Edmonds-Karp Algorithm

- Use a **shortest** augmenting path  
  (via BFS in residual graph)

- Time: $O(n m^2)$

**bfs/shortest-path lemmas**

Distance from $s$ in $G_i$ is never reduced by:

- **Deleting** an edge
  Proof: no new (hence no shorter) path created

- **Adding** an edge $(u,v)$, **provided** $v$ is nearer than $u$
  Proof: BFS is unchanged, since $v$ visited before $(u,v)$ examined
key lemma

Let $f$ be a flow, $G_f$ the residual graph, and $P$ a shortest augmenting path. Then no vertex is closer to $s$ after augmentation along $P$.

Proof: Augmentation along $P$ only deletes forward edges, or adds back edges that go to previous vertices along $P$.

augmentation vs BFS

G: $G_f$ $G_f'$

G: $G_f$ $G_f'$

theorem

The Edmonds-Karp Algorithm performs $O(mn)$ flow augmentations.

Proof:
Call $(u,v)$ critical for augmenting path $P$ if it's closest to $s$ having min residual capacity.

It will disappear from $G_f$ after augmenting along $P$.

In order for $(u,v)$ to be critical again the $(u,v)$ edge must re-appear in $G_f$ but that will only happen when the distance to $u$ has increased by 2 (next slide).

It won't be critical again until farther from $s$ so each edge critical at most $n/2$ times.
critical edges in $G_f$

Shortest s-t path $P$ in $G_f$

After augmenting along $P$

For $(u,v)$ to be critical later for some flow $f'$ it must be in $G_r$
since this is a shortest path so must have augmented along a shortest path containing $(v,u)$

Then we must have $d_r(s,u) = d_r(s,v) + 1 \geq d_r(s,v) + 1 = d_r(s,u) + 2$

project selection

- **Given**
  - a directed acyclic graph $G=(V,E)$ representing precedence constraints on tasks (a task points to its predecessors)
  - a profit value $p(v)$ associated with each task $v \in V$ (may be positive or negative)

- **Find**
  - a set $A \subseteq V$ of tasks that is closed under predecessors, i.e. if $(u,v) \in E$ and $u \in A$ then $v \in A$,
  that maximizes $\text{Profit}(A) = \sum_{v \in A} p(v)$

corollary

- Edmonds-Karp runs in $O(nm^2)$ time

project selection graph

Each task points to its predecessor tasks
extended graph $G'$

- Want to arrange capacities on edges of $G$ so that for minimum $s$-$t$-cut $(S, T)$ in $G'$, the set $A = S\setminus\{s\}$
  - satisfies precedence constraints
  - has maximum possible profit in $G$
- Cut capacity with $S = \{s\}$ is just $C = \sum_{v : p(v) \geq 0} p(v)$
  - $\text{Profit}(A) \leq C$ for any set $A$
- To satisfy precedence constraints don’t want any original edges of $G$ going forward across the minimum cut
  - That would correspond to a task in $A = S \setminus \{s\}$ that had a predecessor not in $A = S \setminus \{s\}$
- Set capacity of each of the edges of $G$ to $C+1$
  - The minimum cut has size at most $C$
extended graph $G'$

Cut value
\[= 13 + 3 + 2 + 3 + 4\]
\[= 13 + 3 + 4 - 8 - 10 - 11 - 12 + 14\]

proof of claim

- $A = S \setminus \{s\}$ satisfies precedence constraints
  - No edge of $G$ crosses forward out of $A$ since those edges have capacity $C + 1$
  - Only forward edges cut are of the form $(v, t)$ for $v \in A$ or $(s, v)$ for $v \not\in A$
  - The $(v, t)$ edges for $v \in A$ contribute
    \[\sum_{v \in A} p(v) = \sum_{v \in A} p(v)\]
  - The $(s, v)$ edges for $v \not\in A$ contribute
    \[\sum_{v \not\in A, p(v) \geq 0} p(v) = C \sum_{v \not\in A, p(v) \geq 0} p(v)\]
  - Therefore the total capacity of the cut is
    \[c(S, T) = C - \sum_{v \in A} p(v) = C - \text{Profit}(A)\]

project selection

- **Claim**: Any $s$-$t$-cut $(S, T)$ in $G'$ such that $A = S \setminus \{s\}$ satisfies precedence constraints has capacity
  \[c(S, T) = C - \sum_{v \in A} p(v) = C - \text{Profit}(A)\]

- **Corollary**: A minimum cut $(S, T)$ in $G'$ yields an optimal solution $A = S \setminus \{s\}$ to the profit selection problem

- **Algorithm**: Compute maximum flow $f$ in $G'$, find the set $S$ of nodes reachable from $s$ in $G'$, and return $S \setminus \{s\}$