COMP 3010 Assignment 2

Ameer Karas, 44948956

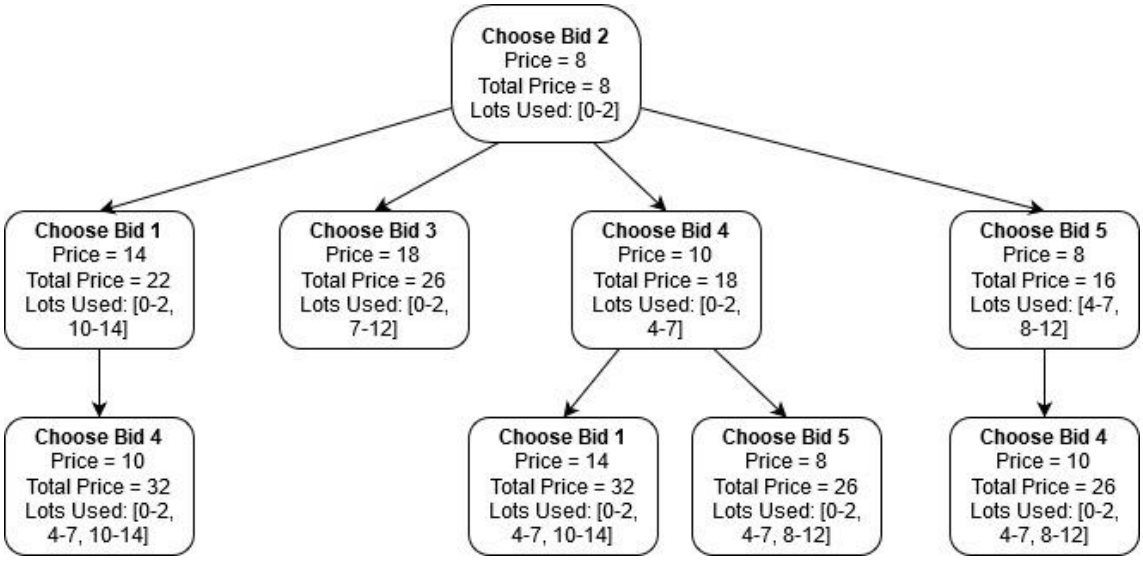
The diagrams and examples used will use the following example from the assignment 2 specification:

```
15 6
 0 0 5 13
 1 10 14 14
 2 0 2 8
 3 7 12 18
 4 4 7 10
 5 8 12 8
```

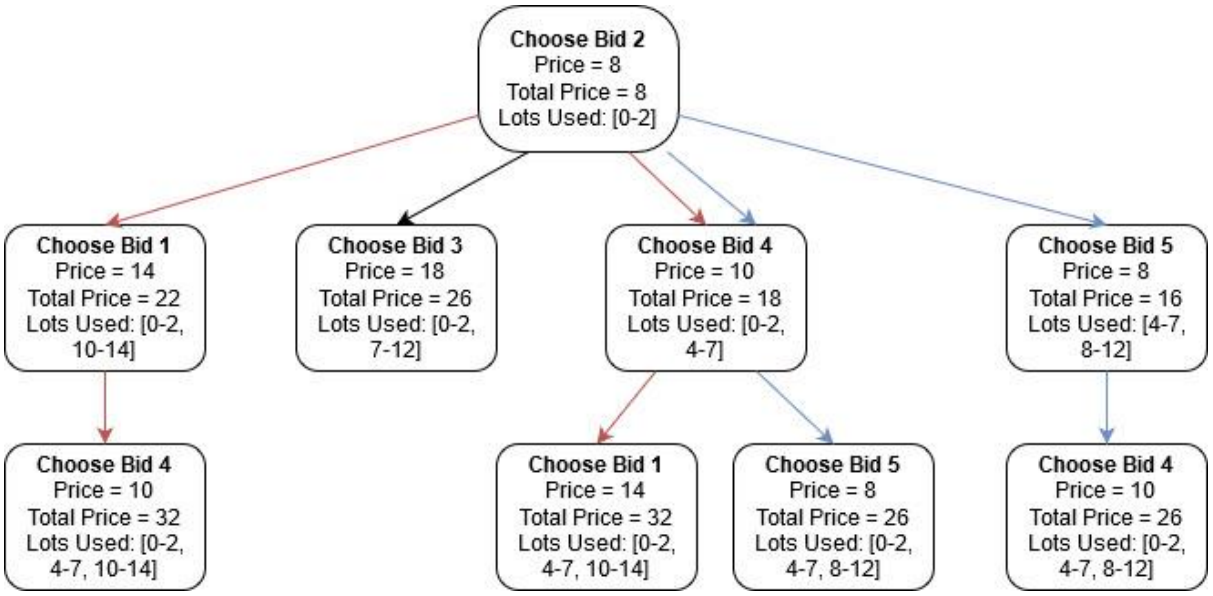
1. Show how you can decompose the problem into subproblems.

