# COMP3010 Algorithm Theory Assignment 1

Chaz Lambrechtsen

45426317

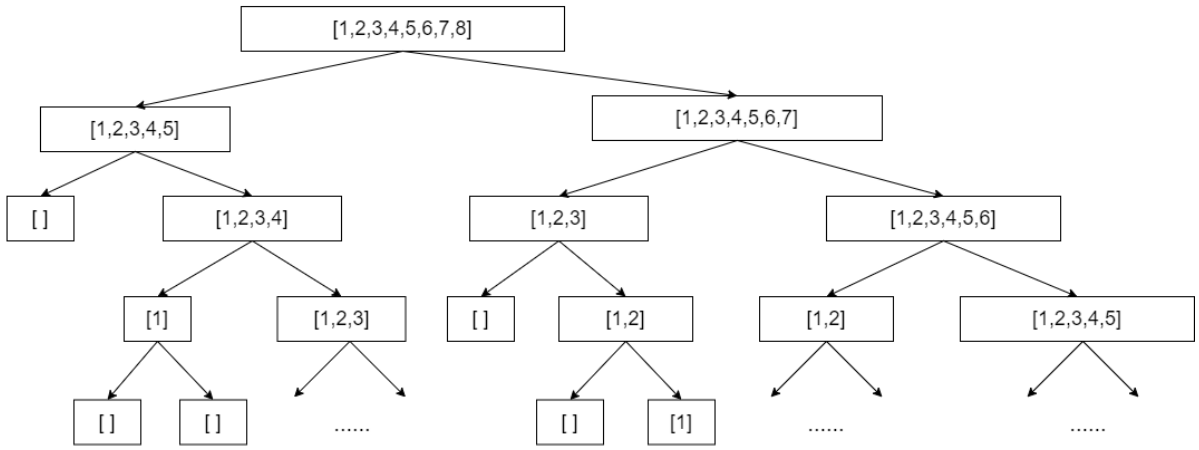
## Contents



## 1. Decomposing the problem into subproblems

The implemented dynamic programming is like divide-and-conquer however this solution solves the problem by combining solutions to sub-problems. This solution will save sub-problems to an array then return the largest sub problem.

Dynamic programming sub-problems:

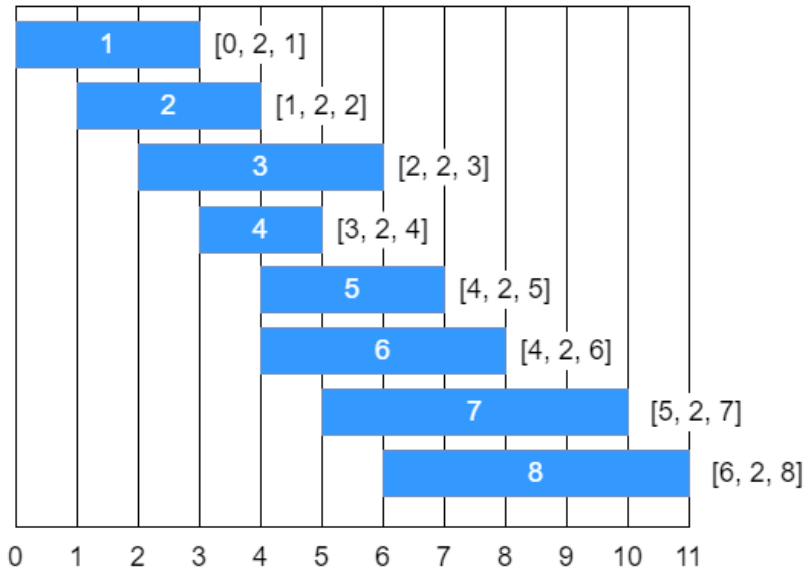



## 2. Overlapping subproblems

Yes, there is cases when there overlapping subproblems while trying to solve a problem by dividing them into several subproblems. For example, in the case of

