Nanyang Technological University

School of Computer Science and Engineering

CZ3005 - Lab 1

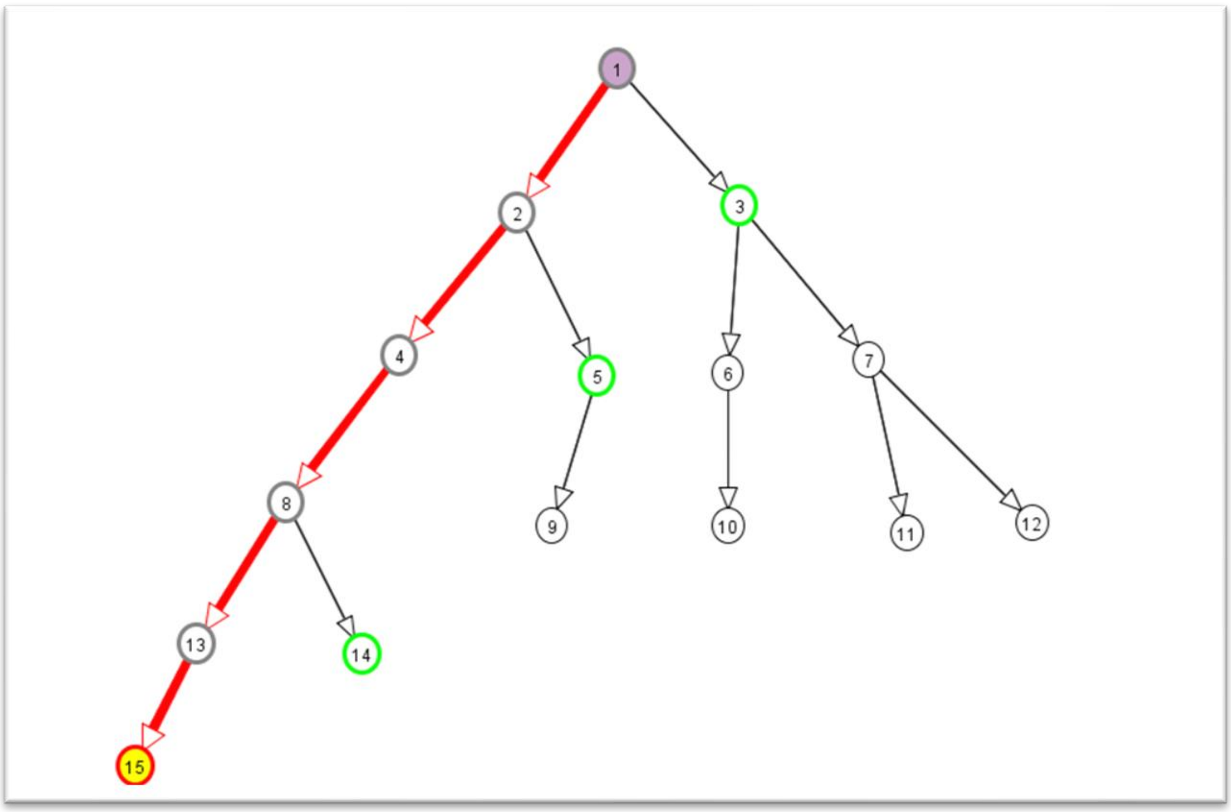
Artificial Intelligence

Phua Jia Sheng

SSP 2

Question 1

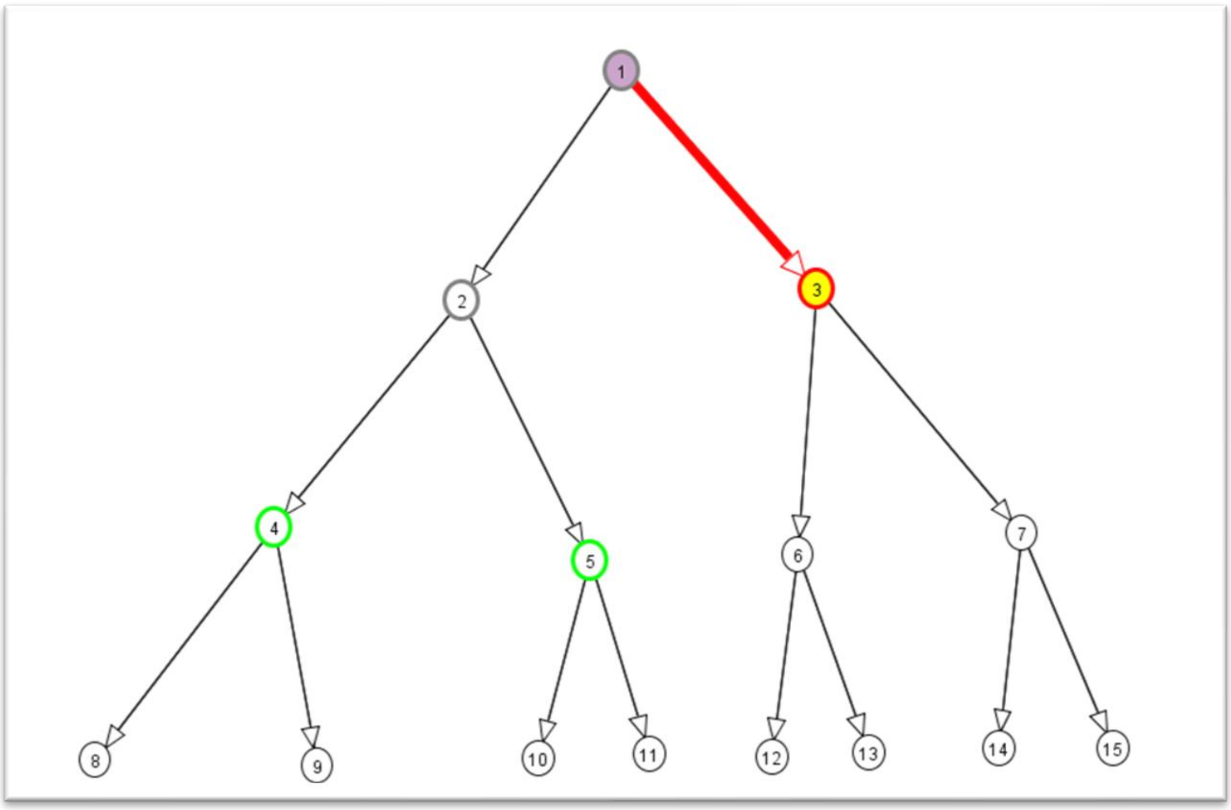
(a) Give a graph where depth-first search (DFS) is much more efficient (expands fewer nodes) than breadth-first search (BFS).



