Artificial Intelligence VBDS1402

MODULE 1

CHAPTER 2

INFORMED SEARCH: A* ALGORITHM

In this session you will learn:

A* informed search

Example of A* search

Algorithm of A* search

Advantages & Disadvantages of A* search

Performance measure of A* search

Types of Informed Search

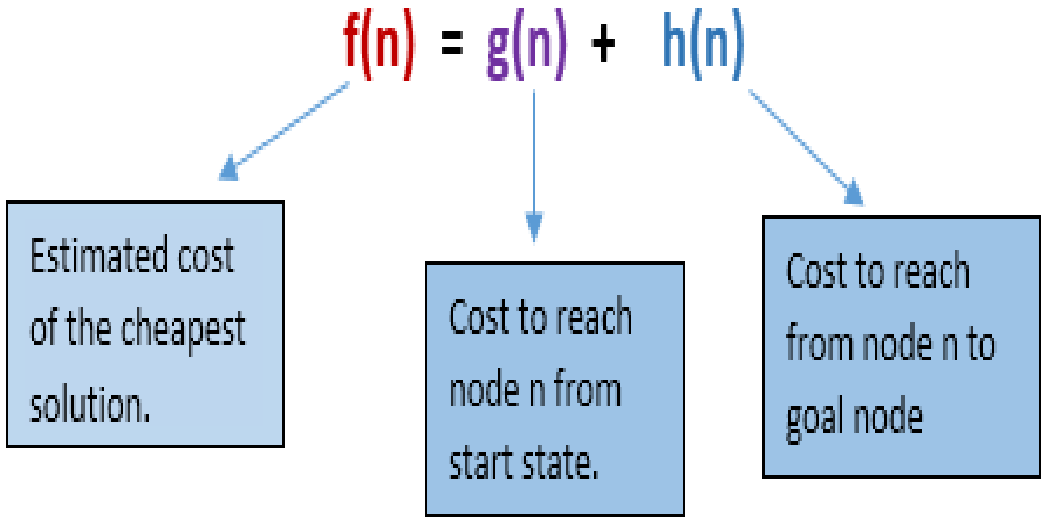
Greedy Best first search

A* search

Memory bounded search

A* search

- Most commonly known form of best-first search.
- Uses h(n) + g(n).
- Features of UCS + greedy best-first search.
- Finds the shortest path through the search space using the heuristic function.
- This search algorithm expands less search tree and provides optimal result faster.
- Uses search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.



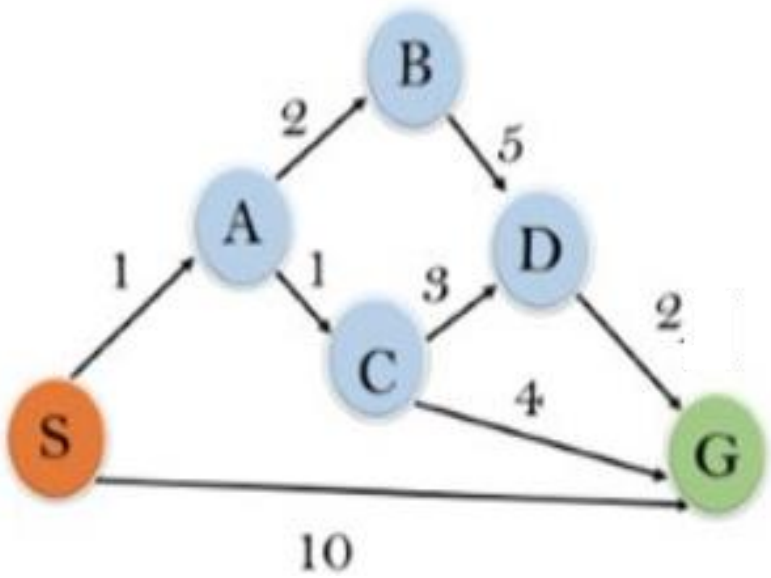
https://www.gatevidyalay.com/a-algorithm-a-algorithm-example-in-ai/

https://www.101computing.net/a-star-search-algorithm/

https://www.codecademy.com/resources/docs/ai/search-algorithms/a-star-search

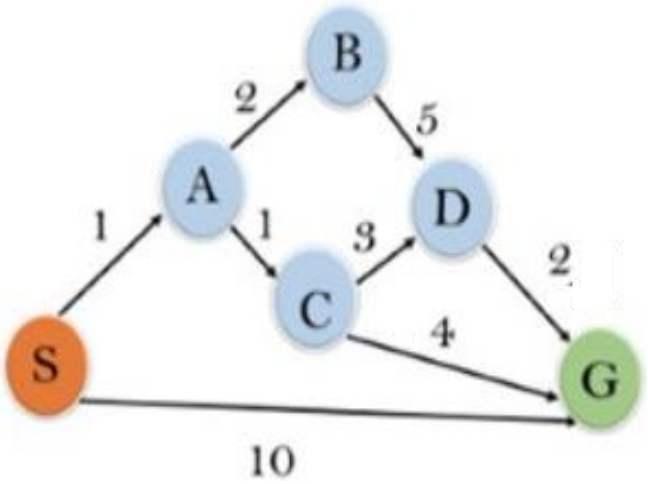
A* search (Example)

- Straight – line heuristics



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

A* search (Example)

