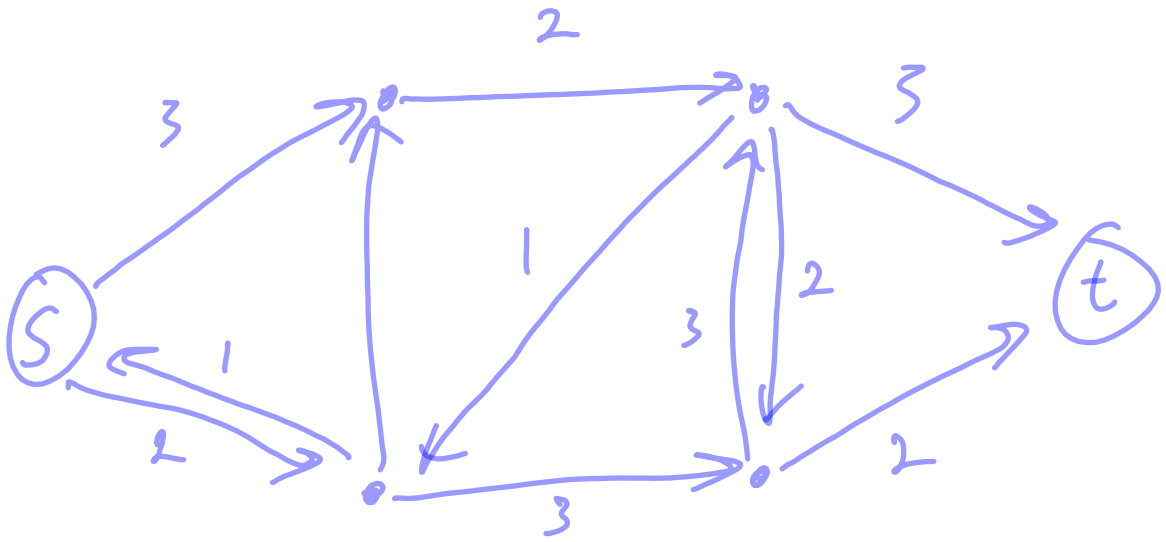
Over the next two lectures, we are going to discuss the **maximum-flow** problem, an important algorithmic problem which can be used to solve many interesting computational problems.

We are given as input a *flow network* which is a directed graph $G = (V, E)$ with two distinguished vertices: a *source* s and a *sink* t. Each edge $(u, v) \in E$ has a nonnegative *capacity* $c(u, v)$ (if there is not an edge from u to v then $c(u, v) = 0$).

Example:



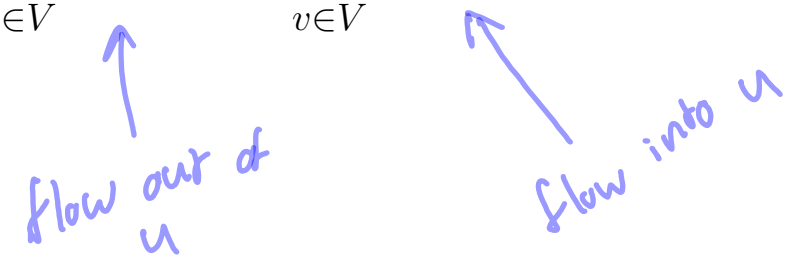
A *flow* in G is a function $f : V \times V \to \mathbb{R}$ satisfying the following:

