• Main Topics
  – Graphs: walks, paths, connectivity.
  – Checking $s \rightarrow t$ reachability in graphs.
  – Shortest path and how to find them.
  – Minimum spanning tree, cut property, and algorithms for finding them.
  – Interface of priority queue.
  – Dijkstra’s algorithm and how to get to $O(m \log n)$.

• NOT included:
  – Speeding up reachability to $O(m)$ using breadth-first-search.
  – Cycle property.
  – How to extract negative cycle from final state of the Bellman-Ford algorithm (you should know that it can be done though).
  – Speeding up Kruskal’s / Prim’s MST algorithms $O(m \log n)$.

• Graphs
  – Vertices, edges, arcs, $G = (V, E, w)$ or $G = (V, E, l)$.
  – Walk: sequence of vertices $v_0 \ldots v_k$ s.t. $v_i \rightarrow v_{i+1} \in E$.
  – Path: Walk with no repeating vertices.
  – If there is a $s \rightarrow t$ walk, there is a $s \rightarrow t$ path.

• Connectivity:
  – Reachability: check if there is path from $s \rightarrow t$
  – Connectivity in $O(nm)$ time by repeatedly propagate ‘reached’ flags.
  – Undirected graph: connected components, vertices are partitioned into connected components.

• $s \rightarrow t$ shortest path:
- Bellman-Ford algorithm: details, correctness, and termination with either paths or a negative cycle.
- Dijkstra’s algorithm: speedup via greedy when all lengths are positive.

• Minimum spanning trees
  - Cut property.
  - Kruskal’s and Prim’s algorithms: proof of correctness by the cut property

### Practice Problems

1. Homework 2 Problems 1 - 3 (solutions are posted).

2. Exercises 4.1. and 4.2. in textbook.

3. 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.

   **SOLUTION:**
   Do a BFS on $T$ to compute the distance from $s$ to all vertices along the tree, let the value to $u$ be $d'[u]$.

   Then check if for all edges $u \rightarrow v$, we have
   \[
   d'[u] + l_{u\rightarrow v} \geq d'[v].
   \]

   If so, $T$ is a shortest path tree, otherwise it’s not.

   Since for each $u$, $d'[u]$ is the length of a path from $s$ to $u$, these are overestimates to distances. So we can use it as an intermediate state in the Bellman-Ford algorithm, which means that these distance are optimal if and only if no more update to $d'$ can be made.

4. 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.
**SOLUTION:**
No, consider the graph on 3 vertices:

- $s \rightarrow a$  \[ l_{s \rightarrow a} = 1, \]
- $a \rightarrow t$  \[ l_{a \rightarrow t} = 1, \]
- $s \rightarrow t$  \[ l_{s \rightarrow t} = 10. \]

Clear the shortest path is $s \rightarrow a \rightarrow t$ with total length 2.

But if we add a constant $c$ to all edge lengths, then the length becomes $2 + 2c$ while the length of $s \rightarrow t$ is $10 + c$. Once $c > 9$, we will incorrectly return $s \rightarrow t$ instead.

5. 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 $e$.

**SOLUTION:**
Remove $e$ from $G$ to form $G'$, compute the shortest path between the endpoints of $e$ in $G'$. Add $e$ to it to complete the cycle.

The runtime is dominated by the cost of computing shortest paths, which is $O(n^2)$. This is correct because any cycle including $e$ is a path between its two endpoints once $e$ is removed, so it suffices to minimize the length of this path.

6. Exercises 5.1. and 5.2. in textbook.

7. Exercise 5.4. in textbook:

Show that if an undirected graph with $n$ vertices has $k$ connected components, then it has at least $n - k$ edges.

**SOLUTION:**
Let $m$ be the number of edges in his graph.

Note that adding an edge between two different connected components reduced the number of connected components by 1. So we can add $k - 1$ edges to this graph to make things connected.

This connected graph has a spanning tree as a subgraph, which has at least $n - 1$ edges, so we have

$$m + (k - 1) \geq n - 1,$$

which simplifies to $m \geq n - k$. 

3
8. 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.

   **SOLUTION:**
   Negate all edge weights, then run the minimum spanning tree algorithm.

9. 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 $H$.

   **SOLUTION:**
   By the cut property, and edge $e$ in $T$ must be the minimum in some cut $E(S, V \setminus S)$ of $G$.
   This same cut on $H$, $E(S \cap H, V_H \setminus)$ contains strictly fewer edges, so if $e$ is in $T \cap H$, the same cut certifies that it’s in the MST of $H$. 