

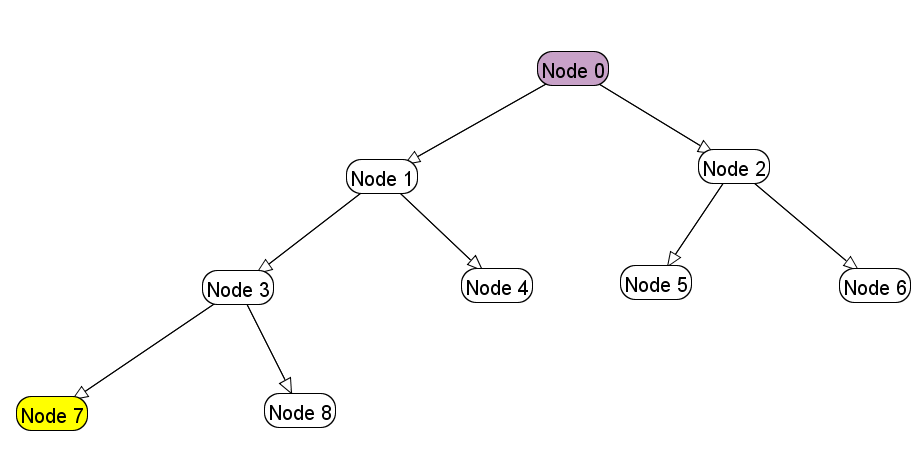
**CZ3005 Artificial Intelligence**

**Assignment 1**

Done by:

U1920095H Claudia Beth Ong

**1a) Give a graph where depth-first search (DFS) is much more efficient (expands fewer nodes) than breadth-first search (BFS). [10 marks]**