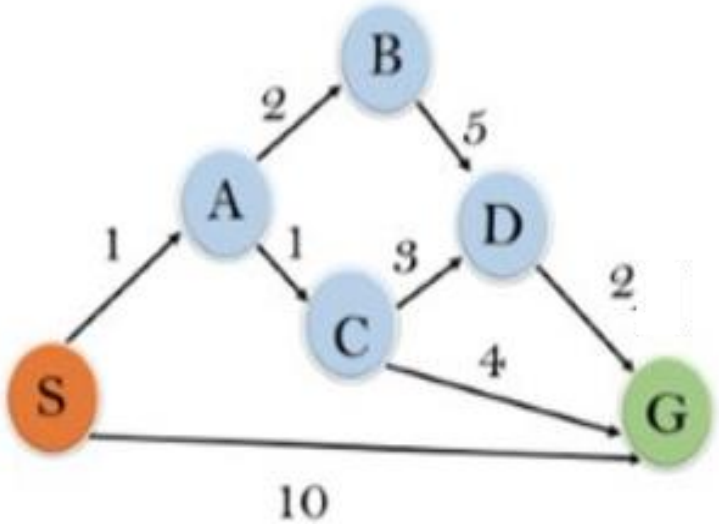


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

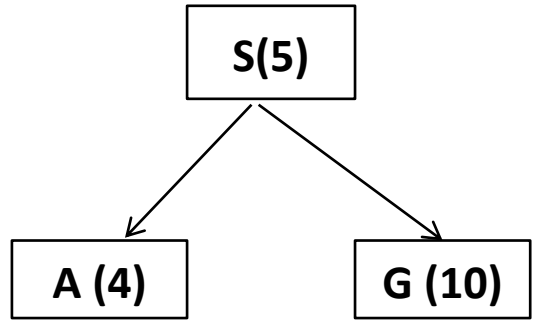
S(5)

Queue: S5

A* search (Example)

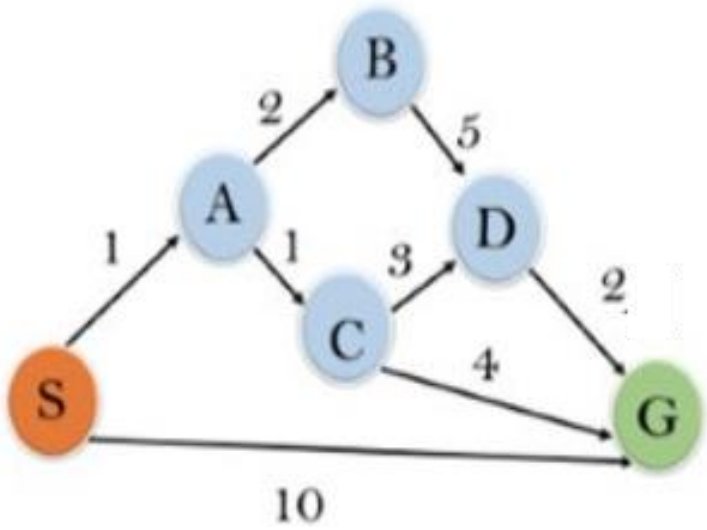


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

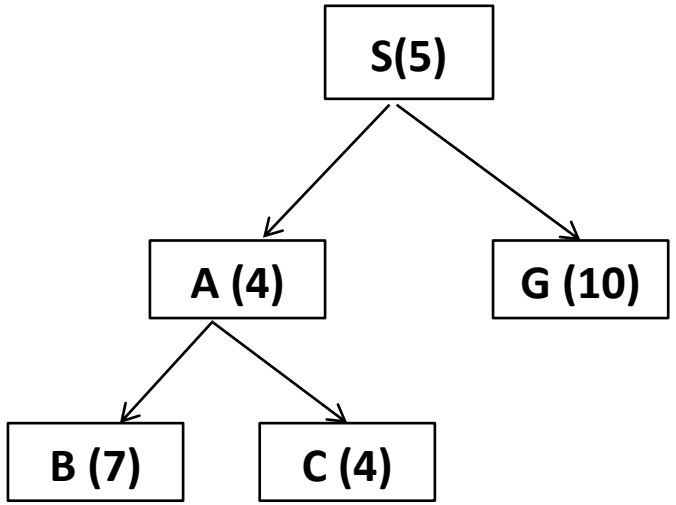


Queue: A4,G10

A* search (Example)

