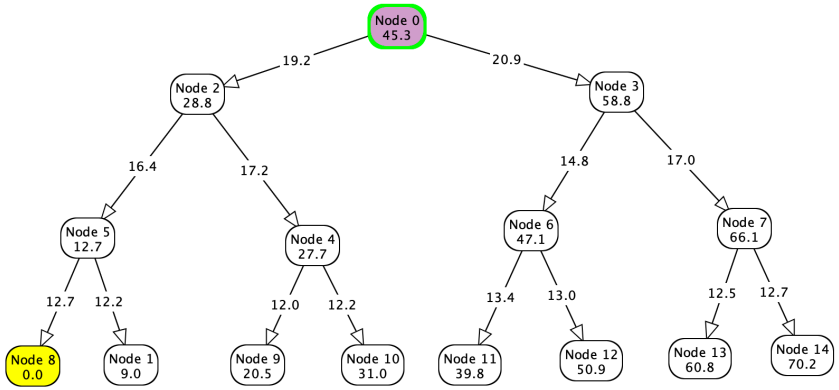
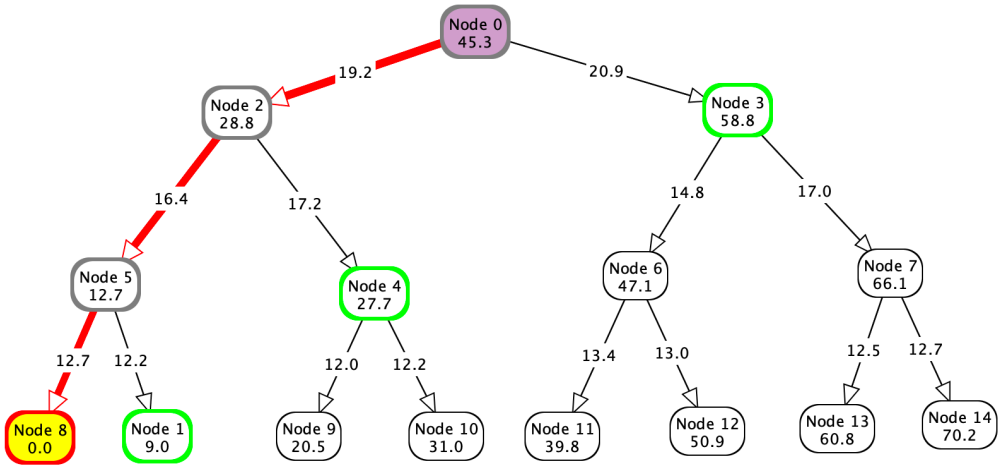
1 Question One

(a) Give a graph where depth-first search (DFS) is much more efficient (expands fewer nodes) than breadth-first search (BFS).

A graph where DFS is more efficient than BFS is:

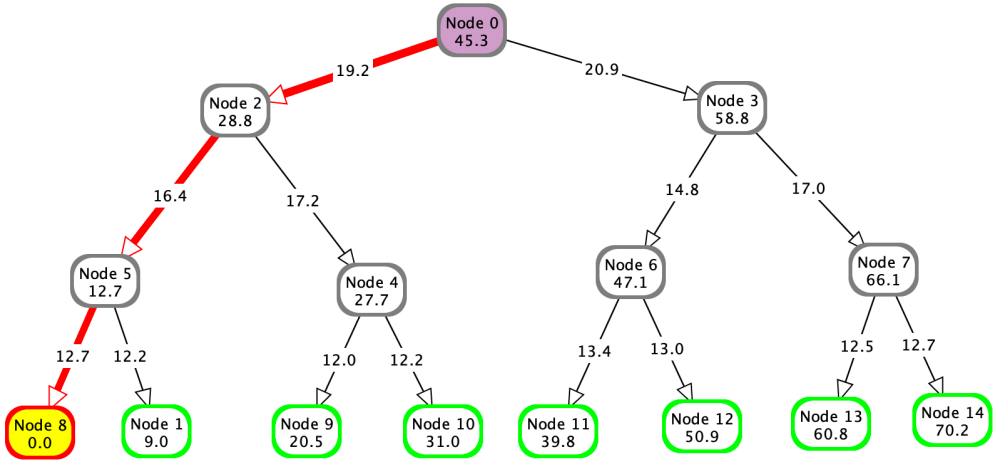


Using depth first search on this tree:



