CSE 421: Algorithms

Winter 2014
Lecture 20: Capacity-scaling and Edmonds Karp

Reading:
Sections 7.3-7.5

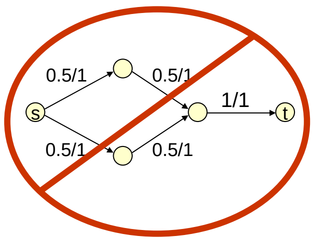
max flow/min cut theorem

Theorem: For any flow **f**, if G_f has no augmenting path then there is some **s-t-cut (A,B)** such that **ν(f)=c(A,B)** (proof on next slide)

- **Corollary:**
  - (1) F-F computes a maximum flow in G
  - (2) For any graph G, the value ν(f) of a maximum flow = minimum capacity c(A,B) of any s-t-cut in G

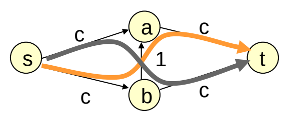
flow integrality theorem

If all capacities are integers
- The max flow has an integer value
- Ford-Fulkerson method finds a max flow in which f(u,v) is an integer for all edges (u,v)



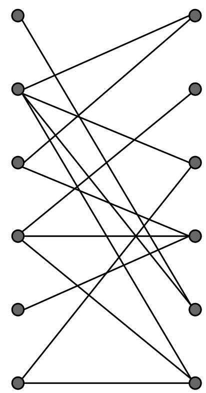
corollaries & facts

- If Ford-Fulkerson terminates, then it has found a max flow.
- It will terminate if c(e) integer or rational (but may not if they're irrational).
- However, may take exponential time, even with integer capacities:

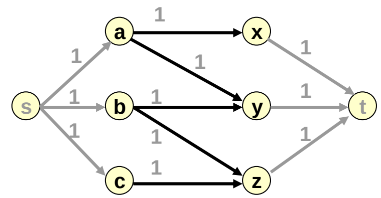


$c = 10^9$, say

bipartite matching



bipartite matching



Integer flows implies each flow is just a subset of the edges

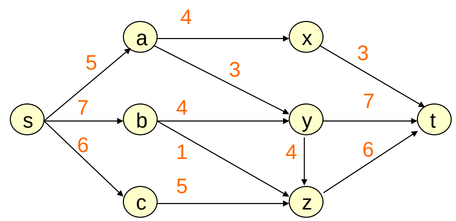
Therefore flow corresponds to a matching

O(mC)=O(nm) running time

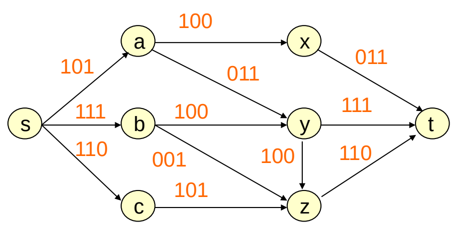
capacity-scaling algorithm

- General idea:
  - Choose augmenting paths **P** with 'large' capacity c_P
  - Can augment flows along a path **P** by any amount $\Delta \leq c_P$
    - Ford-Fulkerson still works
  - Get a flow that is maximum for the high-order bits first and then add more bits later

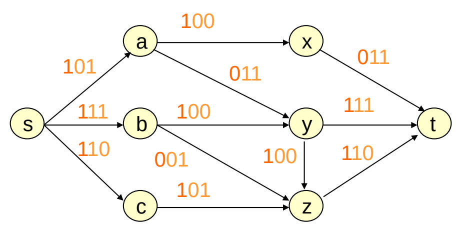
capacity scaling



capacity scaling

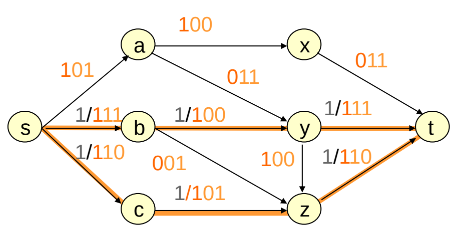


capacity scaling: bit 1



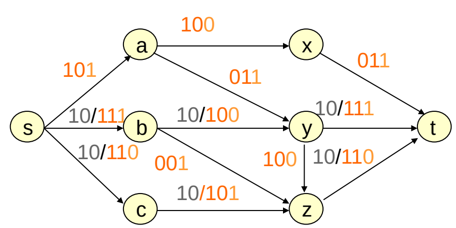
Capacity on each edge is at most **1**
(either **0** or **1** times **Δ=4**)

capacity scaling: bit 1



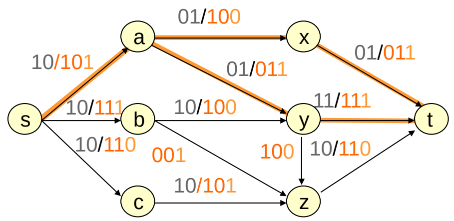
O(nm) time

capacity scaling: bit 2



Residual capacity across min cut is at most **m**
(either **0** or **1** times **Δ=2**)

