1 Main Idea

In the "Maximum Sum Contiguous Subsequence" problem, we are given a sequence S of n numbers, and asked to find the contiguous subsequence with the largest sum. Formally, we want to find the subsequence

$$S[i^* \dots j^*]$$

where $i^*, j^* = \arg\max_{i,j} \sum_{k=i}^{j} S[k]$ for $1 \leq i \leq j \leq n$. To solve this, we can apply dynamic programming, since this problem exhibits optimal substructure and overlapping subproblems. In this algorithm, we use an array to store the maximum sum of a contiguous subsequence ending at each position in the sequence. Specifically, we define the following:

$$\begin{aligned} dp[1] &= S[1] \\ dp[j] &= \max\{dp[j-1] + S[j], S[j]\} \quad \text{for } 2 \leq j \leq n \end{aligned}$$

We then iterate through the sequence, updating the maximum sum and optimal start and end positions of the contiguous subsequence as we go. Finally, we return the contiguous subsequence with the largest sum.

2 Pseudocode

Algorithm 1: Maximum Sum Contiguous Subsequence

Input: A sequence S of n numbers
Output: A contiguous subsequence of S with the largest sum
$dp = [0] * n$;
`// dp is an array of` n `elements initialized to 0`
$dp[1] = S[1]$;
$max = S[1]$;
$i^*, j^*, i = 1$;
for $j = 2$ **to** n **do**
  if $dp[j-1] > 0$ **then**
    $dp[j] = dp[j-1] + S[j]$;
  end
  else
    $dp[j] = S[j]$;
    $i = j$;
  end
  if $dp[j] > max$ **then**
    $max = dp[j]$;
    $i^*, j^* = i, j$;
  end
end
return $S[i^* \dots j^*]$;

3 Proof of Correctness

Proof. □

4 Runtime Analysis