Search Algorithm	Nodes Expanded
DFS	6
BFS	15

Neighbor Ordering: Left to right

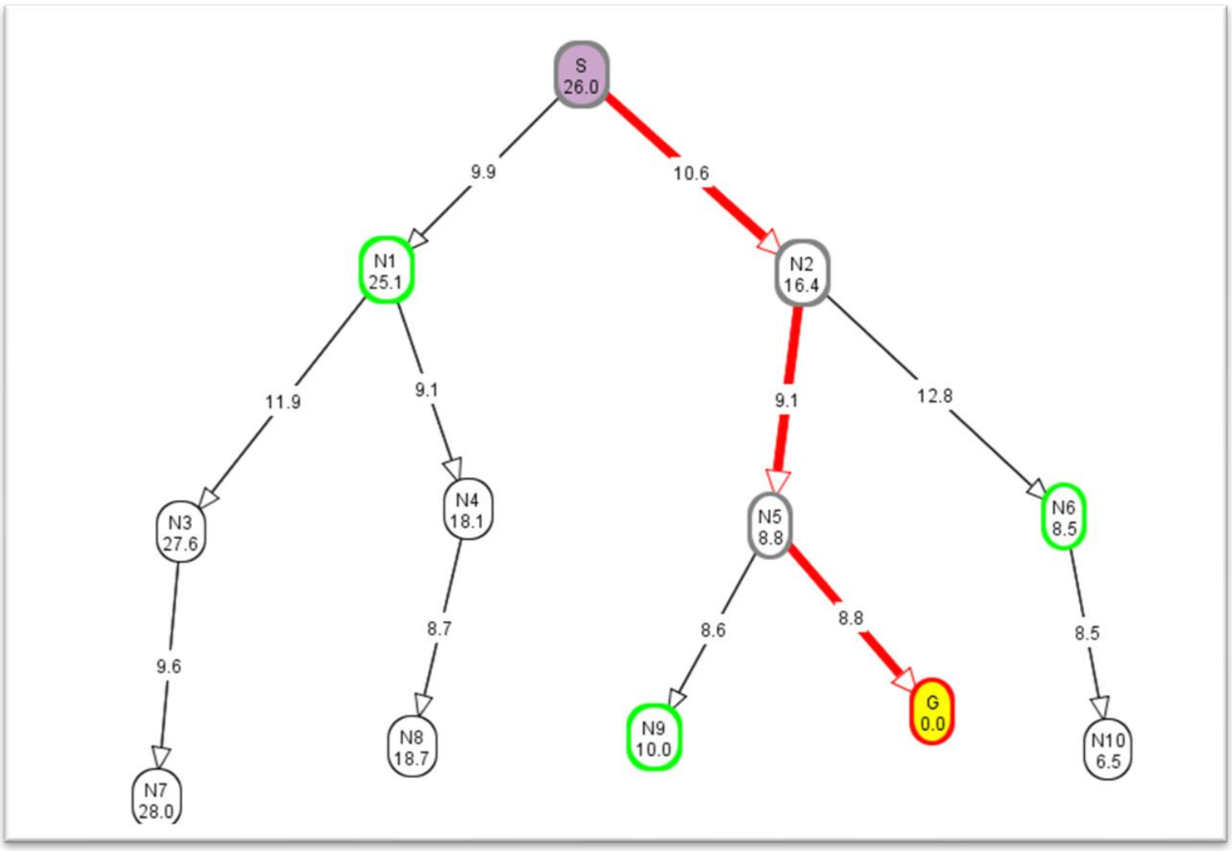
(b) Give a graph where BFS is much better than DFS.



