• **DISCLAIMER:** These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

• Main topics:
  – Maxflow mincut theorem
  – Residual graphs

• Office hour changes this week:
  – Ashwin: 12:30-2:30pm Tuesday Nov 1.
  – Pranathi: 2-3pm Monday Oct 31, (???)4-5pm Wednesday Nov 2(???).

• Ambiguities from last time:
  – Definition of cuts.
  – Goal of algorithms: finding flows in original graph $G$ / residual graph $G_f$, and cuts in $G / G_f$.

Last time we discussed the maximum flow and minimum cut problems. Our description was in terms of unweighted/uncapacitated edges, where the two problems can be described as:

1. Maxflow: find the maximum number of edge-disjoint (directed) paths from $s$ to $t$.
2. Mincut: remove the fewest number of edges so that there are no more (directed) paths from $s$ to $t$.

We now introduce capacities on the edges, which we denote with $c_e$. These can be viewed as having $c_e$ copies of the (unit) edge $e$. Then the flow on an edge can be denoted with $f_e$:

$$0 \leq f_e \leq c_e,$$

we will have use the shorthand of $f_e / c_e$ to denote an edge with capacity $c_e$ where we’re sending $f_e$ units of flow along it.

Flow conservation is still the same: amount of inflow equals to the amount of out flow:

$$\sum_{u \rightarrow v} f_{u \rightarrow v} = \sum_{w \rightarrow u} f_{w \rightarrow u}.$$
In residual graphs we have both types of edges: if we continue with the ‘multiple copies of edges’ view, then an edge \( e \) with \( f_e \) units of flow along it requires us to ‘turn around’ \( f_e \) of these units.

So the formal definition of a residual graph \( G_f \) is for each edge \( e = u \rightarrow v \), we have

1. Edge of capacity \( c_e - f_e \) in the forward, \( u \rightarrow v \), direction.
2. Edge of capacity \( f_e \) in the reverse, \( v \rightarrow u \) direction.

Maxflow algorithms can be described as:

\[
\text{While there is an } s \rightarrow t \text{ path on the residual graph, find such a path, and push flow on it until some edge’s capacity is saturated.}
\]

Such a path in \( G_f \) is called a \textbf{augmenting path}. When such an algorithm terminates, we justify its optimality by showing that there exists a cut whose capacity equals to the value of the flow aka. there exists some \( S \) such that

\[
\sum_{e = u \rightarrow v, u \in S, v \notin S} c_{u \rightarrow v} = \sum_{s \rightarrow v} f_{s \rightarrow v}.
\]

We find such a cut by letting \( S \) be all the edges reachable from \( S \) in \( G_f \). There are two kinds of edges in \( G \) on the peripheral/boundary of \( S \):

1. (outgoing) \( u \rightarrow v \) with \( u \in S \) and \( v \notin S \), for such an edge we must have \( f_{u \rightarrow v} = c_{u \rightarrow v} \).

2. (incoming) \( w \rightarrow u \) with \( w \notin S \) and \( u \in S \). For such an edge we can’t have flow on it, because otherwise \( u \rightarrow w \) must be in the residual graph. So \( f_{w \rightarrow u} = 0 \).

Then the above statement about capacity of cut equaling to flow value comes from summing over the residuals of all vertices \( u \in S \).

This leads to algorithms that find maximum flow and minimum cuts that terminate. Their running time is \( O(m|F|) \) where \( |F| \) is the value of the maximum flow.

It can indeed take this long. Consider the graph

\[
s \rightarrow a, s \rightarrow b, a \rightarrow t, b \rightarrow t, a \rightarrow b
\]

where every edge has capacity \( x \) except \( a \rightarrow b \), which has capacity 1. We can keep on finding augmenting paths involving the edge \( a \rightarrow b \), requiring \( 2x \) steps to finish. Next time we will see how to reduce this iteration count down to \( O(m \log |F|) \).