After sorting the array of bids in increasing order of end time, we have: 2, 0, 4, 3, 5, 1.

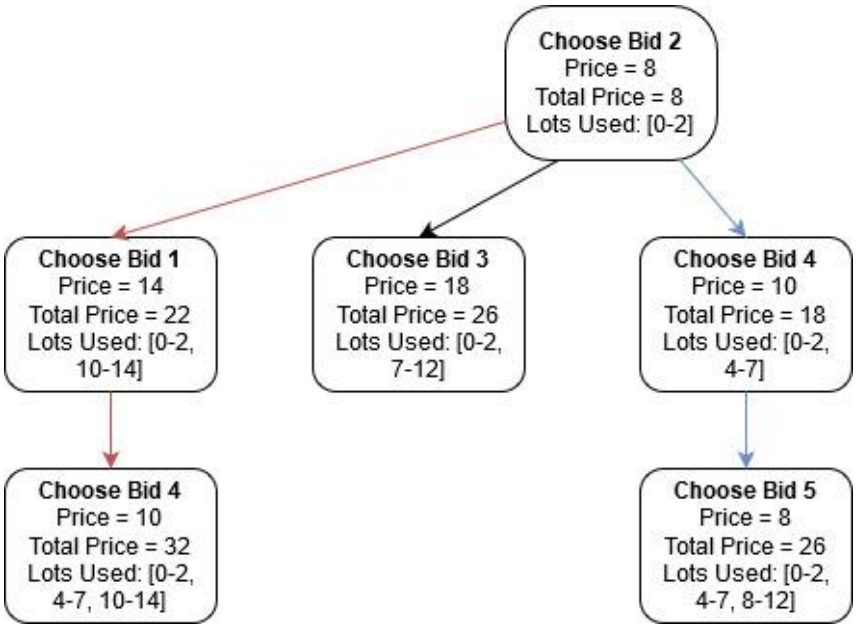
We can derive into subproblems by observing compatible bids for the example from the assignment 2 specification, starting from bid 2:



2. Show that there exists overlapping subproblems (you can use an example for this).

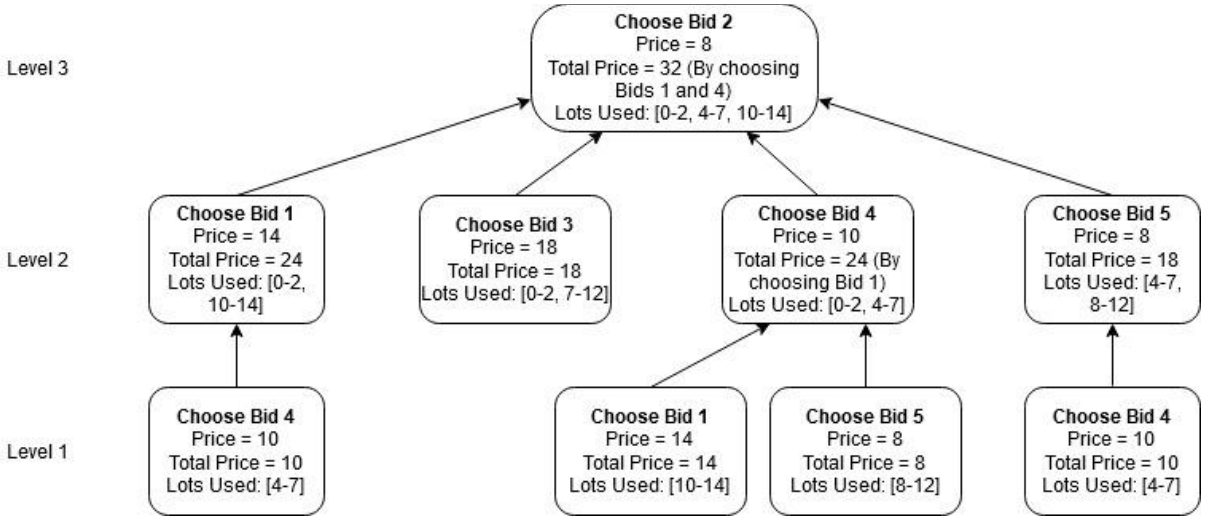


Using the example from part 1, we can observe that there are common calculations in the sub-trees Choose Bid 2 → Choose Bid 1 → Choose Bid 4 and Choose Bid 2 → Choose Bid 4 → Choose Bid 1, as well as Choose Bid 2 → Choose Bid 4 → Choose Bid 5 and Choose Bid 2 → Choose Bid 5 → Choose Bid 4. As we can see that there is an overlap of subproblems, we can now trim parts of the tree to better understand how dynamic programming will minimise the amount of calculations by storing common data.



