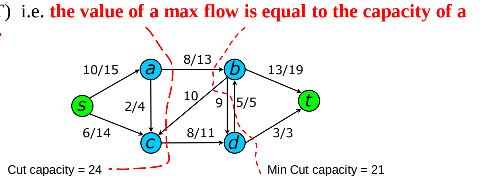
Cuts

- A **cut** is a partition of V into S and T = V – S, such that $s \in$ S and $t \in$ T
- Does the method find the minimum flow?
  - Yes, if we get to the point where the residual graph has no path from *s* to *t*
  - The **net flow** f(S,T) through the cut is the sum of flows f(u,v), where $s \in$ S and $t \in$ T
    - Includes negative flows back from T to S
  - The **capacity** c(S,T) of the cut is the sum of capacities c(u,v), where $s \in$ S and $t \in$ T
    - The sum of positive capacities
  - **Minimum cut** – a cut with the smallest capacity of all cuts.

    |f|= f(S,T) i.e. **the value of a max flow is equal to the capacity of a min cut.**

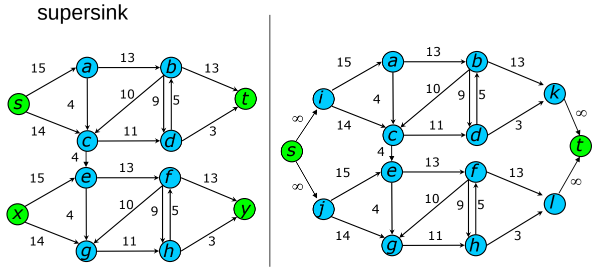


Max Flow / Min Cut Theorem

1. Since $|f| \leq c(S,T)$ for all cuts of (S,T) then if $|f| = c(S,T)$ then c(S,T) must be the min cut of G
2. This implies that f is a maximum flow of G
3. This implies that the residual network G_f contains no augmenting paths.

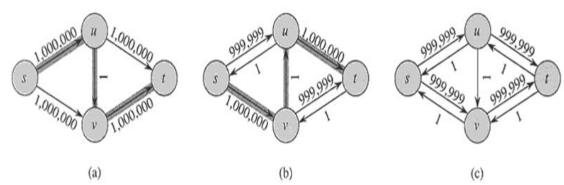
Multiple Sources or Sinks

- What if you have a problem with more than one source and more than one sink?
- Modify the graph to create a single supersource and supersink



Worst Case Running Time

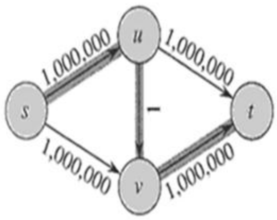
- Assuming integer flow
- Each augmentation increases the value of the flow by some positive amount.
- Augmentation can be done in O(E).
- Total worst-case running time O(E|f*|), where f* is the max-flow found by the algorithm.
- Example of worst case:



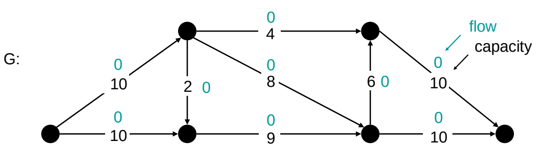
(a) Augmenting path of 1 (b) Resulting Residual Network (c) Resulting Residual Network

Edmonds Karp

- Take **shortest path** (in terms of number of edges) as an augmenting path – Edmonds-Karp algorithm