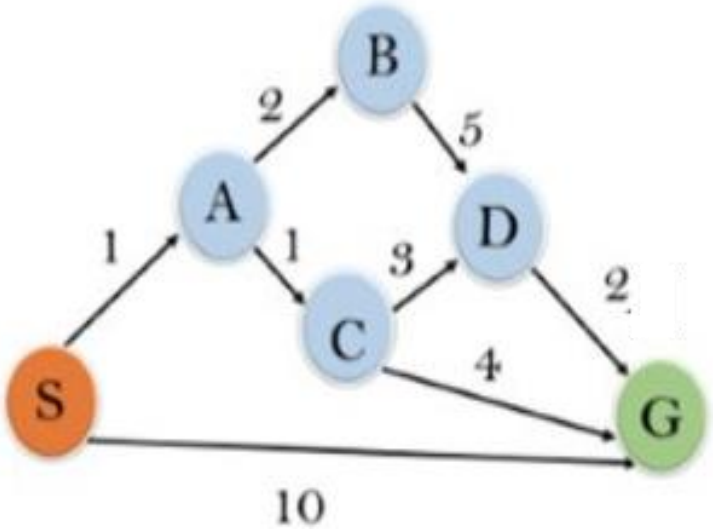


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

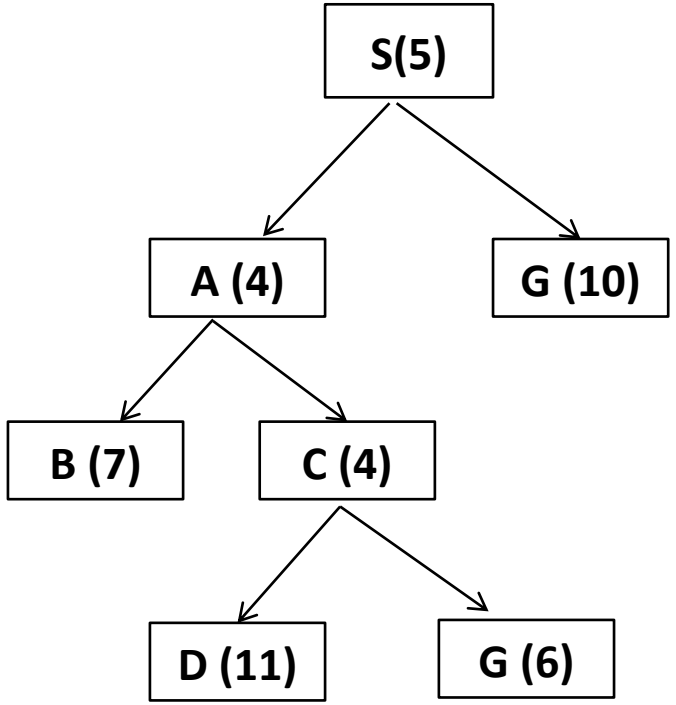


Queue: C4, B7, G10

A* search (Example)

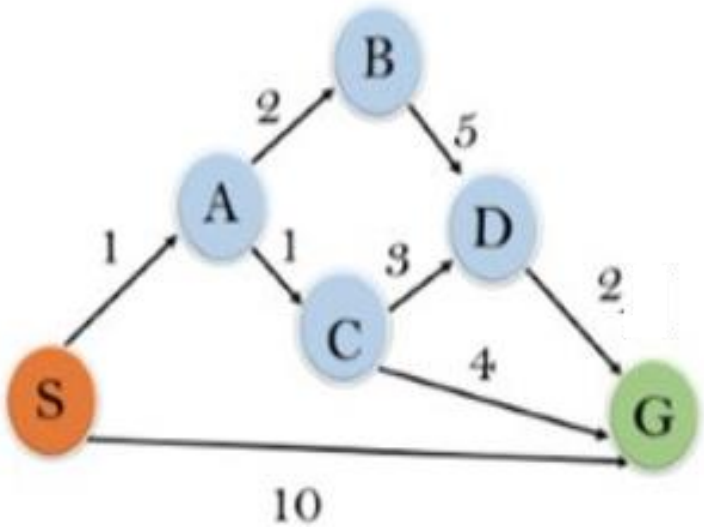


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

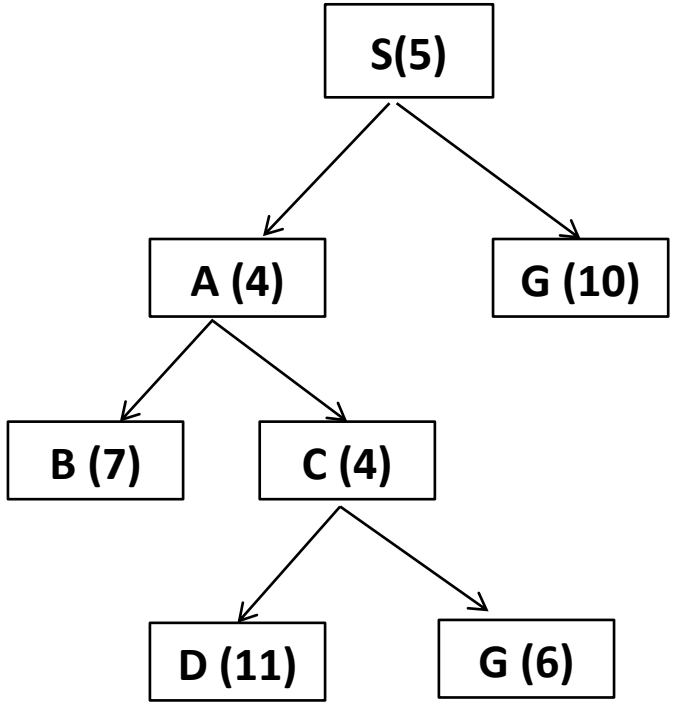


Queue: G6, B7, G10, D11

A* search (Example)



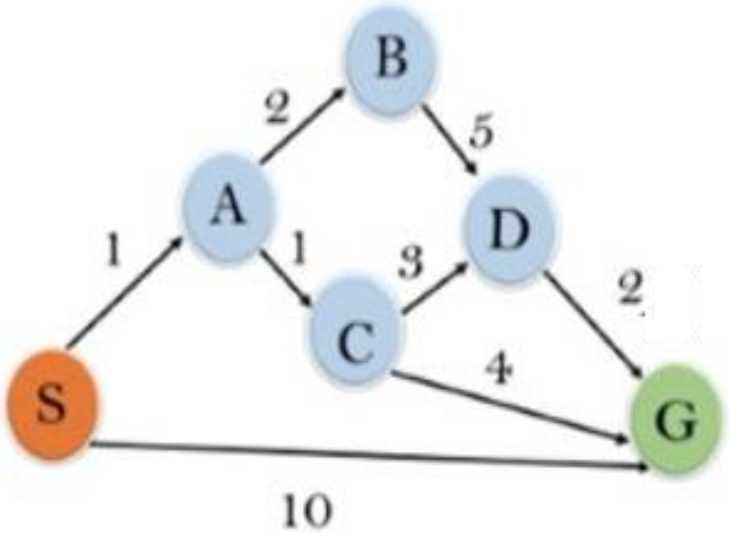
State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



