COMS 4995-8: Advanced Algorithms (Spring'21) Feb 18, 2021

Lecture 12: Max-Flow Algorithms

Instructor: *Alex Andoni* Scribes: *Davepriyan Gunaratnam, Gauri Narayan*

1 Overview

Today we will explore 3 algorithms that improve on the run time of the original Ford-Fulkerson algorithm.

2 Review

Algorithm 1: Ford-Fulkerson algorithm

```
Result: Max flow from $s$ to $t$
Initialize: set flow $f := \vec{0}$ and $G_f = G$ while $\exists$ $s \to t$ path $P \in G_f$ do
    let $\delta = \min_{e \in P} c_e^f > 0$;
    for every $e = (u, v) \in P$ do
        if $(u, v)$ is F-edge then
            $f_{uv} := f_{uv} + \delta$;
        else
            $f_{vu} := f_{vu} - \delta$;
        end
    end
    Reconstruct $G_f$;
end
```

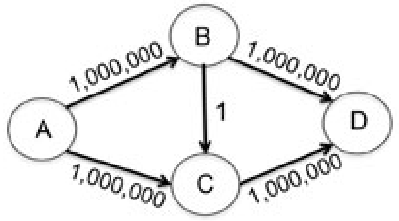
2.1 Run time of the Ford-Fulkerson Algorithm:

Consider a graph G where the capacities on all edges are **integers**.

Claim: FF runtime is $O(|f^*| \cdot m)$ where m is number of edges and $|f^*|$ is the maximum flow.

Proof. Consider the fact that $\delta > 0$ in each iteration by the way we define the residual graph G_f. Since all capacities are integers in the original graph, each time we find an augmenting path and compute a new δ, it must necessarily be integer as well. Since the integrality of δ and all capacities in G_f is maintained at all times, at every iteration $\delta \geq 1$. Thus the total number of iterations is the upper bounded by the max flow $|f^*|$ since we improve by at least 1 unit in each iteration. Finally, each iteration takes $O(m)$ time using BFS or DFS so total time is $O(|f^*| \cdot m)$. □

Problem: We can easily construct a worst case scenario in which this algorithm will increase the flow by exactly 1 unit per iteration. Below is an example of this. If the augmenting path the algorithm finds in every iteration from s to t is the path that that uses the middle edge with capacity 1, then in each iteration we increase the flow by exactly 1. The middle edge capacity causes $\delta = 1$ each time. Thus our run time will be 1,000,000 iterations in this example.



Ultimately this is not a polynomial time algorithm because of its dependence on the size of the maximum flow. We want an algorithm that has run time that is polynomial in n,m, $log(U)$ where U is the maximum capacity edge in G

3 Algorithm 1: Max-Width Path

3.1 Idea

Until now, we have not identified a particular way to pick the augmenting path in G_f. In this solution, each time we pick an augmenting path we choose the one that gives us the largest δ

3.2 Run time Analysis

We'd like to try to bound the number of iterations. Suppose our current flow is f. Define our remaining value v as

$$v := |f^*| - |f|$$

Claim 1. $\delta \geq v/m$

Proof. G_f has a maximum flow value equal to the remaining value, which is v Let the maximum flow on the residual graph G_f be named ϕ^*_f. Thus $|\phi^*_f| = v$ Using **Flow Decomposition Theorem** we can decompose the residual max flow ϕ^*_f as follows:

$\phi^*_f = \sum_{i=1}^k |f_{p_i}| + \sum_{j=1}^l |f_{c_j}|$ where p_i's represent the paths from s to t and c_j's represent the cycles

$\implies |\phi^*_f| = \sum_{i=1}^k |f_{p_i}| \leq k \cdot |f_p| \leq k \cdot \delta$ since the cycles do not contribute to the flow value $|f_p|$ is the augmenting path in G_f with the maximum δ

$\implies v = |\phi^*_f| \leq k \cdot \delta \implies \delta \geq v/k \geq v/m$ □

Thus, after augmenting with this path p, the value remaining is $\leq v - v/m = v(1-1/m)$ since we have increased the flow by δ. Notice that this implies that in every iteration the value remaining decreases by at least a factor of $1 - 1/m$.

After t iterations of this, the remaining flow is: $|f^*| - |f| \leq |f^*| \cdot (1 - 1/m)^t$ Since all quantities are integer $|f^*| \cdot (1 - 1/m)^t < 1 \implies |f^*| \cdot (1 - 1/m)^t = 0$

$|f^*| \cdot (1-1/m)^t \leq m \cdot U \cdot e^{-t/m}$ where U is the max capacity set $t = m \cdot ln(mU)+1$ then $|f^*| \cdot (1-1/m)^t < 1$ Thus $t = O(m \cdot ln(mU))$ where t is the number of iterations needed The total time required is: $t\cdot$[time to find the maximum δ path] We can compute the maximum δ path in two ways: 1. Dynamic Programming

(Dijkstra's Algorithm) 2. Binary search We can do binary search on the value δ of the max-width path P. In order to test whether $\exists$P with $\delta(P) > \theta$ (where θ is a threshold in the binary search) we can simply drop all edges in G_f that have capacity less than θ and check if $\exists$ s$\to$t path. Finally the binary search will converge on the threshold $\theta = \delta$. Thus the time to do this procedure is $O(m \cdot log(U))$

The total time required is: $O(m \cdot ln(mU)) \cdot O(m \cdot log(U)) \implies O(m^2 \cdot log(U) \cdot log(mU))$

4 Algorithm 2: Scaling

4.1 Algorithm

The idea behind this algorithm is to greedily approximate the graph's edge capacities, by starting with each capacity's most significant bits, and scaling the graph up until the least significant bits of the edge capacities have been included.

