CMP(N)302: Design and Analysis of Algorithms

Lecture 09: Flow Networks

Ahmed Hamdy

Computer Engineering Department

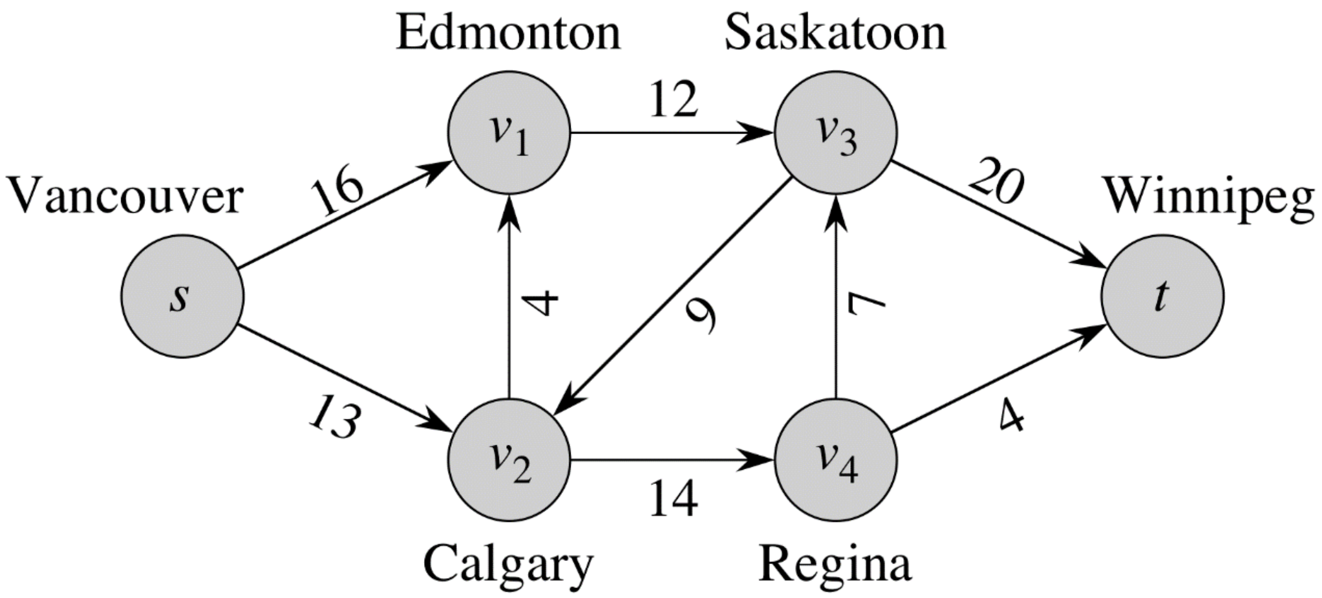
Cairo University

Fall 2019

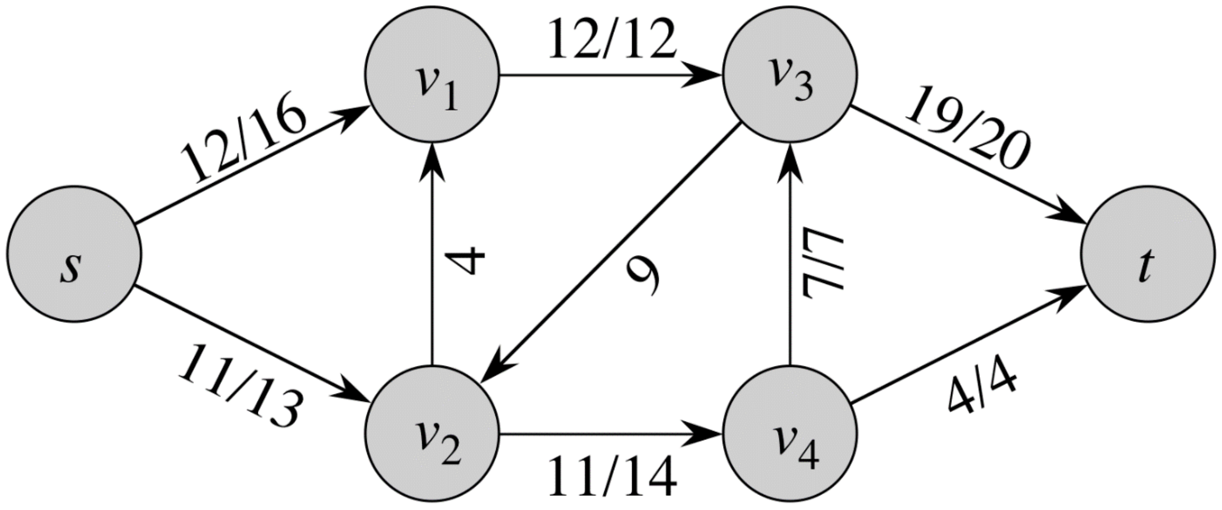
Real life application

- Find max water rate flowing from source (Vancouver) to sink (Winnipeg) in the shown pipe network based on the shown capacities.