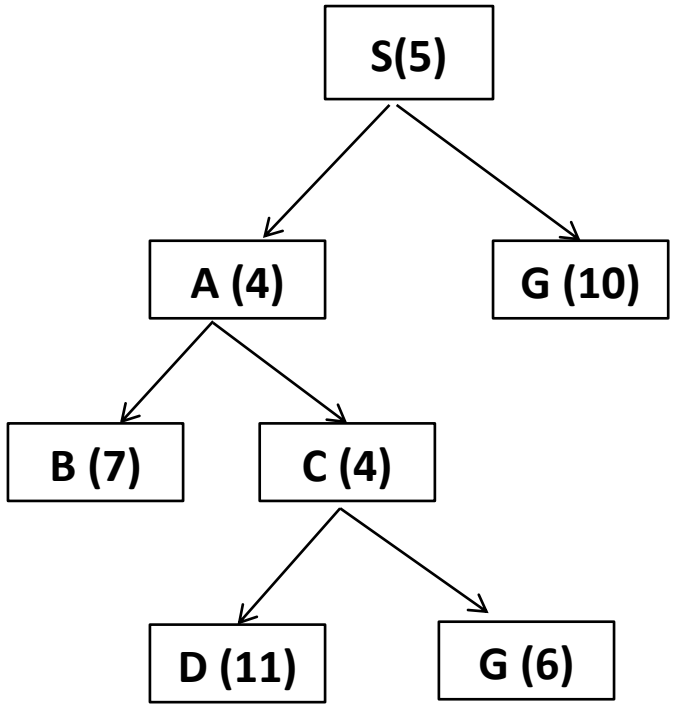
Queue: G6, B7, G10, D11

Path: SACG(6)

A* search (Example)

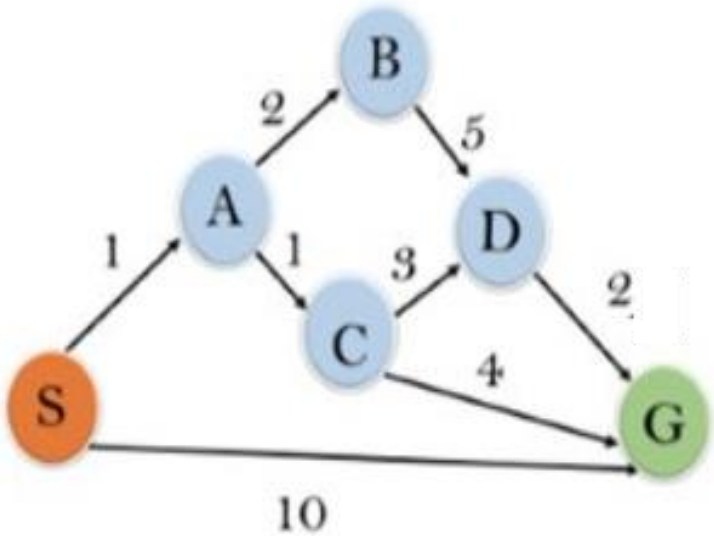


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

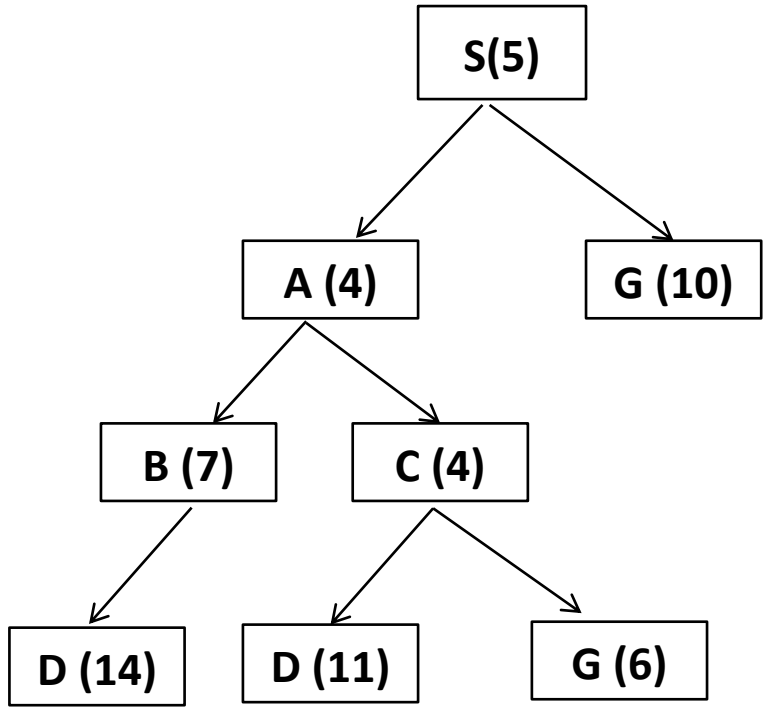


Queue: B7, G10, D11

A* search (Example)