3. Show that the problem exhibits optimal substructure.

A problem exhibits optimal structure if the optimal solution to the problem can be constructed using optimal solutions to subproblems. Using the tree from part 1 we can show optimal substructure by choosing the best option at each level of the tree.



Assume that we are completing the tree in a depth-first search manner. Going down the leftmost subtree from the top node (Choose Bid 2), we can see that the only solution to the subproblem 'Choose Bid 1' is to choose Bid 4. This results in the leftmost branch returning to the root node with a value of 32 for price.

Similarly, the 'Choose Bid 3' subtree has only one choice at each level. This results in this subtree returning to the root node with a value of 26, which is less than our previously found value of 32, so we omit it.

Moving right, we come to the first subtree with more than one choice at level 1. Here we have the options of 'Choose Bid 1' for a price of 14 or 'Choose Bid 5' for a value of 8. As 'Choose Bid 1' has the greater value, we select it as our optimal solution and move up to 'Choose Bid 4', and return to the root node with the formerly found value of 32.

The final and rightmost subtree has only one subproblem per level of the tree, and will return to the root node with a value of 26.

By selecting the optimal choice of the returned subproblems, we can conclude that the maximal value of 32 has been returned by completing the subproblems Choose Bid 4 → Choose Bid 1 → Choose Bid 2 and Choose Bid 1 → Choose Bid 4 → Choose Bid 2.

4. Show the recursive relation that gives the optimal solution, that is, the maximum income that the company can get.

Assume:

- n = bid id
- n_p = bid price
- L(n) = a function that finds the last-ending bid *j* such that:
  - n and j do not overlap
  - j < n
  - The closest start to j.end is n.start
- maxPrice(n) = maximum price possible by selecting bids from the first n intervals.

$$maxPrice(n) = \begin{cases} \mathrm{n_p} & \text{if } \mathrm{n} = 0 \\ \max(\mathrm{n_p} + \mathrm{maxPrice(L(n))},\ \mathrm{maxPrice(n\text{-}1)}) & \text{if } \mathrm{n} > 0 \end{cases}$$

We can check that the recurrence relation holds true. We shall once again use the example values from the previous parts.

n	**0**	**1**	**2**	**3**	**4**	**5**
n_p	$13	$14	$8	$18	$10	$8
L_n	-1	4	-1	0	2	4

As we are sorting the bids in increasing order of their end times, we can extrapolate the data from the above table into a table that determines the maximum possible price.

The new order of the bids is: 2, 0, 4, 3, 5, 1

n	**2**	**0**	**4**	**3**	**5**	**1**
maxPrice(n)	$8	$13	$18	$31	$31	$32

This returns the maximum result $32, which we know to be correct.

5. Show how you can derive the solution that leads to the optimal solution, that is, how to identify the bids that lead to the maximum income.

Assume for this part that the function L(n):

- finds the last-ending bid *j* such that:
  - n and j do not overlap
  - j < n
  - The closest start to j.end is n.start

1. Assume that the current index is j
   a. it will choose the price of index j + 1
2. For any other compatible index:
   a. the addition of the current bid's price value (assume current bid is j) and the price value of L(j) will be calculated; assume this value is x
   b. This value x will be compared with the previous index's maximum price
3. The maximum potential price will be selected after completing all index's and comparing their values.

Using the example from the assignment specification, we sort our list of bids in order of increasing end times (2, 0, 4, 3, 5, 1). We have:
maxPrice(2) = $8
maxPrice(0) = max($13 + $0, $8) = $13
maxPrice(4) = max($10 + $8, $13) = $18
maxPrice(3) = max($18 + maxPrice(2), maxPrice(4)) = $31
maxPrice(5) = max($8 + maxPrice(4), maxPrice(3)) = $31
maxPrice(1) = max($14 + maxPrice(4), maxPrice(5)) = $32

Therefore, we can see that we achieve our maximum price for the list of bids by using Bid 1 ($14), Bid 2 ($8) and Bid 4 ($10).