CS 6200: Algorithmics II	Fall 2020
Instructor: Avah Banerjee, S&T.	
Lecture: 13	**Date: 9/23**

1 Generic Push Relabling

1.1 Observation 8: The number of non-saturated pushes is $O(m^2n)$.

We can define an Φ as $\Phi = \sum_{e(u)>0} h(u)$. Where $e(u)$ represents the overflow of vertex u in some graph G and $h(u)$ represents the height of vertex u. The formula itself represents the sum of all heights of each vertex in graph G. Thus, $\Phi \geq 0$, since all heights must be greater or equal to zero ($h \geq 0$).
To prove this observation we want to show:

- The relabeling on a saturated push increases Φ by O(n)
- A non-saturated push decreases Φ by less than 1

From observation 6 we know that the total number of relabel operations is at most n^2. And from observation 7 we know that the total number of saturated pushes is mn. Thus, $n^2 + mn$ is the total number of events that we can increase Φ by $O(n)$. Thus, the maximum Φ can be is $O(n * (mn + n^2))$, to which we can write as $O(m^2n)$.

2 Minimum Cost Maximum Flow

The objective of Minimum Cost Maximum Flow algorithms is to find the maximum flow of a graph that provides the least cost. Thus, these algorithms can solve problems on graph that not only have flow and flow capacities, but also costs on each edge. Each edge(u, v) has cost cost(u, v) per unit flow.

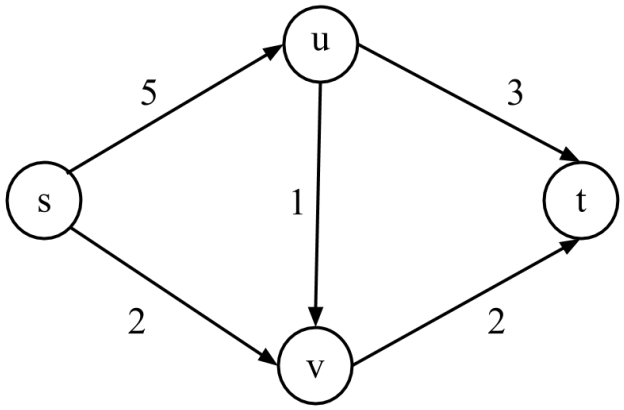

Fig. 1: Graph G with a flow capacity per edge.

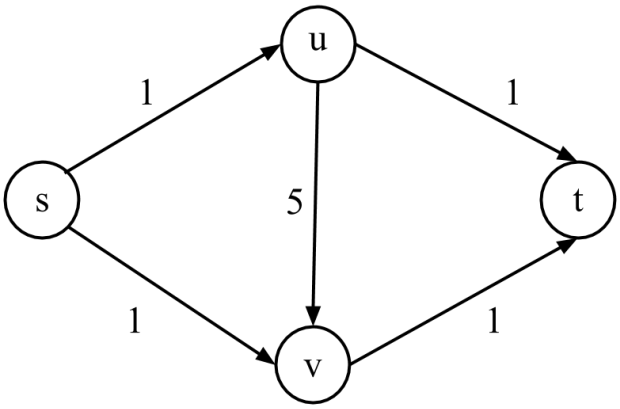

Fig. 2: Graph G with a cost per edge.

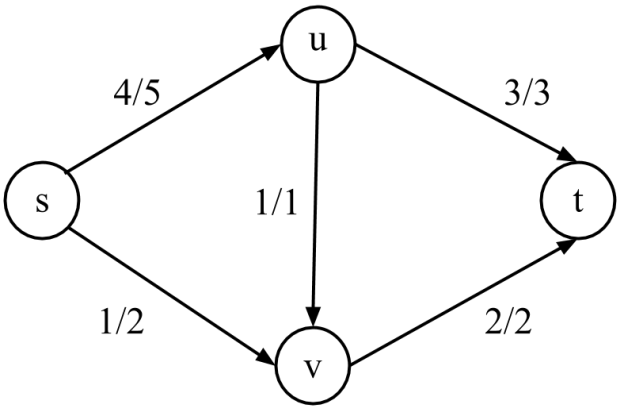

Fig. 3: Graph G flow after Ford-Fulkerson

Figure **??** shows gratph G with each edge's flow capcity while In figure 3 we reach a max flow of 5 using Ford-Fulkerson. Figure 2 shows the costs of each of those edges. Ford-Fulkerson finds the following:

1. $s \to u \to v \to t$ which gives us a flow of 1
2. $s \to u \to t$ which gives us a flow of 4
3. $s \to v \to t$ which gives us a flow of 1

After 3 there are no longer paths to augment and our maximum flow of 5. Based on figure 2 the costs of the maximum flow is 5. Thus, for flow f, $cost(f) = 15$ and $|f| = 5$.