The path is Node 0 → Node2 → Node 5 → Node 8 (Goal) with the path cost being 48.3 and the number of nodes expanded being 4

Using breadth first search on the same tree:

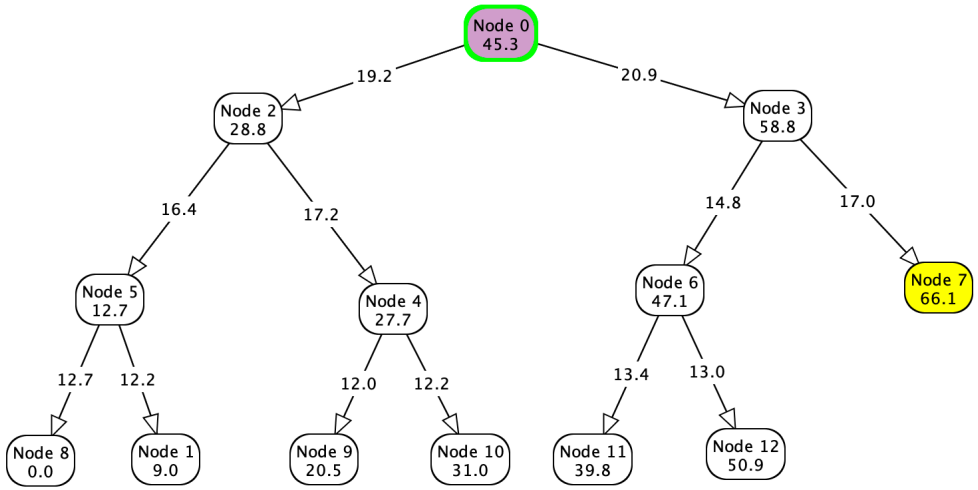


The path is Node 0 → Node2 → Node 5 → Node 8 (Goal) with the path cost being 48.3 and the number of nodes expanded being 8

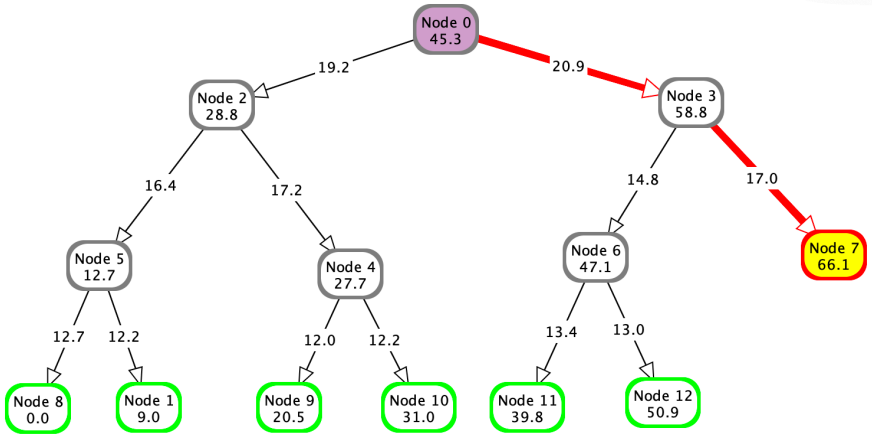
As we can see, the number of nodes expanded is more for BFS, and hence for this example DFS is more efficient than BFS.

(b) Give a graph where BFS is much better than DFS. [15 marks]

A graph where BFS is more efficient than DFS is:

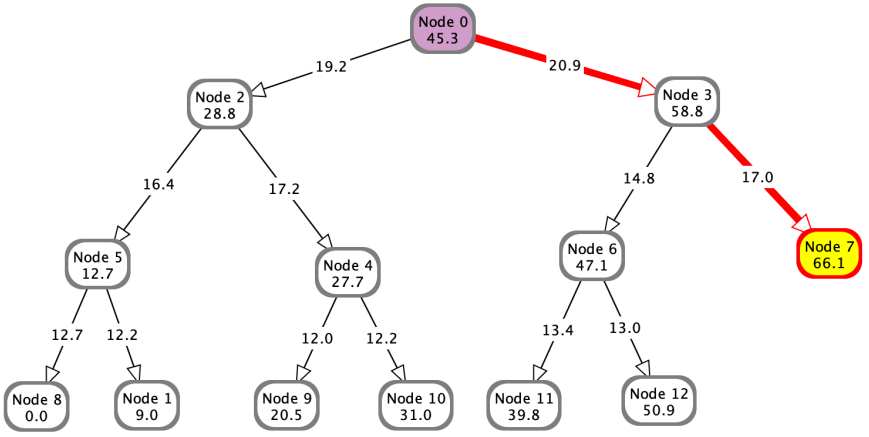


Using BFS on this tree:



