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

$$\arg\max_{i,j} \sum_{k=i}^{j} S[k]$$

where $1 \leq i \leq j \leq n$. To solve this, we can apply dynamic programming, since this problem exhibits optimal substructure and overlapping subproblems. The main idea of this algorithm is keeping track of positive contributions to the sum. First, we initialize variables to keep track of the maximum sum, the current sum, and the indices of the maximum sum subsequence. Then, we loop through the sequence for each ending index $j \in \{1, 2, \ldots, n\}$. On each iteration, we add the current number to the current sum. If the current sum is greater than the maximum sum, we update the maximum sum and the indices of the maximum sum subsequence. If the current sum becomes negative, we reset the current sum to 0 and update the starting index of the subsequence to the next index. At the end of the loop, we use the saved indices to return the maximum sum subsequence.

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
maxSum = −∞;
currentSum = 0;
i_max, j_max, i = 1;
for j = 1 to n do
    currentSum += S[j];
    if currentSum > maxSum then
        maxSum = currentSum;
        i_max = i;
        j_max = j;
    end
    if currentSum < 0 then
        currentSum = 0;
        i = j;
    end
end
return S[i_max : j_max]
```

3 Proof of Correctness

We will prove the correctness of the algorithm by induction.

Proof. Let S be a sequence of n numbers, and let $S[i_{max} : j_{max}]$ be the maximum sum contiguous subsequence of S. We will show that the algorithm returns $S[i_{max} : j_{max}]$.

Base Case: When $n = 1$, the maximum sum contiguous subsequence is the only number in the sequence. The algorithm will return the number, which is correct.

□

4 Runtime Analysis