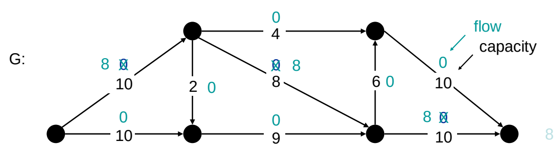


Edmonds-Karp Algorithm

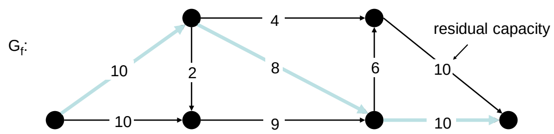


Flow value = 0

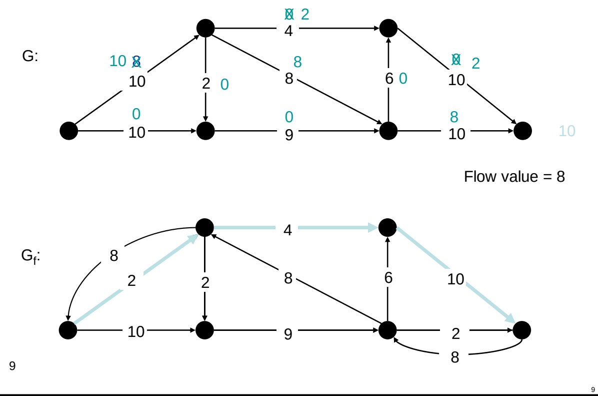
Edmonds-Karp Algorithm

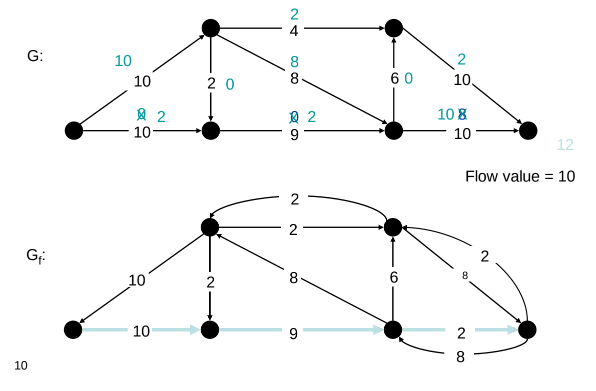


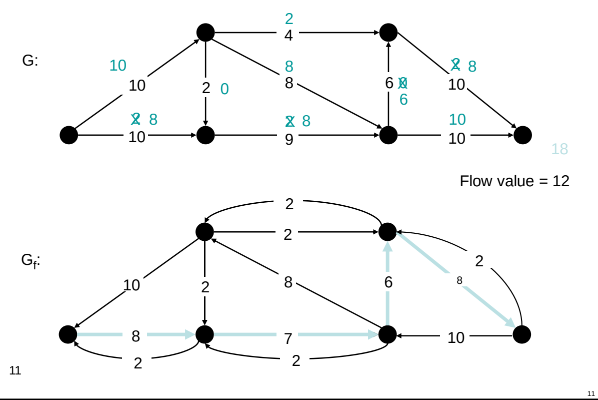
Flow value = 0

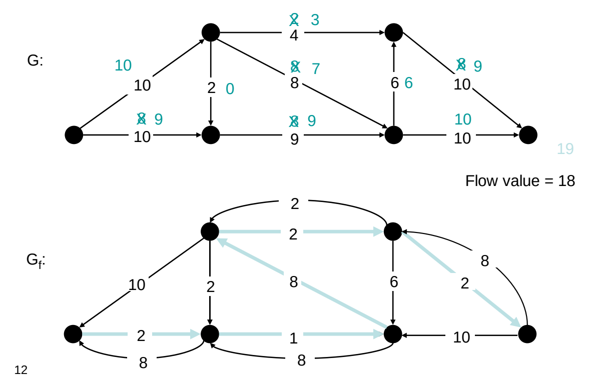


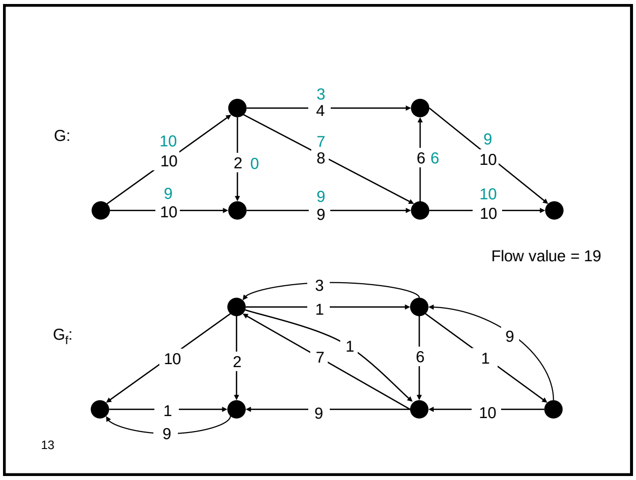
Edmonds-Karp Algorithm

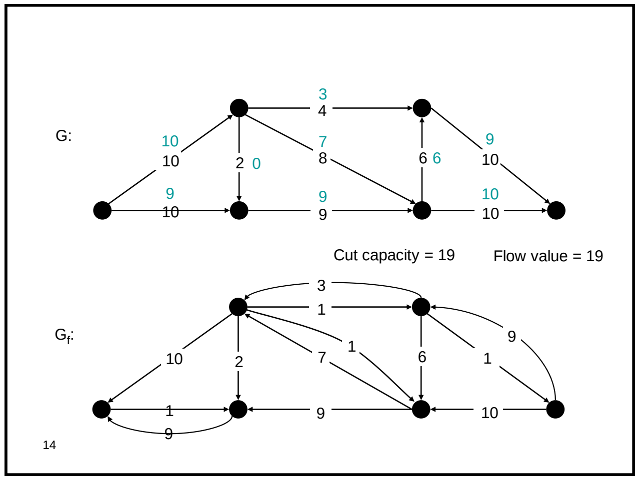









G:
G_f:
Flow value = 19
13


G:
G_f:
Cut capacity = 19
Flow value = 19
14