Subtask5

正解，拓扑排序。

从 x_i 到 y_i 连接一条有向边，执行拓扑排序。

每个位置要填入的数，就是这个点在图上的拓扑序遍历顺序。

根据拓扑排序的性质，x_i 一定在 y_i 之前遍历到。这样，就可以保证 $a_{x_i}<a_{y_i}$。

本 Subtask 的标程为 `permutation_sol4.cpp`。同时，`permutation_sol5.cpp` 是使用了快速读入的版本。

```cpp
#include<cstdio>
#include<algorithm>
#include<queue>
using namespace std;
const int MAXN=1e5,MAXM=5e5;
int n,m;
struct Edge
{
    int to,next;
}edge[MAXM+5];
int edge_cnt;
int head[MAXN+5];
```

```
void add_edge(int u,int v)
{
    edge[++edge_cnt]=(Edge){v,head[u]};
    head[u]=edge_cnt;
}
queue<int>q;
int deg[MAXN+5];
int nowbfn;
int ans[MAXN+5];
int main()
{
    freopen("permutation.in","r",stdin);
    freopen("permutation.out","w",stdout);
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        add_edge(u,v);
        deg[v]++;
    }
    for(int i=1;i<=n;i++)
    {
        if(deg[i]==0)
        {
            q.push(i);
        }
    }
    while(!q.empty())
    {
        int u=q.front();
        q.pop();
        ans[u]=++nowbfn;
        for(int i=head[u];i;i=edge[i].next)
        {
            int v=edge[i].to;
            deg[v]--;
            if(deg[v]==0)
            {
                q.push(v);
            }
        }
    }
    for(int i=1;i<=n;i++)
    {
        if(deg[i])
        {
            printf("-1");
            return 0;
        }
    }
    for(int i=1;i<=n;i++)
    {
        printf("%d",ans[i]);
        if(i!=n)
```

```
        {
            printf(" ");
        }
    }
    return 0;
}
```

版权信息

题解：邓子君

在 CC-BY-NC 4.0 协议下共享。