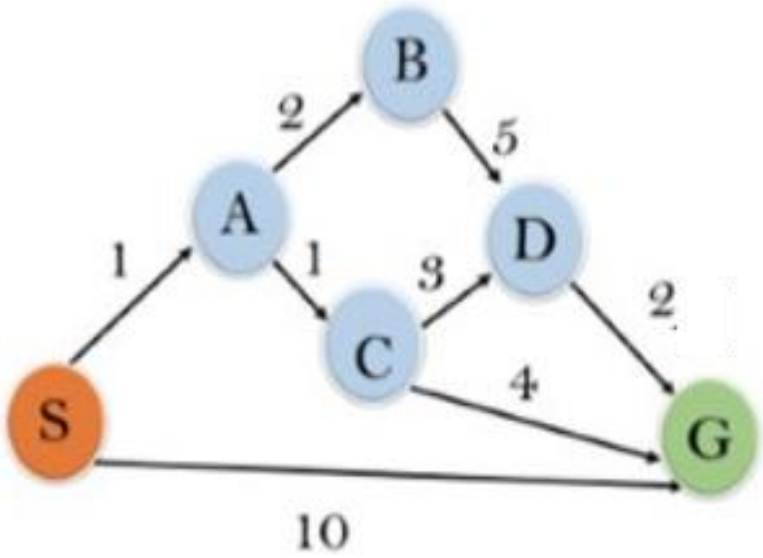


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

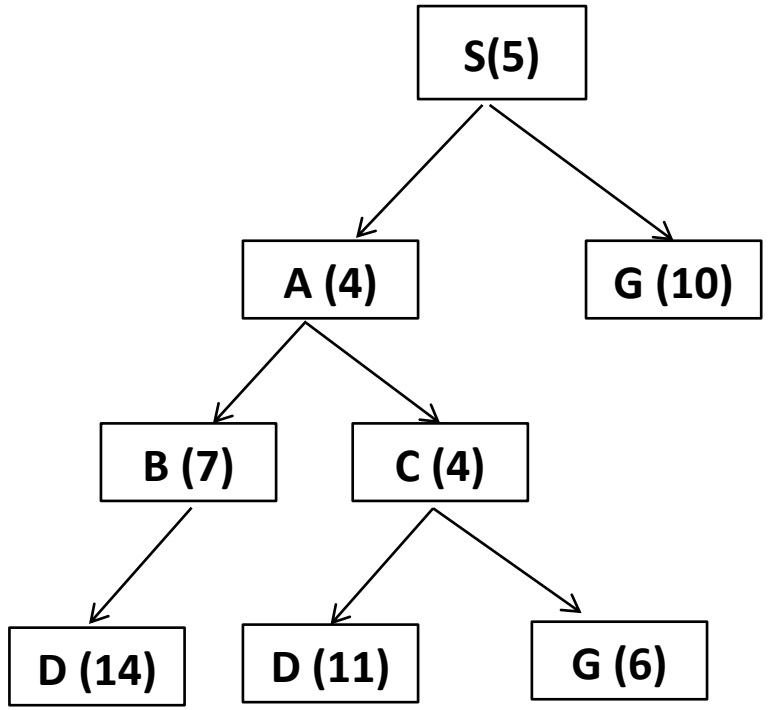


Queue: G10, D11, D14

A* search (Example)



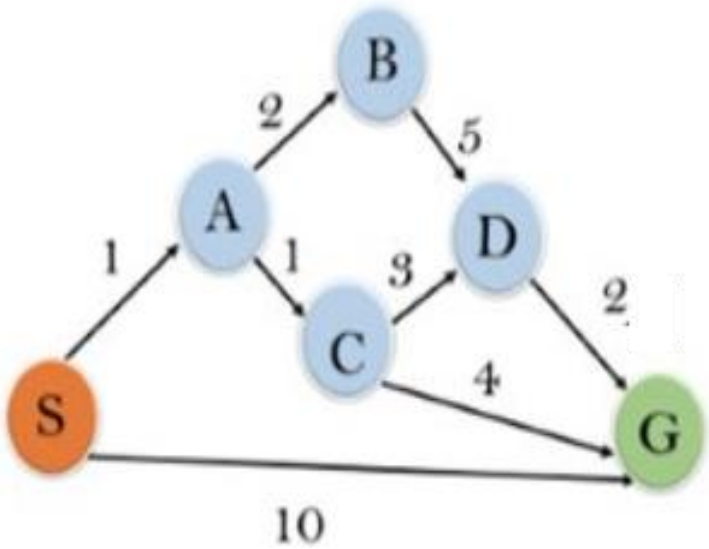
State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



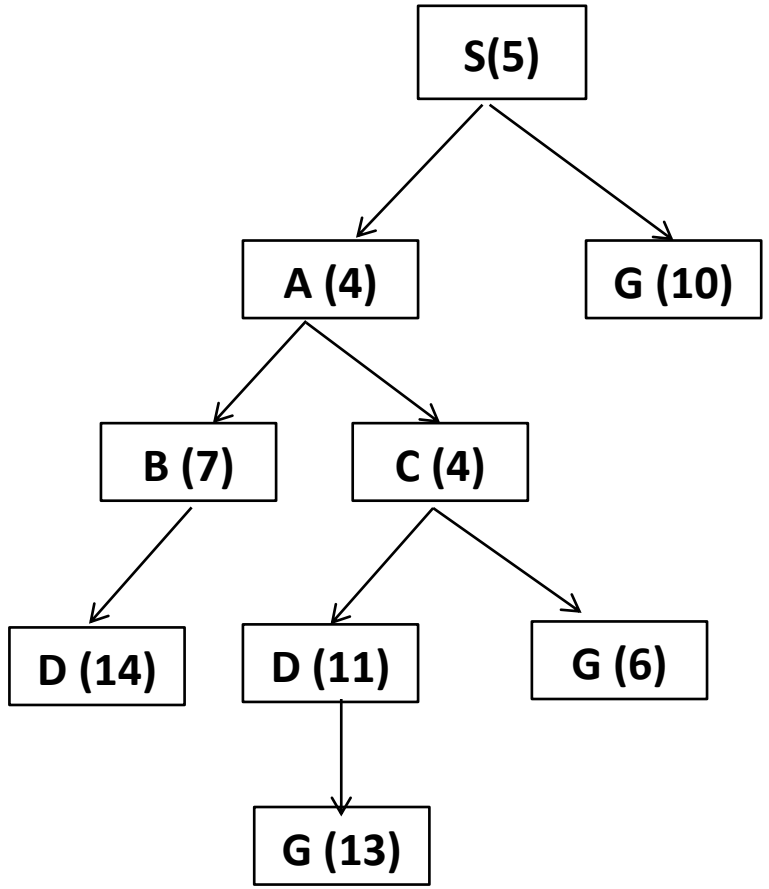
Queue: D11, D14

Path: SACG(6), SG(10)