Even if we reach maximum flow, that does not mean we reach minimum cost. For example, creating a flow through $s \to u \to t$ and $s \to v \to t$ will give us a cost of 4.

To attempt to find a flow that gives us maximum flow but minimum edge, there are many different algorithms. However, we will only go over one.

2.1 Cycle Cancellation

To find the minimum cost maximum flow, we cancel out edges. By finding negative cost cycles through saturated edges, we can reduce the cost of our maximum flow without changing the overall flow value. Once we find a negative cost cycle, we push flow along that cycle to reduce cost.

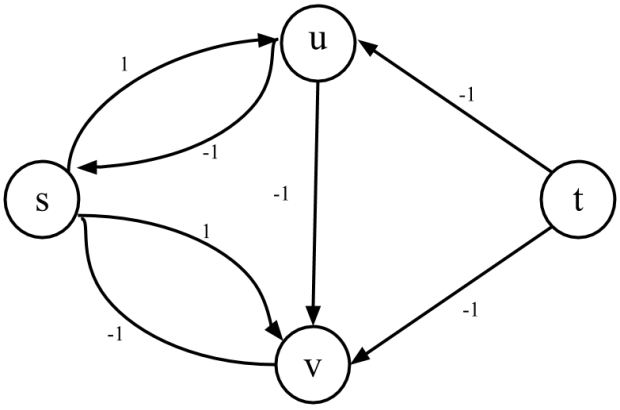

Fig. 4: Graph G costs after finding maximum flow

Figure 4 shows the costs of graph G after running Ford-Fulkerson and finding maximum flow. As seen in figure 3, edges $\{u, t\}$, $\{v, t\}$, and $\{u, v\}$ are saturated, thus figure 4 shows those edges with the option to flow in the negative direction. On the other hand vertices s and u have formed relationship since the edge $\{s, u\}$ is unsaturated thus flow can move in the positive or negative direction. The same applies to the relationship between vertices s and v.

Thus, within figure 4, we find that cycle $s \to v \to u \to s$ generates a cost of -1. Since we now have a negative cost cycle, we can push flow along that cycle to reduce costs of our flow to 10. Therefore, $|f|$ is still 0 while $cost(f) = 10$.

Algorithm 1

1: **Input:** A graph G with costs K and edges m.
2: Find a maximum flow in G
3: **while** there is a negative cycle in C **do**
4: circulate flow along C until some edge in C is saturated
5: **end while**

2.2 Analysis

Now that we have demonstrated a method for finding minimum cost maximum flow, we must analyse the algorithm.

To find a negative cycle in a graph, we can use the Bellman-Ford algorithm that has a run time of $O(mn)$. Where m is the number of edges and n is the number of vertices in a graph. Since we have the run-time to find a negative cycle, we must realize the number of times we need to find

negative cycles and the cost of the flow in a graph. To do this, we need to bound the cost of the flow.

$$cost(f) \leq m(max(c(u,v)) * (max(cost(u,v))) \tag{1}$$

Equation 1 details that the cost of the flow is no more the number of edges m multiplied by the maximum of all capacities multiplied by the maximum of the costs of all edges.
If we label $y \leftarrow max(c(u,v)$ and $z \leftarrow max(cost(u,v))$. We have to find negative cycles myz times. Thus the time complexity is $O(m^2nyz)$.

Theorem 2.1 *A flow f is minimum cost if and only if ($\Longleftrightarrow$) G_f has no negative cost cycle where G is a graph.*

Proof: If G_f has a negative cost cycle C then we can reduce $cost(f)$ by pushing flow around C.
Suppose $\exists f'$ such that $cost(f') < cost(f)$ and $|f| = |f'|$.
Let $f'' = f' - f$ and

1. $|f''| = 0$
2. $cost(f") < 0$

Thus, f'' is a set of circular flow with negative cost. Since the summation of the costs of f'' is less than 0, there must be at least one cycle with a cost less than zero. ■