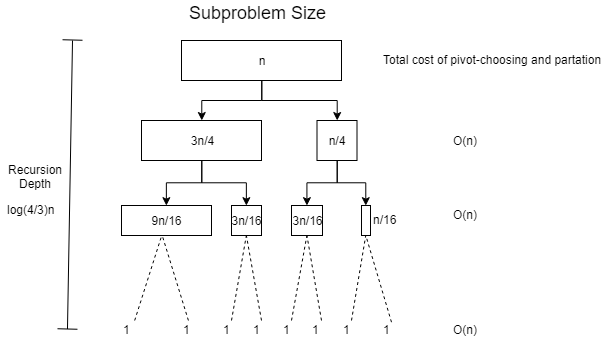
**Question 4**

a) The cost of quicksort can be decomposed into three 4 parts as the following recurrence relation shows

The cost of choosing a pivot is content time O(1). The partition based on the pivot required one comparison between each element and the pivot. Thus, its time complexity is O(n). The recurrence relation can be simplified to

The time complexity of quicksort is related to the depth of recursion.

Given that the oracle algorithm can choose a pivot that is guaranteed to lie in between top 25% and top 75%, the worst time complexity occurs when pivot is either at the 25% or the 75% of the elements. In this case, the problem is divided into two subproblem of size and of the original problem. Thus, the recursion depth is .



The cost of pivot-choosing and partition at each depth is O(n).

Hence, the total cost of quicksort using the oracle algorithm in the worst case is

If we convert log (4/3) to log2, we get

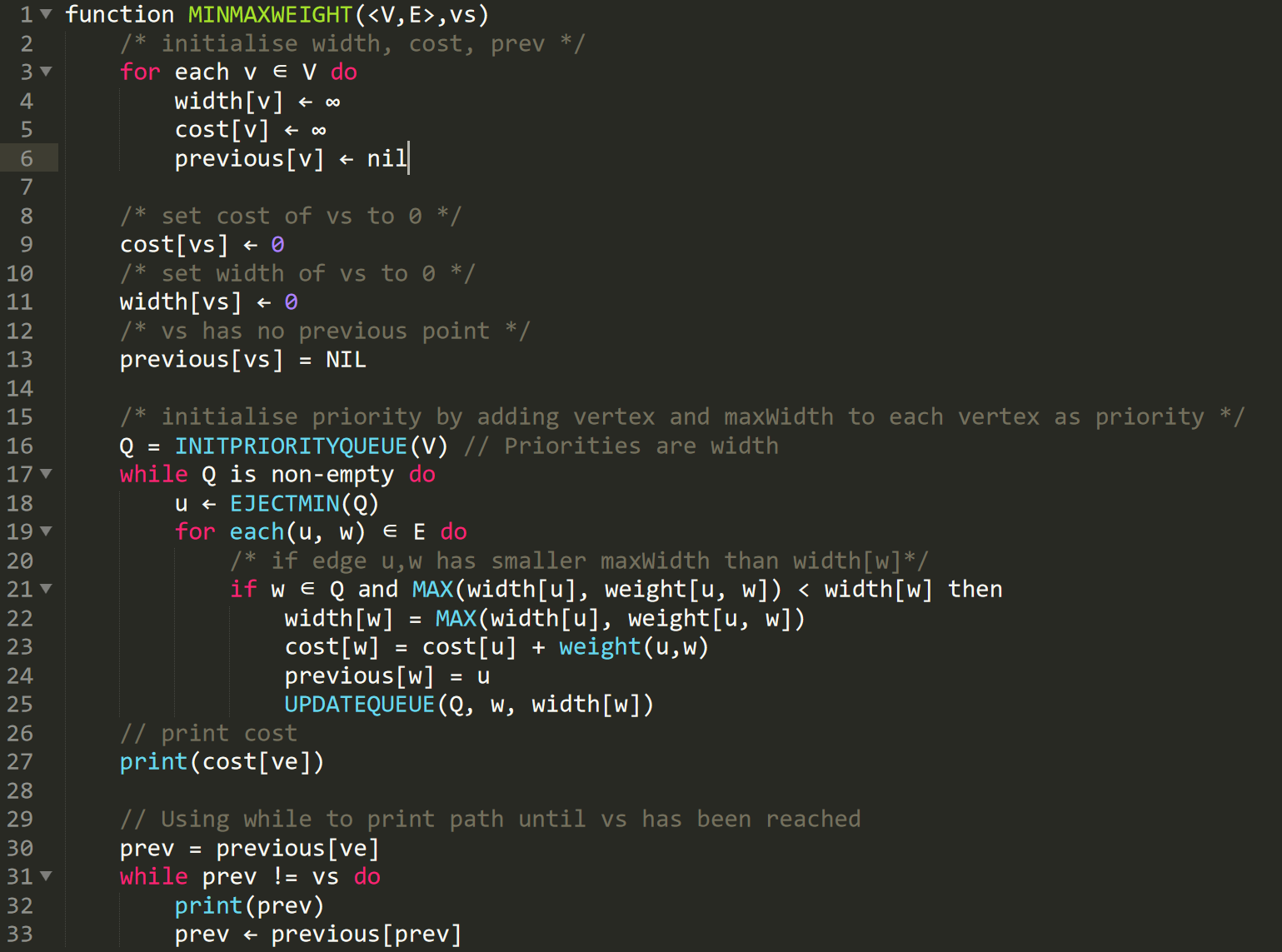
Thus, Quicksort does not perform worse than using the oracle as a pivot selection strategy.

b) The probability that a randomly selected element of the array falls in between the top

25% and top 75% is 0.5.

**Question 5**

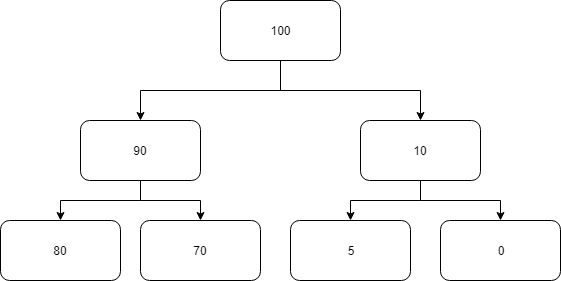
The following screenshot shows the pseudocode to find MinMaxWeightPath. It adapted the idea of greedy algorithm from Dijkstra’s algorithm. Comparing to Dijkstra’s algorithm to find shortest path, the criteria of selecting the next point to visit is max weight along the path from vs. Its worst case time complexity with heap as priorityqueue and Adjacency List is

****

**Question 7**

**Question 8**

Heap property states that each parent is not smaller than its children. It doesn’t state that each child must be smaller than parent’s “brothers”. In other word, heap property doesn’t guarantee that elements on k layer are greater than all elements on k+1 layer.



As the above heap demonstrates, top-3 elements (100, 90, 80) are not in the first 2 layers.