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

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

```
// N为最大点数，M为最大边数
// 链式前向星存图
// 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}
// 表头h[u]：存u点所在的图层
// 1. bfs() 对点分层，判断是否存在增广路
// 2. dfs() 找到一条最短的增广路，更新残留网络
struct Edge {long long to, c, ne;} edge[M * 2];
int h[N], idx = 2; // 从2,3开始配对
int d[N];
int n, m, S, T; // n点数，m边数，S源点，T汇点

// 建一条a->b容量为c的边和它的反向边
```

```cpp
void add(int a, int b, int c)
{
    edge[idx] = {b, c, h[a]}, h[a] = idx ++ ;
    edge[idx] = {a, 0, h[b]}, h[b] = idx ++ ;
}

bool bfs() // 对点分层，找增广路
{
    memset(d, 0, sizeof d);
    queue<int> q;
    q.push(S), d[S] = 1;
    while (q.size())
    {
        auto u = q.front();
        q.pop();
        for (int i = h[u]; i; i = edge[i].ne)
        {
            auto to = edge[i].to, cap = edge[i].c;
            if (!d[to] && cap)
            {
                d[to] = d[u] + 1;
                q.push(to);
                if (to == T) return true;
            }
        }
    }
    return false;
}

long long dfs(int u, long long mf) // 找到一条最短增广路
{
    if (u == T) return mf;
    for (int i = h[u]; i; i = edge[i].ne)
    {
        auto to = edge[i].to, cap = edge[i].c;
        if (d[to] == d[u] + 1 && cap)
        {
            auto f = dfs(to, min(mf, cap));
            edge[i].c -= f, edge[i ^ 1].c += f; // 更新残留网
            if (f) return f;
        }
    }
    return 0;
}

long long SPA()
{
    long long maxflow = 0; // 存储最大流的流量值
    while (bfs())
        maxflow += dfs(S, inf);
    return maxflow;
}
```

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

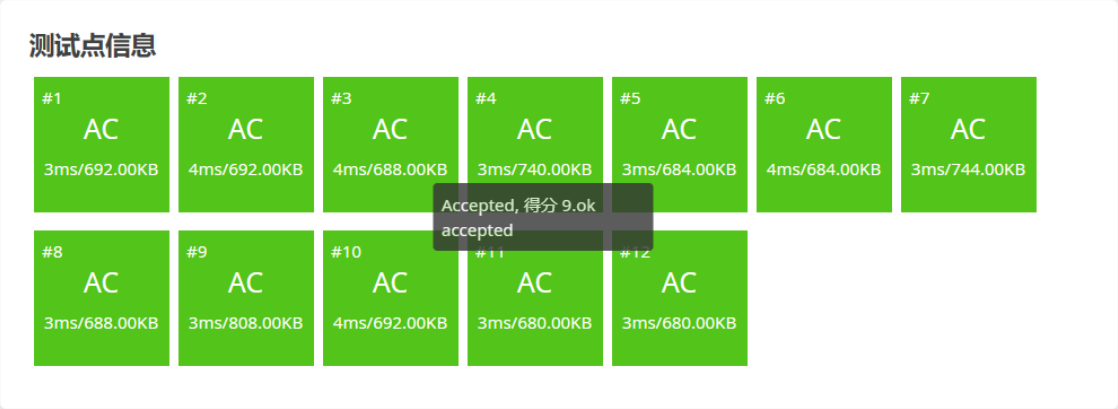
   

   可以看到，`SPA`算法在点数100，边数5000的图没有通过，效率较低，需要优化

   提交记录

2. P2740 [USACO4.2]草地排水

   测试情况

   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`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