Joanna McDonald
Analysis of Algorithms Homework 1

The input sequence array is defined as S = (s_1, s_2, … s_i…s_n), a sequence of n numbers. Let P = (p_1, p_2, … p_i…p_n) be another sequence of numbers that represent the optimal sum for the contiguous subsequences ending at the current index. The main idea of this algorithm is to begin from the first index and traverse through the (sequence) array, creating an array of the possible optimal sums of the contiguous subsequence. At each step, the algorithm will compare the current value of S(i) with the possible sum of the current optimal sum; at the previous index, P(i-1); with the current value of S, S(i). This determines if S(i) (the current position) should be included in the current contiguous subsequence (and thus added to the previous optimal sum at P(i-1) and inserted at P(i)), or if the value of S(i) is greater than the value of combining S(i) with the current optimal sum P(i-1), meaning that S(i) should be the new "mark" for the beginning of the current maximum-sum contiguous subsequence. In that case, S(i) will be inserted at P(i), rather than inserting S(i) + P(i-1) at P(i). By using this method, the algorithm only has to traverse through the sequence S once to create the array P, the array of the maximum sum over all contiguous subsequences ending at the current index in S. This technique allows us to break down the larger problem with many smaller subproblems. Each iteration defines the current subproblem using the previous subproblem. Using this technique the algorithm optimizes time complexity, and this process only takes O(n) time.

The input sequence array is S = (s_1, s_2, … s_i…s_n).

1. The array can be traversed, starting at index 0 where S(0) = 6.
    a. During this traversal, an array of the optimal sum choices made at each index will occur.
    b. I will define this array as P, where P = (p_1, p_2, … p_i…p_n). Let i be the current position in the array.
2. This traversal allows us to make a decision at each index that determines if the value of adding the current S(i) to the current optimal sum in array P(i-1) is higher than the value of the current S(i).
    a. If the current index is 0, then S(0) is automatically added at P(0) as the start of the optimal sums at each index.
        i. Other than this, there are two other conditions to consider at each index.
    b. If P(i-1) + S(i) is greater than S(i), then the value of P(i-1) + S(i) will be input into the array at P(i).
    c. In the other condition, where S(i) is greater than P(i-1) + S(i), then a new starting point for the optimal contiguous subsequence will be defined.
        i. We can define a variable to include the index of the current marked starting point. Then, the traversal will continue.
3. At this point, S(i) is defined as the maximum sum over all contiguous subsequences ending at i.
4. This cycle will continue until P(i) is full with n numbers showing at each i the maximum sum for all contiguous subsequences of S ending at that position.

5. Once P is filled, to find the maximum subsequence, we first find the highest value in array P.
    a. This value is the maximum sum for the contiguous subsequence ending at that position, which will be defined as j.
6. Then, the marked starting point of the current contiguous subsequence will be used to define the subsequence.
    a. That index will be the starting point for the subsequence, so the value of S for that index i will be the starting point, S(i).
    b. The index j we have defined will be the endpoint of the subsequence, at S(j).
7. Thus, the contiguous subsequence will be defined from these indexes of S(i) to S(j).

Here is how the process of finding the contiguous subsequence of the example problem will work with this algorithm, step by step.

Starting with the example input sequence array, S = (6, 16, -29, 11, -4, 41, 11):

1. S(0) = 6.
    - So the first value of the array P, the current maximum sum over all contiguous subsequences ending at this position, is defined as 6.

At this point, P = (6).

2. S(1) = 16.
    - Now the comparison between two conditions will occur.
    - The first condition is the sum of the current value of S with the previous value of P. That is, S(1) + P(0). This is 16 + 6 = 22.
    - The other condition is the current value of S. That is, 16.
    - Now the two values should be compared. As 22 > 16, the first condition is chosen and S(1) will be added to P(0) and this number will be P(1).

At this point, P = (6, 16).

3. S(2) = -29.
    - First condition: S(2) + P(1) = -29 + 16 = -13.
    - Second condition: S(2) = -29.
    - -13 > -29 so, the first condition is chosen and -13 is added to array P at position P(2).

At this point, P = (6, 16, -13).

4. S(3) = 11.
    - First condition: S(3) + P(2) = 11 + (-13) = -2.
    - Second condition: S(3) = 11.
    - 11 > -2, so the second condition is chosen and 11 is added at P(3).
    - In this case, instead of the contiguous sum continuing from index 0, a new starting point for the current contiguous subsequence is defined here at index 3.

At this point, P = (6, 16, -13, **11**).

5. S(4) = -4.
    - First condition: S(4) + P(3) = -4 + 11 = 7.
    - Second condition: S(4) = -4.

   - 7 > -4, so the first condition is chosen and 7 is added at P(4).

At this point, P = (6, 16, -13, **11**, 7).

6. S(5) = 41.
   - First condition: S(5) + P(4) = 41 + 7 = 48.
   - Second condition: S(5) = 41.
   - 48 > 41, so the first condition is chosen and 47 is added at P(5).

At this point, P = (6, 16, -13, **11**, 7, 48).

7. S(6) = 11.
   - First condition: S(6) + P(5) = 11 + 48 = 59.
   - Second condition: S(6) = 11.
   - 59 > 11, so the first condition is chosen and 59 is added at P(6).

At this point, P = (6, 16, -13, **11**, 7, 48, 59).

8. Now the entire array S has been traversed and the possible contiguous subsequence sums have been placed in array P.
9. The next step is to find the maximum value in P, showing the maximum sum for a contiguous subsequence of S.
   - This is 59, at P(6).
   - This (6) will be the index of the endpoint of the contiguous subsequence found in S.
   - So S(6), 11, will be the end of the contiguous subsequence.
10. The next step is to find the current marked starting point for the contiguous subsequence.
    - The index of the current marked starting point is stored as a variable, which is index 3, where P(3) = 11.
    - So, S(3), 11, will be the start of the contiguous subsequence.
11. Now with the start and endpoint defined, we can define the contiguous subsequence..
    - The contiguous subsequence is from S(3) to S(6).
    - **Thus, the contiguous subsequence of S is (11, -4, 41, 11), with a sum of 59.**

Another way to prove this is to look at some possible cases.

If every value in S is positive, then the maximum-sum contiguous subsequence is the same as S.

- For each iteration in S, we will see if S(i) + P(i-1) is greater than or less than S(i). If all values are positive, then S(i) + P(i-1) will always be greater than S(i).
- This indicates that through each iteration, each value in S will be added to the previous sum P(i-1) and added at P(i).
- Therefore, each of these values in S will be included in the maximum-sum contiguous subsequence.
- The sum of this subsequence will be equal to the sum of all values in S.

If every value in S is negative, then the maximum-sum contiguous subsequence is the highest value in S.

- For each iteration in S, there will be varying results for whether S(i) + P(i-1) is greater than S(i) or not.

- Because we want the contiguous subsequence with the highest value, the maximum-sum contiguous subsequence will never be longer than one value. Adding one negative to another negative will only make it more negative, further reducing the maximum-sum.
- Therefore, to find the maximum-sum contiguous subsequence, all that needs to be found is the maximum number in S. This will be the subsequence.

Time complexity:

- The time complexity of the algorithm is O(n). This is because the sequence is traversed n times, n being defined as the numbers included in the original sequence S.
- The other steps in the problem, such as saving the value of the current starting point to a variable, or finding the maximum value in P, have time complexities of O(1) and O(n), respectively.
- So, with the other steps included, the time complexity of the algorithm is still O(n).