Max-Flow I and II Notes

Max-Flow I

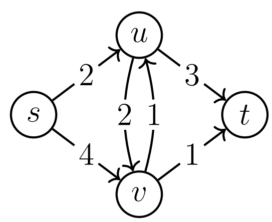
Max-flow is a fundamental problem with many applications in logistics and routing etc.

1 Input

A *flow network* $G = (V, E, s, t, c)$ has:

- Network (i.e. directed graph) (V, E)
- One source node $s \in V$ and one target node $t \in V$
- Edge capacities $c : V^2 \to \mathbb{R}_{\geq 0}$ where $c(u, v) = 0$ if $(u, v) \notin E$

Intuition: Each edge is a one-way road, and the capacity is the number of lanes on that road.



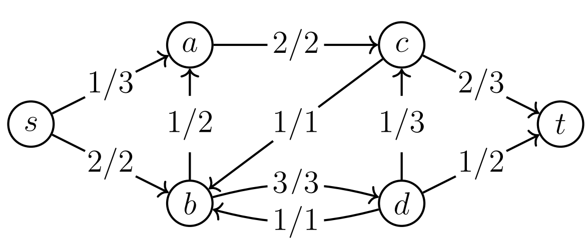
Fundamental question: What is the max value of traffic from s to t that we can support?

1.1 Gross Flow

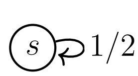
Gross flow is specified by a function $g : E \to \mathbb{R}_{\geq 0}$, satisfying the following properties.

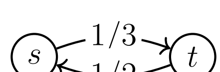
- *Feasibility:* $0 \leq g(e) \leq c(e)$ for all $e \in E$.
- *Flow conservation:* $\displaystyle\sum_{(x,v)\in E} g(x, v) = \sum_{(v,y)\in E} g(v, y)$ for all $v \in V \setminus \{s, t\}$.

Example: Each edge is labelled by $g(e)/c(e)$ where $g(e)$ is the rate and $c(e)$ is the value.
Here we can see all flows $\leq$ capacities and "flow in = flow out" for every node.



However, this definition is not convenient when dealing with flow cycles of length 1 or 2:





1.2 Net flow

Firstly, we won't assume that s is a source and t is a sink (i.e. both can have in- and out- neighbors), so $c(u,v) = 0$ for all $(u,v) \notin E$. We now define *net* flow.

Net flow is specified by a function $f : V^2 \to \mathbb{R}$, satisfying the following properties:

- *Feasibility:* $f(u,v) \le c(u,v)$ for all $u, v \in V$.
- *Flow conservation:* $\sum_{x \in V} f(v,x) = 0$ for all $v \in V \setminus \{s,t\}$.
- *Skew-symmetry:* $f(u,v) = -f(v,u)$ for all $u, v \in V$.

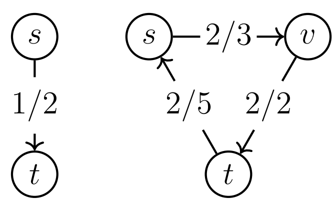
Few corollaries to note:

1. We can have $f(u,v) \le 0$ when $(u,v) \notin E$.
2. $f(v,v) = 0$ for all $v \in V$.
3. If $(u,v),(v,u) \notin E$ then $f(u,v) = 0$.
4. $\sum_{f(x,v)>0} f(x,v) = \sum_{f(v,y)>0} f(v,y)$ for all $v \in V \setminus \{s,t\}$.

We define the **value of flow** f to be $|f| = \sum_{v \in V} f(s,v)$.

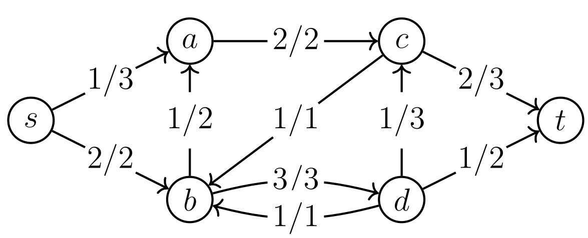
The **Max-Flow Problem** is to find a flow with maximal value for a given flow network G.