Depth-first search

Queue: 0, 1, 3, 7

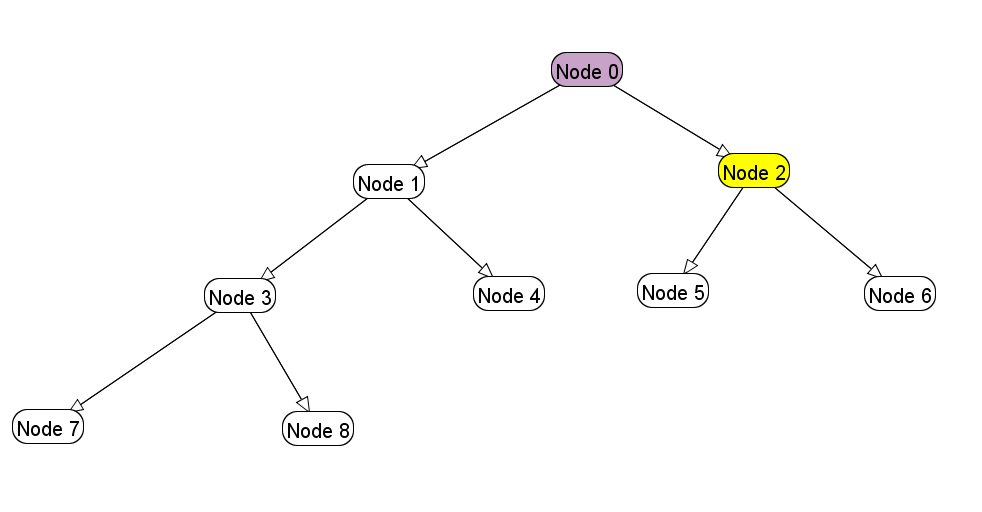
Total nodes expanded: 4

Breadth-first search

Queue: 0, 1, 2, 3, 4, 5, 6, 7

Total nodes expanded: 8

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



Depth-first search

Queue: 0, 1, 3, 7, 8, 4, 2

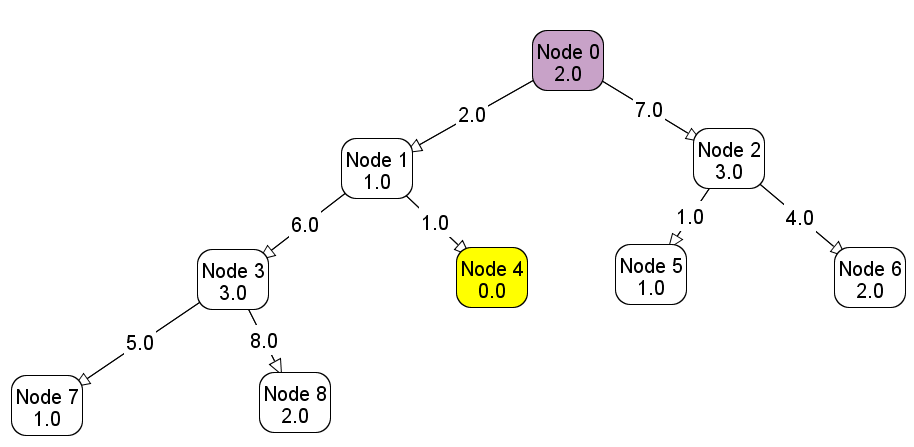
Total nodes expanded: 7

Breadth-first search

Queue: 0, 1, 2

Total nodes expanded: 3

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



A\* Search

Queue: a. 0 (2+0)

b. 1 (1+2), 2 (3+7)

c. 4 (2+1), 2 (3+7), 3 (3+6+2)

d. 2 (3+7), 3 (3+6+2)

Total nodes expanded: 4

Depth-first search

Queue: 0, 1, 3, 7, 8, 4

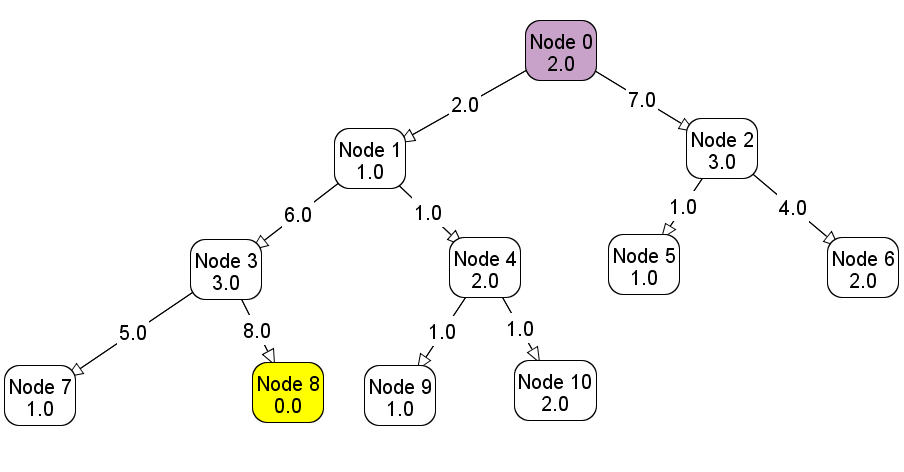
Total nodes expanded: 6

Breadth-first search

Queue: 0, 1, 2, 3, 4

Total nodes expanded: 5

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



A\* Search

Queue: a. 0 (2+0)

b. 1 (1+2), 2 (3+7)

c. 4 (2+1+2), 2 (3+7), 3 (3+6+2)

d. 9 (1+1+1+2), 10(2+1+1+2), 2 (3+7), 3 (3+6+2)

e. 10(2+1+1+2), 2 (3+7), 3 (3+6+2)

f. 2 (3+7), 3 (3+6+2)

g. 5 (1+1+7), 3 (3+6+2), 6 (2+4+7)

h. 3 (3+6+2), 6 (2+4+7)

i. 6 (2+4+7), 7 (1+5+6+2), 8 (2+8+6)

j. 7 (1+5+6+2), 8 (2+8+6)

k. 8 (2+8+6)