Search Algorithm	Nodes Expanded
DFS	9
BFS	3

Neighbor Ordering: Left to right

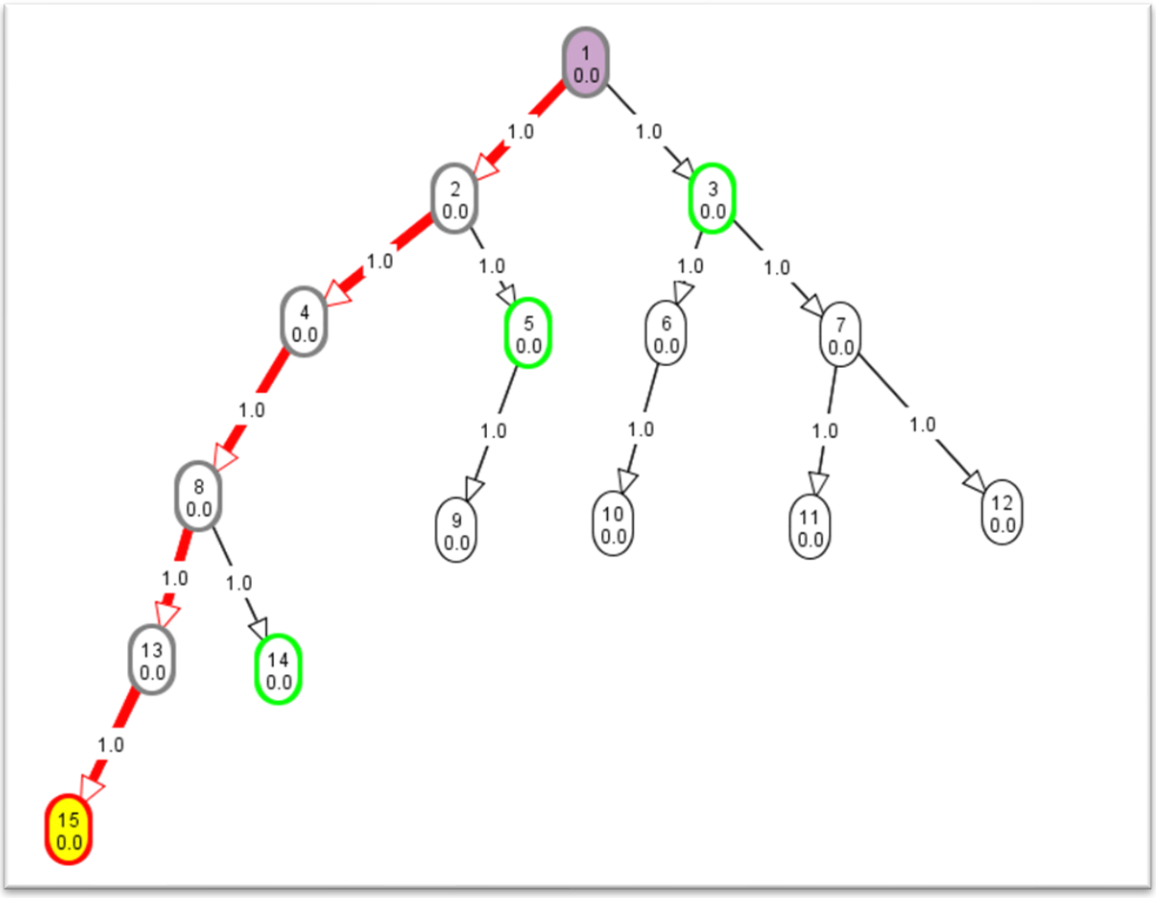
(c) Give a graph where A* search is more efficient than either DFS or BFS.



