Introduction to network flows

Ivan Smirnov
ifsmirnov@yandex.ru
t.me/ifsmirnov

April 21, 2020
Moscow Pre-finals Workshop 2020

1 Notations and definitions

- Graph is denoted as $G = (V, E)$. We consider $n = |V|$, $m = |E|$.
- Maximum flow problem is introduced for a tuple (G, c, s, t), where G is some directed graph such that if arc vu belongs to E, arc uv also belongs to E. c is a capacity function. s is a starting node, called *source*, while t is the target node called *sink*.
- Capacity constraint means $f(u, v) \leq c(u, v)$ for any pair (u, v). If there is no edge uv in E, $c(u, v)$ is considered to be zero.
- Skew symmetry constraint means $f(u, v) = -f(v, u)$ for any pair (u, v).
- Flow conservation means $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$ for any node u except s and t. In other words no flow is "stuck" in any intermediate node.
- $\sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t) = |f|$ is called the value of the flow f.
- Residual capacity of an arc is defined as $c'(v, u) = c(v, u) - f(v, u)$. It indicates how much the flow function can be increased on this edge.
- Residual network is a tuple (G, c', s, t), where c' stands for residual capacity.
- Augmenting path is a path $v_0, v_1, v_2, \ldots, v_k$ such that $v_0 = s$, $v_k = t$ and all residual capacities $c(v_i, v_{i+1}) > 0$.
- Level graph G_L is a graph constructed from a given graph G by running breadth-first search in a residual network (i.e. BFS uses only edges that have positive residual capacity) and removing all edges except those which go from some level to the next level.

2 Lecture plan (optimistic)

1. Introduction to maximum flow problem.
2. Ford-Fulkerson theorem: maximum flow equals minimum cut.
3. Ford-Fulkerson algorithm: find any augmenting path while it exists.
4. Finding some partition that yields the minimum cut.
5. Application: closure problem.
6. Application: maximum matching.
7. Scaling technique. An $O(m^2 \log C)$ time bound for combination of DFS and scaling.
8. Flow decomposition.
9. Blocking flow. Dinic's algorithm. $O(n^2 m)$ time bound.

3 Algorithm details

- To find an augmenting path, run a DFS from s to t using only edges with positive residual capacity (that is, $f_{uv} < c_{uv}$). Note that the edge can be present in the residual network even it was not in the original graph: it is the case of reverse edges, for them $c_{uv} = 0$ and $f_{uv} < 0$ because flow was pushed in the opposite direction.

- When an augmenting path was found, find minimal residual capacity among its edges. Let the path be $s = v_0, v_1, \ldots, v_{k-1}, v_k = t$. Then you should take the value $v = \min(c_{v_0 v_1} - f_{v_0 v_1}, c_{v_1 v_2} - f_{v_1 v_2}, \ldots)$. Push the flow through the path, adding v to flow on the edges in the same direction ($v_0 v_1, v_1 v_2, \ldots$) and subtracting it from the flow on edges in the opposite ($v_1 v_0, v_2 v_1, \ldots$).

- To find some minimal cut, find a maximum flow and run the DFS from s over residual network (the graph with edges with positive residual capacity). The visited vertices will form exactly one part of a minimum cut.

- Tricky point: in directed graphs only edges from the s-part of the cut to the t-part of the cut are counted. Edges in the opposite direction do not contribute to the capacity of the cut.

- To find the flow decomposition, you should first find the maximum flow. After that repeatedly run a DFS from s along the edges with positive flow (and not residual capacity, as before). Each DFS over those edges would terminate at t without backtracking and yield a path for the decomposition. When the path is found, take the minimum flow along it and subtract it from the values of the flow on edges along the path.

- Blocking flow is a flow such that the residual graph contains no $s - t$ path (it is not possible to augment the flow without cancelling existing flow).
- During one phase of Dinic algorithm we build a level graph and find the blocking flow in it. To do it efficiently, during the whole iteration we keep a pointer for each vertex that tracks which neighbours were already visited during this iteration.

4 Theoretical problems to think about

1. Construct the testcase that proves Ford-Fulkerson algorithm is not polynomial even if the order we consider all neighbours in DFS algorithm is the same at each iteration.
2. If capacities may be non-integer, construct the testcase on which Ford-Fulkerson algorithm does not terminate in finite time.
3. Prove that any maximum flow function f produces the same canonical cut.
4. Given a flow network and some maximum flow function f, find in linear time all edges, such that increasing capacity function of these edges will increase the minimum cut. In other words, find all edges that belong to each minimum s-t cut.
5. For a graph with n vertices and m edges, consider the maximum flow decomposition into minimum number of paths. How many paths will there be in worst case? Find a bound and show that it is tight (to show tightness, for each (n, m) present a graph and show that the decomosition in it must have at least certain number of paths).
6. Prove that if $c(v, u) = 1$ (unit network) for any $vu \in E$ the worktime of Dinic's algorithm is bound by $O(m\sqrt{m})$.