CS180 Practice Midterm 2 Review

Practice Midterm 2: Breakdown

1: Majority Problem - as seen in lecture

2: Interval Related

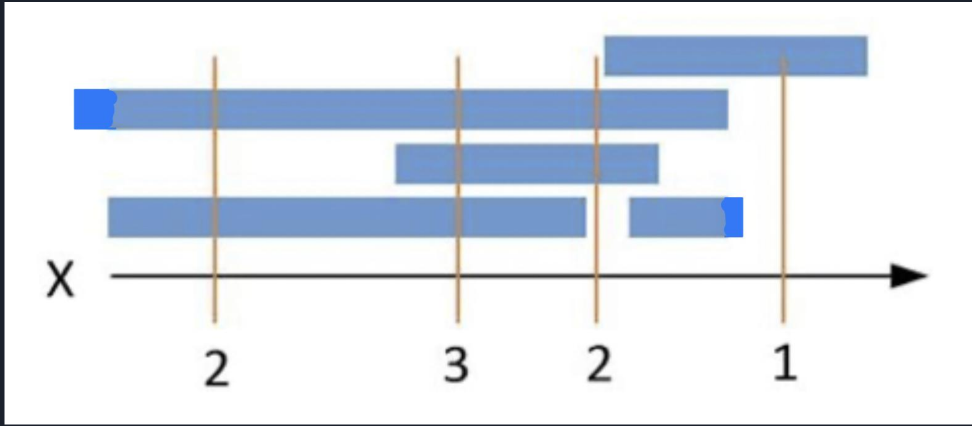
3: Two-colorable - as seen in lecture

4: DAG Related

5: Proof of BFS - as seen in lecture

Practice Midterm 2: Problem 2

Consider a set of intervals/tasks. Design an algorithm that finds the maximum number of mutually overlapping intervals/tasks.



Practice Midterm 2: Problem 2

Algorithm:

- Sort all interval start times and end times into a singular array. Keep track of which times are starting times and which are ending times.
  - depending on how we consider intervals that start and end at the same time, we need to modify this step.
- Initialize two variables to keep track of active_intervals and global_max.
- For every time in the sorted array:
  - if we see a start time, increment active_intervals by one.
  - if we see an end time, decrement active_intervals by one.
  - update global_max as the max of global_max and active_intervals
- return global_max

Practice Midterm 2: Problem 2

Proof:

- Proof by contradiction, assume we return an answer that is not equal to the correct global_max.
- At any given time, the number of active intervals can be calculated by subtracting the number of intervals that have ended from the number that have started.
- Then this must have happened: the algorithm must have missed a start time, or we missed an end time.
- Since the start/end times are sorted, this is not possible.

Practice Midterm 2: Problem 2

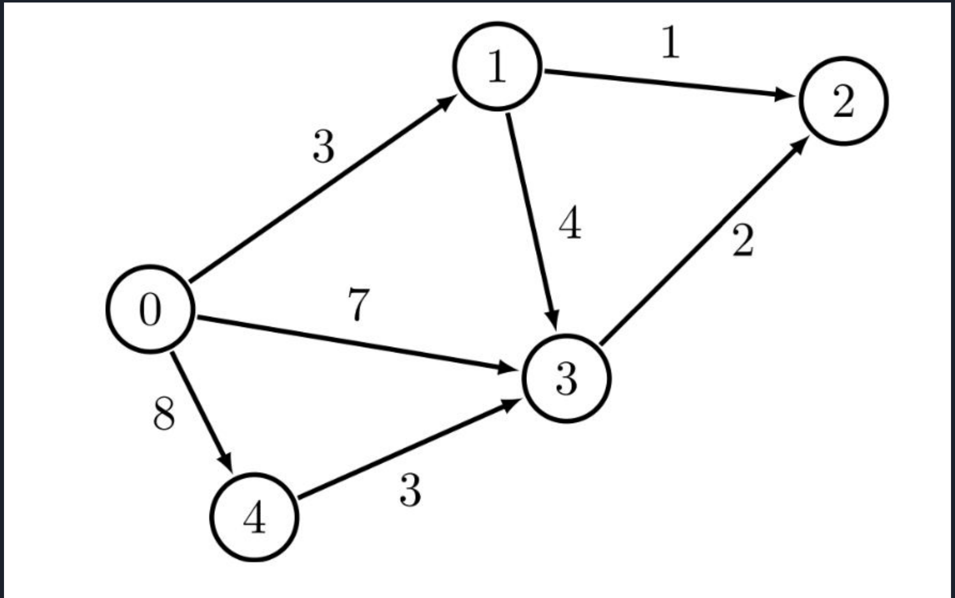
Time complexity:

