Maximum Flow

Flow Networks
Ford-Fulkerson Method
Edmonds-Karp Algorithm
Push-Relabel Algorithms
A flow network is a directed graph where:
Each edge \((u, v)\) has a capacity \(c(u, v) \geq 0\).
If \((u, v)\) is not an edge, then \(c(u, v) = 0\).
There is a source vertex \(s\) and a sink vertex \(t\).
A flow \(f(u, v)\) satisfies the following constraints:

for all \((u, v) \in V \times V\), \(0 \leq f(u, v) \leq c(u, v)\)

for all \((u, v) \in V \times V\), \(f(u, v) = 0 \lor f(v, u) = 0\)

for all \(u \in V - \{s, t\}\), \(\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)\)
Example Flow Network

Flow networks

Ford-Fulkerson

Minimum cut

Edmonds-Karp

Bipartite matching

CS 5633 Analysis of Algorithms
We want to maximize the flow $F$ from $s$ to $t$:

$$F = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$$

Algorithms for maximum flow can be used for liquids flowing through pipes, parts through assembly lines, current through electrical networks, information through communication networks, the maximum matching in a bipartite graph, and the minimum-size cut of a graph.
Multiple Sources and Sinks

flow networks example
flow networks multiple sources/sinks
ford-fulkerson 1
ford-fulkerson 2
ford-fulkerson 3
ford-fulkerson 4
minimum cut
edmonds-karp 1
edmonds-karp 2
edmonds-karp 3
bipartite matching 1
bipartite matching 2
bipartite matching 3
bipartite matching 4

CS 5633 Analysis of Algorithms
Ford-Fulkerson Method

Let $f(\cdot, \cdot)$ be a flow. Define residual capacity $c_f$

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

Define residual edges $E_f$ to be the edges with positive residual capacity.

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$
Ford-Fulkerson Algorithm

**Ford-Fulkerson** \((G, s, t)\)

- **for** each edge \((u, v) \in G.E\)
  - \(f[u, v] \leftarrow 0\)
- **while** (there is a path \(p\) from \(s\) to \(t\) in \(G_f\))
  - \(x \leftarrow \min\{c_f(u, v) : (u, v) \in p\}\)
  - **for** each edge \((u, v) \in p\)
    - \(y \leftarrow f[u, v] - f[v, u] + x\)
    - \(f[u, v] \leftarrow \max(0, y)\)
    - \(f[v, u] \leftarrow \max(0, -y)\)

A maximum flow is found because every path from \(s\) to \(t\) is at full capacity.
Ford-Fulkerson Example

Flow networks example
flow networks multiple sources/sinks
ford-fulkerson 1
ford-fulkerson 2
ford-fulkerson 3
ford-fulkerson 4
minimum cut
edmonds-karp 1
edmonds-karp 2
edmonds-karp 3
bipartite matching 1
bipartite matching 2
bipartite matching 3
bipartite matching 4
Ford-Fulkerson Example Continued

Flow networks example
Flow networks multiple sources/sinks
Ford-Fulkerson 1
Ford-Fulkerson 2
Ford-Fulkerson 3
Ford-Fulkerson 4
Minimum cut
Edmonds-Karp 1
Edmonds-Karp 2
Edmonds-Karp 3
Bipartite matching 1
Bipartite matching 2
Bipartite matching 3
Bipartite matching 4
A cut \((S, T)\) of a flow network partitions \(V\) into vertices \(S\) and \(T = V - S\) s.t. \(s \in S\) and \(t \in T\).

The capacity \(c(S, T)\) of a cut \((S, T)\) is:

\[
\sum_{u \in S} \sum_{v \in T} c(u, v)
\]

\(f(\cdot, \cdot)\) is a maximum flow iff there is a cut \((S, T)\), where there is no flow from \(T\) to \(S\), and the flow from \(S\) to \(T = c(S, T)\).
In this case, \((S, T)\) is a minimum cut.
Choose \(S = \) vertices reachable from \(s\) in \(G_f\)
The Edmonds-Karp algorithm modifies Ford-Fulkerson by finding the shortest augmenting path (as found by BFS).

Running time is $O(VE^2)$. 
Lemma: Residual path lengths do not decrease.

Suppose the distance from $s$ to $v$ decreased after an iteration.

Let $v$ be the closest vertex where this happens.

Some vertex $u$ became adjacent to $v$ (otherwise $v$ could not be closer).

This implies $(v, u)$ is on augmenting path, but this implies $u$ is farther from $s$ than $v$.

A contradiction, so distances do not decrease.
Lemma: Each edge is critical $O(V)$ times.

Some $c_f(u, v)$ is minimum on an augmenting path. This implies $(u, v)$ removed from residual edges. For $(u, v)$ to reappear as a residual edge, $(v, u)$ must be on augmenting path. Because distance to $v$ cannot decrease, distance to $u$ must have increased. This implies each edge can be minimal $O(V)$ times.
An undirected graph $G$ is a *bipartite graph* if $V = L \cup R$ and all edges are between $L$ and $R$ (no edges within $L$ or within $R$).

A *matching* is a subset of edges $M \in E$ s.t. no two edges share a vertex.

A *maximum matching* is a matching with maximum cardinality.
Suppose you have:
- a set of workers $L$,
- a set of jobs $R$, and
- a set of edges from $L$ to $R$ indicating which workers can do which jobs.

Suppose we can only assign one worker to one job, and we want to maximize the number of workers/jobs that are assigned.

The solution to this problem is a maximum bipartite matching.
Create a directed graph $G'$ with $V' = V \cup \{s, t\}$

Add an edge from $s$ to each vertex in $L$.

For each edge $(l, r) \in E$, add a directed edge $(l, r)$ to $G'$.

Add an edge from each vertex in $R$ to $t$.

All edges have a capacity of 1.

A maximum flow found by Ford-Fulkerson is a maximum matching for $G$. 