- Capacity constraint: for all $u, v \in V$,

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

- Flow conservation: for all $u \in V \setminus \{s, t\}$,

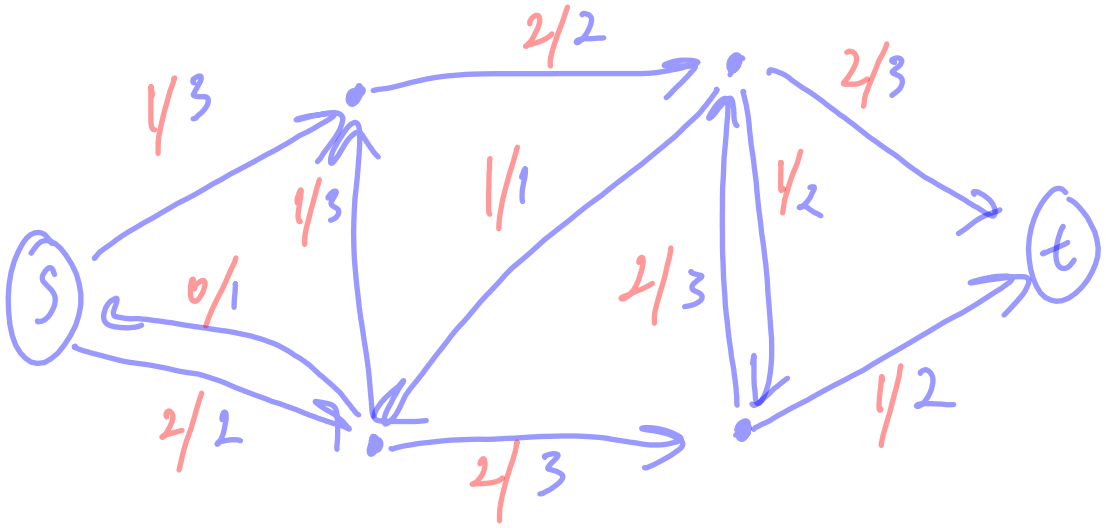
$$\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u) = 0$$



The value of the flow is denoted $|f|$ and is

$$\sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

Example:



$|f| = 3$

Intuitively, we can think of flows as how much "traffic" can move from s to t in the flow network. Given a flow network, we may want to know how much traffic the network can handle.

In the **maximum-flow** problem, we are given a flow network G as input and we want to compute a flow of maximum value on G.

There are two main approaches to computing a maximum flow:

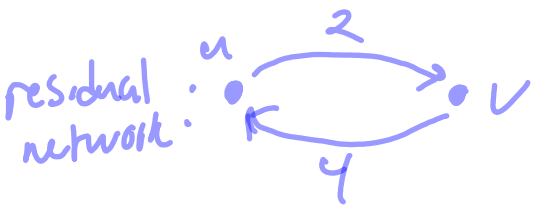
- Ford-Fulkerson method
- Push-relabel

We will consider the Ford-Fulkerson method in class. See Chapter 26.4 in the book if you wish to learn more about push-relabel.

The **Ford-Fulkerson** method utilizes *residual networks*. Intuitively, given a flow network G and a flow f, a residual network G_f consists of edges with capacities that represent how we can change the flow on edges of G.

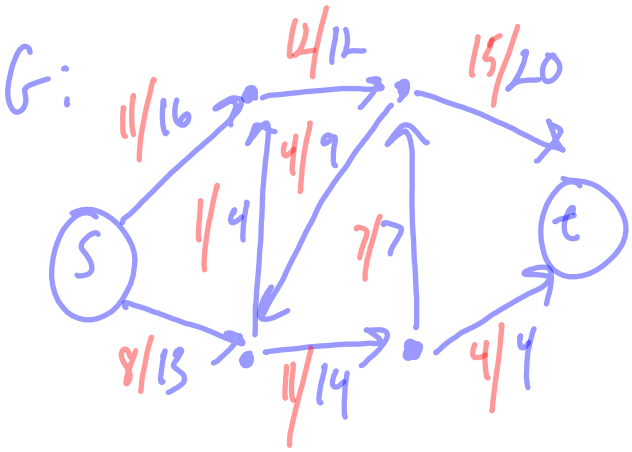
G_f has the same vertex set as G, although the edge set may differ. We define the capacity $c_f(u, v)$ as

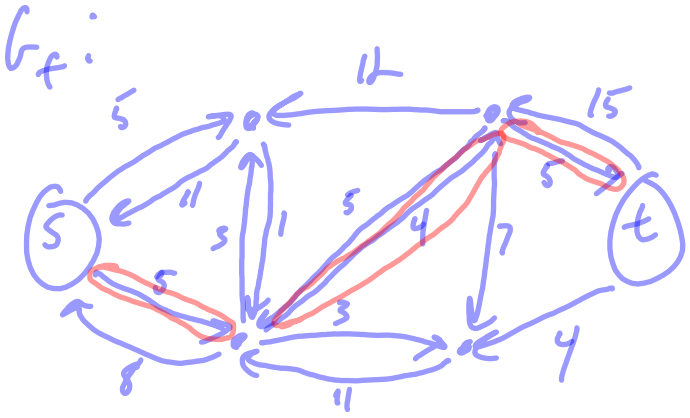
- $c(u, v) - f(u, v)$ if $(u, v) \in E$
- $f(v, u)$ if $(v, u) \in E$
- 0 otherwise