- Can you guess an upper-bound for what we can get from just looking?
- Iteratively how do you come up with the best solution?
- Between iterations, how to simplify and prepare the network for the next iteration?

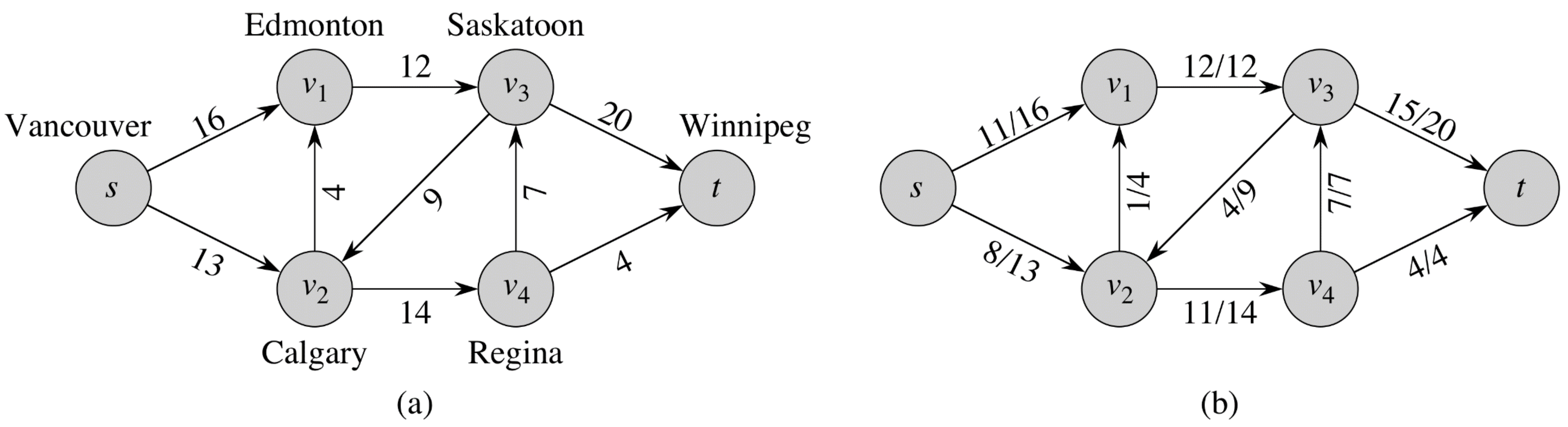


(a)



What is a flow network?

- ***Flow network*** is a directed graph $G(V, E)$
- Each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$. No self-loops.
- If there is $(u, v) \in E$, then there is no edge (v, u) in reverse direction, and $c(v, u) = 0$
- Typically, there is source s and sink t



What is a flow?

- ***Flow*** is a real-valued function $f: V \times V \to \mathbb{R}$ that satisfies:
  - Capacity constraint: For all $u, v \in E$, we require

$$0 \leq f(u,v) \leq c(u,v)$$

  - For all $u, v \in V - \{s,t\}$, we require

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

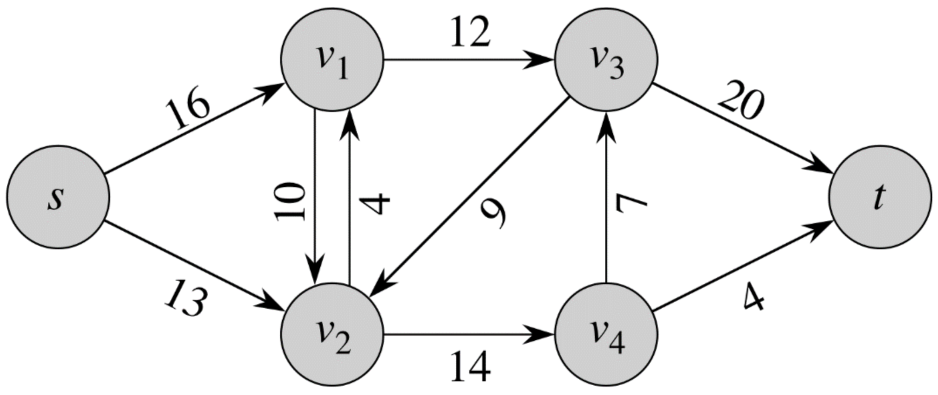
Modeling with antiparallel edges

- Antiparallel edge:

