Comp 251: Practice problems for the Final

Instructor: Jérôme Waldispühl

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

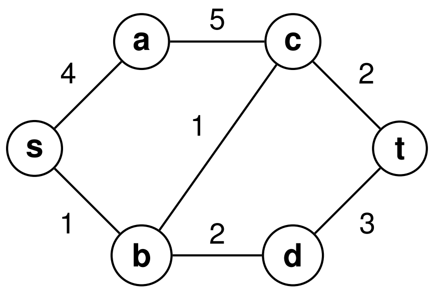
trees to this problem. Indeed, for binary trees of height h, we can only say that the number of nodes at most $2^{h+1} - 1$, which is looser than the bound stated in the question.)

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.

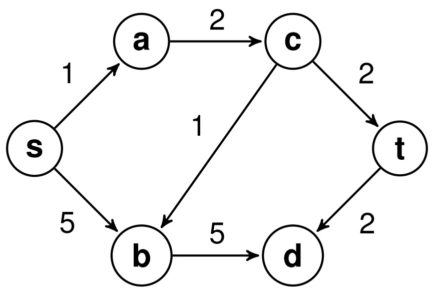


Single-source shortest paths

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.

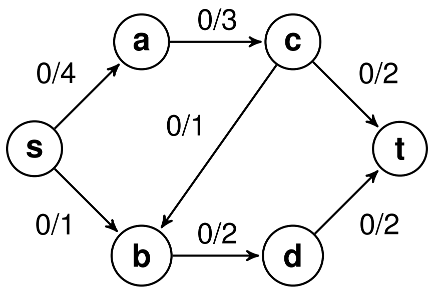
A	A's preferences	B	B's preferences
α_1	β_1, β_2, β_3	β_1	α_3, α_1, α_2
α_2	β_1, β_2, β_3	β_2	α_1, α_3, α_2
α_3	β_1, β_2, β_3	β_3	α_3, α_2, α_1

16. Consider an instance of the stable matching problem in which, for all $\alpha \in A$, α's first choice is β if and only if β's first choice is α. 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.



Dynamic programming

19. The Coin Row Problem: Suppose you have a row of coins with values that are positive integers $c_1, \cdots, c_n$. These values might not be distinct. Your task is to pick up coins have as much total value as possible, subject to the constraint that you don't ever pick up two coins that lie beside each other. How would you solve this using dynamic programming?
Solve the problem for coins with values c_1 to c_6 as follows: $(5, 1, 2, 10, 6, 2)$.

20. The Coin Change Problem: Suppose we have m types of coins with values $c_1 < c_2 < \cdots c_m$ (e.g. in the case of pennies, nickels, dimes, ... we would have $c_1 = 1$, $c_2 = 5$, $c_3 = 10$, $\cdots$). Let $f(n)$ be the minimum number of coins whose values add up to exactly n. Write a recurrence for $f(n)$ in terms of the values of the coins. You may use as many of each type of coin as you wish.
As an example, suppose the coin values c_1, c_2, and c_3 are 1, 3, 4. Solve the problem for $n = 6$ using dynamic programming.

21. What is the optimal substructure of the Neddleman-Wunch algorithm (i.e. optimal pairwise sequence alignment)?

Divide-and-Conquer

22. In Karatsuba multiplication, when you do the multiplication $(x_1+x_0)\cdot(y_1+y_0)$, the two values you are multiplying might be $n/2+1$ digits each, rather than $n/2$ digits, since the addition might have led to a carry e.g. $53+52=105$. Does this create a problem for the argument that the recurrence is $t(n)=3t(n/2)+c_n$?

23. Apply the master method to determine the asymptotic behavior of the function $T(n)$.

    1. $T(n)=2\cdot T(n/4)+n^{0.51}$
    2. $T(n)=0.5\cdot T(n/2)+1/n$
    3. $T(n)=64\cdot T(n/8)-n^2\cdot logn$
    4. $T(n)=\sqrt{2}\cdot T(n/2)+\log n$
    5. $T(n)=6\cdot T(n/3)+n^2\cdot\log n$
    6. $T(n)=3\cdot T(n/3)+n/2$

24. Write a recurrence that describes its worst-case running time of the quicksort algorithm.

Amortized analysis

25. Suppose we perform a sequence of stack operations on a stack whose size never exceeds k. After every k operations, we make a copy of the entire stack for backup purposes. Show that the cost of n stack operations, including copying the stack, is $O(n)$ by assigning suitable amortized costs to the various stack operations.

26. Suppose we perform a sequence of n operations on a data structure in which the i^{th} operation costs i if i is an exact power of 2, and 1 otherwise. Use aggregate analysis or accounting method to determine the amortized cost per operation.