Reminders

- Homework 3 due Monday
  - 7.2, 7.5, 7.9, 7.15 (7.9 and 7.15 b should follow full algorithm write-up guidelines)

- Midterm on Friday

- Project Checkpoint 2 Monday – pass all public tests (2 are up, 1 will be added)

Augmenting Path Algorithm

Augment(f, c, P) {  
  b ← bottleneck(P)  
  foreach e ∈ P {  
    if (e ∈ E) f(e) ← f(e) + b  
    else f(e^R) ← f(e^R) - b  
  }  
  return f
}

Ford-Fulkerson(G, s, t, c) {  
  foreach e ∈ E  
    f(e) ← 0  
  G_f ← residual graph  
  while (there exists augmenting path P) {  
    f ← Augment(f, c, P)  
    update G_f  
  }  
  return f
}

7.3 Choosing Good Augmenting Paths

Use care when selecting augmenting paths.
- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.
- If capacities are irrational, algorithm not guaranteed to terminate!

Goal: choose augmenting paths so that:
- Can find augmenting paths efficiently.
- Few iterations.

Choose augmenting paths with: [Edmonds-Karp 1972, Dinitz 1970]
- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.
Capacity Scaling

Intuition. Choosing path with highest bottleneck capacity increases flow by max possible amount.
- Don’t worry about finding exact highest bottleneck path.
- Maintain scaling parameter $\Delta$.
- Let $G_f(\Delta)$ be the subgraph of the residual graph consisting of only arcs with capacity at least $\Delta$.

```
Capacity Scaling: Correctness
Assumption. All edge capacities are integers between 1 and $C$.
Integrality invariant. All flow and residual capacity values are integral.
Correctness. If the algorithm terminates, then $f$ is a max flow.
Pf.
- By integrality invariant, when $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
- Upon termination of $\Delta = 1$ phase, there are no augmenting paths.
```

```
Capacity Scaling: Running Time
Lemma 1. The outer while loop (while $\Delta \geq 1$) repeats $1 + \lceil \log_2 C \rceil$ times.
Pf. Initially $C < 2 \Delta < 2C$.
- $\Delta$ decreases by a factor of 2 each iteration.

Lemma 2. Let $f$ be the flow at the end of a $\Delta$-scaling phase. Then value of the maximum flow is at most $v(f) + m \Delta$.

Lemma 3. There are at most $2m$ augmentations per scaling phase.
- Let $f$ be the flow at the end of the previous scaling phase.
- Lemma 2 $\Rightarrow v(f_{opt}) \leq v(f) + m (2\Delta)$.
- Each augmentation in a $\Delta$-phase increases $v(f)$ by at least $\Delta$.

Theorem. The scaling max-flow algorithm finds a max flow in $O(m \log C)$ augmentations. Each augmentation can take time $O(m \log C)$ time.
```

```
```

7.5 Bipartite Matching
**Matching**

- **Input:** undirected graph \( G = (V, E) \).
- **Matching:** \( M \subseteq E \) is a matching if each node appears in at most one edge in \( M \).
- **Max matching:** find a max cardinality matching.

**Bipartite Matching**

- **Input:** undirected bipartite graph \( G = (L \cup R, E) \).
- **Matching:** \( M \subseteq E \) is a matching if each node appears in at most one edge in \( M \).
- **Max matching:** find a max cardinality matching.

---

**Theorem.** Max cardinality matching in \( G \) = value of max flow in \( G' \).

**Pf.**

- Given max matching \( M \) of cardinality \( k \).
- Consider flow \( f \) that sends 1 unit along each of \( k \) paths.
- \( f \) is a flow, and has cardinality \( k \).

---

**Theorem.** Max cardinality matching in \( G \) = value of max flow in \( G' \).

**Pf.**

- Let \( f \) be a max flow in \( G' \) of value \( k \).
- **Integrity theorem:** \( k \) is integral and can assume \( f \) is \( 0 \) or \( 1 \).
- Consider \( M := \{ e \in E \mid f(e) = 1 \} \).
- Each node in \( L \) and \( R \) participates in at most one edge in \( M \).
- \( |M| = k \). Consider cut \( (L \cup \{ s \}, R \cup \{ t \}) \).
Perfect Matching

**Def.** A matching $M \subseteq E$ is **perfect** if each node appears in exactly one edge in $M$.

**Q.** When does a bipartite graph have a perfect matching?

**Structure of bipartite graphs with perfect matchings.**
- Clearly we must have $|L| = |R|$.
- What other conditions are necessary?
- What conditions are sufficient?

**Notation.** Let $S$ be a subset of nodes, and let $N(S)$ be the set of nodes adjacent to nodes in $S$ (or the neighbors of $S$).

**Observation.** If a bipartite graph $G = (L \cup R, E)$ has a perfect matching, then $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

**Pf.** Each node in $S$ has to be matched to a different node in $N(S)$.

---

Marriage Theorem

**Marriage Theorem.** (Frobenius 1917, Hall 1935) Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then, $G$ has a perfect matching if $|N(S)| = |S|$ for all subsets $S \subseteq L$.

**Pf.** This was the previous observation.

---

**Proof of Marriage Theorem**

**Pf.** Suppose $G$ does not have a perfect matching. (Contrapositive)
- Formulate as a max flow problem and let $(A, B)$ be min cut in $G'$.
- By max-flow min-cut, $\text{cap}(A, B) = |L|$.
- Define $L_A = L \cap A$, $L_B = L \cap B$, $R_A = R \cap A$.
- $\text{cap}(A, B) = |L_A| + |R_A|$.
- Since min cut can't use $\infty$ edges: $|N(L_A)| \subseteq R_A - \{s\}$.
- $|N(L_A)| = |R_A| = \text{cap}(A, B) - |L_B| - |L_A| - |L_A| = |L_A|$.
- Choose $S = L_A$. ▪

---

Bipartite Matching: Running Time

**Which max flow algorithm to use for bipartite matching?**
- Generic augmenting path: $O(m \cdot \text{val}(f^*)) = O(mn)$.
- Capacity scaling: $O(m^2 \log C) = O(m^2)$.
- Shortest augmenting path: $O(mn^{1/2})$.

**Non-bipartite matching.**
- Structure of non-bipartite graphs is more complicated, but well-understood: (Tutte-Berge, Edmonds-Galai)
- Blossom algorithm: $O(n^5)$ (Edmonds 1965)
- Best known: $O(m n^{1/2})$ [Micali-Vazirani 1980]