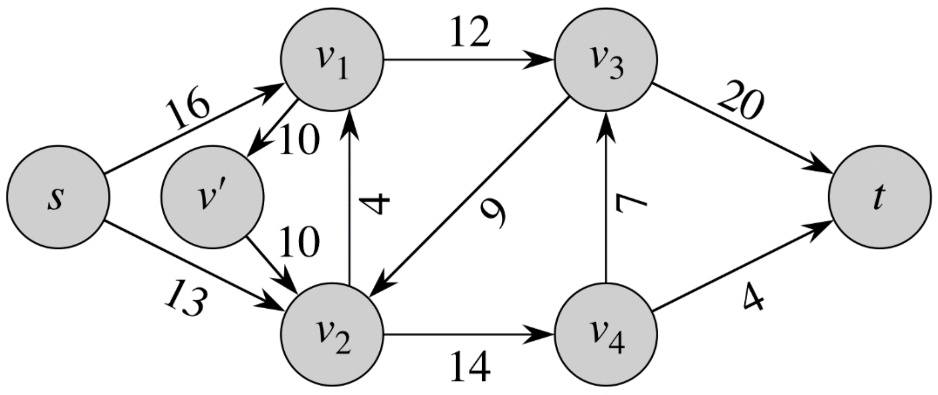
  Edges (v_1, v_2) and (v_2, v_1) are called antiparallel

- Workaround:

  Split one of the edges using a new vertex

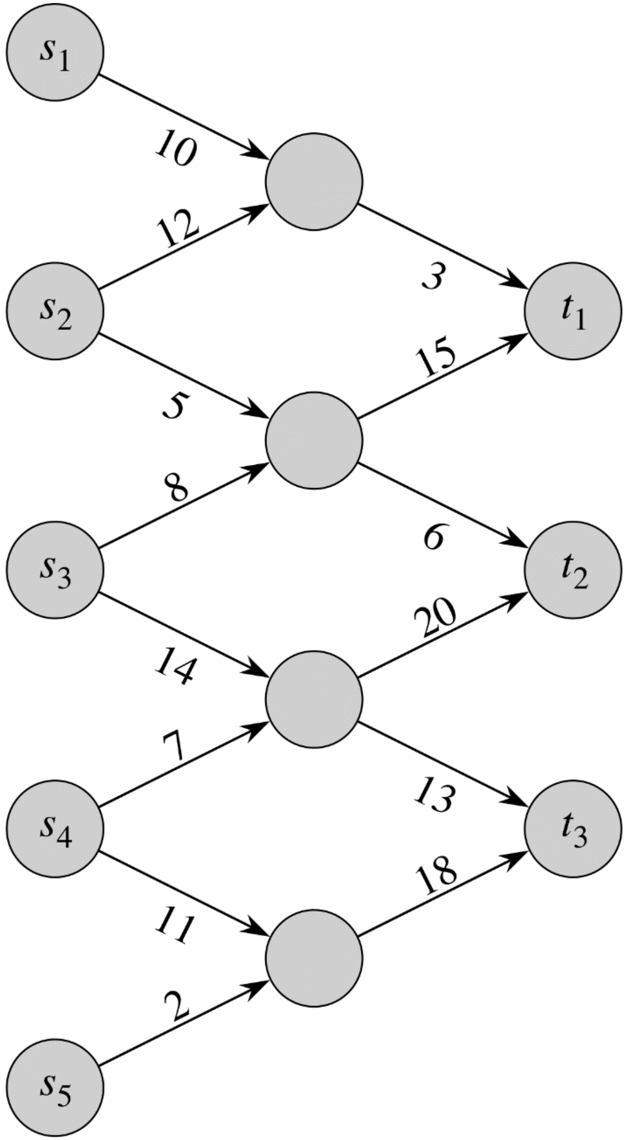


(a)

