Flow networks

What if weights in a graph are maximum capacities of some flow of material?

- Pipe network to transport fluid/gas (e.g., water, oil, natural gas, CO₂)
  - Edges – pipes
  - Vertices – junctions of pipes
- Data communication network
  - Edges – network connections of different capacity
  - Vertices – routers (do not produce or consume data just move it)
- Also used in resource planning, economics, ecosystem network analysis

To do:
[CLRS] 26
#7
Flow Network Definitions

Directed graph $G = (V,E)$

One source and one sink.

Each edge has capacity $c(u,v) \geq 0$.

Each vertex is on some path from $s$ to $t$.

Flow Definitions

How much is currently flowing – $f : V \times V \rightarrow \mathbb{R}$

$$f(V,u) = \sum_{v \in V} f(v,u)$$

$$f(u,V) = \sum_{v \in V} f(u,v)$$

Must satisfy 3 properties:

• Capacity constraint:
  $\forall u,v \in V: f(u,v) \leq c(u,v)$

• Skew symmetry:
  $\forall u,v \in V: f(u,v) = -f(v,u)$

• Flow conservation:
  $\forall u \in V - \{s,t\}: f(u,V) = f(V,u) = 0$

What goes in must go out.

Total value of flow $f$:

$|f| = f(s,V) = f(V,t)$
Example flows

Valid or invalid? Why?

Maximize the flow
Maximum flow: Algorithm idea

- If we have some flow, …

- …and can find an *augmenting path* $s \rightarrow p \rightarrow t$

  can add a constant amount of flow along path:
  $\exists a > 0, \forall (u,v) \in p, f(u,v) + a \leq c(u,v)$

- Then just do it, to get a better flow!

---

Ford-Fulkerson method

Ford-Fulkerson($G, s, t$)

1. initialize flow $f$ to 0 everywhere
2. while there is an augmenting path $p$ do
3. augment flow $f$ along $p$
4. return $f$

- How do we find/choose an augmenting path?
- How much additional flow can we send through that path?
- Does the algorithm always find the maximum flow?
Augmenting

Augmenting path – any path in the *residual network*:
- Residual network: \( G_f = (V, E_f) \)
  \( E_f = \{ (u,v) \in V \times V : c_f(u,v) > 0 \} \)
- Residual capacities:
  \( c_f(u,v) = c(u,v) - f(u,v) \)
- Residual capacity of path \( p \):
  \( c_f(p) = \min_{(u,v) \in p} c_f(u,v) \)

Observe – Edges in \( E_f \) are either edges in \( E \) or their reversals: \(|E_f| \leq 2|E|\).

### Ford-Fulkerson method, with details

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

1. **for** each edge \((u,v)\) \(\in\) \(G.E\) **do**
2. \quad \(f(u,v) = f(v,u) = 0\)
3. **while** \(\exists\) path \(p\) from \(s\) to \(t\) in residual network \(G_f\) **do**
4. \quad \(c_f = \min\{c_f(u,v) : (u,v) \in p\}\)
5. **for** each edge \((u,v)\) in \(p\) **do**
6. \quad \(f(u,v) = f(u,v) + c_f\)
7. \quad \(f(v,u) = -f(u,v)\)
8. **return** \(f\)

Algorithms based on this method differ in how they choose \(p\).
Does it always find a maximum flow?

**Cut**
- a partition of V into S, T such that \( s \in S, \ t \in T \)

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

\[
|f| = f(S, T)
\]

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

**Minimum cut**
- a cut with minimum capacity

---

**Does it always find a maximum flow?**

Max-flow min-cut theorem:

The following are equivalent statements:

1. \( f \) is a maximum flow in \( G \).
2. The residual network \( G_f \) contains no augmenting paths.
3. \(|f| = c(S, T)\), for some cut \((S, T)\) of \( G \).

We will prove three parts:
- 1.
- 3. \( \Rightarrow \) 2.

From this we have 2. \( \Rightarrow \) 1., which means that the Ford-Fulkerson method always correctly finds a maximum flow.
What is the worst-case running time?

