Due: 25th of September 2017 at 11:59pm

COMP 2007 – Assignment 3

All submitted work must be done individually without consulting someone else's solutions in accordance with the University's Academic Dishonesty and Plagiarism policies.

IMPORTANT! Questions 1(a–c) and 2(a–c) should be submitted via Blackboard as a PDF (no handwriting!). The implementation required for Question 2d should be done in Ed, and submitted via Ed.

Questions

1. **[25 points]** Consider a sequence of houses H arranged in a straight line, where all houses have an associated positive value $w : H \to \mathbb{R}^+$ (Fig 1).

   Design an algorithm that chooses the subset of houses with maximum combined value, given the constraint that no two chosen houses can be adjacent. That is, you can't choose both a house and its neighbour.

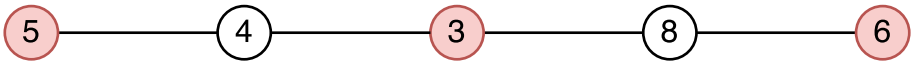

Figure 1: Example input where $n = 5$. The optimum choice of houses is the first, third and fifth house. The value of this selection is $5 + 3 + 6 = 14$.

   (a) Design a dynamic programming algorithm to determine the value of the optimal choice of houses. First derive and justify a recurrence relation and your base cases, then turn it into a bottom-up solution. A more efficient algorithm gives more points. [15 points]

   (b) Argue the correctness of your algorithm. [5 points]

   (c) Prove an upper bound on the time complexity of your algorithm. [5 points]

2. **[75 points]** Now let's examine the same problem on a tree. Formally, let $T = (V, E)$ be a full[1] binary tree, where each node represents a house. All houses have an associated positive value $w : V \to \mathbb{R}^+$. Your task is to find the subset $S \subset V$ that maximises the sum of node values, given that no two nodes in S can be connected by an edge.

   (a) Design a dynamic programming algorithm to determine the value of the optimal choice of houses. First derive and justify a recurrence relation and your base cases, then turn it into a bottom-up olution. A more efficient algorithm gives more points. [35 points]

   (b) Argue the correctness of your algorithm. [10 points]

   (c) Prove an upper bound on the time complexity of your algorithm. [10 points]

   (d) Implement your algorithm (in Ed) and test it on the given input instances. Each instance is given in a text file using the following format (where n is the number of houses):

[1] A full binary tree is one where every node other than the leaves has two children.

```
n
w_0 w_1 w_2 ... w_n
```

Here w_i corresponds to the value for house i. Nodes in the tree are listed from top-to-bottom then left-to-right (Fig 2). Output the *value* of the optimal solution to `stdout` [20 points].

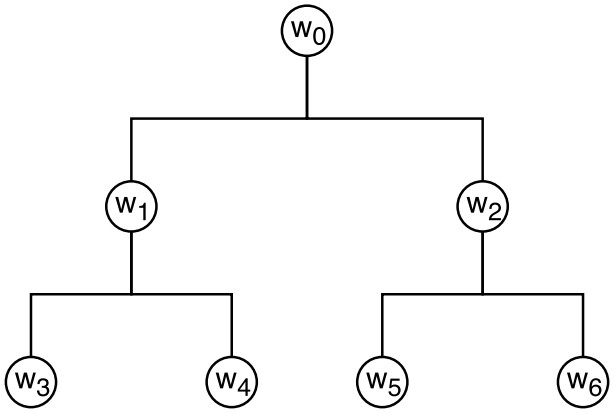

Figure 2: Node labelling scheme for Q2. Nodes are labelled from top-to-bottom then left-to-right. Under this scheme, a node indexed at i has a left and right child indexed at $2i + \{1, 2\}$ respectively. For more information, see `https://en.wikipedia.org/wiki/Binary_tree#Arrays`.