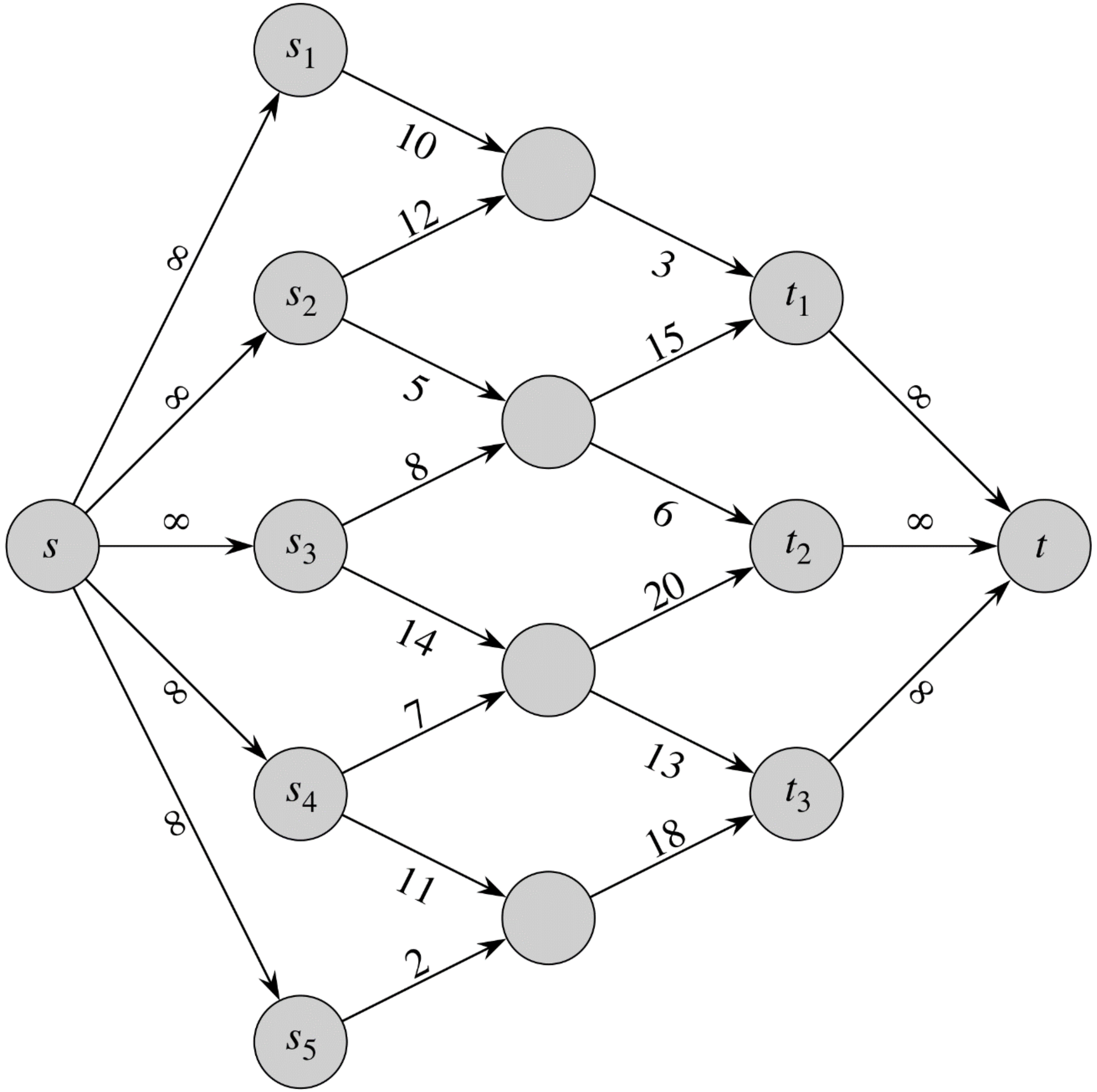


(b)

Modeling with multiple sources/sinks



(a)



(b)

Ford-Fulkerson method

- Method not algorithm because it has several implementations with different complexities
- Greedy, greedy, greedy!!!
- Main ideas
  - Residual networks
  - Augmenting paths
  - Cuts

```
FORD-FULKERSON-METHOD(G, s, t)
  initialize flow f to 0
  while there exists an augmenting path p in the residual network G_f
      augment flow f along p
  return f
```

Residual network

- Residual network G_f contains the residual capacities from G
- It may contain extra edges
  - To allow for decreasing previously-allocated flows
  - Reversed edges for previously-allocated flows
- Kind of similar to a flow network, except it allows for reversed edges
- Now, why there is no reversed edges in flow networks??