Simplest examples of non-zero flow:

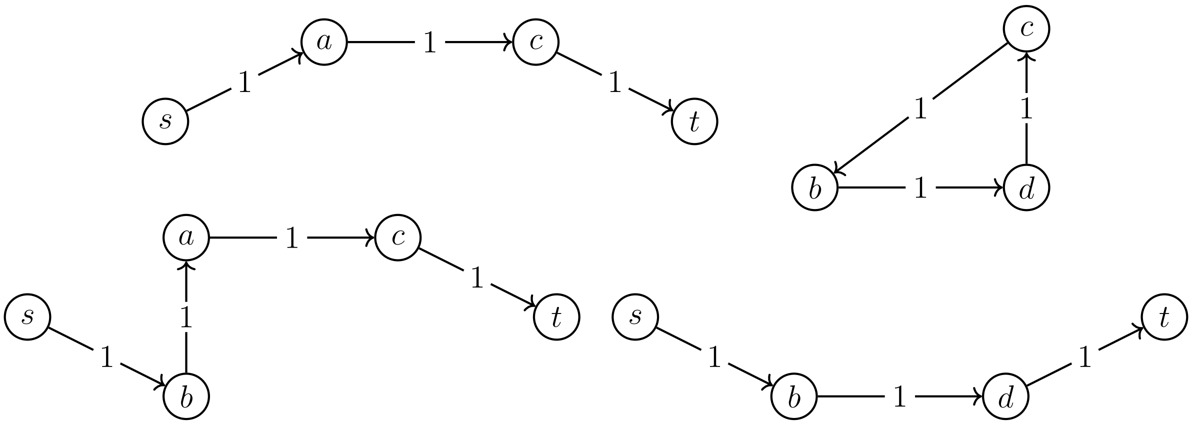


2 Flow Decomposition

An observation: Any flow can be decomposed into a collection of flow cycles and s-t flow paths:



can be decomposed into



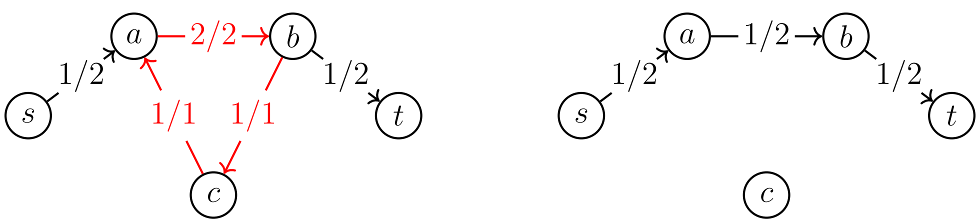
Formally, denote the *support* $\mathrm{supp}_f(G)$ as the subgraph of G with only edges with positive flow.

Flow Decomposition Theorem.
For any flow f with $|f| \geq 0$, $\mathrm{supp}_f(G)$ can be decomposed into a collection of flow cycles and s-t flow paths.

Proof.
We induct on the number of edges in $\mathrm{supp}_f(G)$. The base case is where the support has no edges (so $|f| = 0$), and indeed it can be decomposed into an (empty) set of flow cycles and s-t flow paths. Now assume $\mathrm{supp}_{f'}(G)$ can always be decomposed into flow cycles and s-t flow paths when $|f'| < L$. Let there be a flow f with $|f| = L$.
Suppose that we can find a cycle C in $\mathrm{supp}_f(G)$. Then we can "reduce that cycle" by substracting all the flows on the cycle by $f_{\min}(C) = \min\limits_{e \in C} f(e)$. Formally, we define a new flow f' on the graph:

$$f'(u,v) = \begin{cases} f(u,v) - f_{\min}(C) & (u,v) \in C \\ -f(u,v) + f_{\min}(C) & (v,u) \in C \\ f(u,v) & \text{otherwise} \end{cases}$$

For example,



We can check that f' still satisfies feasibility, flow conservation and skew-symmetry, so f' is a flow.

Now that $\mathrm{supp}_{f'}(G)$ has at least one edge fewer than $\mathrm{supp}_f(G)$, we can apply the inductive hypothesis, decomposing $\mathrm{supp}_{f'}(G)$ and then adding the flow cycle back to get $\mathrm{supp}_f(G)$.

What if there are no cycles? Note that each vertex, that is not s nor t, with an in-edge must have an out-edge by flow conservation, so if we perform DFS starting from s, we are going to reach either t or s again. The latter case forms a cycle, so we instead reach just t.