- Augmentation = ? \(O(E)\)
- How many augmentations?
  - Let’s assume integer flows.
  - Each increases the value of the flow by some integer.
  - \(O(|f^*|)\), where \(f^*\) is the max-flow.
- Total worst-case = \(O(E\times|f^*|)\).

- How an augmenting path is chosen is very important!

---

Edmonds-Karp Algorithm

Use **shortest** augmenting path (in #edges).

Run algorithm on our example:
Edmonds-Karp algorithm analysis: 1

Augmentation = O(E) – Breadth-first search

Will prove: \#augmentations = O(VE).
Let \( d(v) \) be distance from \( s \) to \( v \) in residual network.
Will prove: Every \(|E|\) iterations, \( d(t) \) increases by \( \geq 1 \).
\( d(t) \) can increase at most \(|V|\) times \( \rightarrow O(VE) \) iterations

Total = \( O(VE^2) \)

Edmonds-Karp algorithm analysis: 2

Will prove: Every \(|E|\) iterations, \( d(t) \) increases by \( \geq 1 \).

Consider the residual network in levels according to \( d(v) \):

As long as \( d(t) \) doesn't change, the paths found will only use forward edges.
- Each iteration saturates & removes at least 1 forward edge, and adds only backward edges (so no distance ever drops).
- After removing \(|E| - d(t) + 1\) forward edges, \( t \) will be disconnected from \( s \).

So, within \(|E|\) iterations, either
- \( t \) is disconnected, & algorithm terminates, or
- A non-forward edge used, & \( d(t) \) increased.
Antiparallel edges

Allow:

Disallow:

Flows cancel.

Residual (multi)graph can have parallel edges.

What if we have multiple sources or sinks?
Maximum Flow: Another Algorithm Idea

Greedily fill outgoing edges to capacity. Later edges’ capacity constraints can lead us to reduce those flows.

Push-Relabel Example – Goldberg & Tarjan (1986)
Relabel a

Push Preflow from a to b & c
Relabel b

Push Preflow from b
Relabel c

Push Preflow from c to d
Relabel c

Push Preflow from c to a
Relabel a & Push Preflow from a to c

Relabel & Push c, Relabel & Push a, Relabel & Push c
Relabel a & Push Preflow from a to s

Relabel c & Push Preflow from c to s
Relabel d & Push Preflow from d to t

Relabel d & Push Preflow from d to b
Push Preflow from b to t

Relabel d & Push Preflow from d to c
Push Preflow from c to s

Correctness Overview

Is this a flow?
Is it maximum? – What is the residual graph?
Correctness Depends on Height Invariant

**Invariant:** Residual edges go down in height by at most one.

**But,** height(s) – height(t) = |V|, longer than any s-t path.

---

Running Time Overview

Max height = 2|V| – 1.

Overall algorithm is $O(V^2E)$.
- Requires amortized analysis.
- See text & homework.

Choosing operation order wisely, can reduce to $O(V^3)$. 
Two Other Approaches

Blocking flows:
- Each s-t path in blocking flow contains a saturated edge.
- Dinitz/Dinic (1970)
  \(O(VE \log V)\)
- Goldberg & Rao (1997)
  \(O(\min(V^{2/3},E^{1/2}) \log(V/2E) \log(\max_{(u,v) \in E} c(u,v)))\)

Combinatorial game:
- Cheriyan, Hagerup (1989)
  \(O(VE + V^2 \log^2 V)\) expected
- King, Rao, & Tarjan (1994)
  \(O(VE \log E/(V \log V) V)\)

An Application of Max-Flow

Maximum bipartite matching problem

*Matching* in a graph is a subset \(M\) of edges such that each vertex has at most one edge of \(M\) incident on it. It puts vertices in pairs.

E.g., dating agency
Maximum Bipartite Matching

\[ V = L + R \]
\[ E \subseteq L \times R \]

- How can we reformulate as a max-flow problem?
- What is the running time, using Edmonds-Karp?