# Comp 251: Practice problems

Instructor: Jérôme Waldispühl

This is a collection of problems that you can use to prepare the COMP 251 midterm exam.

- The first part of the exam (40 points) will be composed of multiple choice questions similar to the quizzes following each lecture.
- The second part of the exam (60 points) will be composed of longer problems such are those illustrated below.

## Hashing

1. Suppose that you use a hash table and a hash function implementing the division method. The following keys are inserted: 5, 28, 19, 15, 20, 33, 12, 17, 10 and m = 9 (for simplicity, here we do not distinguish a key from its hashcode, so we assume h(key)=key). In which slots do collisions occur?

2. Now suppose you use open addressing with linear probing and the same keys as above are inserted. More collisions occur than in the previous question. Where do the collisions occur and where do the keys end up?

## Balanced binary search trees

3. What is the maximum number of nodes in an AVL tree of a given height h?

4. Starting with an empty tree, construct an AVL tree by inserting the following keys in the order given: 2, 3,5, 6, 9, 8, 7, 4, 1. If an insertion causes the tree to become unbalanced, then perform the necessary rotations to maintain the balance. State where the rotations were done.

## Heaps

5. Suppose you have a heap which supports the usual add() and removeMin() operations, but also supports a changePriority (name, new Priority) operation. How could you combine these operations to define a remove(name) operation?

6. Suppose you used an ordered array to implement a priority queue. Give the O( ) time for the operations removeMin(), add(element, key), findMin() take?

## Disjoint sets

7. Consider the set of all trees of height h that can be constructed by a sequence of "union-by-height" operations. How many such trees are there?

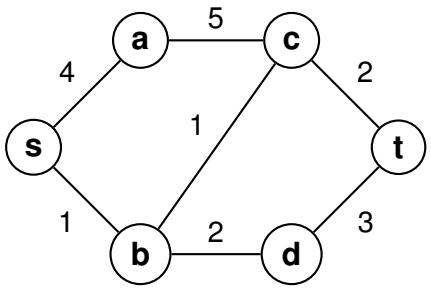
8. Consider a tree formed by union-by-height operations, without path compression. Suppose the tree has n nodes and the tree is of height h. Show that n is greater than or equal to $2^h$. (Note that the tree need not be a binary tree, and so we cannot just apply properties of binary trees to this problem. Indeed, for binary trees of height h, we can only say that the number of nodes at most $2^{h+1} - 1$, which is looser than the bound stated in the question.)

## Minimum spanning-trees

9. Prove that for any weighted undirected graph such that the weights are distinct (no two edges have the same weight), the minimal spanning tree is unique.

10. If a connected undirected graph has n vertices, then any spanning tree has n-1 edges.

11. Consider the flow graph below. Apply the Kruskal algorithm to calculate the minimum spanning tree.

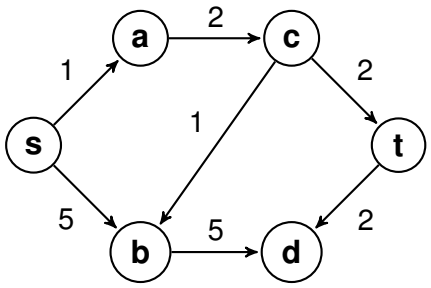



## Single-source shortest paths

12. In breadth first search, each vertex has a "visited" field which is set to true before the vertex is put in the queue. What happens if BFS instead sets the visited field to true when the vertex is removed from the queue? Does the algorithm still work? Does it run just as fast? What if we want to find the shortest path between every pair of vertices in the graph ?

13. Dijkstra's algorithm assumes the edges have non-negative weights. (Where does this come up in the proof of correctness?) But suppose we have a graph with some negative weights, and let edge e be such that cost(e) is the smallest (most negative). Consider a new graph in which we add cost(e) to all the edge weights, thus making all weights in the new graph non-negative. Now the conditions hold for Dijkstra to find the shortest paths, so we could now run Dijkstra. Is this a valid way to solve for shortest paths in the case that some edges have negative weights? Justify your answer.

14. Consider the flow graph below. Apply the Dijkstra's algorithm to calculate the shortest paths from $s$.




# Bipartite graphs

15. Given the preferences shown here, use the Gale-Shapley algorithm to find a stable matching.

| A | A's preferences | B | B's preferences |
|---|---|---|---|
| $\alpha_1$ | $\beta_1, \beta_2, \beta_3$ | $\beta_1$ | $\alpha_3, \alpha_1, \alpha_2$ |
| $\alpha_2$ | $\beta_1, \beta_2, \beta_3$ | $\beta_2$ | $\alpha_1, \alpha_3, \alpha_2$ |
| $\alpha_3$ | $\beta_1, \beta_2, \beta_3$ | $\beta_3$ | $\alpha_3, \alpha_2, \alpha_1$ |

16. Consider an instance of the stable matching problem in which, for all $\alpha \in A$, $\alpha$'s first choice is $\beta$ if and only if $\beta$'s first choice is $\alpha$. In this case, there is only one stable matching. Why?

# Flow networks

17. Suppose the capacities in a network flow are not integers. How would this change the O( ) runtime of the Ford Fulkerson algorithm? Does the algorithm terminate?

18. Consider the flow graph below. Apply the Ford-Fulkerson algorithm to calculate the maximum flow.