Now we have an s-t path P, we reduce it by subtracting the flow by $f_{\min}(P) = \min_{e \in P} f(e)$. In other words, we define again a new flow

$$f'(u,v) = \begin{cases} f(u,v) - f_{\min}(P) & (u,v) \in P \\ -f(u,v) + f_{\min}(P) & (v,u) \in P \\ f(u,v) & \text{otherwise} \end{cases}$$

and check that it still satisfies it being a flow. Then at least one edge is removed, and we can use the inductive hypothesis similarly. $\square$

Exercise. There is always a flow decomposition with at most $|E|$ flow cycles and s-t flow paths.

Exercise*. If $|f| > 0$, then there must be at least one s-t flow path in $\text{supp}_f(G)$.

3 Cuts

Let f^* be a max flow of G (need not be unique), and denote $F^* = |f^*|$.

Warmup. How to check whether $F^* > 0$ using flow decomposition?

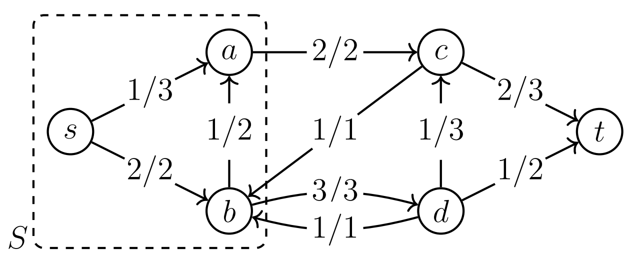
Let G^* be the subgraph of G with only edges with positive capacities. Note: $\text{supp}_f(G)$ is the subgraph with positive *flow* edges, and hence depends on f; whereas G^* is the subgraph with positive *capacity* edges, and hence is independent of flow.

By flow decomposition, $F^* > 0$ *if and only if* there is an s-t path in G^*.

So how do we certify when there is no s-t path in G^*? We use a *cut*.

Let $\overline{S} = \{v \in V \mid \text{exists } s\text{-}v \text{ path in } G^*\}$. Then when $F^* = 0$, $s \in \overline{S}$ but $t \in V \setminus \overline{S}$.

More generally, we define an s-t **cut** to be a cut $(S, V \setminus S)$ such that $s \in S$ and $t \in V \setminus S$.



We also define the *capacity of a cut* $c(S) = c(S, V \setminus S) = \sum_{u \in S} \sum_{v \in V \setminus S} c(u,v)$. In other words, $c(S)$ is the total capacity of edges leaving S. Then

$$F^* = 0 \iff \text{no } s\text{-}t \text{ path in } G^* \iff \text{exists } s\text{-}t \text{ cut } S \text{ with } c(S) = 0.$$

Therefore, s-t paths and s-t cuts are dual to each other!

Minimum s-t cut problem: Given G, find an s-t cut of minimum capacity.

Let $S^* = \underset{S\ s\text{-}t \text{ cut}}{\operatorname{argmin}}\, c(S)$ be a cut with minimum capacity. Hence $F^* = 0 \Leftrightarrow c(S^*) = 0$.

Given a flow f, we can also define a *flow across a cut* $f(S) = f(S, V \setminus S) = \sum_{u \in S} \sum_{v \in V \setminus S} f(u, v)$. By feasibility, $f(S) \le c(S)$ for any s-t cut S.

Claim. For any two s-t cuts S, S', we have $f(S) = f(S')$.

Proof. By flow decomposition, f is a collection of flow cycles and s-t flow paths. For any cut S,

- Flow cycles contribute 0 to both $f(S)$ because of skew-symmetry and the fact that cycles will zig-zag into and out of S.
- s-t flow paths contribute to $f(S)$ the amount equal to the flow of the path itself, because paths will zig-zag into and out of S, cancelling out except for the last zig, which contributes the flow.

In both cases, $f(S)$ did not depend on S, so $f(S) = f(S')$. $\square$

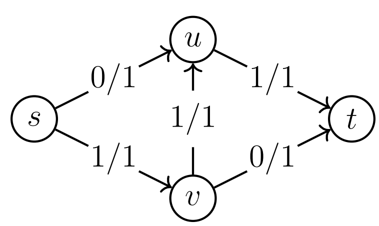
Therefore, for any s-t cut S, $f(S) = f(\{s\}) = |f|$.

Weak Duality (Maxflow $\le$ Mincut). $|f^*| = f(S) \le c(S) \le c(S^*)$.