A* search (Example)

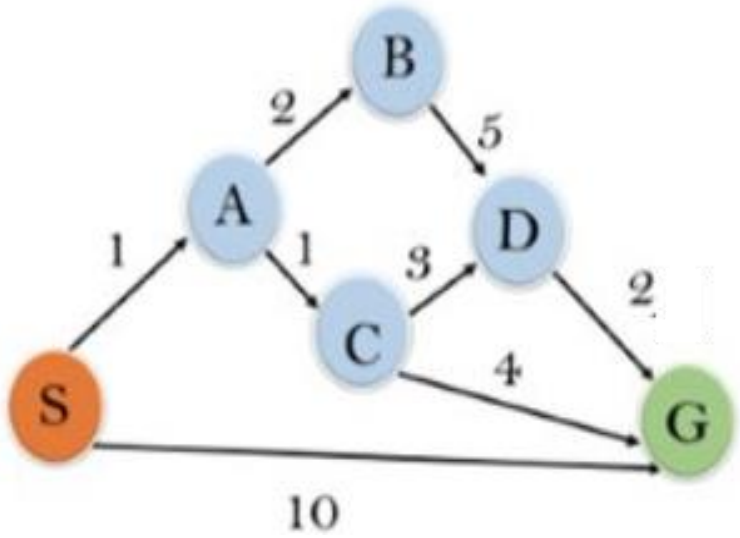


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

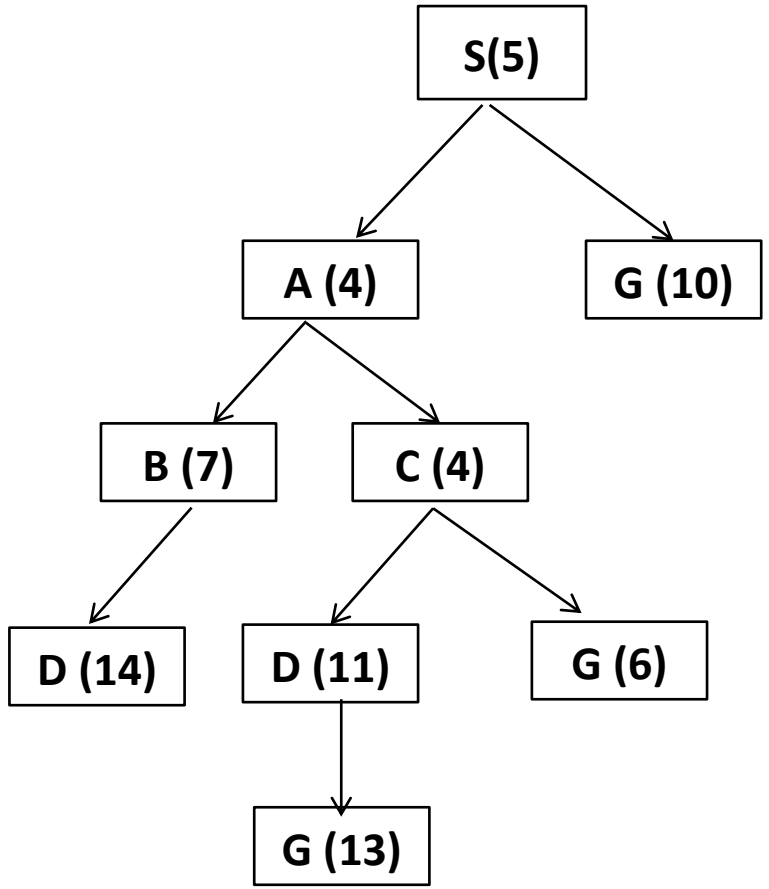


Queue: G(13), D(14)

A* search (Example)



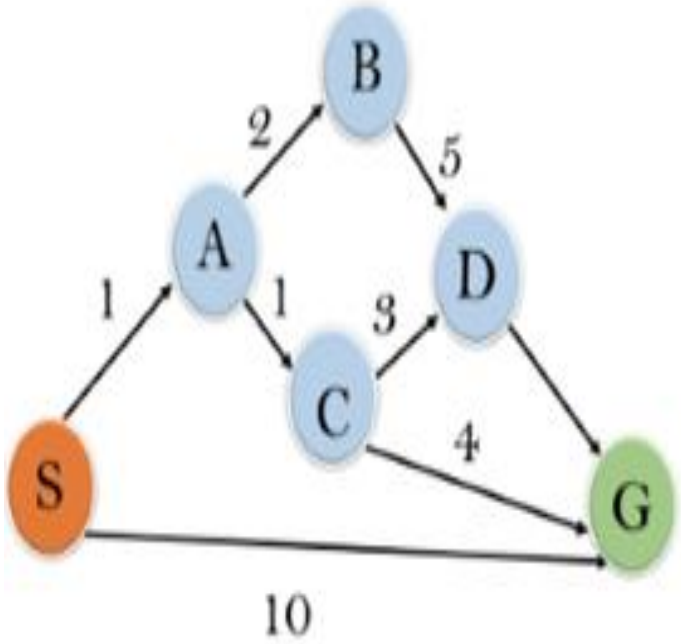
State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



