Exercise of Algorithms

Problem 1

Let $S = (s_1, s_2, \cdots, s_n)$ be a sequence of n numbers. A "contiguous subsequence" of S is a subsequence $s_i, s_{i+1}, \cdots, s_j$ (for some $1 \leq i \leq j \leq n$), which is made up of consecutive elements of S. (For example, if $S = (6, 16, -29, 11, -4, 41, 11)$, then 16, -29, 11 is a "contiguous subsequence", but 6, 16, 41 is not.) Given a sequence S, our objective is to find a contiguous subsequence whose sum if maximized. (For example, if $S = (6, 16, -29, 11, -4, 41, 11)$, such a contiguous subsequence would be 11, -4, 41, 11, whose sum is 59.)

Our "Maximum-Sum Contiguous Subsequence Problem" is defined as follows:

Input: A sequence of numbers $S = (s_1, s_2, \cdots, s_n)$.

Output: A contiguous subsequence of S whose sum is maximized.

Your task: Design an algorithm of time complexity $O(n)$ for the above problem. (Remember: as we emphasized in class, whenever you design an algorithm, you need to: (1) explain the main idea of the algorithm, (2) present its pseudo-code, (3) prove its correctness, (4) analyze its time complexity. For dynamic programming, (1) and (3) can often be the same.)

(**Hint:** For each $j \in \{1, 2, \cdots, n\}$, consider contiguous subsequences ending exactly at position j.)