

Edmonds-Karp Algorithm

Advanced Algorithms (CSE 794)

Instructor: Tamal K. Dey

Scribe: Anand Joseph *

April 30, 2009

1 Analysis of Ford Fulkerson Algorithm

Consider the flow network G , given in Figure 1.

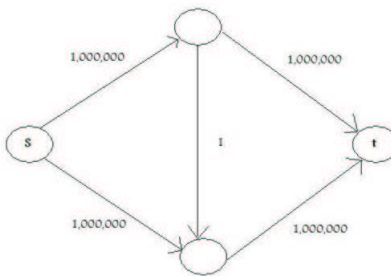


Figure 1: Flow Network G with given flow capacities

If the Ford-Fulkerson algorithm is used to choose an augmenting path as shown in Figure 2, the residual network will be as shown in Figure 3. The total flow $|f|$ in the network after selecting one augmenting path is 1.

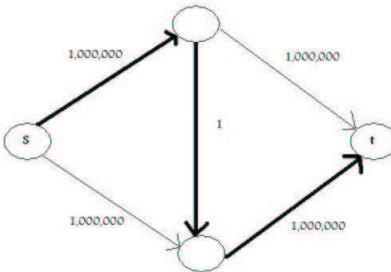


Figure 2: Augmenting Path

*Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA.

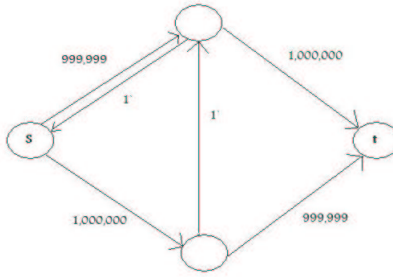


Figure 3: Residual Graph

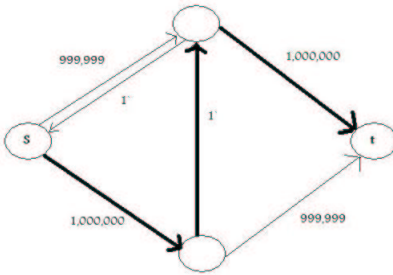


Figure 4: The second Augmenting Path

And from the residual network shown in Figure 3, an augmenting path may be selected as in Figure 4, the total flow $|f| = 2$. Therefore, if these two augmenting paths are chosen alternatively, the maximum flow $|f^*| = 2,000,000$ will be obtained only after 2,000,000 steps.

But if the augmenting paths are selected as shown in Figures 5 and 6, the maximum flow is obtained after just two steps.

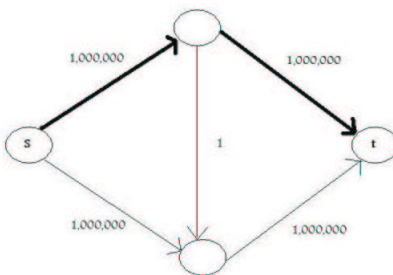


Figure 5: Efficient selection of Augmenting Path: Step 1

So, the efficiency of Ford-Fulkerson algorithm is highly dependent on the choice of augmenting paths. From the previous example, we can see that the upper bound for the running time of the algorithm is $O(m|f^*|)$.

where $m = |E|$ and $|f^*|$ is the maximum flow.

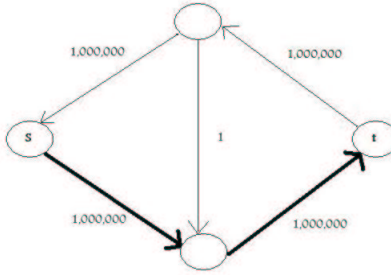


Figure 6: Efficient selection of Augmenting Path: Step 2

If the flow constraints are all integral, the algorithm always converges to give the maximum flow. But the same cannot be guaranteed for a graph with non-integral flows.

2 Edmond-Karp Algorithm

The Edmond-Karp Algorithm is exactly same as the Ford-Fulkerson algorithm, except for the fact that the augmenting path is selected from a breadth first search of the residual network instead of a depth first search. Because of the improvement made in the choice of augmenting paths the running time complexity of the Edmond-Karp Algorithm is an improved $O(nm^2)$.

where $m = |E|$ and $n = |V|$.

Lemma 1. *If Edmond-Karp Algorithm is run on a flow network $G : (V, E)$ with a source S and sink T , then $\forall v \in V - \{s, t\}$, the shortest distance $\delta_f(s, v)$ in G_f increases monotonically with each flow augmentation.*

Proof. Consider that there was some augmentation that reduced the shortest distance $\delta_f(s, v)$ for some v . And we also assume that v is the first node whose shortest distance from s is reduced by an augmentation. Let f and f' be the flows before and after the augmentation which decreased the distance.