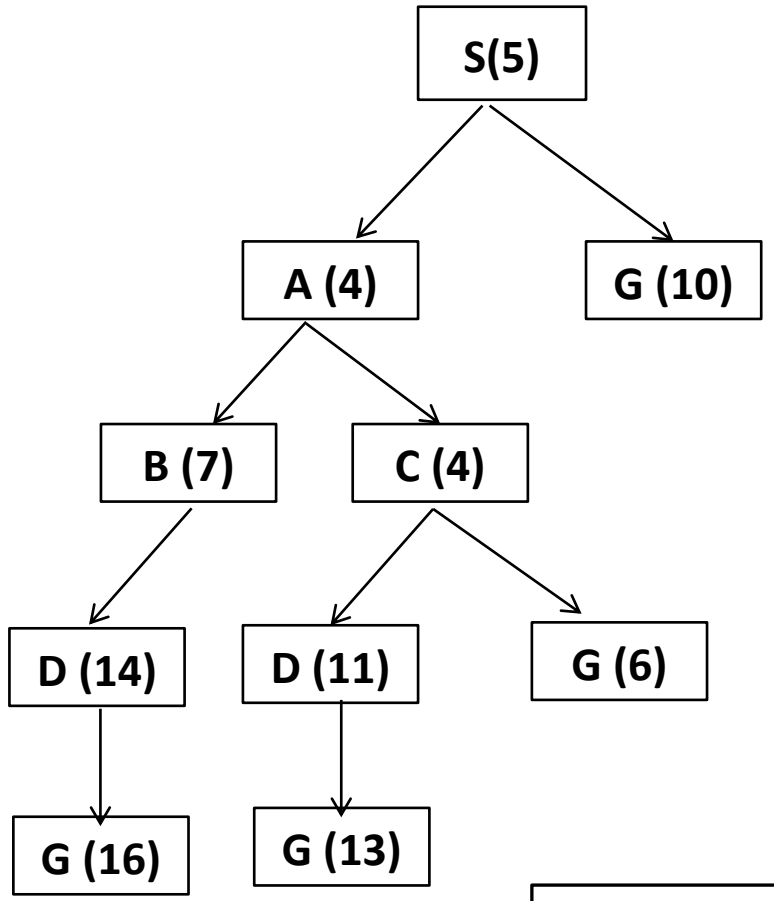
Queue: D(14)

Path: SACG(6), SG(10), SACDG(13)

A* search (Example)



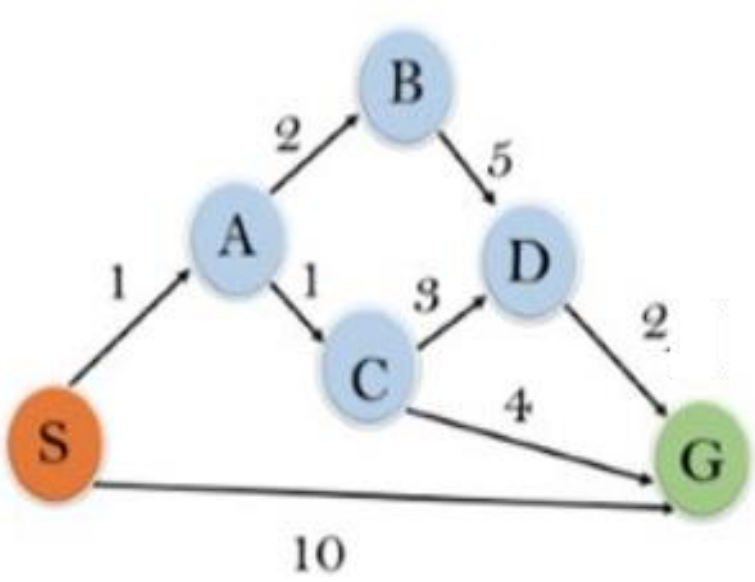
State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



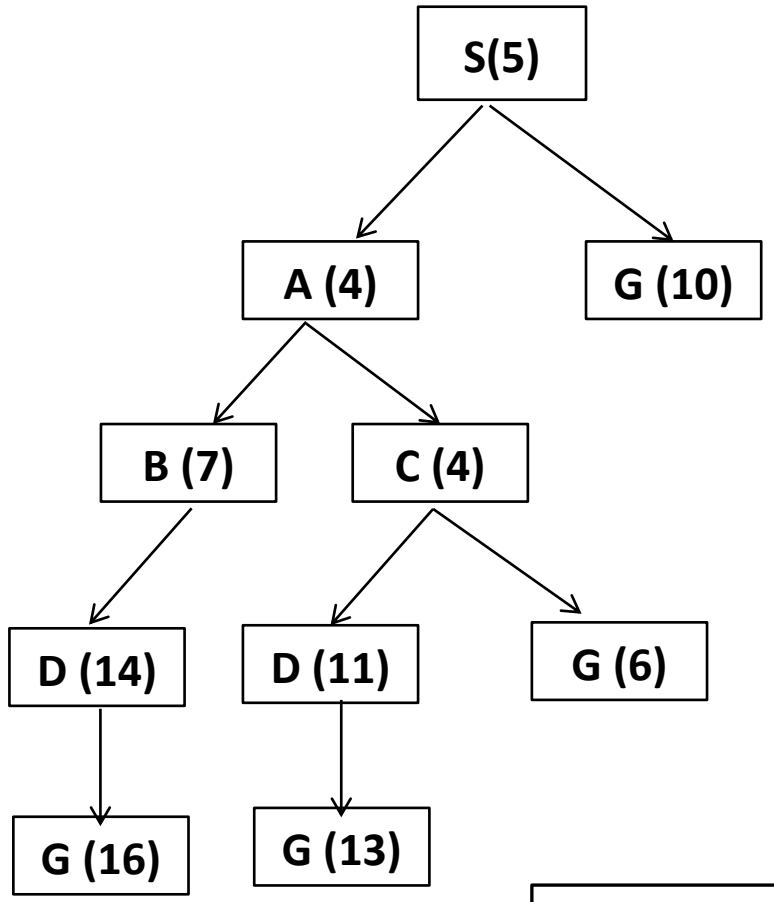
Queue: G(16)