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E\ , \\ f(v,u) & \text{if } (v,u) \in E\ , \\ 0 & \text{otherwise}\ . \end{cases}$$

Augmenting paths

- Augmenting path p is a simple path from s to t in the residual network G_f

A path is simple if all the vertices on the path are distinct.

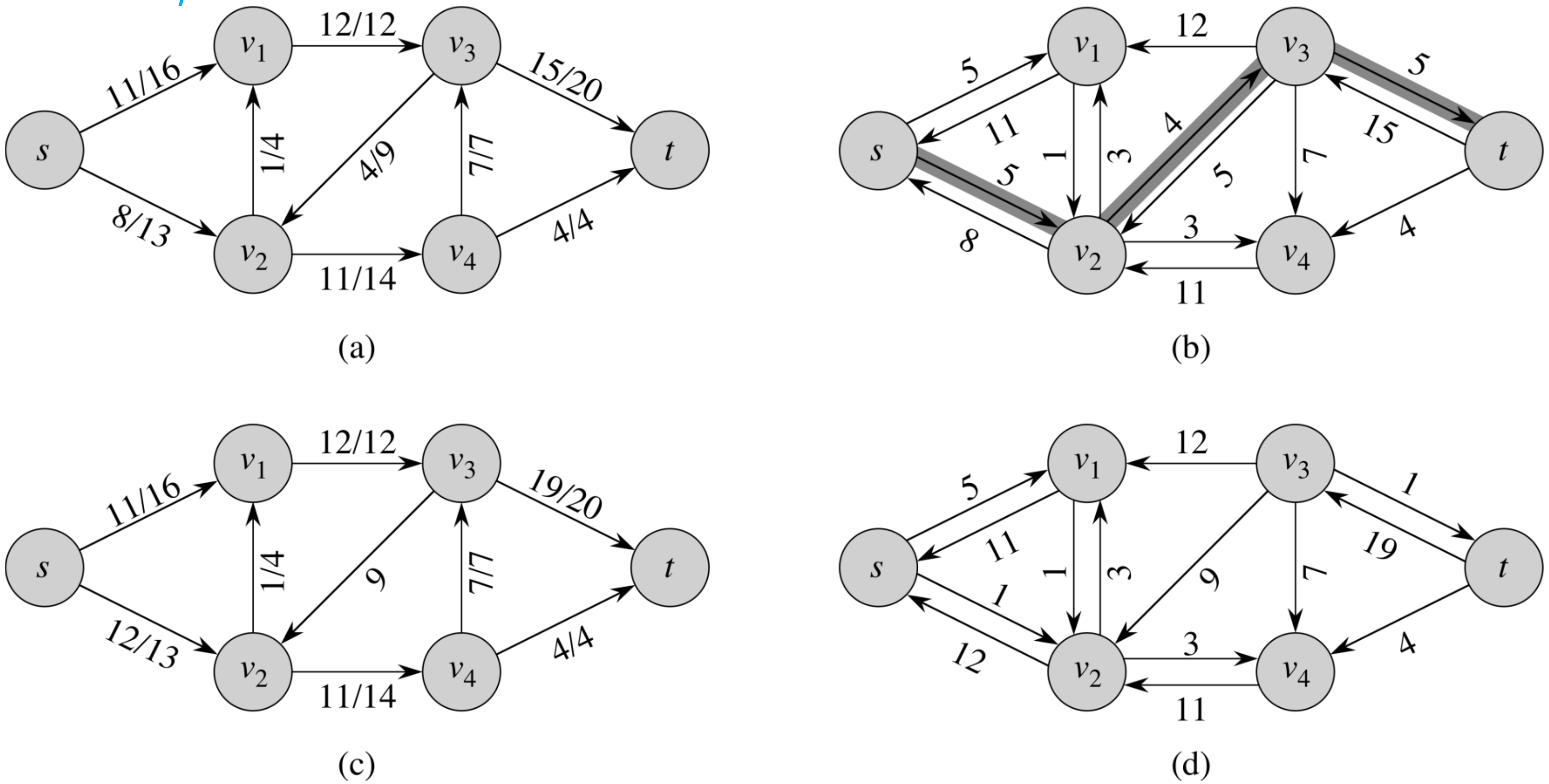
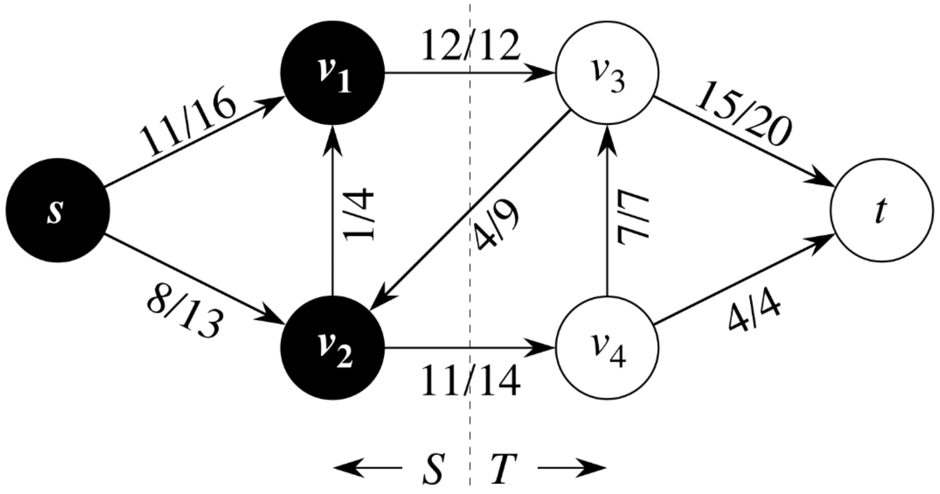