Search Algorithm	Nodes Expanded
DFS	10
BFS	11
A*	4

Neighbor Ordering: Left to right

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



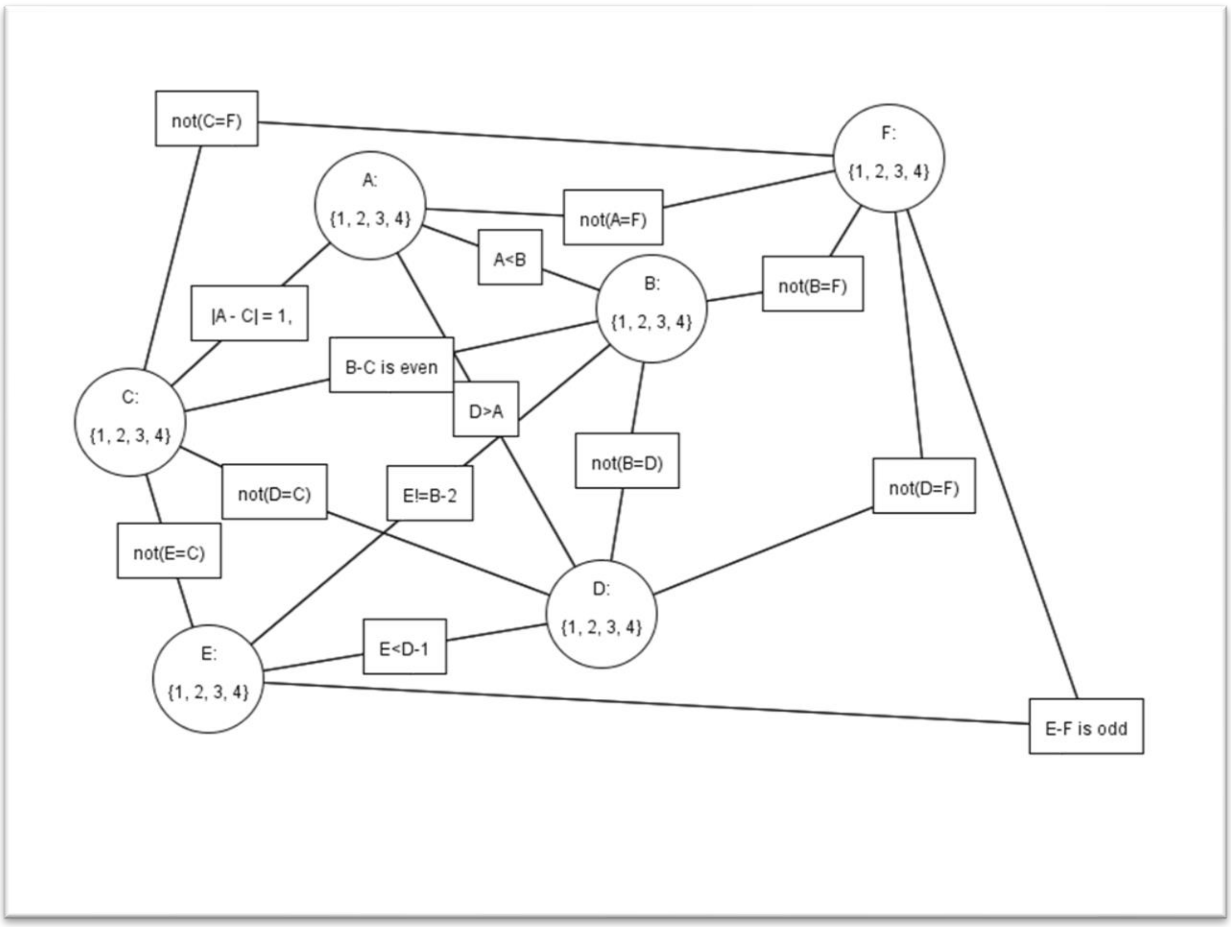
Search Algorithm	Nodes Expanded
DFS	6
BFS	15
A*	15

Neighbor Ordering: Left to right

Note: Since A* Search uses F(n)=G(n)+H(n), when H(n)=0, A* Search degrades into Uniform-Cost Search. As arc costs are all the same, it further degrades into BFS. Although A* and BFS both expand the same number of nodes, since A* search maintains a priority queue, which has greater time complexity than a regular queue, A* search ends up being less efficient than BFS in this scenario.

Question 2

(a) Draw this graph in AIspace.org as a CSP problem (constraint graph).



(b) For the first 5 instances of arc consistency, show which elements of a domain are deleted at each step, and which arc is responsible for removing the element.

Step 1
Arc selected: (B, A<B)
Consistent?: No
Element '1' deleted from domain of B.

Step 2
Arc selected: (A, A<B)
Consistent?: No
Element '4' deleted from domain of A.

Step 3
Arc selected: (C, |A-C|=1)
Consistent?: Yes

No element deleted from domain of C.

