CS 7800: Advanced Algorithms

Class 3: Greedy Algorithms II

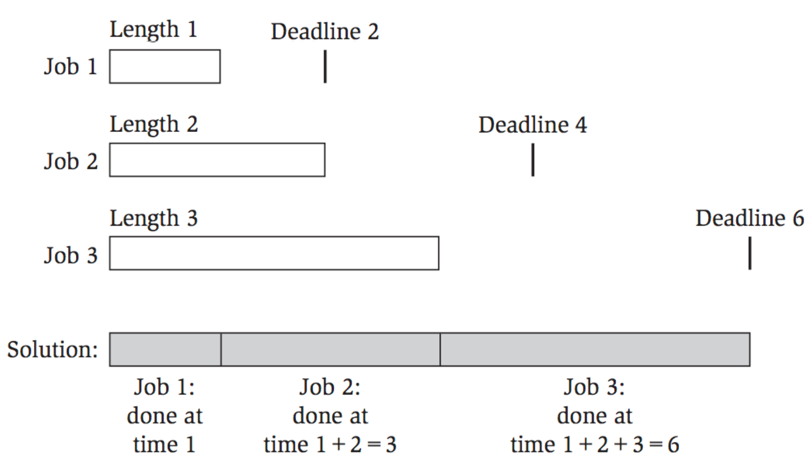
- Finish Minimum Lateness Scheduling
- Minimum Spanning Tree

Jonathan Ullman

September 9, 2025

Minimum Lateness Scheduling

- Input: n jobs with length t_i and deadline d_i
  - Simplifying assumption: all deadlines are distinct
- Output: a minimum-lateness schedule for the jobs
  - Job i starts at s_i finishes f_i, no jobs overlap
  - The lateness of job i is $\max\{f_i - d_i, 0\}$
  - The lateness of a schedule is $\max_i\{\max\{f_i - d_i, 0\}\}$



Possible Greedy Rules

Greedy Algorithm: Earliest Deadline First

- Sort jobs so that $d_1 \leq d_2 \leq \cdots \leq d_n$
- For i = 1, … , n:
  - Schedule job i right after job $i - 1$ finishes

Exchange Argument

Exchange Argument

Exchange Argument

Exchange Argument

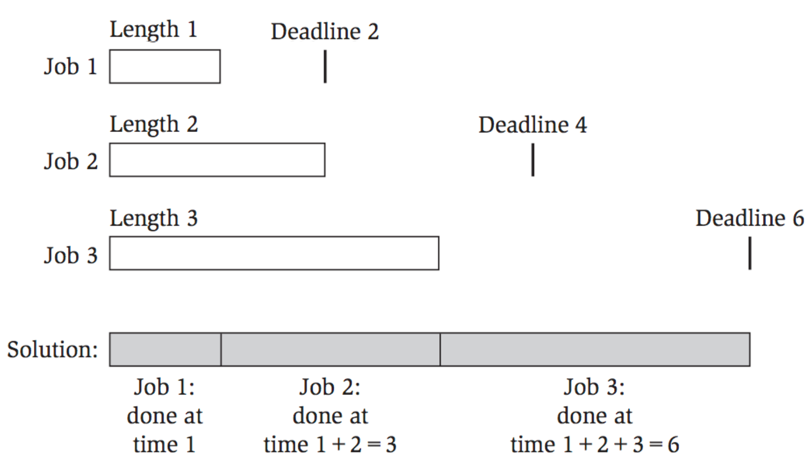
Exchange Argument

Exchange Argument

- **Putting the steps together (a thought experiment)**
  - (1) The greedy schedule G has no inversions
  - (2) While O is **not** equal to G
    - (2a) O has at least one inversion
    - (2b) O has a pair of consecutive jobs i, j that are inverted
    - (2c) Swap the order of i, j to fix the inversion
  - (3) Now O is equal to G but its lateness didn't increase, so O started at least as late as G