Figure 26.4 **(a)** The flow network G and flow f of Figure 26.1(b). **(b)** The residual network G_f with augmenting path p shaded; its residual capacity is $c_f(p) = c_f(v_2, v_3) = 4$. Edges with residual capacity equal to 0, such as (v_1, v_3), are not shown, a convention we follow in the remainder of this section. **(c)** The flow in G that results from augmenting along path p by its residual capacity 4. Edges carrying no flow, such as (v_3, v_2), are labeled only by their capacity, another convention we follow throughout. **(d)** The residual network induced by the flow in (c).

Cuts of flow networks

- How to know when the algorithm terminates??
- A cut (S, T) of flow network $G(V, E)$ is a partition of V into S and $T = V - S$
- ***Net flow*** $f(S, T)$ across the cut (S, T) is defined as

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$



Cuts of flow networks

- The ***capacity of the cut*** (S, T) is

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

- A ***minimum cut*** of a network is a cut whose capacity is minimum over all cuts of the network

Max-flow min-cut

Theorem 26.6 (Max-flow min-cut theorem)
If f is a flow in a flow network $G = (V, E)$ with source s and sink t, then the following conditions are equivalent:

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.

Basic Ford-Fulkerson algorithm

FORD-FULKERSON(G, s, t)

for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path p from s to t in the residual network G_f
    $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
    for each edge (u, v) in p
        if $(u, v) \in E$
            $(u, v).f = (u, v).f + c_f(p)$
        else $(v, u).f = (v, u).f - c_f(p)$

- Path-finding is performed using either BFS or DFS, thus $O(V + E) = O(E)$
- The while loop is executed $|f^*|$ (if every iteration just adds on unit value), where f^* is the maximum flow
- Overall complexity is $O(E|f^*|)$

