

最大流

Maximum Flow

Ford-Fulkerson 方法

Concepts:

- 剩余容量 Residual Capacity: 一条边的容量与流量之差, $c_f(u, v) = c(u, v) - f(u, v)$
- 残量网络 Residual Network: 所有剩余容量大于 0 的边的生成子图
- 增广路 Augmenting Path: 原图 G 中, 一条从源点到汇点的由剩余容量都大于 0 的边构成的路径

Idea: 不断寻找增广路直到找不到为止。

Edmonds-Karp

Idea: bfs 寻找增广路。

Complexity: $O(VE^2)$

ATT: 链式前向星存储时, `edgeNum` 初始化为 1; 建图时建流为 0 的反向边。

Code:

```
1  int pre[N], minFlow[N];
2  int bfs(){
3      queue<int> q;
4      for(int i = 1; i <= n; i++){
5          pre[i] = 0;
6          minFlow[i] = INF;
7      }
8      q.push(src);
9      while(!q.empty()){
10         int cur = q.front(); q.pop();
11         for(int i = head[cur]; i; i = edge[i].nxt){
12             if(edge[i].flow && !pre[edge[i].to]){
13                 pre[edge[i].to] = i;
14                 minFlow[edge[i].to] = min(minFlow[cur], edge[i].flow);
15                 q.push(edge[i].to);
16             }
17         }
18     }
19     if(pre[dst] == 0) return -1;
20     return minFlow[dst];
21 }
22
23 int EK(){
24     int flow = 0, maxflow = 0;
25     while((flow = bfs()) != -1){
26         int t = dst;
27         while(t != src){
28             edge[pre[t]].flow -= flow;
29             edge[pre[t]^1].flow += flow;
30             t = edge[pre[t]^1].to;
31         }
32         maxflow += flow;
33     }
34     return maxflow;
35 }
```

Dinic

Idea: bfs 将图分层, dfs 按分层图寻找增广路。

Optimization: 当前弧优化。

Complexity: $O(V^2E)$

ATT: 链式前向星存储时, `edgeNum` 初始化为1; 建图时建流为 0 的反向边。

Code:

```
1 // s refers to source, t refers to destination
2 bool inq[N];
3 int dep[N];
4 bool bfs(){
5     for(int i = 1; i <= n; i++)
6         dep[i] = INF, inq[i] = 0;
7     queue<int> q;
8     q.push(s);
9     inq[s] = 1;
10    dep[s] = 0;
11    while(!q.empty()){
12        int cur = q.front(); q.pop();
13        inq[cur] = 0;
14        for(int i = head[cur]; i; i = edge[i].nxt){
15            if(dep[edge[i].to] > dep[cur] + 1 && edge[i].flow){
16                dep[edge[i].to] = dep[cur] + 1;
17                if(!inq[edge[i].to]){
18                    q.push(edge[i].to);
19                    inq[edge[i].to] = 1;
20                }
21            }
22        }
23    }
24    if(dep[t] != INF) return 1;
25    return 0;
26 }
27 int dfs(int x, int minFlow){
28     int flow = 0;
29     if(x == t) return minFlow;
30     for(int i = head[x]; i; i = edge[i].nxt){
31         if(dep[edge[i].to] == dep[x] + 1 && edge[i].flow){
32             flow = dfs(edge[i].to, min(minFlow, edge[i].flow));
33             if(flow){
34                 edge[i].flow -= flow;
35                 edge[i^1].flow += flow;
36                 return flow;
37             }
38         }
39     }
40     return 0;
41 }
42 int Dinic(){
43     int maxFlow = 0, flow = 0;
44     while(bfs()){
45         while(flow = dfs(s, INF))
46             maxFlow += flow;
47     }
48     return maxFlow;
49 }
```

Code (当前弧优化) :

```

1 namespace FLOW{
2
3     int n, s, t;
4     struct Edge{
5         int nxt, to;
6         LL flow;
7     }edge[M<<1];
8     int head[N], edgeNum = 1;
9     void addEdge(int from, int to, LL flow){
10         edge[++edgeNum].nxt = head[from];
11         edge[edgeNum].to = to;
12         edge[edgeNum].flow = flow;
13         head[from] = edgeNum;
14     }
15     void ae(int from, int to, LL flow){
16         addEdge(from, to, flow), addEdge(to, from, 0);
17     }
18
19     bool inq[N];
20     int dep[N], curArc[N];
21     bool bfs(){
22         for(int i = 1; i <= n; i++){
23             dep[i] = 1e9, inq[i] = 0, curArc[i] = head[i];
24         }
25         queue<int> q;
26         q.push(s);
27         inq[s] = 1;
28         dep[s] = 0;
29         while(!q.empty()){
30             int cur = q.front(); q.pop();
31             inq[cur] = 0;
32             for(int i = head[cur]; i; i = edge[i].nxt){
33                 if(dep[edge[i].to] > dep[cur] + 1 && edge[i].flow){
34                     dep[edge[i].to] = dep[cur] + 1;
35                     if(!inq[edge[i].to]){
36                         q.push(edge[i].to);
37                         inq[edge[i].to] = 1;
38                     }
39                 }
40             }
41             if(dep[t] != 1e9) return 1;
42             return 0;
43         }
44     LL dfs(int x, LL minFlow){
45         LL flow = 0;
46         if(x == t) return minFlow;
47         for(int i = curArc[x]; i; i = edge[i].nxt){
48             curArc[x] = i;
49             if(dep[edge[i].to] == dep[x] + 1 && edge[i].flow){
50                 flow = dfs(edge[i].to, min(minFlow, edge[i].flow));
51                 if(flow){
52                     edge[i].flow -= flow;
53                     edge[i^1].flow += flow;
54                     return flow;
55                 }
56             }
57         }
58         return 0;
59     }
60     LL Dinic(){
61         LL maxFlow = 0, flow = 0;
62         while(bfs()){
63             while(flow = dfs(s, INF))
64                 maxFlow += flow;
65         }

```

```
66         return maxFlow;
67     }
68
69     void init(){
70         edgeNum = 1;
71         for(int i = 1; i <= n; i++){
72             head[i] = 0;
73         }
74     }
75 }
```

ISAP

预流推进 Push-Relable