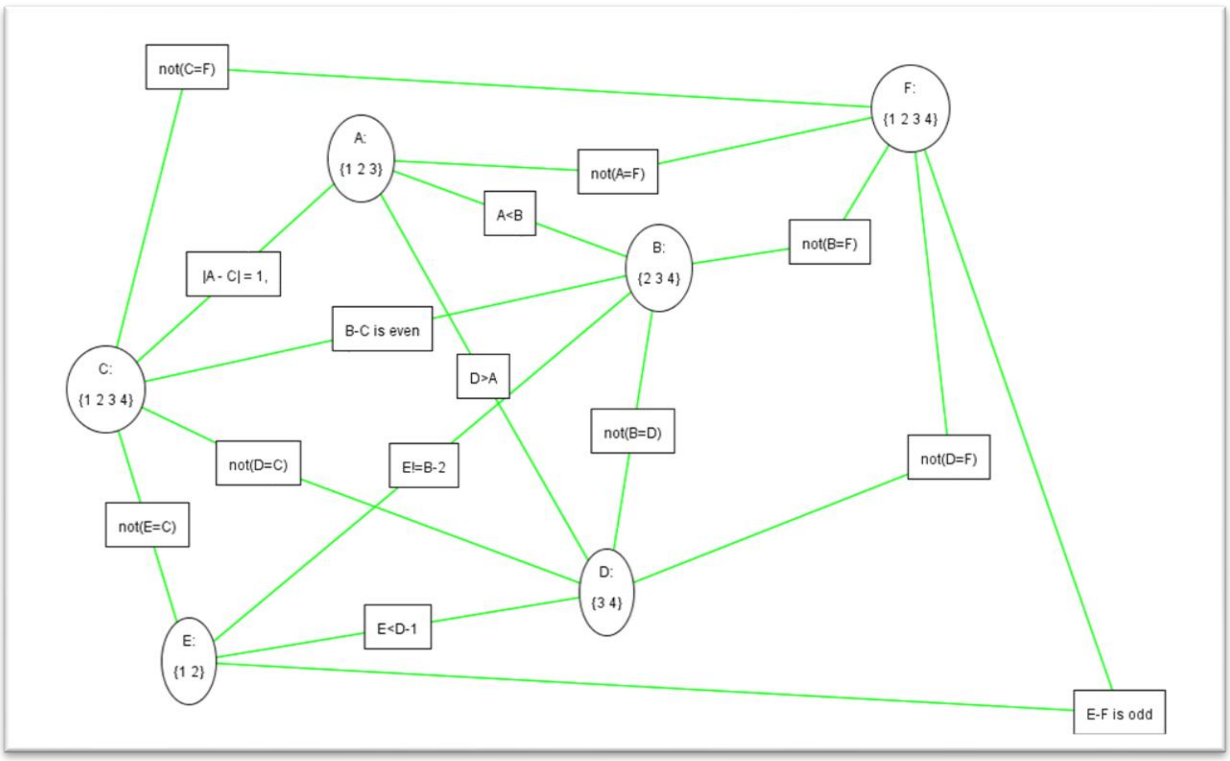
Step 4

Arc selected: (A, |A-C|=1)
Consistent?: Yes
No element deleted from domain of A.

Step 5

Arc selected: (C, B-C is even)
Consistent?: Yes
No element deleted from domain of C.

(c) Show explicitly the constraint graph after arc consistency has stopped.



(d) Show how splitting domains can be used to solve this problem. Draw the tree of splits and show the solutions.

After arc consistency is achieved for the whole graph, there are still several variables with |domain|>1, so domain splitting needs to be used to find a solution.

```
DOMAIN-SPLITTING HISTORY:

D in {3}
   Solution found:  A = 1, B = 2, C = 2, D = 3, E = 1, F = 4
D in {4}
   A in {1}
      Cannot split variable A
   A in {2}
      Solution found:  A = 2, B = 3, C = 3, D = 4, E = 2, F = 1
      Solution found:  A = 2, B = 3, C = 3, D = 4, E = 2, F = 1
```

Question 3

Conjecture:

(1) The closer H(n) is to the actual distance, the more efficient A* search is.
(2) As long as H(n) is admissible (an underestimate of actual distance, the closeness of H(n) to the actual distance has no effect on the accuracy of A* search (it will always optimal and complete).