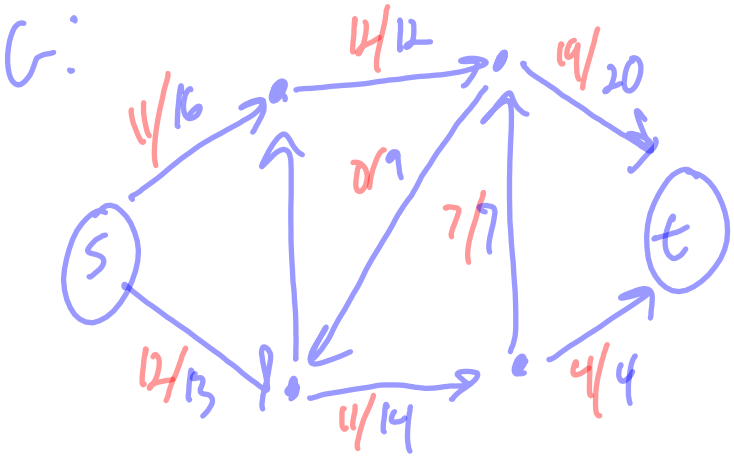


A path from s to t in G_f is called an **augmenting path**. The key property of these paths is that it represents a way to obtain a flow in G that is better than the flow f.

Example:


G:
12/12
15/20
11/16
4/9
1/4
7/7
s
t
8/13
11/14
4/4

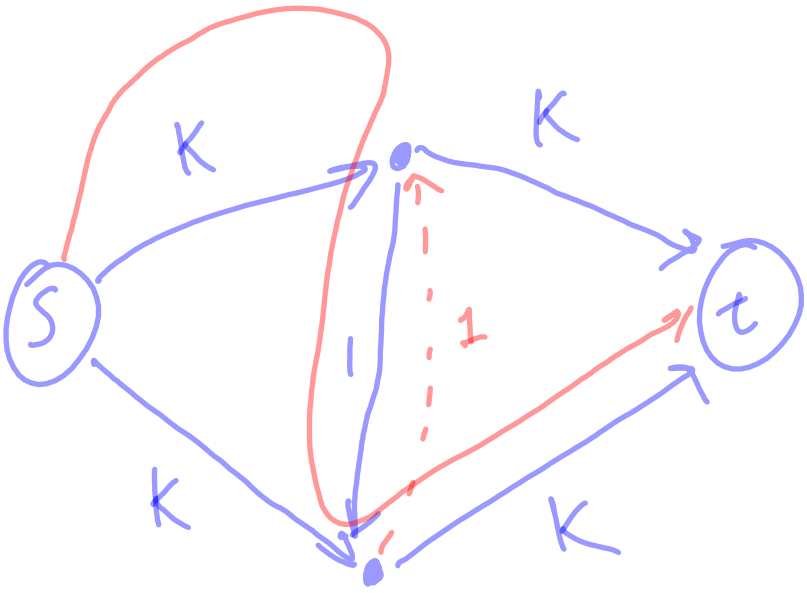

G_f:
12
5
15
11
3
1
5
4
7
5
s
t
5
3
8
11
4


G:
12/12
19/20
11/16
8/9
7/7
s
t
12/13
11/14
4/4

The Ford-Fulkerson method starts with an "empty flow", and then repeats the following as long as there is an augmenting path in the residual network:

- Find an augmenting path, and augment the flow based on this path.