4 Increasing Flow

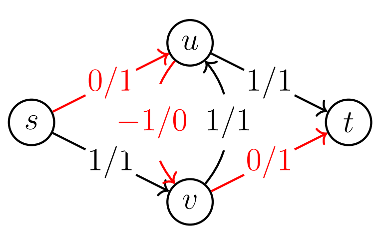
Question: Given a flow f, how to increase its value or conclude that $|f| = F^*$?

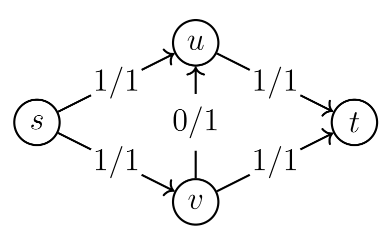
Our first idea is to find another s-t path and push more flow along it. But this does not always work. Consider the following example:



There are no more s-t paths to push flow, yet $|f| = 1$ is not the max flow!

New idea: *Undo some existing flows.* We can think of (u, v) as being there with capacity 0, so we can push flow along $s \to u \to v \to t$, undoing the flow:





Now we have $|f| = 2$! How do we ensure that 2 is a maximum? We can use weak duality and choose an appropriate cut S. By choosing $S = \{s\}$, we get $2 = |f| \le F^* \le c(S^*) \le c(\{s\}) = 2$, confirming maxflow.

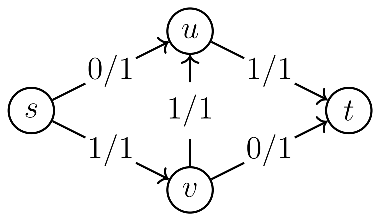
5 Residual Network

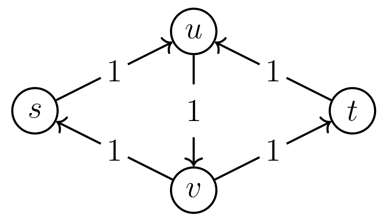
In light of the previous example, we introduce a **residual network** to find pushable flows.

Given a flow f on a flow network G, we define the **residual network** $G_f = (V, E_f, s, t, c_f)$ where

- $c_f(u,v) = c(u,v) - f(u,v)$
- $(u,v) \in E_f \iff c_f(u,v) > 0$

For the previous example, here is how the original flow network and its residual network look like:



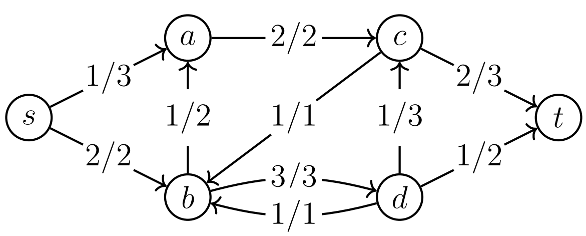


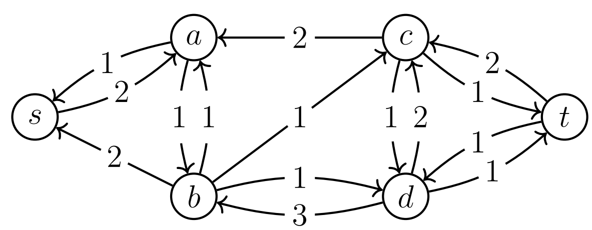
Notice that a non-edge $(u,v) \notin E$ can be in E_f if $f(v,u) > 0$. This is because

$$c_f(u,v) = c(u,v) - f(u,v) = 0 + f(v,u) > 0.$$

Therefore this definition allows us to undo flow!

Another example of flow vs residual:





Observation: If f, f' are flows in G, G_f respectively, then $f + f'$ is a flow in G.

So to improve f, we just need to find a non-zero flow in G_f.

What if there is no non-zero flow in G_f? Then by setting $\overline{S}$ as the set of vertices reachable from s in G_f, we have $c_f\left(\overline{S}\right) = 0$, which rearranges to

$$\sum_{u \in \overline{S}} \sum_{v \in V \setminus \overline{S}} (c(u,v) - f(u,v)) = 0 \Longrightarrow c\left(\overline{S}\right) = f\left(\overline{S}\right).$$

Applying weak duality, $c\left(\overline{S}\right) = f\left(\overline{S}\right) = |f| \leq c(S^*) \leq c\left(\overline{S}\right)$, so f is a max-flow and $\overline{S}$ is a min-cut!