Experiment Set-up

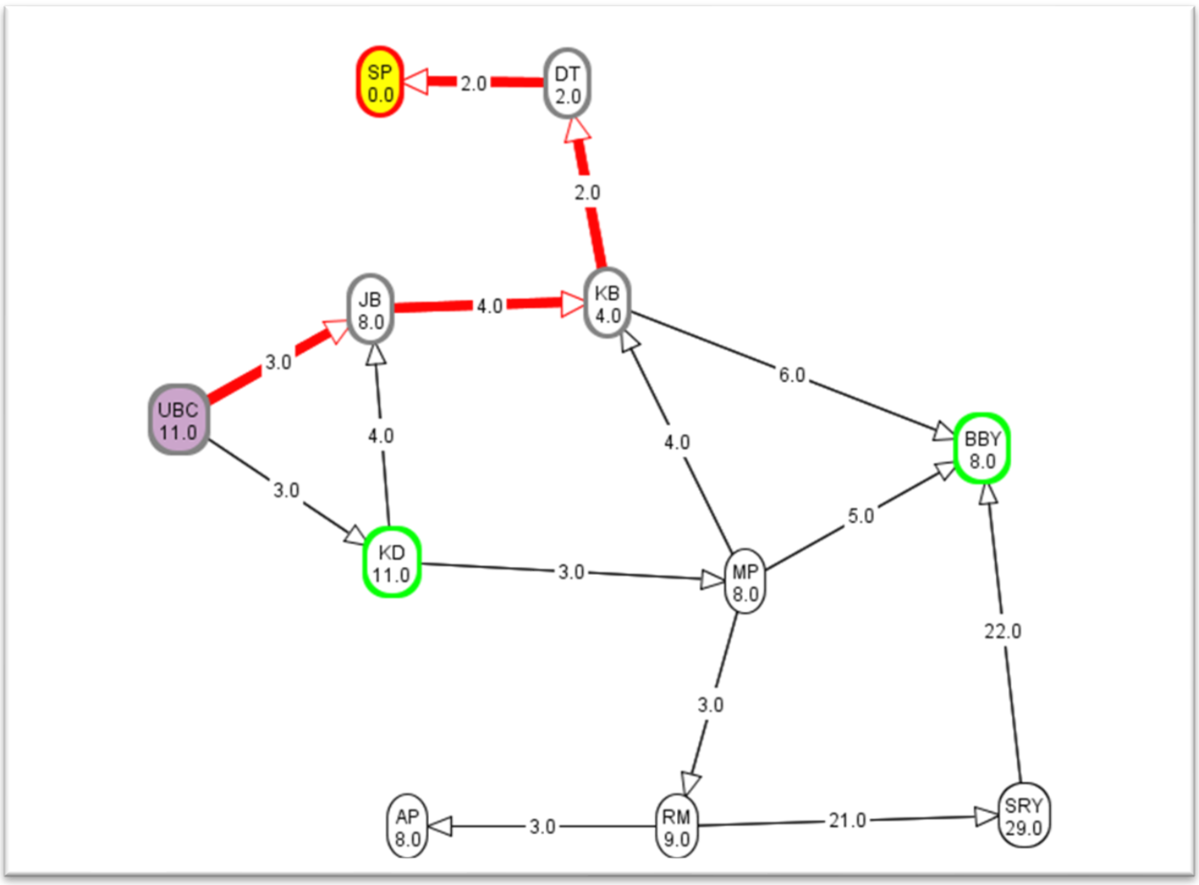


Figure 1: The base case used for the experiment, in which all nodes with a path to the goal node has H(n) which is equal to the actual cost-distance to the goal node.

Experimental Data

H(n)	Nodes Expanded (A*)	Nodes Expanded (Lowest-Cost-First)	Goal Node Found?	Shortest Path Found?
Base Case (Accurate H(n))	5	12	Y	Y
Underestimate (*0.75)	5		Y	Y
Underestimate (*0.5)	8		Y	Y
Underestimate (*0.25)	9		Y	Y
Underestimate (H(n)=0 for all nodes)	12		Y	Y
Underestimate (* -0.25)	14		Y	Y
Underestimate (* -0.5)	15		Y	Y
Underestimate (* -0.75)	17		Y	Y
Overestimate (*1.25)	5		Y	Y
Overestimate (*1.5)	5		Y	Y
Overestimate (*1.75)	5		Y	Y
Overestimate (*25)	5		Y	Y