- Sorting: would O(nlogn), since we are sorting 2n timesteps
- We have a linear traversal of the timesteps, for each timestep we do some arithmetic, and take the max of two numbers => thus we do a constant number of operations.
- Thus whole algorithm is O(nlogn)

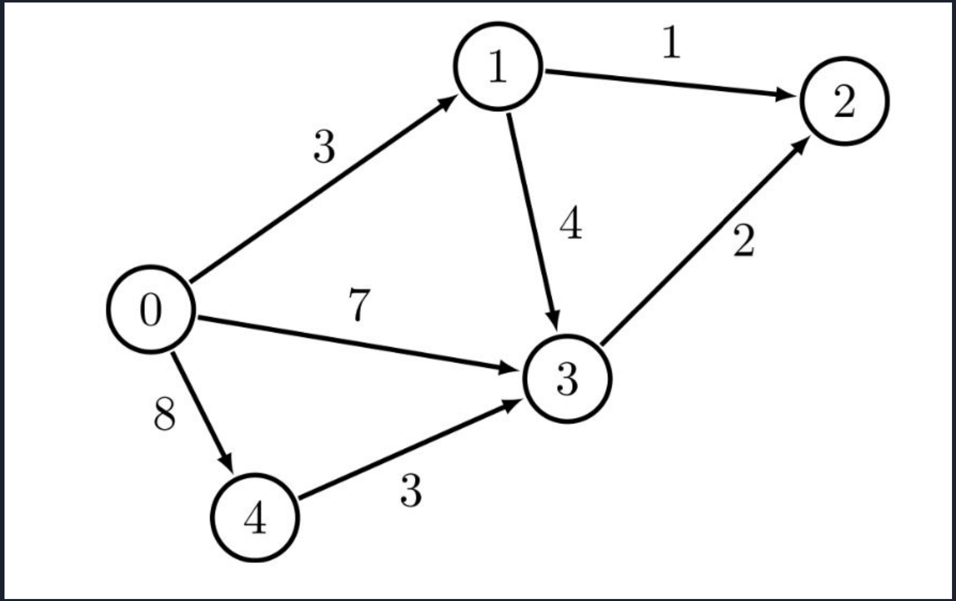
Practice Midterm 2: Problem 4

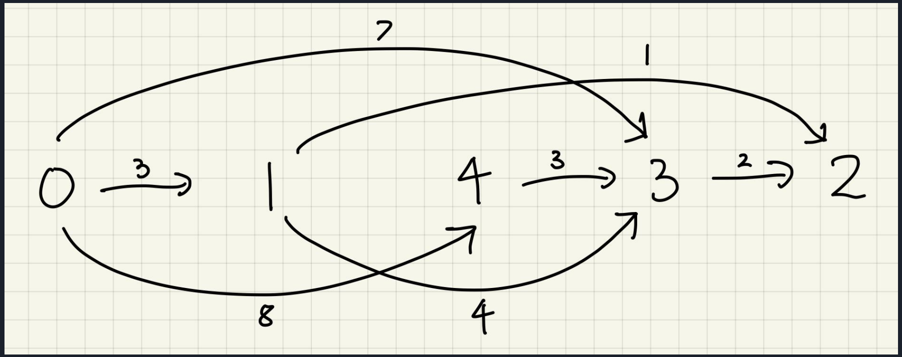
Design an O(V+E) algorithm that finds the shortest path between two vertices in a connected DAG, where V is the number of vertices and E is the number of edges.