In conclusion, we have

$$|f| = F^* \iff \text{no } s\text{-}t \text{ path in } G_f$$

6 Towards an Algorithm

Define an *augmenting path* as a directed s-t path in G_f. We have shown that we should always push additional flow along such a path P up to the (residual) bottleneck capacity $c_f(P) = \min_{e \in P} c_f(e)$, increasing $|f|$ by $c_f(P)$. When there is no more augmenting path, we have also shown that f is a max-flow. We can write this as the FORD-FULKERSON algorithm:

Algorithm 1 FORD-FULKERSON (1956)

```
Start with zero flow
while G_f has augmenting path, do
    Find augmenting path P in G_f via DFS          ▷ O(|E|)
    Augment flow by pushing c_f(P)                 ▷ O(|E|)
end while
```

Runtime. If all capacities are integers in $[0, C]$ then

$$\text{no. of augmentations} \le |f| \le c(\{s\}) \le |V| \cdot C$$

and thus the runtime is $O(|E| \cdot |f|) = O(|E| \cdot |V| \cdot C)$. This is *pseudopolynomial*: It is polynomial in the *numerical value* C, i.e. polynomial in the *unary* coding $(1, 11, 111, \cdots)$ of the numbers instead of binary.

By this algorithm, we also have the flow integrality theorem.

Flow Integrality Theorem. If all capacities are integers, there exists an integral max-flow.

Note that if the capacities are irrational, this algorithm may potentially take infinite runtime!

Max-Flow II

7 Maxflow-Mincut Theorem

Maxflow-Mincut Theorem (Strong Duality). The following are equivalent:

- There is an s-t cut $(S, V \setminus S)$ such that $c(S) = f(S) = |f|$.
- f is a max flow.
- There is no s-t path in G_f.

Proof.
(1) $\Rightarrow$ (2). Use weak duality: $|f| \le |f^*| \le c(S^*) \le c(S) = f(S) = |f|$
(2) $\Rightarrow$ (3). Contrapositive: If there is an s-t path in G_f, push the flow!
(3) $\Rightarrow$ (1). Proven in section 5.

8 Other Algorithms

8.1 Max Bottleneck Path Algorithm (MBP)

Algorithm 2 MAX BOTTLENECK PATH

```
1: Start with zero flow
2: while G_f has augmenting path, do
3:     Find augmenting path P in G_f that has maximum c_f(P)      ▷ O(|E| log |V|)
4:     Augment flow by pushing c_f(P)                             ▷ O(|E|)
5: end while
```

Runtime. It turns out that the number of iterations is $O(|E| \log |f|)$, so the runtime is

$$O(m \log n) \cdot O(m \log |f|) = O(m^2 \log n \log(nC))$$

which is *weakly polynomial*.

8.2 Edmonds-Karp

Algorithm 3 EDMONDS-KARP

```
1: Start with zero flow
2: while G_f has augmenting path, do
3:     Find augmenting path P in G_f with minimal number of edges
4:     Augment flow by pushing c_f(P)
5: end while
```

Runtime. $O(m^2 n)$. Polynomial (no proof given)

