

# 实验五报告

程序语言: C++

姓名: 郑子睿

学号: 22920212204317

专业: 计算机科学与技术

内容: 实现课本146页的 ShortestPathAugmentation 算法

## 一、问题描述

实现课本146页的 ShortestPathAugmentation 算法

```
1 ShortestPathAugmentation(G, s, t)
2   1 for each edge (u, v) in E do
3     2 f(u, v) <- 0
4   3 Gf <- G
5   4 find the level graph GL of Gf
6   5 while t is a vertex in GL do
7     6 while there is a path p from s to t in GL do
8       7 let cf(p) be the bottleneck capacity on p
9       8 augment the current flow f by cf(p)
10      9 update GL and Gf along the path p
11     10 use Gf to compute a new level graph GL
```

## 二、算法思想

ShortestPathAugmentation 算法开始先初始化流为0流, 并设网络的剩余网络 $G_f$ 为原始图 $G$ , 并计算其层次图(level graph), 然后执行 while 循环。while 循环分阶段进行, 每个阶段由下面两步组成:

(1) 只要 $G_L$ 中有从 $s$ 到 $t$ 的路径 $p$ , 就用 $f_p$ 对当前的流 $f$ 进行增广, 即 $f + f_p$ , 从 $G_L$ 和 $G_f$ 中移去饱和边, 并相应地更新 $G_L$ 和 $G_f$ 。

(2) 根据剩余网络 $G_f$ 计算出层次图 $G_L$ , 若 $t$ 不在 $G_L$ 中, 则停止, 否则继续。

## 三、描述算法

```
1 // N为最大点数, M为最大边数
2 // 链式前向星存图
3 // 边e[i]: 存第i条出边{边的终点to, 边的容量c, 起点相同的下一条边ne}
4 // 表头h[u]: 存u点所在的图层
5 // 1. bfs() 对点分层, 判断是否存在增广路
6 // 2. dfs() 找到一条最短的增广路, 更新残留网络
7 struct Edge {long long to, c, ne;} edge[M * 2];
8 int h[N], idx = 2; // 从2,3开始配对
9 int d[N];
10 int n, m, S, T; // n点数, m边数, S源点, T汇点
11
12 // 建一条a->b容量为c的边和它的反向边
```

```

13 void add(int a, int b, int c)
14 {
15     edge[idx] = {b, c, h[a]}, h[a] = idx ++ ;
16     edge[idx] = {a, 0, h[b]}, h[b] = idx ++ ;
17 }
18
19 bool bfs() // 对点分层, 找增广路
20 {
21     memset(d, 0, sizeof d);
22     queue<int> q;
23     q.push(S), d[S] = 1;
24     while (q.size())
25     {
26         auto u = q.front();
27         q.pop();
28         for (int i = h[u]; i; i = edge[i].ne)
29         {
30             auto to = edge[i].to, cap = edge[i].c;
31             if (!d[to] && cap)
32             {
33                 d[to] = d[u] + 1;
34                 q.push(to);
35                 if (to == T) return true;
36             }
37         }
38     }
39     return false;
40 }
41
42 long long dfs(int u, long long mf) // 找到一条最短增广路
43 {
44     if (u == T) return mf;
45     for (int i = h[u]; i; i = edge[i].ne)
46     {
47         auto to = edge[i].to, cap = edge[i].c;
48         if (d[to] == d[u] + 1 && cap)
49         {
50             auto f = dfs(to, min(mf, cap));
51             edge[i].c -= f, edge[i ^ 1].c += f; // 更新残留网
52             if (f) return f;
53         }
54     }
55     return 0;
56 }
57
58 long long SPA()
59 {
60     long long maxflow = 0; // 存储最大流的流量值
61     while (bfs())
62         maxflow += dfs(S, inf);
63     return maxflow;
64 }

```

## 四、验证算法

本实验是实现一个求网络最大流的模板，用洛谷上的两道模板题验证。

## 1. [P3376【模板】网络最大流](#)

测试情况

测试点信息						
#1 AC 4ms/692.00KB	#2 AC 4ms/692.00KB	#3 AC 4ms/680.00KB	#4 AC 4ms/732.00KB	#5 AC 4ms/680.00KB	#6 AC 4ms/692.00KB	#7 AC 5ms/684.00KB
#8 AC 15ms/808.00KB	#9 AC 573ms/736.00KB	#10 TLE 1.16s/680.00KB	#11 AC 3ms/680.00KB	#12 AC 3ms/680.00KB		

可以看到，SPA 算法在点数100，边数5000的图没有通过，效率较低，需要优化

[提交记录](#)

## 2. [P2740 \[USACO4.2\]草地排水](#)

测试情况

测试点信息						
#1 AC 3ms/692.00KB	#2 AC 4ms/692.00KB	#3 AC 4ms/688.00KB	#4 AC 3ms/740.00KB	#5 AC 3ms/684.00KB	#6 AC 4ms/684.00KB	#7 AC 3ms/744.00KB
#8 AC 3ms/688.00KB	#9 AC 3ms/808.00KB	#10 AC 4ms/692.00KB	#11 AC 3ms/680.00KB	#12 AC 3ms/680.00KB		

可以看到，SPA 算法点数和边数小于200的图上可以顺利通过

[提交记录](#)

# 五、结论

给定一个网络 $G$ ，用  $\text{ShortestPathAugmentation}(G, s, t)$  算法找到最大流所需要的时间是  $O(|V||E|^2)$ ，可以加入多路增广，当前弧优化，残枝优化，余量优化等改进为 Dinic 算法，效率更高。[Dinic算法模板](#)