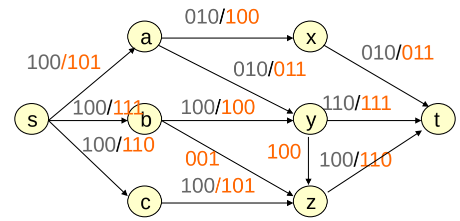
capacity scaling: bit 2



Residual capacity across min cut is at most **m**

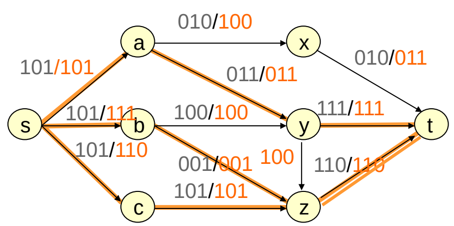
⇒ ≤ **m** augmentations

capacity scaling: bit 3



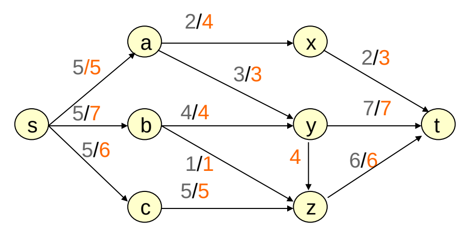
Residual capacity across min cut is at most **m**
(either **0** or **1** times **Δ=1**)

capacity scaling: bit 3

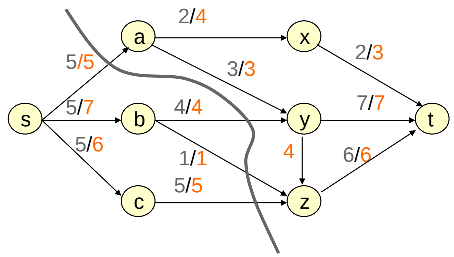


After ≤ **m** augmentations

capacity scaling: final flow



capacity scaling: min cut

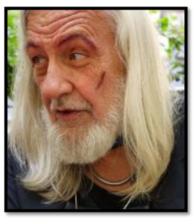


total time for capacity scaling

- $\log_2 U$ rounds where U is largest capacity
- At most m augmentations per round
  - Let c_i be the capacities used in the i^{th} round and f_i be the maxflow found in the i^{th} round
    For any edge (u,v), $c_{i+1}(u,v) \le 2c_i(u,v)+1$
  - $i+1^{st}$ round starts with flow $f = 2\, f_i$
  - Let (A,B) be a min cut from the i^{th} round
    $\nu(f_i)=c_i(A,B)$ so $\nu(f)=2c_i(A,B)$
  - $\nu(f_{i+1}) \le c_{i+1}(A,B) \le 2c_i(A,B)+m = \nu(f)+m$
- $O(m)$ time per augmentation
- Total time $O(m^2 \log U)$

Edmonds-Karp Algorithm

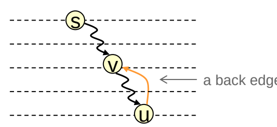
- Use a shortest augmenting path
  (via BFS in residual graph)
- Time: $O(n\, m^2)$

bfs/shortest-path lemmas

Distance from s in G_f 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

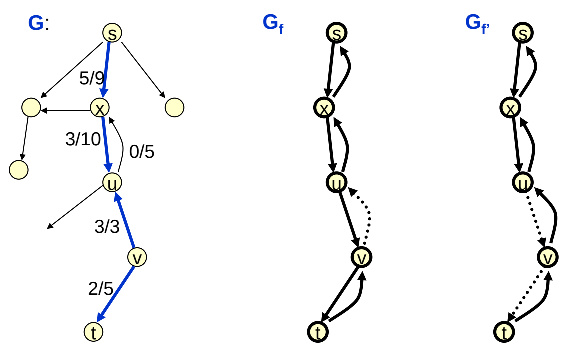
Let **f** be a flow, **G_f** the residual graph, and **P** a shortest augmenting path. Then no vertex is closer to **s** after augmentation along **P**.

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



theorem

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

Proof:

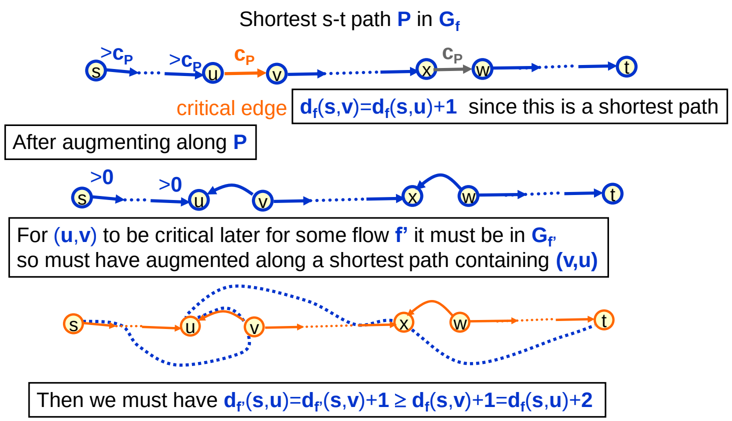
Call (u,v) critical for augmenting path **P** if it's closest to **s** having min residual capacity