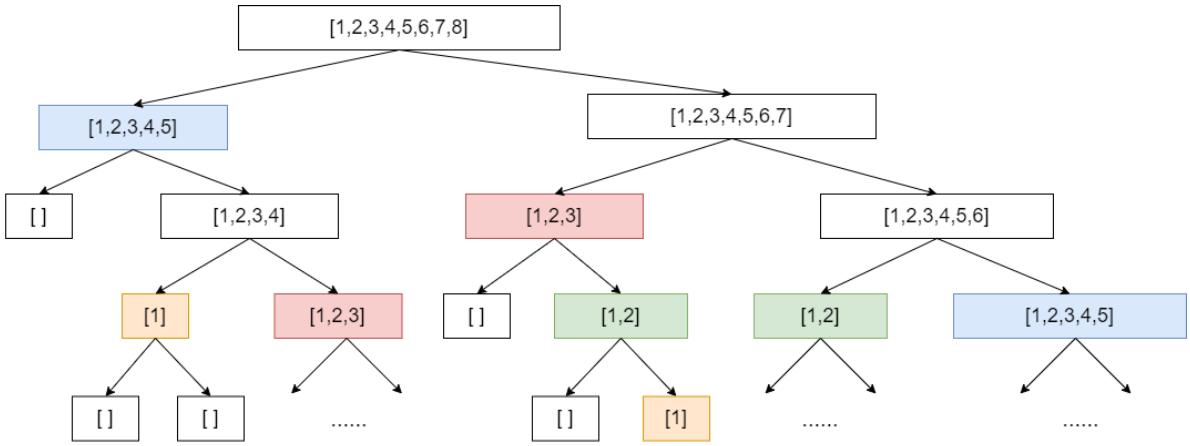
dynamic programming, the results that have been obtained can be stored and used later and hence it is a classic case of overlapping subproblems.

Example Bids: (Bids are stored in the format: [start, end, amount])




Tree of sub-problems for this set of bids:

In this example overlapping sub-problems are highlighted with colours blue, orange, red, green




## 3. Optimal substructure

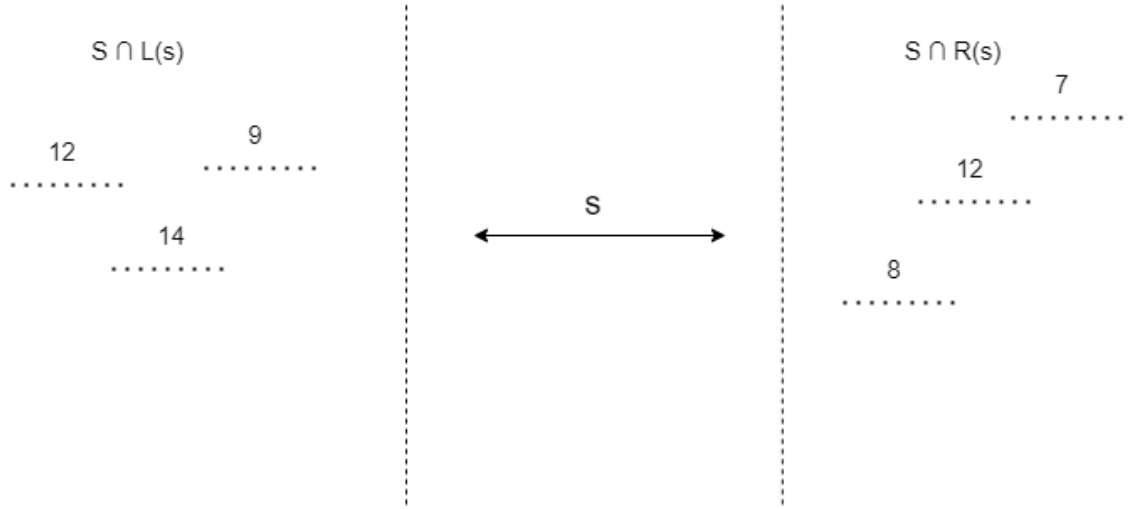
In a set of bids B is non-empty, an optimal substructure $S \subseteq B$ must contain some bid because the bid b is positive. So, consider some $s \in S$ (any). Then all the bids in S are either left of s or right of s because bids of S do not overlap. So:

$L(s) = \{a \in B : a \cap s = \emptyset \text{ and a is left of s}\}$

$R(s) = \{a \in B : a \cap s = \emptyset \text{ and a is right of s}\}$

Hence usage of binary search to solve this problem.

