Edmonds-Karp algorithm

Edmonds-Karp: modify **algFordFulkerson** so it always returns the shortest augmenting path in $\mathbf{G}_f$.

Definition

For a flow f, let $\delta_f(v)$ be the length of the shortest path from the source s to v in the residual graph $\mathbf{G}_f$. Each edge is considered to be of length 1.

Assume the following key lemma:

Lemma

$\forall v \in V \setminus \{s, t\}$ *the function* $\delta_f(v)$ *increases.*

The disappearing/reappearing lemma

Lemma

During execution **Edmonds-Karp**, *edge* $(u \to v)$ *might disappear/reappear from* $\mathbf{G}_f$ *at most* $n/2$ *times,* $n = |V(\mathbf{G})|$.

Proof.

1. iteration when edge $(u \to v)$ disappears.
2. $(u \to v)$ appeared in augmenting path π.
3. Fully utilized: $c_f(\pi) = c_f(uv)$. f flow in beginning of iter.
4. till $(u \to v)$ "magically" reappears.
5. ... augmenting path σ that contained the edge $(v \to u)$.
6. g: flow used to compute σ.
7. We have: $\delta_g(u) = \delta_g(v) + 1 \geq \delta_f(v) + 1 = \delta_f(u) + 2$
8. distance of s to u had increased by 2. QED.

□

Comments...

1. $\delta_f(u)$ might become infinite
2. Then u is no longer reachable from s.
3. By monotonicity, the edge $(u \to v)$ will never appear again.

Observation

For every iteration/augmenting path of **Edmonds-Karp** *algorithm, at least one edge disappears from the residual graph* $\mathbf{G}_f$.

Edmonds-Karp # of iterations

Lemma

Edmonds-Karp *handles* $O(nm)$ *augmenting paths before it stops.*
Its running time is $O\left(nm^2\right)$, *where* $n = |V(\mathbf{G})|$ *and* $m = |E(\mathbf{G})|$.

Proof.

1. Every edge might disappear at most $n/2$ times.
2. At most $nm/2$ edge disappearances during execution **Edmonds-Karp**.
3. In each iteration, by path augmentation, at least one edge disappears.
4. **Edmonds-Karp** algorithm perform at most $O(mn)$ iterations.
5. Computing augmenting path takes $O(m)$ time.
6. Overall running time is $O\left(nm^2\right)$.

Shortest distance increases during Edmonds-Karp execution

Lemma

Edmonds-Karp run on $\mathbf{G} = (V, E)$, s, t, *then* $\forall v \in V \setminus \{s, t\}$, *the distance* $\delta_f(v)$ *in* $\mathbf{G}_f$ *increases monotonically.*

Proof

1. By Contradiction. f: flow before (first fatal) iteration.
2. g: flow after.
3. v: vertex s.t. $\delta_g(v)$ is minimal, among all counter example vertices.
4. v: $\delta_g(v)$ is minimal and $\delta_g(v) < \delta_f(v)$.

Proof continued...

1. $\pi = s \to \cdots \to u \to v$: shortest path in $\mathbf{G}_g$ from s to v.
2. $(u \to v) \in \mathbf{E}(\mathbf{G}_g)$, and thus $\delta_g(u) = \delta_g(v) - 1$.
3. By choice of v: $\delta_g(u) \geq \delta_f(u)$.
   (i) If $(u \to v) \in \mathbf{E}(\mathbf{G}_f)$ then

$$\delta_f(v) \leq \delta_f(u) + 1 \leq \delta_g(u) + 1 = \delta_g(v) - 1 + 1 = \delta_g(v).$$

This contradicts our assumptions that $\delta_f(v) > \delta_g(v)$.

Proof continued II

(ii) f $(u \to v) \notin \mathbf{E}(\mathbf{G}_f)$:

1. π used in computing g from f contains $(v \to u)$.
2. $(u \to v)$ reappeared in the residual graph $\mathbf{G}_g$ (while not being present in $\mathbf{G}_f$).
3. $\implies$ π pushed a flow in the other direction on the edge $(u \to v)$. Namely, $(v \to u) \in \pi$.
4. Algorithm always augment along the shortest path. By assumption $\delta_g(v) < \delta_f(v)$, and definition of u:
   $$\delta_f(u) = \delta_f(v) + 1 > \delta_g(v) = \delta_g(u) + 1,$$
5. $\implies$ $\delta_f(u) > \delta_g(u)$
   $\implies$ monotonicity property fails for u.
   But: $\delta_g(u) < \delta_g(v)$. A contradiction. ■