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



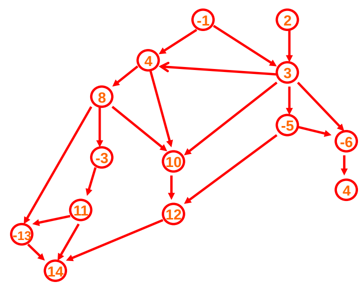
corollary

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

project selection

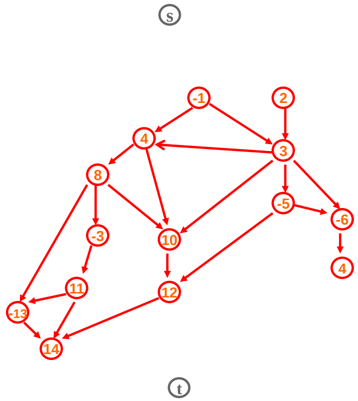
- Given
  - a directed acyclic graph $G=(V,E)$ representing precedence constraints on tasks (a task points to its **predecessors**)
  - a profit value $p(v)$ associated with each task $v \in V$ (may be positive or negative)
- Find
  - a set $A \subseteq V$ of tasks that is closed under predecessors, i.e. if $(u,v) \in E$ and $u \in A$ then $v \in A$, that maximizes $\text{Profit}(A)=\sum_{v \in A} p(v)$

project selection graph



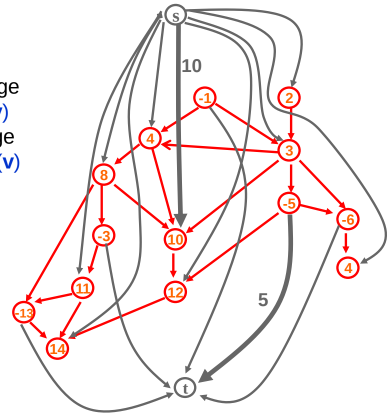
Each task points to its predecessor tasks

extended graph



extended graph G'

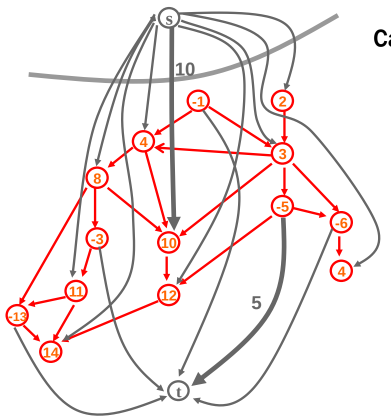
For each vertex **v**
If **p(v)≥0** add **(s,v)** edge
with capacity **p(v)**
If **p(v)<0** add **(v,t)** edge
with capacity **–p(v)**



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-{s}**
  - satisfies precedence constraints
  - has maximum possible profit in **G**
- Cut capacity with **S={s}** is just $C=\sum_{v:\,p(v)\geq 0} p(v)$
  - Profit(A) ≤ 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-{s}** that had a predecessor not in **A=S-{s}**
- Set capacity of each of the edges of **G** to **C+1**
  - The minimum cut has size at most **C**

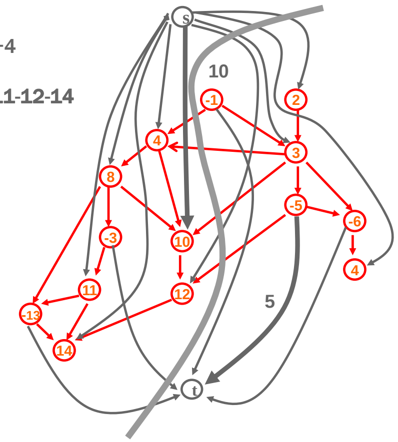
extended graph G'



Capacity **C**

extended graph G'

Cut value
=13+3+2+3+4
=13+3
+C-4-8-10-11-12-14



project selection

- **Claim**: Any s-t-cut (S,T) in G' such that A=S-{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-{s} to the profit selection problem
- **Algorithm**: Compute maximum flow f in G', find the set S of nodes reachable from s in G'_f and return S-{s}

proof of claim

- A=S-{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 $v\notin A$
  - The (v,t) edges for $v\in A$ contribute
    $\sum_{v\in A:p(v)<0} -p(v) = - \sum_{v\in A:p(v)<0} p(v)$
  - The (s,v) edges for $v\notin A$ contribute
    $\sum_{v\notin A:\, p(v)\geq 0} p(v)=C-\sum_{v\in A:\, p(v)\geq 0} p(v)$
  - Therefore the total capacity of the cut is
    $c(S,T) = C - \sum_{v\in A} p(v) = C-\text{Profit}(A)$