Last Time

• Shortest Path problems
  – Relaxation
  – Dijkstra’s Algorithm
  – Acyclic Shortest Path
  – Bellman-Ford Algorithm
Greedy Algorithms

• **Greedy algorithms** are a class of algorithms that always take the best available solution at each step, regardless of future consequences.
  – Prim’s Algorithm (Lazy and Eager)
  – Dijkstra’s Algorithm
  – Huffman coding
  – Selection sort
Greedy Algorithms

• Need to show two things:
  – **Correctness:** The algorithm produces an acceptable output (nothing new introduced, no data manipulated).
    • Usually trivial.
  – **Optimality:** The algorithm produces the best solution (no possible “better” output).
    • Somewhat more difficult.
Proof of Greedy Correctness

- **Theorem:** Algorithm A is correct/optimal.
- **Proof:** (by contradiction)
  - There exists some input $i$ on which A doesn’t produce acceptable output.
  - Consider $A(i)$, the output of algorithm A on input $i$, and $OPT(i)$, the optimal/correct output for $i$.
  - Produce $OPT'(i)$ such that:
    - $OPT'(i)$ agrees with $A(i)$ for one more step.
    - $OPT'(i)$ is still an optimal/correct output.
  - $OPT'$ is definitionally correct for one further step, so create an $OPT''$ that meets the same goals as $OPT'$.
  - Eventually, you have a chain $OPT \rightarrow OPT' \rightarrow OPT'' \rightarrow OPT''' \rightarrow \ldots \rightarrow A$. (Maintaining correctness, you get more like greedy with every new step.)
  - Now, look at $OPT$ globally as “the correct solution that agrees with A for the most number of steps” (assumes that A is incorrect).
  - But, $OPT'$ agrees with A for one more step than OPT, the “best” optimal solution. A contradiction! ■

- This is called an **Exchange Argument**.
Minimum Average Waiting Time

- **Input:** Jobs with processing times $P_1, P_2, P_3, \ldots, P_n$.
- **Output:** Order to execute the jobs to minimize average waiting time.

**Example:**
- Input: {7, 2, 4, 11}
  - Possible Output: (4, 11, 7, 2)
    - Waiting times: 4, 15, 22, 24; avg wait = 16.25
  - Optimal Output: (2, 4, 7, 11)
    - Waiting times: 2, 6, 13, 24; avg wait = 11.25
Minimum Average Waiting Time

• **Theorem:** Shortest Job First is correct.

• **Proof:** Assume to reach a contradiction that there exists an input $i$ on which shortest job first is not correct.

• Let us consider:
  – $\text{SJF}(i) =$ output of Shortest Job First on input $i$
  – $\text{OPT}(i) =$ an optimal output
  – $\text{OPT}'(i) =$ a new optimal output that agrees with $\text{SJF}(i)$ for one further step
  – $k =$ the first greedy choice that $\text{OPT}(i)$ disagreed with
Minimum Average Waiting Time

- SJF(i)
- OPT(i)
- OPT'(i)_swap
- OPT'(i)_shift
Minimum Average Waiting Time

- **Correctness**: OPT’(i) is still a permutation of the input i.

- **Optimality**: Need to argue that the average waiting time hasn’t increased.
  - Jobs before the swap/shift won’t have different waiting times.
  - Jobs after the swap/shift won’t have different waiting times.
  - Only need to consider jobs between A and B.
Minimum Average Waiting Time

- \( \text{OPT}(i) \)
  - B
  - A
  - X
  - A
  - B

- \( \Delta X = (\text{size}A - \text{size}B) \leq 0 \)
  - \( \text{size}A \leq \text{size}B \) by definition of Shortest Job First

- \( \Delta A = -X - B \)
- \( \Delta B = +X + A \)
- \( \Delta A + \Delta B = -X + X + A - B = A - B \leq 0. \)
- Therefore, \( A \leq B \). So, \( \text{OPT}'(i) \) is still correct.
Minimum Average Waiting Time

• So, we showed that:
  – OPT’(i) is both correct and optimal.
  – OPT’(i) matched A(i) for at least one more step than OPT(i) did.
  – We defined OPT(i) to be the solution that matched A(i) the longest.
  – OPT(i) != OPT’(i)

• Contradiction! □
Minimum Spanning Tree Problem

• Theorem: Kruskal’s Algorithm is correct.
• Proof: Assume to reach a contradiction that there exists an input $i$ on which Kruskal’s Algorithm is not correct.
• Let us consider:
  – $\text{OPT}(i) =$ the optimal solution that agrees with Kruskal the longest
  – $\text{KRUSKAL}(i) =$ Kruskal’s output for input $i$
  – $e_1, e_2, \ldots, e_n =$ ordered edges such that $e_1 \leq e_2 \leq \ldots \leq e_n$
  – $k =$ first edge that $\text{KRUSKAL}(i)$ and $\text{OPT}(i)$ disagree on
  – $l =$ edge removed from $\text{OPT}(i)$ to remove a cycle
  – $\text{OPT}'(i) = \text{OPT}(i) + e_k - e_l$
Minimum Spanning Tree Problem

• Why do KRUSKAL$(i)$ and OPT$(i)$ disagree at $k$?

1. Kruskal skipped an edge that OPT took.
   • Can’t happen, because Kruskal skipped it for a reason.
   KRUSKAL$(i)$ and OPT$(i)$ chose the same edges up to point $k$, so if Kruskal skipped an edge, it’s because there already exists a path between the two endpoints and a cycle would occur.

2. Kruskal picked an edge that OPT skipped.
   • Sure, this can definitely happen.

• Add $e_k$ and remove $e_l$ from OPT$(i)$ to form OPT’$(i)$.
Minimum Spanning Tree Problem

• Cases to consider:

  1. There exists an $e_l$ on the cycle formed by $\text{OPT}(i) + e_k$ such that $|e_l| \geq |e_k|$.  
     • This is a contradiction, because $\text{OPT}'(i)$ agrees with $\text{KRUSKAL}(i)$ for one more step, breaking our definition of $\text{OPT}(i)$.

  2. All $e_l$ on the cycle formed by the addition of $e_k$ satisfy $|e_k| < |e_l|$.  
     • This is a contraction, because Kruskal would definitionally have chosen $e_k$.

• Both cases result in a contradiction. ■
Any Questions?