Path: SACG(6), SG(10), SACDG(13)

A* search (Example)



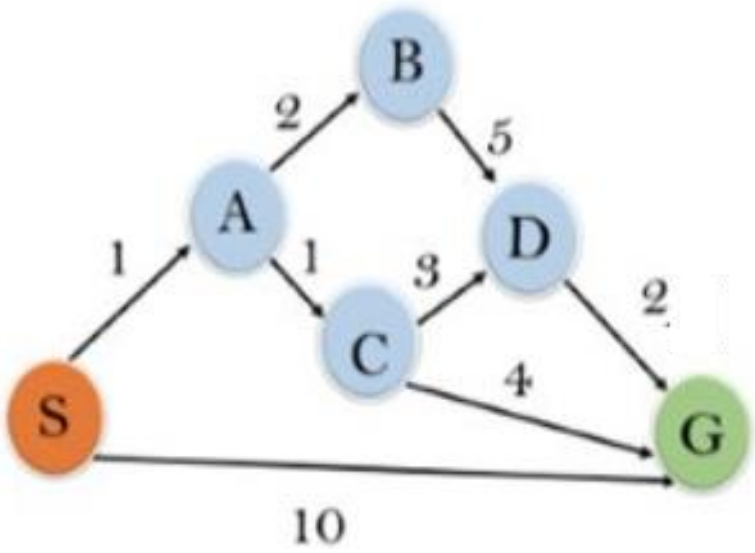
State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



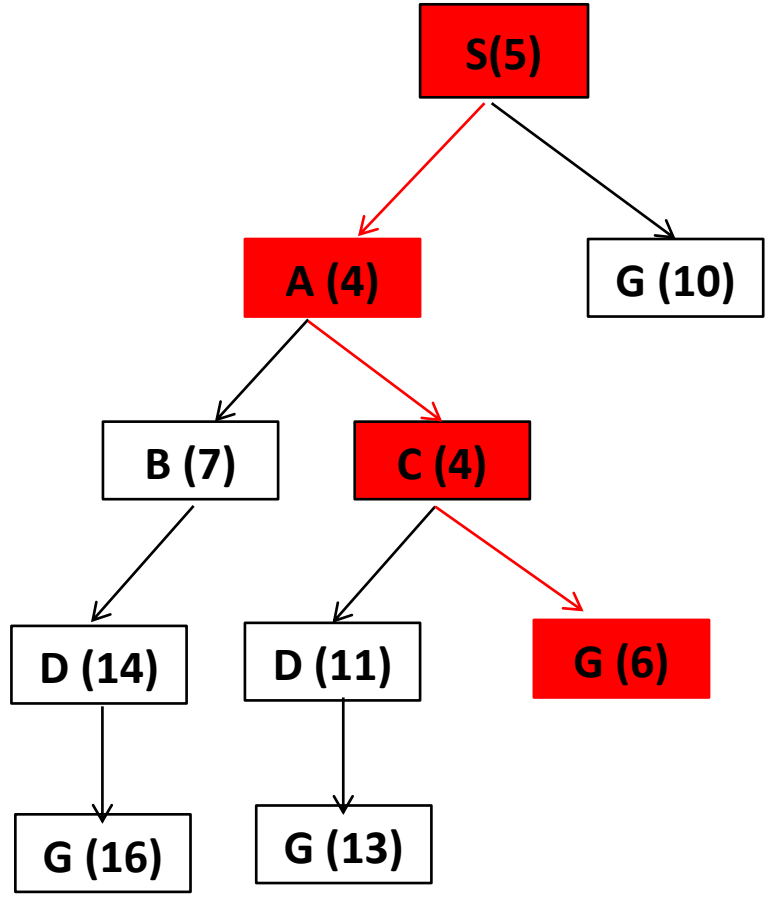
Queue: G(16)

Path: SACG(6), SG(10), SACDG(11), SABDG(16)

A* search (Example)



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



Queue: G16

Path: SACG(6), SG(10), SACDG(13), SABDG(16)

Admissibility

- h(n) is said to be admissible if it underestimates the cost of any solution that can be reached from n.
- If C*(n) is the cost of the cheapest solution path from n to a goal node, and if h is admissible.

- h(n) <= C*(n).
- A* is admissible.
- It means if a solution exists, the first solution found by A* is an optimal solution.

Algorithm for A*

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to **Step 2**.

Advantages

A* search algorithm is the best algorithm than other search algorithms.

A* search algorithm is optimal and complete.

This algorithm can solve very complex problems.

Disadvantages

It does not always produce the shortest path as it mostly based on heuristics and approximation.

A* search algorithm has some complexity issues.

The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

Take home task

Apply A* search