l. empty queue

Total nodes expanded: 11

Depth-first search

Queue: 0, 1, 3, 7, 8

Total nodes expanded: 5

Breadth-first search

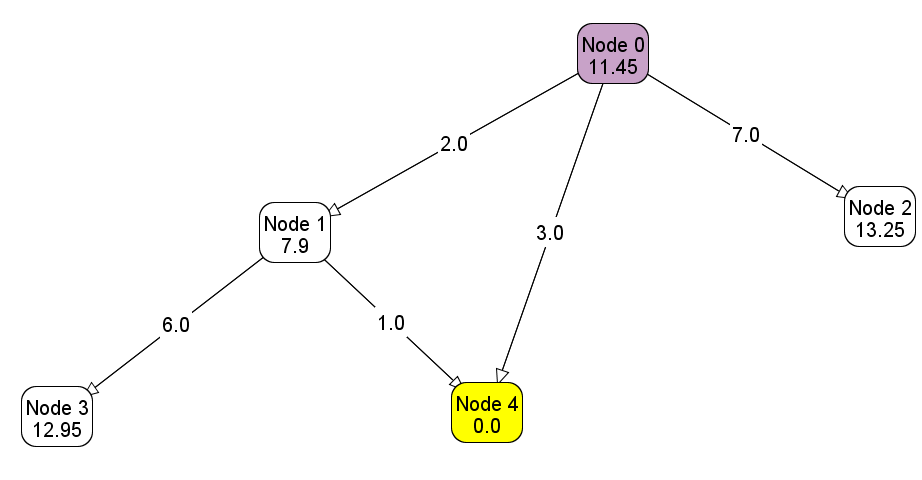
Queue: 0, 1, 2, 3, 4, 5, 6, 7, 8

Total nodes expanded: 9

**2(a) What is the effect of reducing h(n) when h(n) is already an underestimate? [15 marks]**

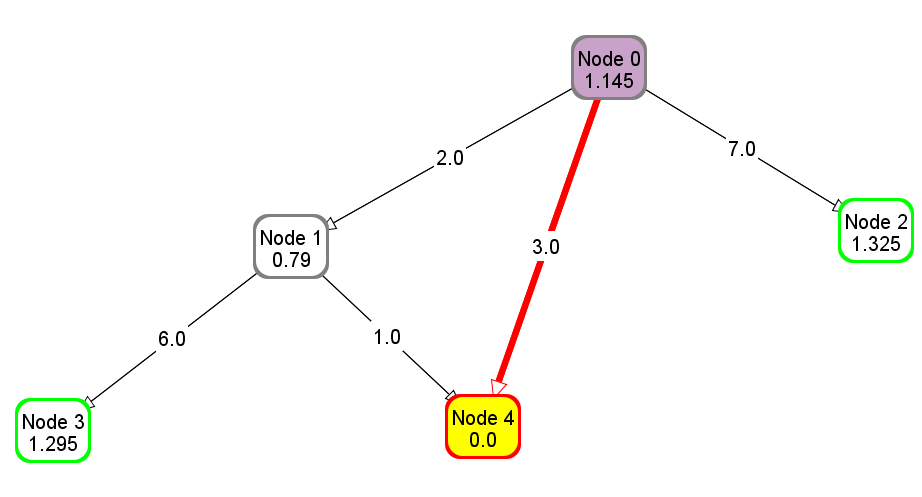
As h(n) decreases, the algorithm increasingly resembles an uninformed uniform cost search. Therefore, as with UCS, the algorithm will definitely return the most optimal path, but may be more inefficient in finding this path.

