CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGG.

Bachelor of Engineering (Computer Science & Engineering)

Principles of Artificial Intelligence (20CST-258)

A* Search Algorithm

DISCOVER . LEARN . EMPOWER

Outline

- Type of Search Strategies
- Pure Heuristic Search
    - Best First Search Algorithm(Greedy search)
    - A* Search Algorithm
- Hill Climbing:
    - Simple hill Climbing
    - Steepest-Ascent hill-climbing
    - Stochastic hill Climbing

Heuristic Search

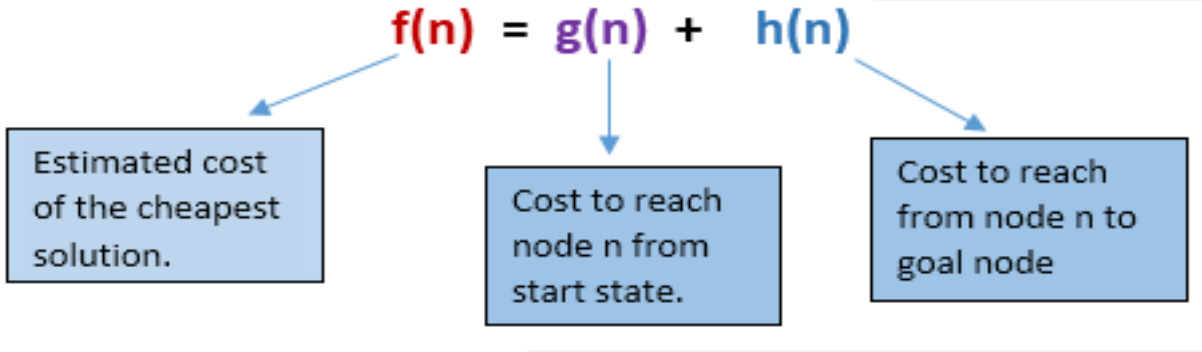
- **Pure Heuristic Search**
  - In the informed search, there are two main algorithms:
    - Best First Search Algorithm(Greedy search)
    - A* Search Algorithm
- **Hill Climbing:**
    - Simple hill Climbing
    - Steepest-Ascent hill-climbing
    - Stochastic hill Climbing

A* Search Algorithm

- A* search is the most commonly known form of best-first search.
- It uses heuristic function h(n), and cost to reach the node n from the start state g(n).
- It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently.
- A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster.

A* Search Algorithm

- A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).
- In A* search algorithm, we use 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**.



A* Search Algorithm

- Algorithm of A* search:
  - **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

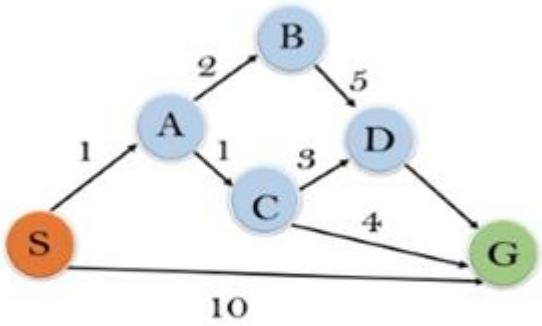
- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

Example:

- In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula :

$$f(n)= g(n) + h(n),$$

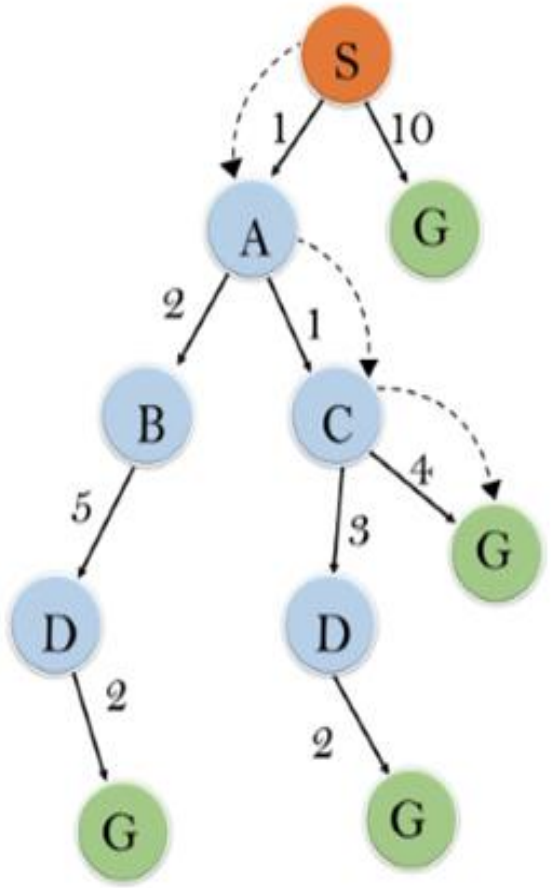
- where g(n) is the cost to reach any node from start state.
- Here we will use OPEN and CLOSED list.



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

Solution

- **Initialization:** {(S, 5)}
- **Iteration1:** {(S--> A, 4), (S-->G, 10)}
- **Iteration2:** {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}
- **Iteration3:** {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}
- **Iteration 4** will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.



Key Points

- **Complete:**
  - A* algorithm is complete as long as:
  - Branching factor is finite.
  - Cost at every action is fixed.
- **Optimal:** A* search algorithm is optimal if it follows below two conditions:
- **Admissible:** the first condition requires for optimality is that h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.
- **Consistency:** Second required condition is consistency for only A* graph-search.
- If the heuristic function is admissible, then A* tree search will always find the least cost path.
- **Time Complexity:** The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.
- **Space Complexity:** The space complexity of A* search algorithm is **O(b^d)**

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.

THANK YOU