COSC3320: Homework #2
Smart recursion, greedy, data structures, set operations, graphs

Instructor: Carlos Ordonez
University of Houston

1 Introduction

This is a written homework emphasizing theory: correctness and time complexity. You need to produce a PDF in latex.

2 Homework requirements

1. Team-based: same as HW #1. Students who lost their partner are required to inform TAs to assign a new partner. Working alone is discouraged.
2. PDF Format: IEEE double column
3. Notation: following textbooks. You must use different symbols for variable assignment and comparison, to avoid ambiguity with math symbols and programming symbols (e.g.$x \stackrel{\circ}{=} 1$, $x \leftarrow 1$) and comparison with equality (=).
4. Arrays and vertices are indexed $1 \ldots n$.
5. Functions: arrays passed by reference, functions may return several values as a list.
6. You must write the algorithm in math notation. You cannot copy/paste source code into the latex document.
7. It is a good idea to program the algorithms in C++, Java or Python, but making sure arrays are indexed 1 to n (declare them $[n+1]$).
8. Derive tight time complexity $\Theta(g(n))$ bounds when possible. Otherwise, $O(f(n)) = g(n)$.
9. References: I know you will start searching the answers in Google. If you find anything beyond the textbooks, you must disclose the specific reference in your report to avoid misunderstandings.

3 Problems

1. Consider the set cover problem on U with n elements. Explain for which inputs the greedy algorithm can be $O(n)$ (best case) and for which inputs it can be $O(2^n)$ (worst). How far is the greedy algorithm solution from the optimal solution?
2. Assume you have a string of length n with k symbols, possibly repeated. Justify why Huffman codes can be built in time $\Theta(n \log(n))$. Explain in which cases RLE is better than Huffman codes (lower length of encoded string). Assume the Huffman binary tree is already built. Show the algorithms and their $O()$ to encode a string with n symbols and decode a string with m bits.

3. For the DNA sequence alignment problem show the top down algorithm with smart recursion (not shown in Gopal's textbook). Explain how this algorithm is transformed into the classic "dynamic programming" iterative algorithm using loops. Justify why it is $O(mn)$ for input strings of length m, n. Compare similarities and differences between this algorithm and the EditDistance algorithm in Jeff's textbook.

4. Write a table summarizing O() for these operations: insert, delete, search one element, get max(), list all elements in order of these data structures: sorted linked list, unsorted array, sorted array stack, circular queue, binary search tree (balanced), heap (max). If an operation does not make sense write "NA" (not available).

5. For the matrix chain multiplication problem compare the smart recursion (DP) solution with these alternative solutions: (1) exhaustive: consider all potential parenthesis placements (2) greedy: pick each product with largest intermediate sizes. (3) greedy: pick the product with the smallest matrix output size. Write the algorithm for each alternative and show its $O()$.

6. Consider two large sets of integers S_1 and S_2, stored on files and a limited amount of RAM. Write efficient algorithms to compute $S_1 \cap S_2$, $S_1 \cup S_2$, $S_1 \subseteq S_2$, assuming only a block (subset) of each set can be loaded in RAM as the data sets are scanned. What is the fastest algorithm you can come up? Show its $O()$.

7. Contrast Kruskal and Prim algorithms to find the MST of a connected undirected graph G. Compare the $O()$. Now, consider the dynamic case where edges or vertices may be inserted or deleted continuously. Which algorithm is better to handle a batch of changes?

8. Explain how binary search trees become 2-3 trees and then how to 2-3 trees are generalized to B-trees with 3 or more keys per node. When does the B-tree grow one level? When does the B-tree shrink one level? Explain how you split a node. Write algorithms to insert a key, search a key and their $O()$. Explain how to exploit the B-tree to compute the set operations above.

9. For an undirected graph G determine which vertices are reachable fron any vertex. The output should be a binary matrix (1=reachable). Show best, average and worst cases for time complexity, considering graphs of different density amd connectivity (e.g. isolated components, a tree, a complete graph). Consider the more general version of the problem where edges have distances (weighted graph). Write an algorithm that finds the distance of the shortest path between each pair of vertices.

10. For an undirected graph G write an algorithm that enumerates all triangles in the graph, without repetition. Justify all triangles are found (no triangle missing). Show best, average and worst cases for time complexity, considering graphs of different density and connectivity. Explain how to find cliques of 4 and 5 vertices. Discuss examples on social networks.