Original graph:



Nodes expanded: 0, 4

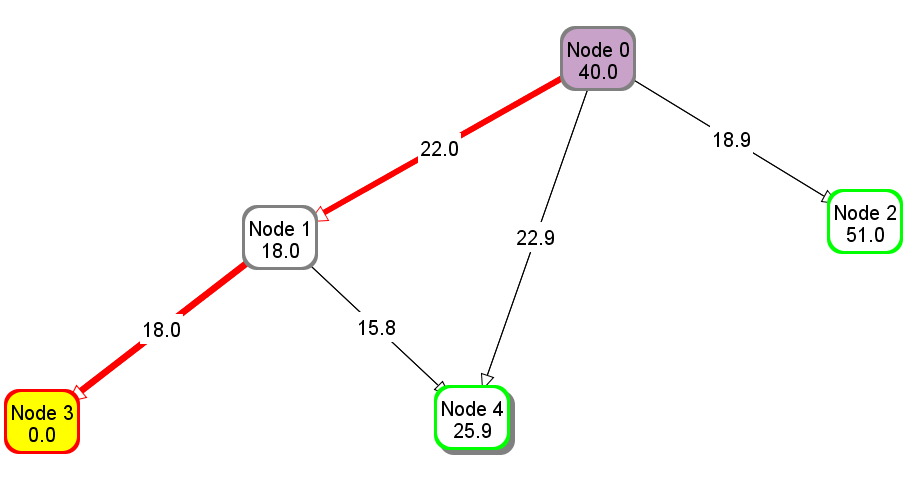
New graph (h2(n) = h1(n)/5):



Nodes expanded: 0, 1, 4

**(b) How does A\* perform when h(n) is the exact distance from n to a goal? [15 marks]**

A\* will perform perfectly if h(n) is the exact distance, as this means that the predicted distance to the goal is the actual distance. Thus the algorithm will follow the shortest path to the goal, and only follow this path, making it extremely fast.



Queue: a. 0 (40.0)

b. 1 (40.0), 2 (69.9)

c. 3 (40.0), 4 (63.7), 2 (69.9)

d. 4 (63.7), 2 (69.9)

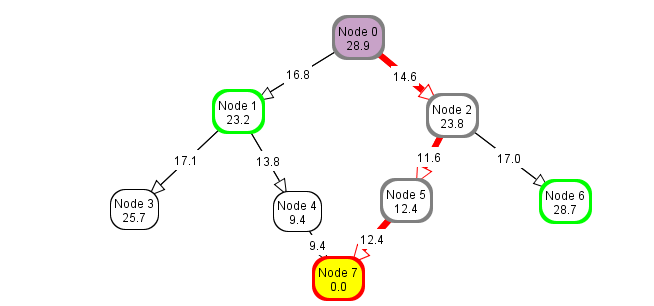
Nodes expanded: 3

**(c) What happens if h(n) is not an underestimate? You can give an example to justify your answer. [15 marks]**

When h(n) is not an underestimate, it means that it is either the exact distance to the goal (in part b), or it is an overestimate.

If h(n) is an overestimate, the estimated distance to the goal is longer than the actual distance. Since the A\* algorithm is essentially g(n) (cost at the previous step) + h(n) (estimated distance to the goal), if h(n) is increased and is an overestimate, the algorithm will start to function more like a greedy best-first search. This results in the algorithm possibly returning a path that is not the optimal path to the goal, since the algorithm places priority on the distance to the goal (ie. the future cost) and gives less weightage to the cost that has already been accumulated.

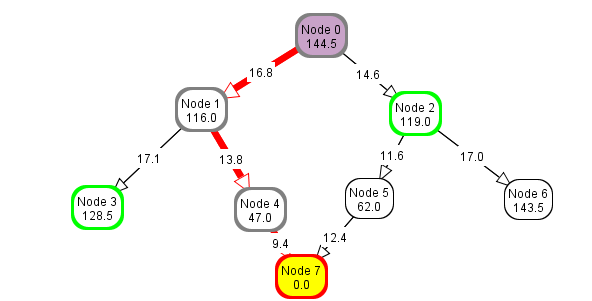
Original graph:



Nodes expanded: 0, 1, 2, 5

Total path cost: 38.6

New graph (h2(n) = h1(n)\*5):



Nodes expanded: 0, 1, 2, 4

Total path cost: 40.0