COMP160: Algorithms, **Homework 10 part 2**

- To obtain full credit, you must justify your answers. When describing an algorithm, do not forget to **state any assumptions that you make**, analyze its running time and explain why the algorithm is correct.
- Although not specifically stated, You can assume that we look for algorithms that are as fast as possible, and bounds as tight as possible.
- You may discuss these problems with others, but remember to write the answers on your own. In case of doubt, cite any source you used to do the assignment.
- Remember to submit each question in a separate page.
- **You do not have to do ALL questions. For full credit, attempt at least 4 problems.**
- Doing additional questions is great for reviewing the content of the course

1. Simplify the following expressions:
   - $O(n \log_3 n) + O(n \log_2 n) + O(n \log_n n)$.
   - $O(n^2 \log_2 n) + O(n(\log_2 n)^2)$.
2. Analyze the recurrence $T(n) = 3T\left(\frac{n}{4}\right) + \sqrt{n}$ using the Master Method. Draw the recurrence tree.
   [*Optional:* Show a linear upper bound using substitution.]
3. What is a stable sorting algorithm? Which algorithms (as presented in class) are stable?
4. Does Selection work if we group elements into groups of 3 (instead of the usual 5 as described in class)? That is, can it find the k-th smallest element?
5. Use IRV to compute the expected number of times a randomly generated 16-digit binary number will have four 1's in a row (e.g., 0011111000111111 has the pattern 1111 five times).
6. A new computer game makes you guess a random 12 bit natural number with YES/NO questions. You search for a good strategy and in an online forum, user `EliteHacker10One` claims to have a strategy that can find the number with at most 10 questions. What do you deduce from this?
7. Selection, Quicksort, and RandomizedSelection all call some version of Partition. What pivot does each one use?
8. Does CountingSort work when we have an array of (potentially both positive and negative) integers? Do you need to do any modifications to the default algorithm? What about RadixSort?

9. Build a max-heap using the input $[20, 1, 3, 6, 35, 12, 7, 24, 5]$, using both the top-down and the bottom-up methods. Extract the maximum and re-heapify. Delete 20 and re-heapify.

10. Build a binary search tree using the same input as the previous problem. Extract the maximum and rebuild the tree. Delete 6 and rebuild the tree.

11. Build an AVL tree using the same input.

12. For each of the following DP problems from lecture: **(i)** give the recursion, **(ii)** give the base cases, and **(iii)** analyze the time complexity:

    (a) Number of paths from top-left to bottom-right in an $m \times n$ table

    (b) LCS

    (c) LIS (covered in recitation)

    (d) Rod-cutting

13. What is the purpose of amortized analysis? Specifically, what can we do that is not possible without this technique?

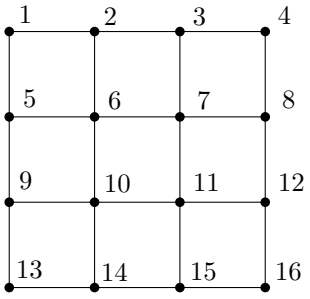
14. Explain the principle behind amortized analysis. That is, why can we have an amortized runtime that is smaller than the corresponding worst case bound.

15. Review the algorithm for computing the rank of an element in a balanced binary search tree. In class we discussed an algorithm that augments each node by storing the number of nodes in the subtree under it coupled with a slightly complicated algorithm. What would happen if we instead store the rank of the element instead? Wouldn't this drastically simplify the search algorithm?

16. What is the difference between the top-down and bottom-up approach in a dynamic problem?

17. What is the difference between Simple Uniform Hashing and Uniform Hashing? When should we use each of the assumptions?

18. Using the graph below, draw the trees resulting from BFS and DFS. Assume that both procedures consider the vertices in numerical order. For DFS clearly denote the two numbers associated to the traversal of each vertex.



19. What is the key Lemma that guarantees correctness of Prim's algorithm? What about Kruskal?

20. Dijkstra's algorithm is known to have several runtimes depending on several parameters. Draw a decision tree that shows the best bound for each case (say, if the graph is given in adjacency matrix/list, if G has few or many edges, and so on).

21. State the condition for a vertex to be a cut vertex in G as a function of the discovery times of a DFS traversal.

22. Give an example of a graph G that Bellman-Ford needs $n-1$ relaxation rounds until we find the shortest path tree. In what order did you traverse the edges? In what order would you need to traverse it to need only 1 relaxation round?

23. Give an example of a (weighted undirected) graph and a source s whose minimum spanning tree is different than the shortest path tree from s

---

Extra summary questions. We strongly recommend you write down the answers and be capable of reconstructing them fast. They should be helpful in the exam (and in the future).

24. Go back to each of the previous questions and list the topic it is associated to. Which questions are related to topics of critical importance? Is there anything of that topic that you do not feel fully comfortable with?

25. Make a list of all sorting algorithms discussed in class. For each algorithm list its worst-case runtime, its expected runtime (if different than worst-case), if it is in place or not, the runtime recurrence (if any), any case in which it works better, and a 1 sentence summary of how it works.

26. Make a list of all data structures/techniques discussed in the second block of class. For each one list a 1 sentence summary, what are they generally good for, a sample application, and a case in which they cannot be used

27. Make a list of all graph algorithms described in the third block of class. For each one list a 1 sentence summary, the graph requirements (if any), the space/time complexity, and comparison with other similar algorithms.