The path is Node 0 → Node 3→ Node 7 (Goal) with the path cost being 37.9 and the number of nodes expanded being 7

Using DFS on the same tree:

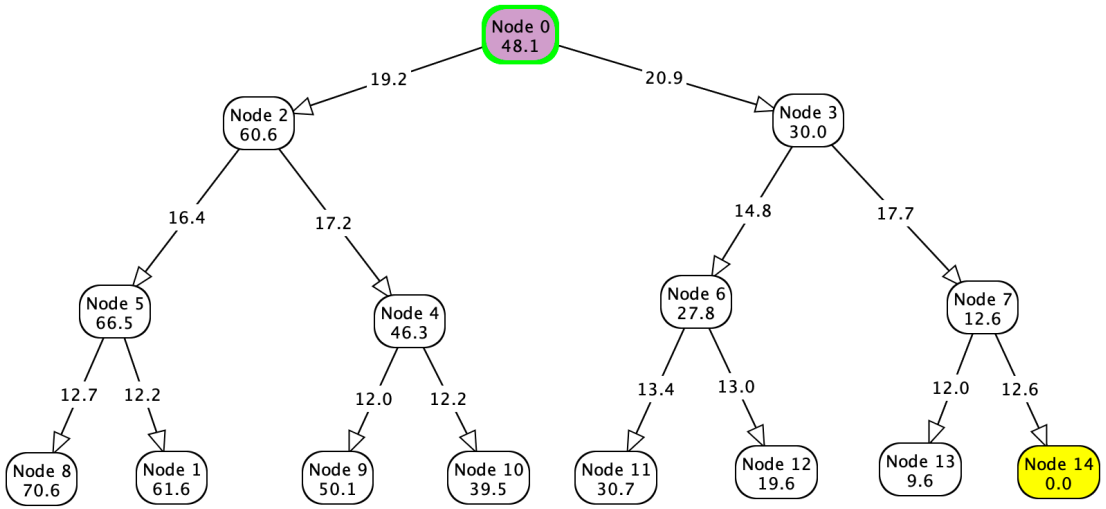


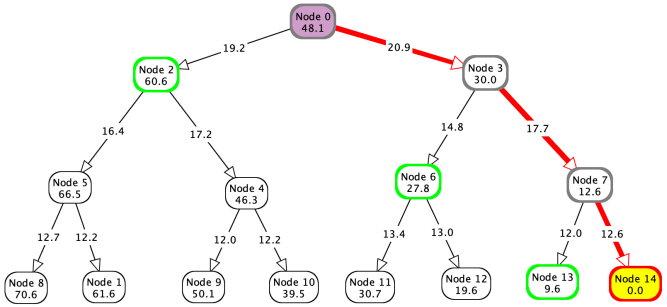
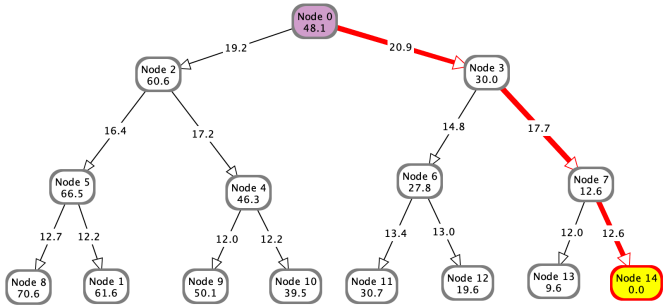
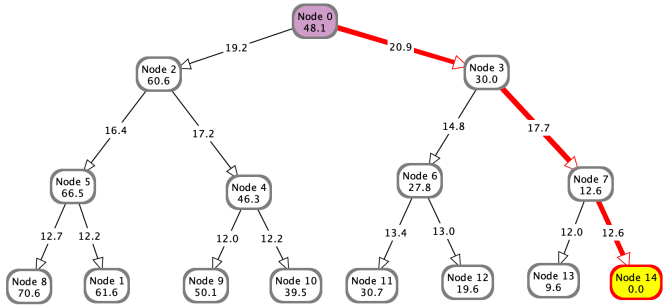
The path is Node 0 → Node3 → Node 7 (Goal) with the path cost being 37.9 and the number of nodes expanded being 13

As we can see, the number of nodes expanded is more for DFS, and hence for this example BFS is more efficient than DFS.

