二分图的最大匹配

<[861. 二分图的最大匹配 - AcWing题库]>

> 一个顶点，则称 M 是一个匹配。
>
> 二分图的最大匹配：所有匹配中包含边数最多的一组匹配被称为二分图的最大匹配，其边数即为最大匹配数。

输入格式

第一行包含三个整数 n_1、 n_2 和 m。

接下来 m 行，每行包含两个整数 u 和 v，表示左半部点集中的点 u 和右半部点集中的点 v 之间存在一条边。

输出格式

输出一个整数，表示二分图的最大匹配数。

数据范围

$1 \leq n_1, n_2 \leq 500,$
$1 \leq u \leq n_1,$
$1 \leq v \leq n_2,$
$1 \leq m \leq 10^5$

输入样例：

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

输出样例：

```
2
```

挑战模式 C++

匈牙利算法的思路：

- 最关键的是：有机会上，没机会创造机会也要上 ， 队友当前男生，如果其所喜欢的女生的集合中存在没有对象的，那么我们可以直接将其分配给这个男生，否则，我们通过递归 看能不能通过改变之前的已经确定的匹配关系 ， 来给这个男生腾出来一个女生当做其对象。st 数组的作用就是，我们在递归上层预定的女生 不能在递归下层重复使用了。

```
//match[j] = a,表示女孩 j 的 现有配对男友 是 a
int match[N] ;
// st函数的作用就是在匹配的过程中，先将当前的这个女孩进行预定，防止在递归的过程中重复使用。
int st[N] ;
//这个函数的作用是用来判断,如果加入x来参与模拟配对,会不会使匹配数增多
int find(int x)
{
    //遍历自己喜欢的女孩
    for(int i = h[x] ;  ~i ;i = ne[i])
    {
        int j = e[i];
        if( !st[j] )//如果在这一轮模拟匹配中,这个女孩尚未被预定
```

```
        {
            st[j] = true; //那x就预定这个女孩了
            //如果女孩j没有男朋友，或者她原来的男朋友能够预定其它喜欢的女孩。配对成功,更新
match
            if(!match[j]||find(match[j]))
            {
                match[j] = x;
                return true ;
            }

        }
    }
    //自己中意的全部都被预定了。配对失败。
    return false;
}

//记录最大匹配
int res = 0;
for(int i = 1; i <= n1 ;i ++)
{
    //因为每次模拟匹配的预定情况都是不一样的所以每轮模拟都要初始化
    memset(st,false,sizeof st);
    if(find(i))
        res++;
}
```

AC代码:

```
#include <iostream>
#include <cstring>
using namespace std ;
const int N = 1e5 + 10 , M = 510  ;
int h[N] , e[N] , ne[N] , idx ;
bool st[M] ; // st[j] 为真的时候表示这个点已经被搜过了。
int match[M] ;
int n1 , n2 , m ;
void add(int a , int b )
{
    e[idx] = b , ne[idx] = h[a] , h[a] = idx ++ ;
}

bool find(int x )
{
    for(int i = h[x] ; ~i ; i =  ne[i] )
    {
        int j = e[i] ;
        if( st[j] ) continue ;
        st[j] = true ;
        if( !match[j] || find(match[j] ) )
        {
```

```
            match[j] = x ;
            return true ;
        }
    }
    return false ;
}

int main()
{
    cin >> n1 >> n2 >> m ;
    memset(h , -1 , sizeof h ) ;
    for(int i = 0 ; i < m ; i ++ )
    {
        int a , b ;
        cin >> a >> b ;
        add(a , b ) ;
    }
    int ans = 0 ;
    for(int i = 1 ; i <= n1 ; i ++ )
    {
        memset(st , 0 , sizeof st ) ;
        if(find(i)) ans ++ ; // 如果这个点能够找到对象。
    }
    cout << ans << endl ;
    return 0 ;
}
```