8.3 Later Work

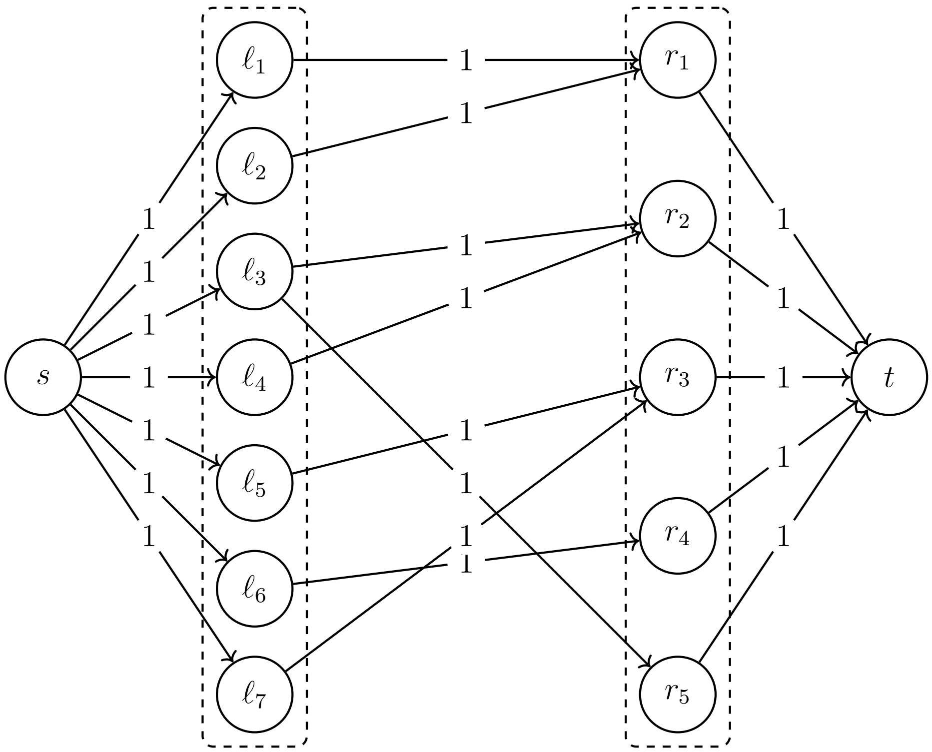
KING-RAO TARJAN (1994):	$O(mn \log_{m/n \log n} n)$
ORTIN (2013):	$O(mn)$
GOLDBERG-RAO (1998):	$O(\min\left(m^{3/2}, mn^{2/3}\right) \log(n^2/m) \log C)$
LEE-SIDFORD (2014):	$O(m\sqrt{n} \log^{O(1)} n \log C)$
MADRY (2013):	$O(m^{10/7} \log^{O(1)} n)$ for $C = 1$
CHEN-KYNG-LIU-PENG-GUTENBERG-SACHDEVA (1998):	$O(m^{1+o(1)} \log C)$

9 Application

A *matching* in a graph G is a set of edges that do not share end points. The size $|M|$ of a matching M is the number of edges in M.

Max Bipartite Matching Problem: Given a bipartite graph G (a graph on vertices $L \cup R$ disjoint, with edges $E \subseteq L \times R$), output a matching of maximum size.

Solution. The method is to create a source s that connects to all vertices in L, and let all vertices in R to connect to a target t. Then we assign a capacity of 1 to all the edges.



1. By flow integrality, since all capacities are integers, there is a max flow with 0 or 1 values on every edge and Ford-Fulkerson will find it.

2. If the maxflow f^* has value F^* then F^* of the edges (s, x) get flow 1 and the rest get 0.

3. The max matching is $\geq F^*$ because given any maxflow f^*, no edges in $L \times R$ with flow 1 have a common node (otherwise the flow in/out is > 1 but the capacities are all 1), and hence we can find a matching with that number of edges, i.e. F^*. The best case is at least as good as this.

4. F^* is at least the max-matching because given any max-matching we can extend the matching to s and t and give flow 1 to all of them, giving a valid flow. The maxflow must be better than this.

5. Therefore $F^* =$ max-matching.

Runtime via Ford-Fulkerson. $O(m|f|) = O(m|M|) = O(mn)$.

10 Optional: Running Time of MBP

Let f_i be the flow computed after iteration i, decomposable into $\leq |E(G_{f_i})|$ flow cycles and s-t flow paths.
Let f_i^* be the max flow in G_{f_i} after i-th iteration.
At the beginning, $|f_0^*| = F^*$.
(Exercise) $f^* = f_i + f_i^*$ for all i.

Since $|E(G_{f_i})| \leq 2m$, there is always an s-t flow path P with flow value $f_i^*(P) \geq \frac{|f_i^*|}{2m}$. For this path P in G_{f_i}, we must have $c_{f_i}(P) \geq f_i^*(P) \geq \frac{|f_i^*|}{2m}$.

Therefore after iteration $(i+1)$, the residual flow

$$f_{i+1}^* \leq |f_i^*| - \frac{|f_i^*|}{2m}$$

which solves to

$$f_{i+1}^* \leq \left(1 - \frac{1}{2m}\right)^i F^*.$$

Consider $I = (4m \ln F^*) + 1$ iterations, then

$$\begin{aligned}
|f_I^*| &\leq \left(1 - \frac{1}{2m}\right)^{4m \ln F^*} F^* \\
&< \left(\frac{1}{e}\right)^{2 \ln F^*} F^* \\
&= \frac{1}{(F^*)^2} F^* \leq 1
\end{aligned}$$

and hence $f_I^* = 0$. Therefore we need $\leq O(m \log F^*)$ iterations. $\square$