Minimum-Lateness Scheduling Recap

- There is an $O(n \log n)$ greedy algorithm for the minimum-lateness scheduling problem
  - Sort by earliest deadline and schedule jobs consecutively with no gaps
  - Analyze via an exchange argument



Network Design

- **Build a cheap, connected graph**

- We are given
  - A set of nodes $V = \{v_1, \ldots, v_n\}$ and edges $E \subseteq V \times V$
  - a weight function on the edges w_e
- Want to build a network to connect the nodes
  - Every v_i, v_j must be connected
  - Must be as cheap as possible

- **Many variants of network design**

Minimum Spanning Trees (MST)

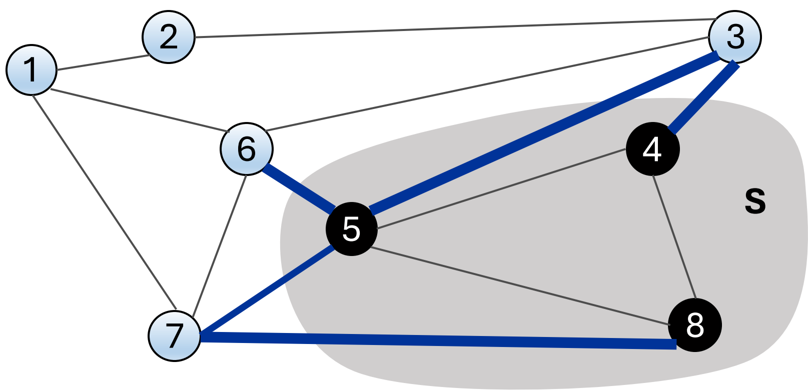
- **Input:** a weighted graph $G = (V, E, \{w_e\})$
  - Undirected, connected, weights may be negative
  - All edge weights are distinct

- **Output:** a spanning tree T of minimum cost
  - A spanning tree of G is a subset of $T \subseteq E$ of the edges such that (V, T) forms a tree
  - Cost of a spanning tree T is the sum of the edge weights

Minimum Spanning Trees (MST)

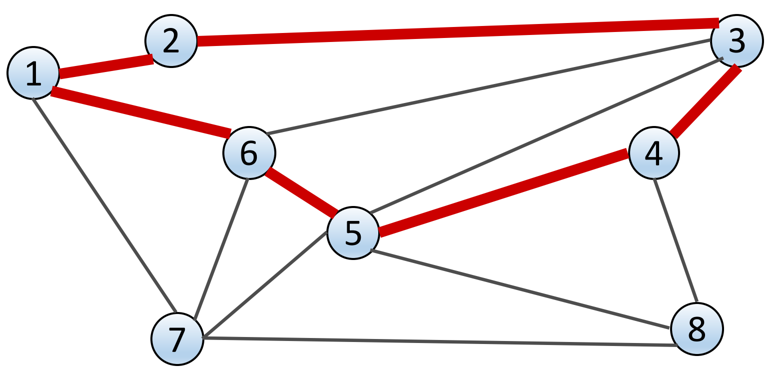
Cuts and Cycles

Cut: a subset of nodes S **Cutset:** edges w/ 1 endpoint in cut



Cut S = {4, 5, 8}
Cutset of S = (5,6), (5,7), (3,4), (3,5), (7,8)

Cycle: a set of edges $(v_1, v_2), (v_2, v_3), \ldots, (v_k, v_1)$



Cycle C = (1,2),(2,3),(3,4),(4,5),(5,6),(6,1)

Cut Property

The “Only” MST Algorithm

Borůvka's Algorithm

Borůvka's Algorithm

Borůvka's Algorithm

Kruskal’s Algorithm

MST Recap

- There is an $O(m \log n)$ greedy algorithm for finding a minimum spanning tree
  - There are actually several such algorithms
  - Bespoke analysis using structural properties of MSTs