Practice Midterm 2: Problem 4

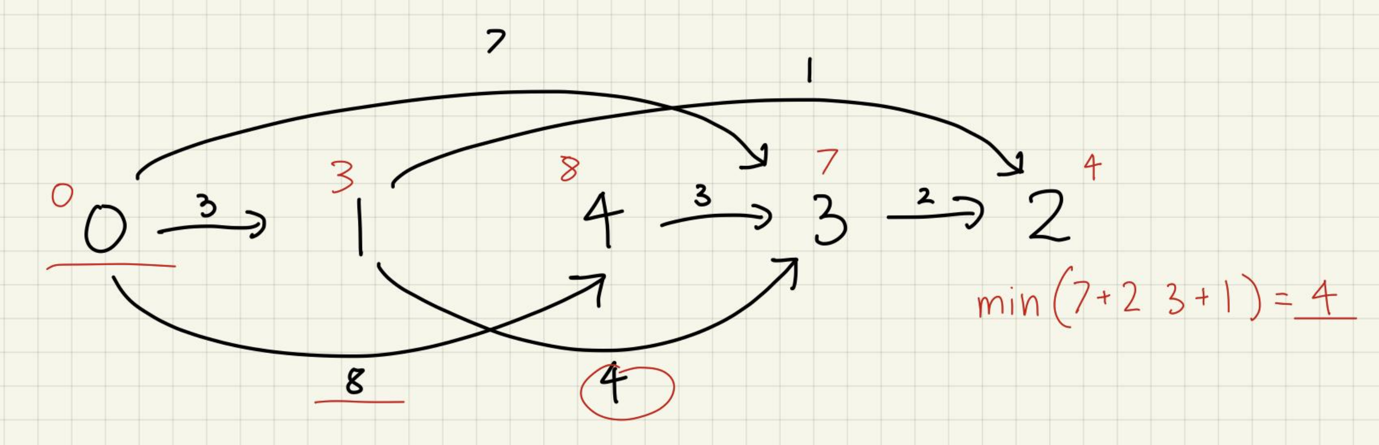


Practice Midterm 2: Problem 4





Practice Midterm 2: Problem 4



Practice Midterm 2: Problem 4

Algorithms:

- Keep track of temp_min for every node, initialize all to infinity.
- set min_dist to source node to 0.
- Run a modify topo sort on the DAG. We disregard all nodes that come before our source node of interest in the topological ordering:
    - for each directed edge to the next source node being processed, update temp_min to be min(temp_min, min_dist of incoming node + edge weight).
- Output the min_dist for the target node.

Practice Midterm 2: Problem 4

Proof: Since algorithm is greedy, lets try using induction:

- base case:
    - trivially, the distance from root to itself is 0.
- assume that we know the shortest distance for first n in the topological sort.
    - since we know the minimum distance for first n nodes, the min path to this node must be from one of those nodes.
    - Because we have a DAG and a topological sort, all the edges pointing into the n+1 node must be from a node we know the minimum distance to.
    - By case analysis we find the minimum distance for the n+1 node by considering all those and take the min path from one of those nodes.

Practice Midterm 2: Problem 4

Time complexity:

- Topological sort runtime:
  - First we count indegrees for every node O(E), and for each node we store their indegrees O(V)
  - for each source node, we remove the outgoing edges O(E)
  - also we keep track of each nodes indegree at each step O(V)
  - topo sort has runtime O(V+E)
- we can use an edge based accounting method to count the number of updates we do: O(E)
- we kept track of min_distance to each node, O(V)
- Thus whole algorithm has runtime O(V+E)