FFA

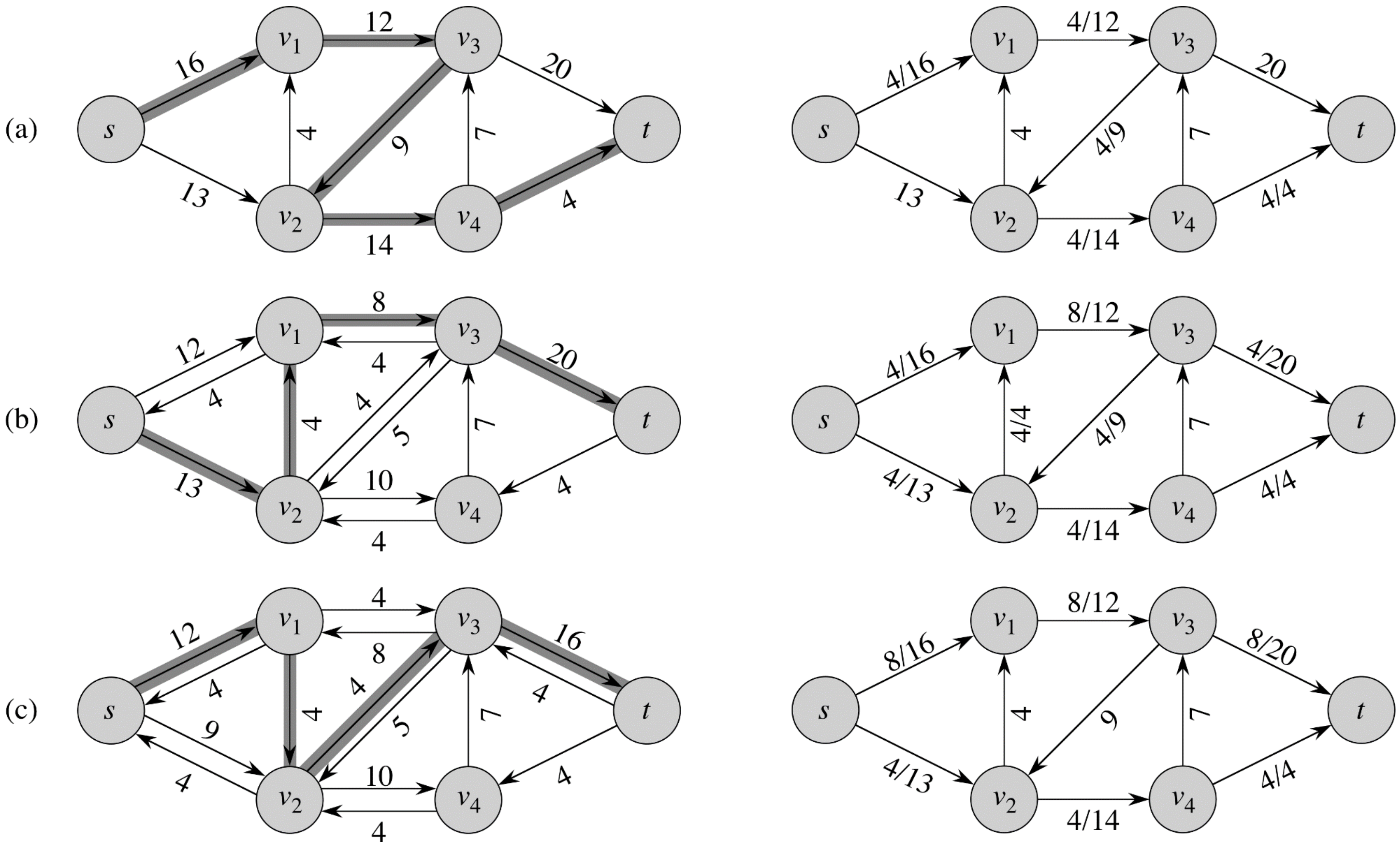
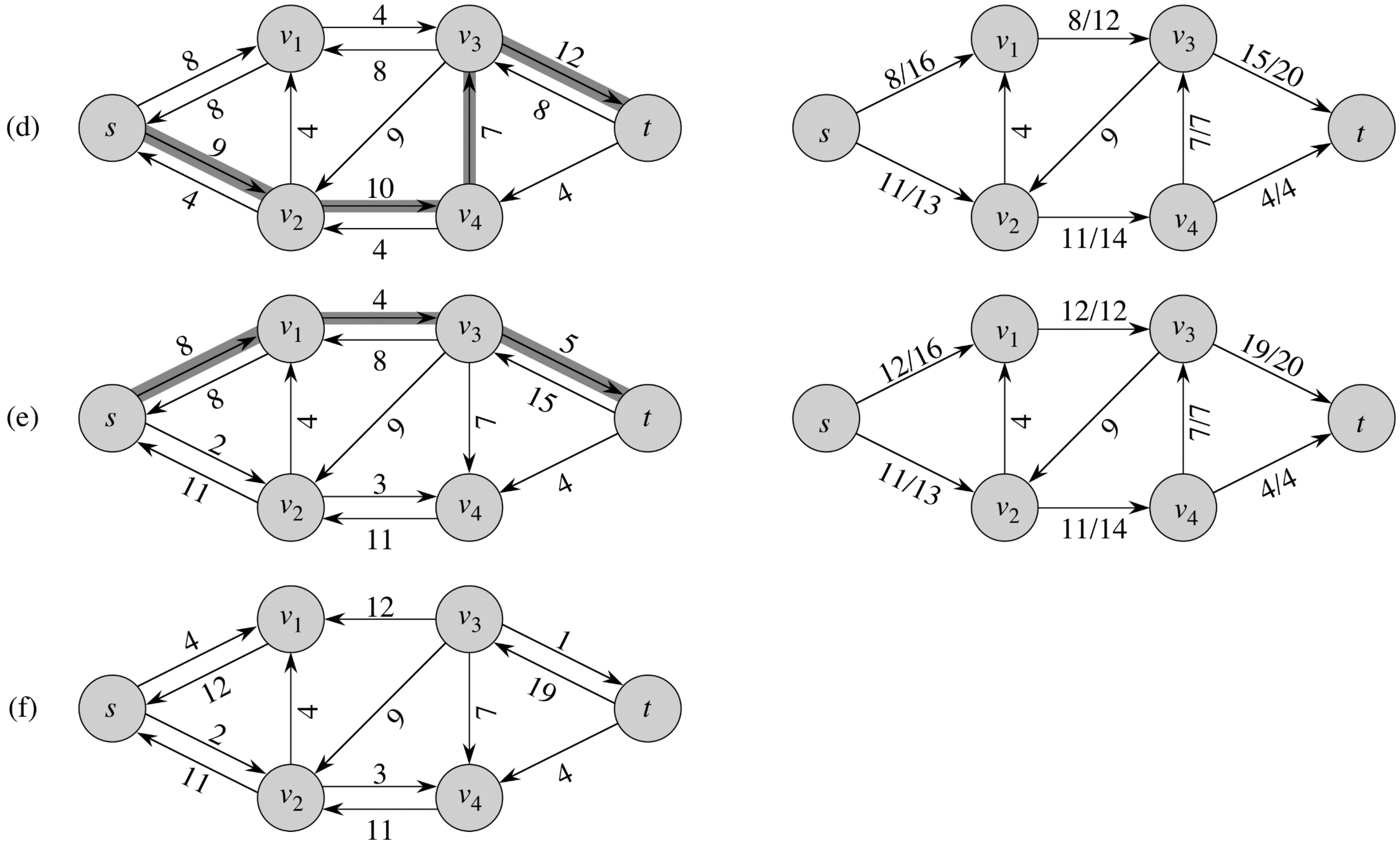


