CS161 SU24: Homework 4 (Due July 29 11:00am)

❖ Problem 1

Divide and conquer was a key technique we used to speed up algorithms for sorting, with the prime example of Mergesort. Mergesort was also a recursive algorithm, which might hint to us that dynamic programming could help speed up the algorithm even further.

1. Consider the following permutation of the first 16 integers:

   $$3\ 6\ 4\ 1\ 10\ 15\ 14\ 12\ 8\ 11\ 5\ 7\ 13\ 2\ 16\ 9 \tag{1}$$

   Draw a tree where each node is an input to mergesort, and a node's children represent the recursive call made from that input.

2. Based on what you drew, which of the two elements of dynamic programming that we discussed in class are not present? Briefly explain why dynamic programming fails to speed up mergesort based on this.

❖ Problem 2

In lecture, we saw that the longest simple path problem does not have optimal substructure. However, if we make a slight change and require the graph G to be a DAG (directed acyclic graph, in other words, a directed graph with no cycles), the situation seems to change.

Problem (DIRECTEDLSP)**.**

INPUT: An unweighted directed acyclic graph $G = (V, E)$ and the labels u, v of two vertices in G. You can assume that v is reachable from u.

OUTPUT: A path from u to v consisting of as many edges in G as possible. The path must be simple, meaning each vertex can only be used once.

1. Draw an instance of DIRECTEDLSP on a graph with 10 nodes. Label two vertices u and v, then write down the longest simple path from u to v.

2. Identify optimal substructure in this problem. To do this, suppose you have the solution to the problem, then show that breaking the solution into two pieces yields solutions to two subproblems. Remember, the subproblems must be valid instances of the main problem!

3. Define the recursion in this problem. From the starting node u, what are the recursive calls we want to make? How do we select the best option out of these recursive calls?

4. Based on the recursion you've defined, are there overlapping subproblems for this problem? If yes, draw an instance where this occurs.

5. Write an algorithm to solve DIRECTEDLSP. You must use dynamic programming, but you can do either of top-down or bottom-up. State which option you will use as well.

6. What is the runtime of your algorithm?

❖ Problem 3

Consider the following problem.

Problem (REDSQUARE)**.**

INPUT: A two dimensional array with m rows and n columns (basically an m by n grid), whose elements are either RED or BLUE.

OUTPUT: The size of the largest square which can be found in this array where every element in the square is RED.

1. Does this problem exhibit optimal substructure? State the series of choices that are made and the subproblems that emerge from these choices. If these are valid instances of the original problem AND can be used to construct the final solution, there is optimal substructure.

2. Define the recursion in this problem. What are the recursive calls that we want to make? How do we select the best option out of these recursive calls?

3. Based on the recursion you've defined, are there overlapping subproblems for this problem? If yes, draw an instance where this occurs.

4. Write an algorithm to solve REDSQUARE. You must use dynamic programming, but you can do either of top-down or bottom-up. State which option you will use as well.

5. What is the runtime of your algorithm?

❖ Leetcode

Here is a sample of some Leetcode problems related to dynamic programming that you should be able to start attempting now. As mentioned at the beginning of class, set a timer to try solving these on your own, then once the timer is up check a solution and try to understand why that works.

For any solution you write, try to identify optimal substructure, overlapping subproblems, and state the runtime of your algorithm.

- Minimum Path Sum [Medium]

  https://leetcode.com/problems/minimum-path-sum/description/

- Maximum Subarray [Medium]

  https://leetcode.com/problems/maximum-subarray/description/

- Jump Game II [Medium]

  https://leetcode.com/problems/jump-game-ii/description/

- Trapping Rain Water [Hard]

  https://leetcode.com/problems/trapping-rain-water/description/