References for Today’s Lecture
References for Today’s Lecture

• Required reading
  – Sections 21.1–21.2

• References
  – AMO Chapter 6
  – CLRS Sections 26.1–26.2
Labeling Algorithm (Ford and Fulkerson (1956))

- Fill in details of generic augmenting path algorithm
  - how to identify augmenting path (or show no path exists)
  - whether algorithm terminates in finite number of iterations
  - whether final flow value is maximal

- The labeling algorithm is the most straightforward variant.

- The cost to find the augmenting path is low, but the number of augmentations can be high.

- Depth-first search is a special case.
Identifying an Augmenting Path

- Use search technique to find a directed path in $G(x)$ from $s$ to $t$
  - At any step, partition nodes into labeled and unlabeled
  - Iteratively select a labeled node and scan its arc adjacency list in $G(x)$ to reach and label additional nodes
  - When sink becomes labeled, augment flow, erase labels and repeat
  - Terminate when all labeled nodes have been scanned and sink remains unlabeled
Labeling Algorithm

Input: A network $G = (N, A)$ and a vector of capacities $u \in \mathbb{Z}^A$
Output: $x$ represents the maximum flow from node $s$ to node $t$
  label node $t$
  while $t$ is labeled do
    unlabel all nodes
    \[ \text{pred}(j) \leftarrow 0 \ \forall j \in N \]
  label node $s$ and set $LIST \leftarrow \{s\}$
  while $LIST \neq \emptyset$ and $t$ is unlabeled do
    remove a node $i$ from $LIST$
    for each arc $(i, j)$ in the residual network do
      if node $j$ is unlabeled then
        $\text{pred}(j) \leftarrow i$
        label node $j$ and add $j$ to $LIST$
      end if
    end for
  end while
  if $t$ is labeled then
    augment
  end if
end while
Example of Labeling Algorithm
Correctness of Labeling Algorithm

Claim 1. When the algorithm terminates, the current flow $x$ is a maximum flow.

Proof:
Note that in each iteration of the while loop, the algorithm either (i) performs an augmentation or (ii) terminates. Therefore, we need to show that the current flow $x$ is a maximum flow when (ii) occurs.
Max-Flow Min-Cut Theorem

Theorem 1. [6.3] The maximum value of the flow from a source node $s$ to a sink node $t$ in a capacitated network equals the minimum capacity among all $s - t$ cuts.

Proof: Follows from the Correctness of the Labeling Algorithm.
Augmenting Path Theorem

**Theorem 2. [6.4]** A flow $x^*$ is a maximum flow if and only if the residual network $G(x^*)$ contains no augmenting path.

**Proof:**
Integrality Theorem

Theorem 3. [6.5] *If all arc capacities are integer, the maximum flow problem has an integer maximum flow.*

Proof:
Complexity of the Labeling Algorithm


Proof:
At each iteration of the while loop, how much work is done?

How many augmentations are done?
Flows with Lower Bounds

• Suppose that we add non-negative lower bounds on the arc flows to the maximum flow problem:

\[ l_{ij} \leq x_{ij} \leq u_{ij}, \forall (i, j) \in A. \]

• Zero flow is no longer always a feasible solution.

• Objective: determine if the problem is feasible and, if so, establish a maximum flow.

• Approach: first, determine a feasible flow and then determine a maximum flow.
Determining a Feasible Flow

- Transform max flow into circulation (max flow has feasible flow if and only if circulation has feasible flow)

- Identify an infeasible arc \((p, q)\) (one that violates lower bound).

- Start with the zero flow and then augment flow around cycles with \((p, q)\) as a forward arc.

- The algorithm terminates with either a feasible circulation or a proof that no such circulation exists.

**Theorem 5. [6.11]** A circulation problem with non-negative lower bounds on the arc flows is feasible if and only if, for every set \(S\) of nodes,

\[
\sum_{(i,j) \in (\bar{S}, S)} l_{ij} \leq \sum_{(i,j) \in (S, \bar{S})} u_{ij}.
\]
Determining a Maximum Flow

• Suppose that we have a feasible flow $x$ in the network.

• To obtain a maximum flow, we can modify any maximum flow algorithm to accommodate non-negative lower bounds.

• Define the residual capacity of an arc $(i, j)$ to be

$$r_{ij} = (u_{ij} - x_{ij}) + (x_{ji} - l_{ji})$$

• From optimal residual capacities, we can construct a maximum flow.

• Theorem 6.10 is a generalized version of the Max-Flow Min-Cut Theorem for networks with both lower bounds and upper bounds on the arc flows.
Application: Network Connectivity

- Two directed paths from $s$ to $t$ are *arc disjoint* if they do not have any arc in common.

- Given a directed network $G = (N, A)$ and two specified nodes $s$ and $t$:
  - What is the maximum number of arc-disjoint directed paths from node $s$ to node $t$?
  - What is the minimum number of arcs that we should remove from the network so that it contains no directed paths from $s$ to $t$?

**Theorem 6.** [6.7] *The maximum number of arc-disjoint paths from node $s$ to node $t$ equals the minimum number of arcs whose removal from the network disconnects all paths from $s$ to $t.*
Application: Matchings and Covers in a Bipartite Network

Given a directed bipartite network \( G = (N, A) \), where \( N = N_1 \cup N_2 \):

- A subset \( A' \subseteq A \) is a matching if no two arcs in \( A' \) are incident to the same node.
- A subset \( N' \subseteq N \) is a node cover if every arc in \( A \) is incident to one of the nodes in \( N' \).

**Theorem 7. [6.9]** In a bipartite network \( G = (N_1 \cup N_2, A) \), the maximum cardinality of any matching equals the minimum cardinality of any node cover of \( G \).