算法设计与分析

作业（八）

姓　　名	熊恪峥
学　　号	22920202204622
日　　期	2022年4月20日
学　　院	信息学院
课程名称	算法设计与分析

作业（八）

目录

1 题9.2

源点：教授家

汇点：学校

顶点：街道拐角

边：若u和v之间有街道，连一权值为1的边

则若存在一条权值≥ 2的流，就可以上同一所学校。

2 题9.5

证明：$c_f(u,v)+c_f(v,u)=c(u,v)+c(v,u)$. 由对称性$f(u,v)=f(v,u)$

$$\begin{aligned} c_f(u,v)+c_f(v,u) &= c(u,v)-f(u,v)+c(v,u)-f(v,u) \\ &= c(u,v)-f(u,v)+c(v,u)-f(u,v) \\ &= c(u,v)+c(v,u) \end{aligned}$$

□

3 题9.7

Proof. 原命题等价于若一个网络中所有的容量之都不同，则存在一个唯一的最大流

则最大流$|f|$有

$$\begin{aligned} |f| &= \max \sum_{v\in V} f(s,v) \\ &= f(s,v_1)+f(s,v_2)+\cdots+(s,v_k)+\cdots+f(s,v_n) \end{aligned}$$

存在$v_k \in V$使得$f(s,v_k)=c(s,v_k)$

若有两个最大流，设$f(s,v_k)=c(s,v_k)<f^{'}(s,v_k^{'})=c^{'}(s,v_k^{'})$ 则在f中用$(s,v_k^{'})$替代(s,v_k)可以使得$|f|$更大，且$|f^{'}|=|f|$ 则存在一个唯一的最大流。由最大流最小割定理得有唯一的最小割。 □

4 题9.8

在剩余网络中从源点进行广度优先搜索，并记录路径，若到达汇点则找到一条增广路径。

5 题9.9

为了找到最大瓶颈容量的增广路，可以从源点到汇点进行深度优先搜索，然后选择瓶颈容量最大的一条。如算法 ??

6 题9.18

将信i与除j之外剩余的$n-1$个信封连边构成二分图，求该二分图的最大匹配，就能尽可能多地正确装入信封中

算法 1 找到最大瓶颈容量的增广路

Input: 有向无环图G,源点s，汇点t

procedure FIND(G, s, t)
  $path = \emptyset$
  $length = 0$
  for $u \in G.vertics$ **do**
    for $v \in G.vertics$ **do**
      $p, len =$ DFS(G,u,v,u,$\emptyset$)
      if $len > length$ **then**
        $path = p$
  return $path$
procedure DFS($G, s, t, c, len, p = \emptyset$)
  if $c == t$ **then**
    return p
  for $v \in G.vertics$ **do**
    if $G.edge$(c,v) *and not* $v.visited$ **then**
      $v.visited = True$
      return DFS(G,s,t,max($len, G.edge$(c,v).c),v,$p + v$)

7 实现匈牙利算法

代码 1: 匈牙利算法

```
constexpr int NMAX = 510;
bool g[NMAX][NMAX]{}, vis[NMAX];
int p[NMAX]{};

void reset()
{
        memset(g, 0, sizeof(g));
        memset(vis, 0, sizeof(vis));
        memset(p, 0, sizeof(p));
}

bool match(int i, const int N)
{
        for (int j = 1; j <= N; j++)
        {
                if (g[i][j] && !vis[j])
                {
                        vis[j] = true;
                        if (!p[j] || match(p[j], N))
                        {
                                p[j] = i;
                                return true;
                        }
                }
        }
        return false;
}

int hungarian(const int M, const int N)
{
        int ret = 0;
        for (int i = 1; i <= M; i++)
```

```
33          {
34                  memset(vis, 0, sizeof(vis));
35                  if (match(i, N))
36                  {
37                          ret++;
38                  }
39          }
40          return ret;
41  }
```

8 实现SPA

代码 2: 匈牙利算法

```
1
2   int n, m, s, t, last[MAXN], flow[MAXN];
3   int bfs()
4   {
5       memset(last, -1, sizeof(last));
6       queue<int> q;
7       q.push(s);
8       flow[s] = INF;
9       while (!q.empty())
10      {
11          int p = q.front();
12          q.pop();
13          if (p == t)
14              break;
15          for (int eg = head[p]; eg; eg = edges[eg].next)
16          {
17              int to = edges[eg].to, vol = edges[eg].w;
18              if (vol > 0 && last[to] == -1)
19              {
20                  last[to] = eg;
21                  flow[to] = min(flow[p], vol);
22                  q.push(to);
23              }
24          }
25      }
26      return last[t] != -1;
27  }
28
29  int EK()
30  {
31      int maxflow = 0;
32      while (bfs())
33      {
34          maxflow += flow[t];
35          for (int i = t; i != s; i = edges[last[i] ^ 1].to)
36          {
37              edges[last[i]].w -= flow[t];
38              edges[last[i] ^ 1].w += flow[t];
39          }
```

```
40      }
41      return maxflow;
42  }
```