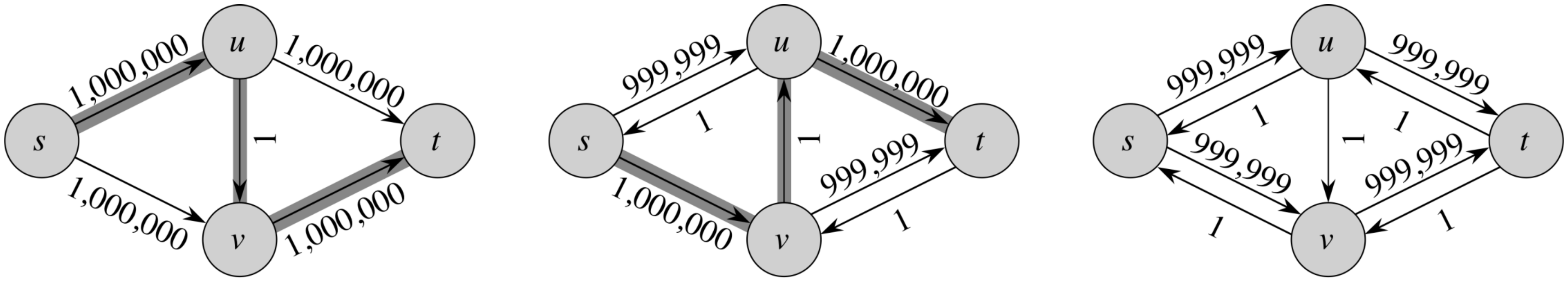
Figure 26.6 The execution of the basic Ford-Fulkerson algorithm. **(a)–(e)** Successive iterations of the **while** loop. The left side of each part shows the residual network G_f from line 3 with a shaded augmenting path p. The right side of each part shows the new flow f that results from augmenting f by f_p. The residual network in (a) is the input network G.

FFA



Worst case for FFA

- $|f^*|$ in this case is 2,000,000



(a) (b) (c)

Edmonds-Karp algorithm

- Uses BFS for augmenting path (shortest path)

- Total number of flow augmentations is $O(VE)$, thus overall complexity $O(VE^2)$

- How does it perform with the worst case for FFA?