Main Topics

- Graphs: walks, paths, connectivity.
- Dynamic programming: longest increasing subsequence, knapsack, more complicated states.
- Shortest path and how to find them.
- Minimum spanning tree, cut property, and Kruskal’s algorithm.

NOT included:

- How to extract negative cycle from final state of the Bellman-Ford algorithm (you should know that it can be done though).
- Intervals as states.
- Subsets as states.
- Prim’s algorithm.
- Shortest path / path finding algorithms faster than Bellman-Ford ($O(nm)$), e.g. Dijkstra’s algorithm.
- Cycle property.

Dynamic programs:

- State, transition, base case.
- Order in which to compute the states.
- Path finding / counting on grids.
- Longest common subsequence: states are pairs of locations.
- Knapsack: state is remaining budget.

Graph algorithms

- reachability / connectivity.
- Bellman-Ford algorithm: details, correctness, and termination with either paths or a negative cycle.
- Minimum spanning trees, cut property.
- Kruskal’s algorithm and proof of correctness by the cut property.
Practice Problems

1. Homework 2 (solutions will be posted at Friday Oct 13, 3pm).

2. Problems from Tests 2 and 3 of Fall 2016.

3. Exercises 6.3 of textbook:
   Given \( n \) potential locations \( m_1 \ldots m_n \), and potential profits for each location \( p_i \), find the maximum profit if we can only choose locations that are at least \( k \) apart.

4. Exercise 6.11 in textbook: find the longest common subsequence of two strings of length \( n \), \( x \) and \( y \), in \( O(n^{10}) \) time or better.

5. Exercises 6.17 of textbook: is it possible to make a total value of \( v \). This was covered at the start of Lecture 8 on Sep 25.

6. Exercise 6.18: make change where each coin value can be used at most once. This was covered in Lecture 10 on Oct 2.

7. Exercises 4.1. and 4.2. of the textbook, but both with the Bellman-Ford algorithm (Dijkstra’s algorithm was optional).

8. Exercise 4.7 in textbook:
   You are given a directed graph \( G = (V,E) \) with (possibly negative) weighted edges, along with a specific node \( s \in V \) and a tree \( T = (V,E'), E' \subseteq E \). Give an algorithm that checks whether \( T \) is a shortest-path tree for \( G \) with starting point \( s \). Your algorithm should run in linear time.

9. Exercise 4.8. in textbook:
   Professor F. Lake suggests the following algorithm for finding the shortest path from node \( s \) to node \( t \) in a directed graph with some negative edges: add a large constant to each edge weight so that all the weights become positive, then run Dijkstra’s algorithm starting at node \( s \), and return the shortest path found to node \( t \). Is this a valid method? Either prove that it works correctly, or give a counterexample.

10. Exercise 4.12 in textbook:
    Give an \( O(n^2) \) time algorithm for the following task.
    Input: An undirected graph \( G = (V,E,l) \); edge lengths \( l_e > 0 \); an edge \( e \in E \). Output: The length of the shortest cycle containing edge \( e \).

11. Exercises 5.1. in textbook.

12. Exercise 5.2. in textbook, but only with Kruskal’s algorithm (we did not cover Prim’s algorithm).
13. Exercise 5.7. in textbook:
   Show how to find the maximum spanning tree of a graph, that is, the spanning tree of largest total weight.

14. Exercise 5.10 in textbook (modified to remove duplicates):
   Let $G$ be an undirected, unweighted graph where all edges have distinct weights. Let $T$ be a MST of graph $G$. Given a connected subgraph $H$ of $G$, show that $T \cap H$ is contained in the MST of $H$. 