We start by setting the number of scaling iterations to be

$$b = \lceil log_2(U) \rceil$$

And G^i to be a modification of graph G with the capacities $c^i(e)$ being equal to the i most significant bits of $c(e)$.

$$c^i(e_j) = \lfloor \frac{c_{e_j}}{2^{b-1}} \rfloor$$

Therefore, $G^b = G$.

The full algorithm is

Algorithm 2: Scaling

```
Initialize: set flow $b := \lceil log_2(U) \rceil$ and $f^0 = 0$ for $i$ = 0, ..., $b$ do
    find max flow $f^i$ in $G^i$ using FF algorithm, starting from flow $2f^{i-1}$;
end
return$f^b$
```

4.2 Correctness

The scaling algorithm is correct. We can show this by proving that each flow $2 \cdot f^{i-1}$ is a valid flow, and therefore f^b is the max flow, since it is computed for the full capacities of each edge.

Claim 2. *Flow $2 \cdot f^{i-1}$ is a valid flow in G^i.*

Proof. By induction.

Base case: Initialize flow=0.

Inductive Hypothesis: Assume $0 \leq f^{i-1} \leq c^{i-1}(e)$ and f^{i-1} satisfies flow conservation.

Inductive step:

By flow conservation of the inductive hypothesis, $0 \leq 2f^{i-1}(e) \leq 2c^{i-1}(e)$.

Because of how we defined c^i, we know that $2c^{i-1}(e) \leq c^i(e)$.

Therefore, flow conservation holds since $2f^{i-1}(e) \leq c^i(e)$.

And so $2f^{i-1}$ is a valid flow in G^i. □

4.3 Runtime

We know that the number of scaling iterations is $b = \lceil log_2(U) \rceil$.

The time for each scaling iteration is equal to the time to run FF in G^i starting from $2f^{i-1}$. This is upper bounded by the max flow in the residual graph $G^i_{2f^{i-1}}$, multiplied by m.

Claim 3. *The remaining flow in G^i is $\leq m$*

Proof. $2F^{i-1}$ is max flow in graph with capacity $2c^{i-1}(e)$.

We know that

$$c^i(e) \in \{2c^{i-1}(e), 2c^{i-1}(e) + 1\}$$

We can use the Max-flow Min-cut Theorem, which states that $\forall f, \forall s$

1)

$$|f| \leq c(s)$$

2)

$$\max_f |f| = \min_s c(s)$$

Using part 2) of the theorem, there exists an s-t cut on G^{i-1} such that

$$|f^{i-1}| = c^{i-1}(s)$$

We know that

$$c^i(s) \leq 2c^{i-1}(s) + (\#\text{edges } s \to \bar{s})$$
$$\leq 2|f^{i-1}| + m$$

Using part 1) of the theorem,

$$|f^i| \leq 2|f^{i-1}| + m$$
$$|f^i| - 2f^{i-1} \leq m$$

Since $|f^i| - 2f^{i-1}$ is the remaining flow in graph G^i when starting with flow $2f^{i-1}$, we have proven our claim. □

Therefore, the total runtime is $O(b * m * m) = O(logU \cdot m \cdot m) = \boxed{O(m^2 log(U))}$

5 Algorithm 3: Shortest Paths

The idea behind this algorithm is to use the Ford-Fulkersen algorithm, but choosing P to be the shortest path in G^f each time. We c

5.1 Real RAM Model

We will assume that we are working with the Real RAM model, in which all registers/cells in memory contain real numbers. We can further assume that we can perform the following operations in constant runtime:

1. Add/Delete
2. Multiplication/Division
3. Max/Min

5.2 Correctness

Proof of correctness follows from the proof of correctness of the original Ford-Fulkerson algorithm, since we are essentially just replacing all nonzero capacities with 1.

5.3 Runtime

We can show that the runtime is $O(m^2 n)$ time total.

Definition 4. *$d_f(u,v) = d_{G_f}(u,v)$ is the shortest path in G_f from u to v.*

Claim 5. *Fix flow f. Let P be the shortest path augmenting f to f'. Then,*

$$d_{p'}(s,v) \geq d_f(s,v)$$

Proof. By contradiction:
Setting

$$A = \{v : d_{f'}(s,v) < d_f(s,v)\} \neq 0$$

$$v = \min_{v \in A} d_{f'}(s,v)$$

Consider the shortest path P in $G_{f'}$ from s to v. Then, with w being the node immediately preceding v, since $w \notin A$,

$$d_{f'}(s,w) \geq d_f(s,w)$$

$$\Rightarrow d_f(s,v) > d_{f'}(s,v) = d_{f'}(s,w) + 1 \geq d_f(s,w) + 1$$

Therefore,

$$d_f(s,v) > d_f(s,w) + 1$$

This implies that that

$$(w,v) \notin G_f$$

since $(w,v) \in G_f$ implies that $d_f(s,v) \leq d_f(s,w) + 1$, which contradicts the above inequality.
Therefore, we know

$$(w,v) \in G_{f'}$$

This means that the shortest augmenting path in G_f would pass from $v \to w$.
However, this is a contradiction to the above inequality $d_f(s,v) > d_f(s,w) + 1$.

□

Proof of runtime to be concluded next lecture.