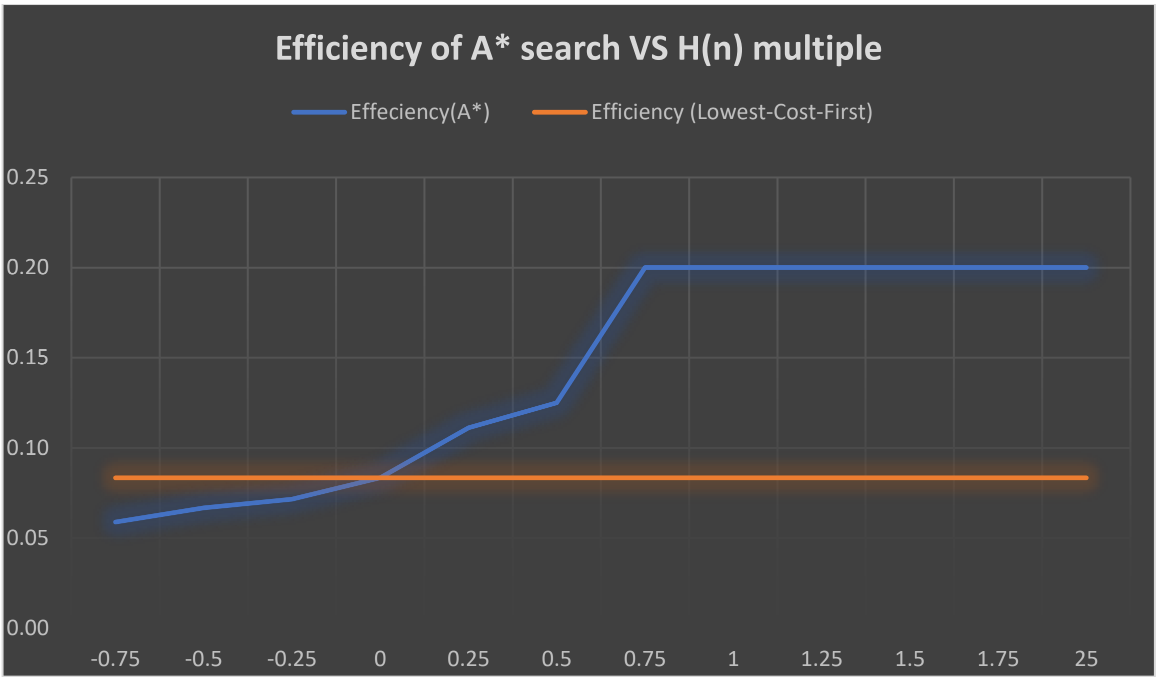


Figure 2: Data for Cyclic Graph

Experiment 2

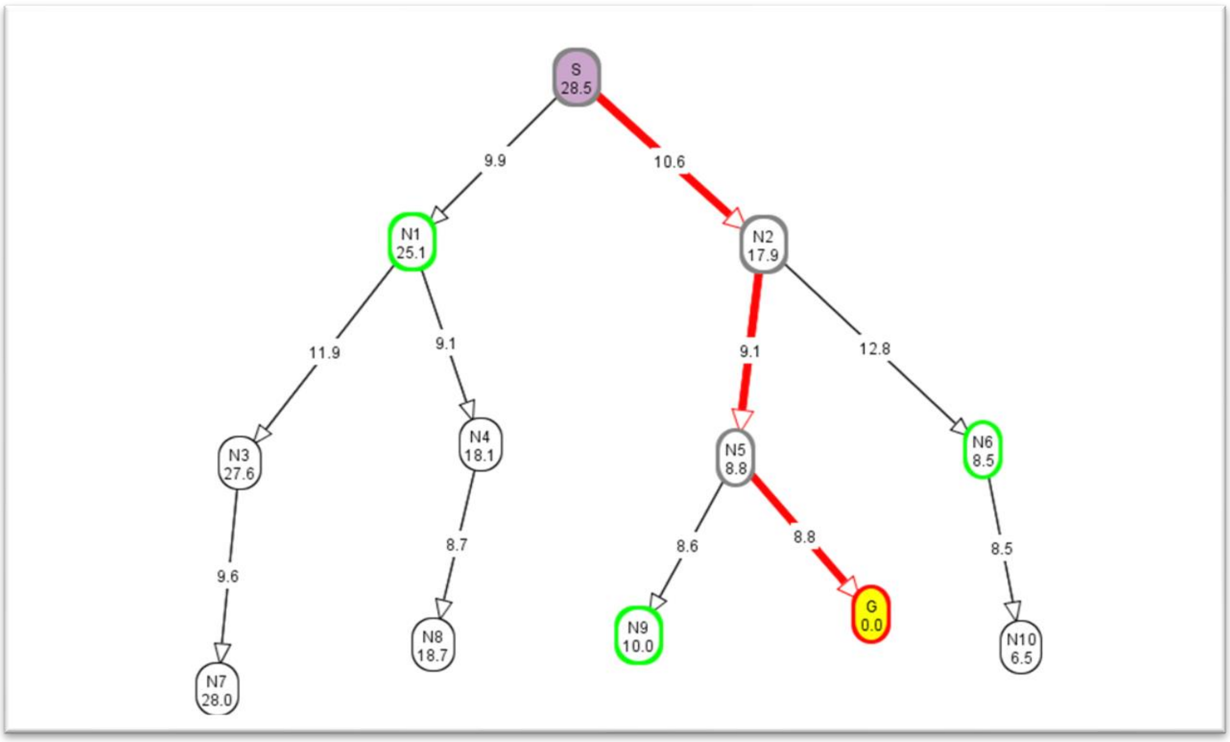


Figure 3: The experiment is repeated with a tree (acyclic), only path to goal node is also shortest path.

Experiment 2 Data

H(n)	Nodes Expanded (A*)	Nodes Expanded	Goal Node Found?

		(Lowest-Cost-First)	
Base Case (Accurate H(n))	4		Y
Underestimate (*0.75)	4		Y
Underestimate (*0.5)	7		Y
Underestimate (*0.25)	7		Y
Underestimate (H(n)=0 for all nodes)	10		Y
Underestimate (*-0.25)	11	10	Y
Underestimate (*-0.5)	11		Y
Underestimate (*-0.75)	12		Y
Overestimate (*1.25)	4		Y
Overestimate (*1.5)	4		Y
Overestimate (*1.75)	4		Y
Overestimate (*25)	6		Y

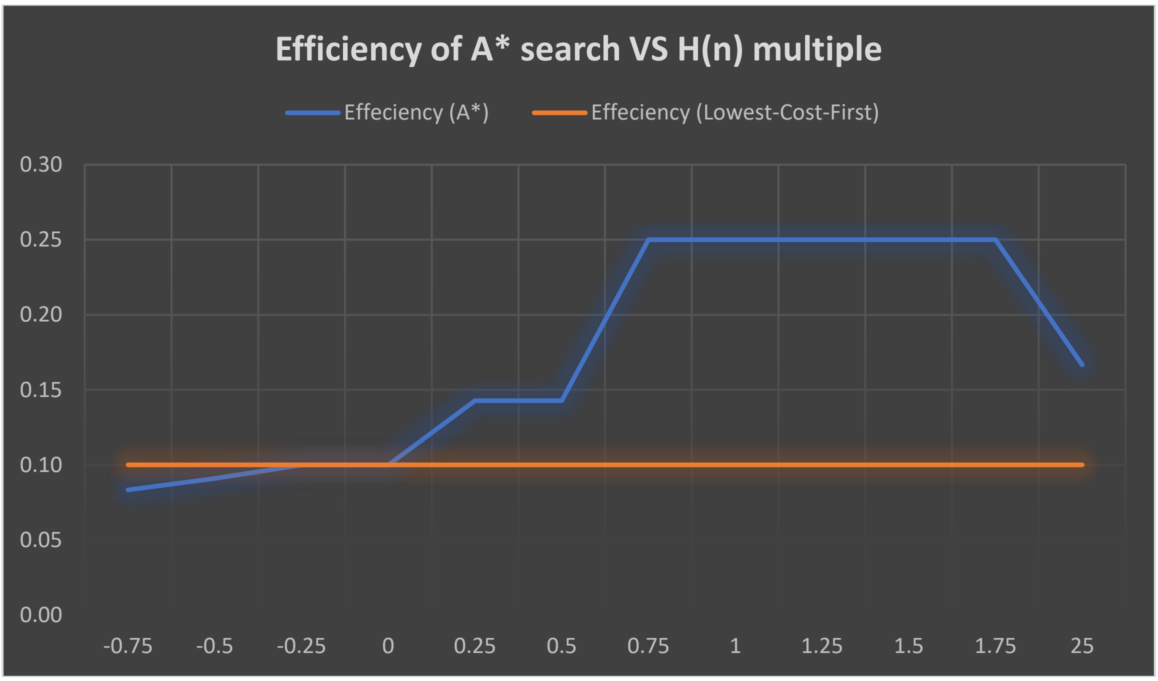


Figure 4: Data for acyclic graph

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

As seen from the data above, if H(n) is already an underestimate, reducing H(n) will worsen the performance of A* search. This is consistent with the theory of heuristics dominance, which states that if for all n, $H_2(n) >= H_1(n)$, then $H_2(n)$ is said to dominate $H_1(n)$ and is better for search. At H(n)=0, since A* search essentially degrades into Uniform Cost

