A picture containing text, font, screenshot, algebra

Description automatically generated

**Problem Formulation**

T[i] = Contiguous Subsequence of Maximum Sum in a1….ai, which ends at ai

**Recurrence**

T[i] = we know that ai is going to be a part of the subsequence, either we add it to the existing subsequence till previous element i.e. T[i-1] + ai   
or  
we take only this element i.e. ai  
so we find the max out of the two i.e. max( (T[i-1] + ai), ai)  
  
therefore,   
T[i] = max((T[i-1] + ai), ai)

**Base case**  
T[0] = 0

**Sample Run**

S = 5, 15, -30, 10, -5, 40, 10  
T = 5, 20, -10, 10, 5, 40, 55  
  
**Return value**

For the maximum sum, we find max(T)   
and for the subsequence, the index of maximum sum is the end of subsequence and we can keep track of the start of subsequence using an array K where,  
K[i] = i , if ai > T[i-1]+ai  
 K[i-1], otherwise   
hence, K[argmax(T)] is the start of subsequence

**PseudoCode**

T[0] = 0  
K[0] = 0

for i = 1->n ----- (1)  
 if (T[i-1] >= 0)  
 T[i] = T[i-1] + S[i]  
 K[i] = K[i-1]  
 else  
 T[i] = S[i]  
 K[i] = i

maxsum = max(T) ---- (2)

end = argmax(T) ---- (3)  
start = K[argmax(T)] ---- (4)  
  
maxsumsub = S[start…end] ---- (5)  
  
return maxsumsub, maxsum

**Runtime Complexity**

1 takes O(n) time, 2 takes O(n) time, 3 takes O(n) time, 4 and 5 take O(1) time  
Hence total runtime complexity = O(n)