CS314: Algorithms
Homework 2

1. Suppose you are given an array $A[1..n]$ of integers. Describe and analyze an algorithm that finds the largest sum of elements in a contiguous subarray $A[i..j]$. For example, if the array contains $(-6, 12, -7, 0, 14, -7, 5)$, the largest sum of contiguous entries is $19 = 12 - 7 + 0 + 14$.

2. We define a *subsequence* as anything that can be obtained from a sequence (or list of things) by extracting a subset of the elements but keeping them in the same order. For example, the strings C, YAIOAI, and DYNAMICPROGRAMMING are all subsequences of the string DYNAMICPROGRAMMING.

   Let $A[1..m]$ and $B[1..n]$ be to arbitrary arrays. A *common subsequence* of A and B is another sequence that is a subsequence of both A and B. Describe an efficient algorithm to compute the length of the *longest* common subsequence of A and B.

3. Consider a graph with n vertices. Recall that a subset of the vertices is called *independent* if no two of them are joined by an edge. Finding large independent sets is difficult in general, as we discussed in class, but can be done on some simple classes of graphs.

   Call a graph a *path* if its vertices can be written as $v_1, v_2, \ldots, v_n$ with an edge between each v_i and v_{i+1} (but no other edges). With each vertex v_i, we associate a weight w_i.

   Our goal in this problem is to find the largest weight independent set. (Note that this is different from the largest independent set, since here we take the weights into account!)

   (a) Construct an example showing why the following simple *greedy* algorithm does NOT always work.

   ```
   S ← ∅
   While G is not empty:
     Pick a node v_i of maximum weight
     Add v_i to S
     Delete v_i and its neighbors from G
   Return S
   ```

   (b) Construct an example showing why the following different simple *greedy* algorithm does NOT always work.

   ```
   S_1 ← {v_i with i odd}
   S_2 ← {v_i with i even}
   oddsum ← sum of all weights in S_1
   evensum ← sum of all weights in S_2
   if evensum > oddsum
     return S_2
   else
     return S_1
   ```

   (c) Give an algorithm that takes an n-vertex path G with weights and returns an independent set of maximum total weight. Your running time should be polynomial in n.