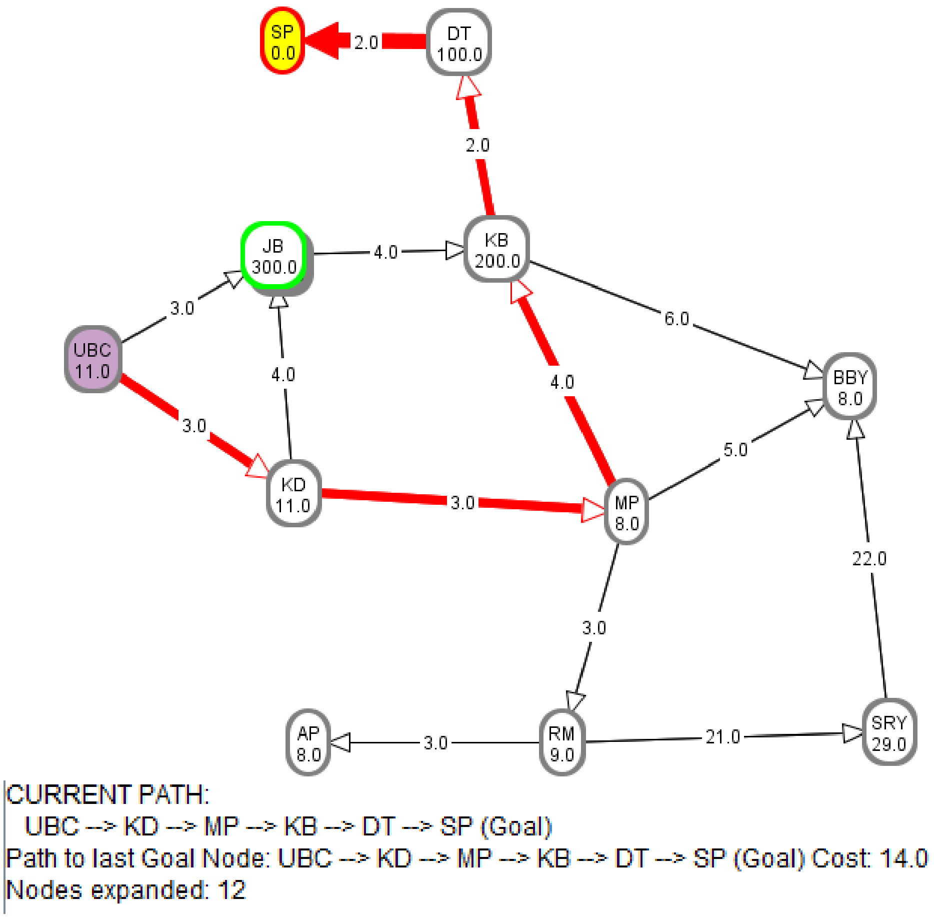
Search(UCS), as supported by the data from both experiments 1 and 2. When H(n) becomes negative, the performance of A* search becomes even worse than UCS.

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

As seen from the data from both experiments, A* performs best when H(n) is the exact distance from n to goal.

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

The data above seems to suggest that H(n) still performs optimally when it is an overestimate, that is plausibly explained by the fact that H(n) for all n have been inflated by the same factor from an exact H(n). However, the extreme value of *25H(n) for experiment 2 suggests that A* search performance can degrade when H(n) is an overestimate. Additionally, when H(n) is manipulated to be intentionally misleading, the performance of A* search will degrade and A* is no longer guaranteed to be optimal, as shown below, using the same graph from Experiment 1 with deliberately misleading H(n).



```
CURRENT PATH:
  UBC --> KD --> MP --> KB --> DT --> SP (Goal)
Path to last Goal Node: UBC --> KD --> MP --> KB --> DT --> SP (Goal) Cost: 14.0
Nodes expanded: 12
```

*Figure 5: Lower efficiency when A*search is deliberately fooled by misleading H(n) - 12 nodes expanded vs 5 expanded when perfectly accurate H(n) is used, non-optimal path with 14 cost found when path with 11 cost exists.*

Conclusion

The trend from the experimental data seems to support the part (1) of the conjecture, while there is no disproving evidence that goes against part (2) of the conjecture from the limited experimental data. Although decreasing H(n) when it is already admissible decreases the performance of A* search, there seems to be no impact on accuracy (completeness and optimal route found), with the only criteria for accuracy seeming to be usage of an admissible heuristic function.

Question 4

Question	Time Spent (hour(s))
1	1
2	2
3	4