Then S ∩ L(s), S ∩ R(s), and {s} partition S.




So, S ∩ L(s) is an optimal solution the problem restricted to L(s) which contains the optimal sub structure because of the recursive relation in 4.

## 4. Recursive relation that gives the optimal solution

In this problem, we want to find the maximum-bid subset of nonoverlapping lots, given a set L of lots that have bids associated with them. Lot i ∈ L has a start time $s_i$, an end time $e_i$, and a bid $b_i$. We seek to find an optimal subset O of non-overlapping lots in L with the maximum possible sum of bids. Formally represented as:

$$O = \max \mathrm{O} \subseteq \mathrm{L}; \forall \mathrm{i}, \mathrm{j} \in \mathrm{O}, \text{either } e_i \leq s_j \text{ or } e_j \leq s_i \sum_{i \in O} b_i$$

## 5. Derive the solution that leads to the optimal solution

1. Sort bids in ascending order of starting time.

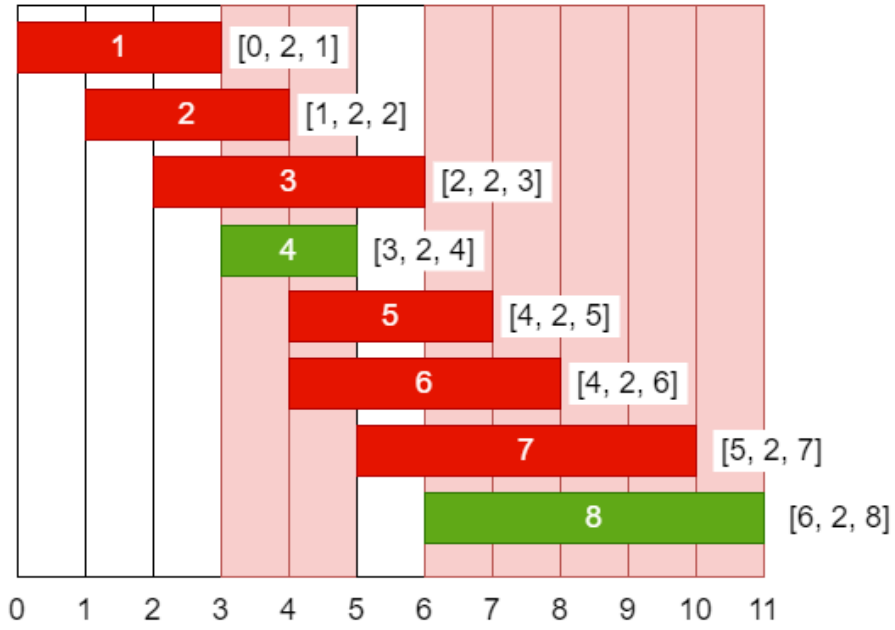
2. For 1 ≤ i ≤ n, find the smallest j > i such that bid j doesn't overlap with bid i.

3. For i = n up to size, compute value(i) using formula from 4. and store it in sub-problems array.

4. The bids with optimum yield value is value(size - 1).

Because we want to recover the optimal solution from the array, we add the values of value(i) as well as the option that led to it to a sub-problems array.

For example, applying these steps to the previous example we get the following result:


1
[0, 2, 1]
2
[1, 2, 2]
3
[2, 2, 3]
4
[3, 2, 4]
5
[4, 2, 5]
6
[4, 2, 6]
7
[5, 2, 7]
8
[6, 2, 8]
0 1 2 3 4 5 6 7 8 9 10 11