(c) **Give a graph where A* search is more efficient than either DFS or BFS. [15 marks]**

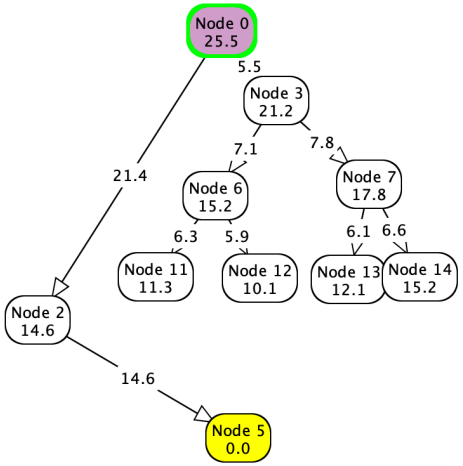
A graph where A* search is more efficient than DFS or BFS is:

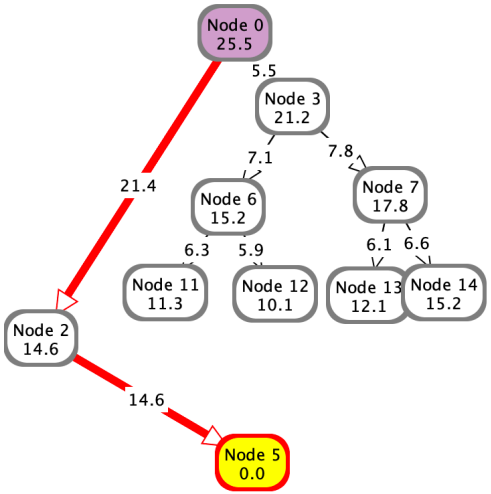
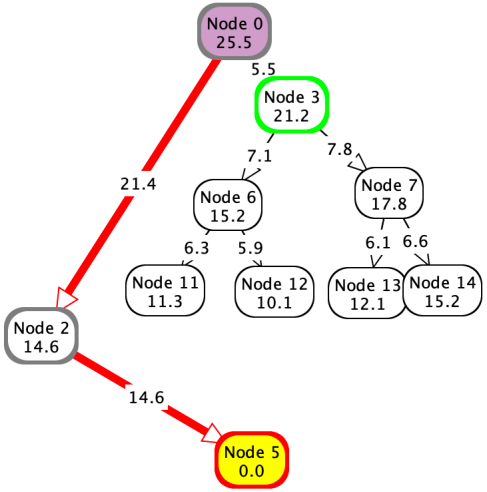
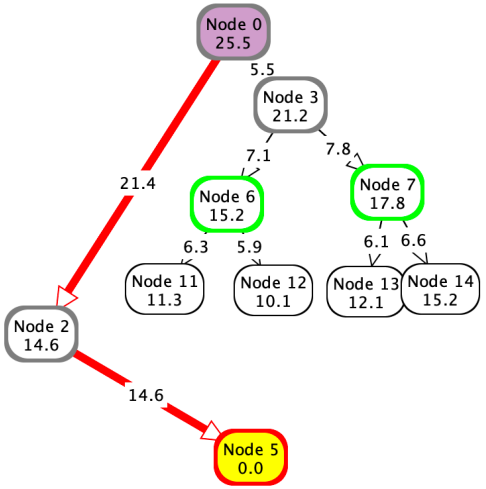


Using A* on this tree:	Using DFS on the same tree:	Using BFS on the same tree:
		
The path is Node 0 → Node 3→ Node 7 → Node 14 (Goal) with the path cost being 51.2 and the number of nodes expanded being 4	The path is Node 0 → Node 3→ Node 7 → Node 14 (Goal) with the path cost being 51.2 and the number of nodes expanded being 15	The path is Node 0 → Node 3→ Node 7 → Node 14 (Goal) with the path cost being 51.2 and the number of nodes expanded being 15

As we can see, the number of nodes expanded is more for DFS and BFS, and hence for this example A* is more efficient than DFS or BFS.

