Khrystian Clark

Assignment 4

CS 325 – Winter 2022

Assignment: Dynamic Programming & Backtracking

Note: These problems are to be discussed as part of the Group Assignment. (Check this week's Group Assignment on Canvas for details).

The questions asked in this assignment – code implementation and time complexity of your code should be done individually based on the problem-solving strategy discussed within your group.

1. **Solve Dynamic Programming Problem and find its optimal solution.**
   Given a list of numbers, return a subset of non-consecutive numbers in the form of a list that would have the maximum sum.

   Example 1: Input: [7,2,5,8,6]
   Output: [7,5,6] (This will have sum of 18)
   Example 2: Input: [-1, -1, 0]
   Output: [0] (This is the maximum possible sum for this array)
   Example 3: Input: [-1, -1, -10, -34]
   Output: [-1] (This is the maximum possible sum)

   a. Implement the solution of this problem using dynamic Programming. Name your function **max_independent_set(nums)**. Name your file **MaxSet.py**
      a. Strategy
         i. Base case
            1. Find the list of non-consecutive numbers that returns the greatest max sum.
         ii. Get count for how many numbers in the input list, create an empty array to put the new values in, and establish a base-case max value
            1. Count = len(input)
            2. Newlist = []
            3. Maxnum = -1000
         iii. Iterate for through each non-consecutive value and create a new list with the integer(s) while storing the sum.
            1. Num1 = 0
            2. Num2 = 2
            3. For i in (nums): #Outside loop iterates through from a starting value
               a. Inside loop iterates through follow-on non-consecutive values

                    b. Given that the starting value is less than the initial
                    c. If num2 > num1
                        i. Save it into a new placeholder
                4. Replace maxnum as needed and append the newlist when a higher value is created
            iv. Return maxnum, and newlist

   b. What is the time complexity of your implementation?
      a. O(n^2)

2. **Implement a backtracking algorithm**
   a. Write the implementation to solve the powerset problem discussed in the exercise of the exploration: Backtracking. Name your function **powerset(inputSet)**. Name your file **PowerSet.py**
      a. This one is a little different than the permutations problem, as integer order matters now and you can have less values in it than the initial array. I, along with the group I worked with did not find a more time efficient method than that of the example given in the module.
      b. If the pointer is greater than 0
         i. Initialize an empty array
            1. Result = []
         ii. Run the powerset_helper (below)
         iii. Add choices_made to the input[pointer]
         iv. Recurse back through the function after decrementing the pointer value.
         v. Pop the last element in choices_made
         vi. Recurse back through the function after decrementing the pointer value.
         vii. Return “result”
   b. What is the time complexity of your implementation?
      a. O(2^n)
         i. For each value “n”, at each index, we have a two choice decision.