算法设计与分析

实验报告（三）实现匈牙利算法

姓　　名	熊恪峥
学　　号	22920202204622
日　　期	2022年6月1日
学　　院	信息学院
课程名称	算法设计与分析

实验报告（三）实现匈牙利算法

目录

1 问题描述

实现匈牙利算法解决二分图匹配算法

2 问题分析

匈牙利算法由匈牙利数学家于1965年提出，以Hall定理为依据，通过寻找增广路来寻找二分图最大匹配。

由König定理：

定理 1. 一个二分图中的最大匹配数等于这个图中的最小点覆盖数。

该方法还可以用来处理最小点覆盖数问题。也就是想找到最少的一些点，使得二分图所有的边都至少有一个端点在这些点中。

匈牙利算法的执行过程是

- 从点集$\mathcal{V}$中找到未匹配点
- 寻找增广路
- 如果找到，就记录匹配
- 如果没有找到，就从$\mathcal{V}$中找下一个未匹配的点

在这个过程中，匈牙利算法不断地寻找增广路，直到无法找到新的增广路。因此可以找到最大匹配。

A 实现代码

代码 1: 红包发放

```
#include<iostream>

using namespace std;

constexpr int NMAX = 1010;
constexpr int EMAX = 1000010;
int n = 0, m = 0, e = 0, ans = 0;

int edge[EMAX], head[NMAX], nxt[EMAX], tot = 0;
int dfn[NMAX << 1], match[NMAX], x = 0, y = 0, ti = 0;

void add(int x, int y)
{
        edge[++tot] = y;
        nxt[tot] = head[x];
        head[x] = tot;
}

int hungrain(int x, int ti)
{
        for (int i = head[x]; i; i = nxt[i])
        {
                int j = edge[i];
```

```
24                  if (dfn[j] != ti)
25                  {
26                          dfn[j] = ti;
27                          if (!match[j] || hungrain(match[j], ti))
28                          {
29                                  match[j] = i;
30                                  return 1;
31                          }
32                  }
33
34          }
35          return 0;
36  }
37
38  int main()
39  {
40          cin >> n >> m >> e;
41          for (int i = 1; i <= e; i++)
42          {
43                  cin >> x >> y;
44                  if (x > n || y > m)
45                  {
46                          continue;
47                  }
48                  add(x, y);
49          }
50
51          for (int i = 1; i <= n; i++)
52          {
53                  if (hungrain(i, ++ti))
54                  {
55                          ans++;
56                  }
57          }
58
59          cout << ans << endl;
60
61          return 0;
62  }
```