(d) Give a graph where DFS and BFS are both more efficient than A* search. [15 marks]



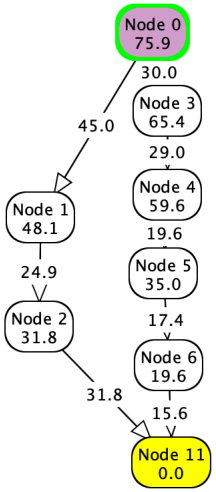
Using A* on this tree:	Using DFS on the same tree:	Using BFS on the same tree:
		
The path is Node 0 → Node 2→ Node 5 (Goal) with the path cost being 36.0 and the number of nodes expanded being 10	The path is Node 0 → Node 2→ Node 5 (Goal) with the path cost being 36.0 and the number of nodes expanded being 3	The path is Node 0 → Node 2→ Node 5 (Goal) with the path cost being 36.0 and the number of nodes expanded being 4

As we can see, the number of nodes expanded is more for A*, and hence for this example BFS or DFS is more efficient than A*.

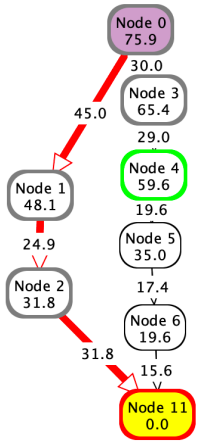
Question 2

a. What is the effect of reducing *h(n)* when *h(n)* is already an underestimate?

The smaller the h(n), the more preferred the node would be during A* search, so if h(n) is reduced further when it is already an underestimate, more nodes will be expanded, given the tree is deep and there are nodes to be expanded before goal state is found.
For example,

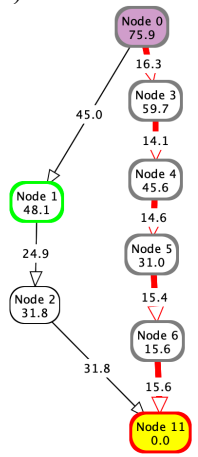


If we use A* search on that, we will get this:



The path is Node 0 → Node 1 → Node 2 → Node 11 (Goal) with the path cost being 101.70 and the number of nodes expanded being 5

However, if we reduce h(n) when it is already an underestimate, we get:

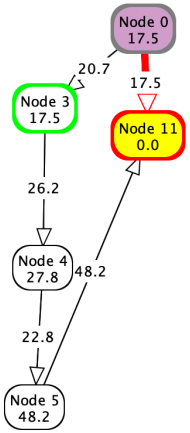


The path is Node 0 → Node 3 → Node 4 → Node 5 → Node 6 → Node 11 (Goal) with the path cost being 76.0 and the number of nodes expanded being 6

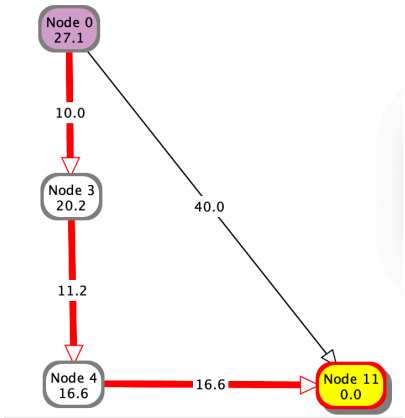
As we see here, more nodes were explored.

b. How does A^* perform when *h(n)* is the exact distance from *n* to a goal?

A*'s performance when h(n) is the exact distance from n to a goal depends on whether that is the optimal is not.
For example,



In this case, only 2 nodes were expanded, with a total path cost of 17.5.
However, if there are other optimal paths, like this:



We can see that it preferred an alternate route instead of going directly to the goal state.

c. What happens if *h(n)* is not an underestimate?

If h(n) is not an underestimate then it might not find an optimal path, because it would not know when it can stop exploring a path, as there can be paths with lower actual cost but higher estimated cost than the best currently known path to the goal.