Let $P : S \rightarrow v$ be the shortest path in $G_{f'}$ so that $(u, v) \in E_{f'}$ (Note: By shortest path we mean the minimum number of edges in the traversal from a source to a destination. In other words, every edge is assumed to have unit distance)

Therefore $\delta_f(s, u) = \delta_f(s, v) - 1$ (because the shortest path from s to v passes through u and (u, v) is an edge)

We know that $\delta_{f'}(s, u) \geq \delta_f(s, u)$ (because v is the first node whose shortest path from s has been reduced by an augmentation)

CLAIM: $(u, v) \notin E_f$

Justification:

If we had $(u, v) \in E_f$, $\delta_f(s, v) \leq \delta_f(s, u) + 1 \leq \delta_{f'}(s, u) + 1 = \delta_{f'}(s, v)$ which is a contradiction of the assumption, $\delta_f(s, v) > \delta_{f'}(s, v)$

Therefore $(u, v) \notin G_f$

So, after the augmentation, some increased flow has gone through (v, u) (from v to u)

This means that the shortest path from s to u in G_f has (v, u) as its last edge.

Therefore, $\delta_f(s, v) = \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v) - 2$

i.e., $\delta_f(s, v) < \delta_{f'}(s, v)$.

This contradicts our initial assumption, $\delta_{f'}(s, v) < \delta(s, v)$

Therefore the shortest distance from the source to each monotonically increases after each augmentation. \square

Theorem 2. *If Edmond-Karp algorithm runs on a flow network $G = (V, E)$ with source s and sink t , there are at most $O(nm)$ flow augmentations.*

Proof. Goal: Each edge can become critical at most $(n/2 - 1)$ times.

(By a critical edge, we mean the edge which has minimal residual capacity in the augmenting path. Note: the critical edge disappears after augmentation).

If (u, v) is critical for the first time,

$$\delta_f(s, v) = \delta_f(s, u) + 1 \text{ (obvious)}$$

(u, v) reappears only if (v, u) appears in an augmenting path. Since (u, v) disappeared after being a critical edge, the residual capacity of (v, u) will be equal to the flow capacity of (u, v) . Now, if (v, u) appears in the augmenting path, it implies that the flow from u to v is reduced. So, the edge (u, v) will reappear.

If f' is the flow when this event occurs,

$$\begin{aligned} \delta_f(s, v) &\leq \delta_{f'}(s, v) \text{ (by Lemma 1)} \\ \delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 &\geq \delta_f(s, v) + 1 = \delta_f(s, u) + 2 \end{aligned}$$

Therefore for each time an edge reappears in the residue graph, the shortest path from the source to that edge increases by at least 2.

So, any edge, (u, v) can reappear at most $(n/2 - 1)$ times. (because the longest simple path in a graph on n vertices cannot be longer than $n-1$)

Therefore, each edge can be a critical edge at most $(n/2 - 1)$ times. There are m edges. Therefore there are at most $O(nm)$ flow augmentations. \square

Goldberg-Tarjan algorithm

Also known as Push-Relabel algorithm. The main change made in this algorithm is that, while constructing the max flow the nodes are permitted to have excess flows, i.e., the inflow of a node can be greater than

the outflow. Such a flow is called as a preflow. When the algorithm converges, the preflow becomes the maximum flow satisfying all the constraints for a flow.

PREFLOW A preflow f is defined as,

$f : vXv \rightarrow R$ that satisfies skew symmetry and capacity constraint.

EXCESS FLOW: $e(u) = f(V, u)$ for $u \neq s$

u overflows when $e(u) > 0$

HEIGHT: $h : v \rightarrow N$

$h(s) = n$

$h(t) = 0$

$h(u) \leq h(v) + 1$

for each edge $(u, v) \in E_f$

Lemma 3. Let f be a preflow in G and h be a height function.

If $h(u) > h(v) + 1$ then (u, v) is not an edge in G_f

Proof. By definition of height function,

If $(u, v) \in E_f$, $h(u) \leq h(v) + 1$

Therefore it is obvious that if $h(u) > h(v) + 1$ then (u, v) cannot be an edge in G_f

□

PUSH OPERATION: *Push*(u, v) condition: u is overflowing,

$c_f(u, v) > 0$ and

$h(u) = h(v) + 1$

Actions,

Push $d_f(u, v) = \min(e(u), c_f(u, v))$

$f(u, v) = f(u, v) + d_f(u, v)$

$f(v, u) = -f(u, v)$

$e(u) = e(u) - d_f(u, v)$

$e(v) = e(v) + d_f(u, v)$

NON-SATURATING PUSH: If $c_f(u, v) = 0$ after the push.

Lemma 4. After a non-saturating push, $e(u) = 0$ (proof is obvious)

RELABEL OPERATION: Condition: u is overflowing and $\forall v \in V$ such that $(u, v) \in E_f$ we have $h(u) < h(v)$

Action: increase $h(u)$:

$h(u) = 1 + \min(h(v))$ such that $(u, v) \in E_f$