In this lecture we discussed examples of approximation algorithms including

- Greedy and matching vertex cover algorithms for minimum vertex cover
- Greedy approximation for list scheduling

Handouts (posted on webpage): none
Reading: Jeff Erickson’s notes on approximation algorithms

1 Minimum Vertex Cover

Given an undirected graph \( G = (V, E) \), find the smallest set of vertices \( S \subset V \) such that all edges in \( G \) have at least 1 endpoint in \( S \).

**Algorithm 1 Greedy**

```plaintext
repeat
    add vertex \( v \in G \) with highest degree to \( S \)
    remove \( v \) and all incident edges of \( v \) from \( G \)
until no vertices left in \( G \)
```

Guarantee: size of greedy vertex cover \( \leq \log(n) \times \text{OPTVC} \) for all graphs

**Algorithm 2 Matching Vertex Cover**

```plaintext
S = \emptyset 
repeat
    pick edge \((u, v) \in G\) (any edge)
    remove \((u, v)\) from \( G \) and all edges adjacent to \( u \) or \( v \)
    add \( u \) and \( v \) to \( S \)
until no edges left in \( G \)
```

Example: \( S = \{a, b, c, d, e, f\} \)
Claim: MVC is a 2-approximation for vertex cover
Proof: we don’t know how big $\text{OPTVC}(G)$ is but we can lower bound its size

- $|\text{OPTVC}(G)| \geq$ number of edges picked by MVC (because edges selected by MVC form a
  matching and no vertex covers more than 1 edge in a matching)
- number of vertices picked by MVC is $2 \times$ number of edges picked

$\Rightarrow |\text{MVC}(G)| \leq 2 \times |\text{OPTVC}(G)|$

2 List Scheduling (Graham 1966)

Given $n$ jobs and $m$ (identical) machines, where job $i$ must execute uninterruptedly for $p_i$ time units and each machine can work on 1 job at a time, find a schedule of jobs that minimizes completion time.

Greedy algorithm: whenever a job becomes idle, assign the next job to it.

Example: $p_i$’s = 5, 7, 17, 10, 9, 30
   
   Greedy solution:

   ![Greedy Solution Diagram]

   Optimal solution:

   ![Optimal Solution Diagram]

   One benefit of this algorithm is that it can be an online algorithm, whereas an algorithm that requires sorting the jobs can only be performed offline.
Claim: \(|\text{GREEDY}(p_1, p_2, \ldots, p_n)\| \leq (2 - \frac{1}{m})|\text{OPT}(p_1, \ldots, p_n)|\)

Proof:

- \(\text{OPT} \geq p_i, \forall i\)
- \(\text{OPT} \geq \frac{\sum p_i}{m}\)

Let \(k\) be the last job to finish and \(s_k\) be the start time of job \(k\)

\[
\begin{align*}
p_k & \leq \text{OPT} \\
s_k & \leq \frac{\sum_{i \neq k} p_i}{m} \\
& \leq \frac{1}{m} \sum_{i \neq k} p_i + p_k \\
& = \frac{1}{m} \sum_i p_i + (1 - \frac{1}{m})p_k \\
& \leq \text{OPT} + (1 - \frac{1}{m}) \times \text{OPT} \\
& = (2 - \frac{1}{m}) \times \text{OPT}
\end{align*}
\]