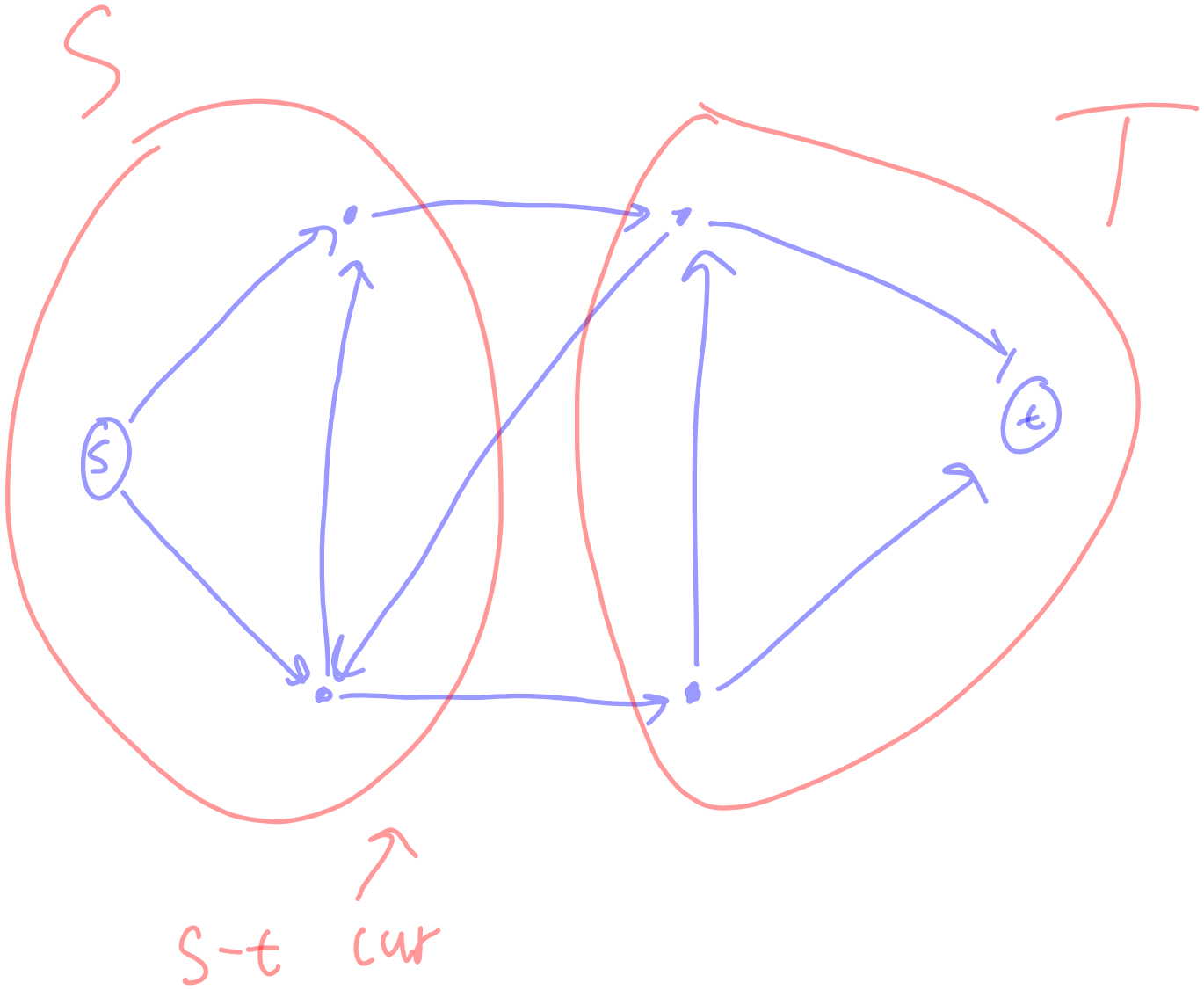
When there are no more augmenting paths in the residual network, then flow is a maximum flow (which we will show next class). However, if we do not carefully choose the residual paths, the running time of the algorithm can be arbitrarily bad:



It can even be shown that there are examples for which the algorithm may not terminate.

It can be seen that if we repeatedly choose an augmenting path with the smallest number of vertices possible, then the algorithm runs in polynomial time in $|V|$ and $|E|$. Such a path can be computed with breadth-first search.

When computing the augmenting paths in this manner, the algorithm is known as the **Edmonds-Karp** algorithm. The running time is $O(|V| \cdot |E|^2)$.



Capacity of a cut is the sum of the capacities of edges connecting a vertex in S to a vertex in T.

Capacity of a cut is an upper bound on the value of any feasible flow.