1. Consider this game. We are playing on an \( n \times n \) board with a number in each square, possibly negative. When you visit a square, you get points equal to the number on that square. Each square also has three moves associated with it, that is, three squares to which you can visit from here. Our goal is to get from the square \((0, 0)\) to the square \((n, n)\) with the largest possible score. Assume that you cannot ever return to a square once you leave.

   a) Give a simple recursive procedure to find the maximal path, without any use of memory. What is the time complexity of this approach?

   b) Give an algorithm that uses dynamic programming. How much faster is this approach?

   c) Since you can’t return to a square once you leave, we can model this problem as a graph. There is one node for each box and the three transitions represent edges leaving the node. Give a graph theoretic algorithm to find the optimal set of moves. Is this any faster than the dynamic programming approach?

2. Describe a generalization for the FFT where \( n \) is a power of 3 instead of 2. Analyze the running time of the new algorithm.

3. Consider the change-making problem, that is, given some value of \( n \) and unlimited quantities of coins with denominations \( d_1, d_2, \ldots, d_k \), find the smallest number of coins that add up to \( n \) or indicate that such a value is unfeasible. A real-world example (for the United States) would be denominations 1, 5, 10, 25 and \( n \) something like 233.

   a) Design a greedy approach to the problem and prove it produces the optimal result. What is the maximal runtime (in terms of \( n \) and \( k \))? 

   b) Devise a dynamic programming approach. Is it any different from the greedy algorithm? Prove the optimality of this approach as well.