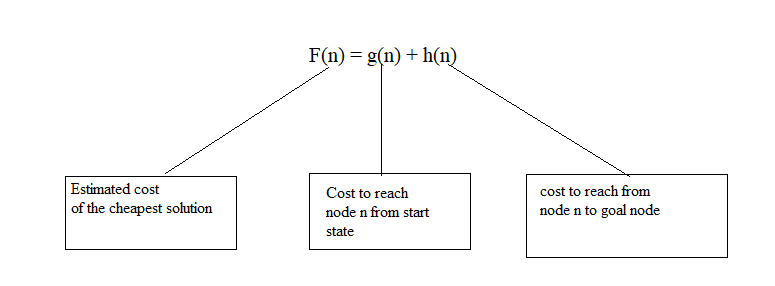
**A \* Search**

The most widely known form of best-first search is called A \* search.  A-star search is one of the most successful search algorithms to find the shortest path between nodes or graphs. It is an informed search algorithm, as it uses information about path cost and also uses [heuristic](https://en.wikipedia.org/wiki/Heuristic_(computer_science))s to find the solution.

It avoids expanding paths that are already expensive, but expands most promising paths first. It evaluates nodes by combining g(n), and h(n). Each time A\* enters a state, it calculates the cost, f(n) to travel to all of the neighbouring nodes, and then enters the node with the lowest value of f(n).

These values are calculated with the following formula:



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

* g(n) is the cost to reach the node n from start state
* h(n) is the estimated cost to get from the node n to the goal
* f(n) is the estimated total cost of path through n to goal node OR estimated cost of the cheapest solution through n .

Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of g(n) + h(n).

**Algorithm of A\* search:**

**Step-1:**

* Define a list OPEN.
* Initially, OPEN consists start node S.

**Step-2:**

* If the list is empty, return failure and exit.

**Step-3:**

* Remove node n with the smallest value of f(n) from OPEN and move it to list CLOSED.
* If node n is a goal state, return success and exit.

**Step-4:**

* Expand node n.

**Step-5:**

* If any successor to n is the goal node, return success and the solution by tracing the path from goal node to S.
* Otherwise, go to Step-6.

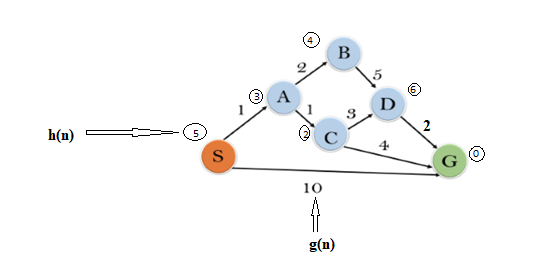
**Step-6:**

* For each successor node,
* Apply the evaluation function f to the node.
* If the node has not been in either list, add it to OPEN.

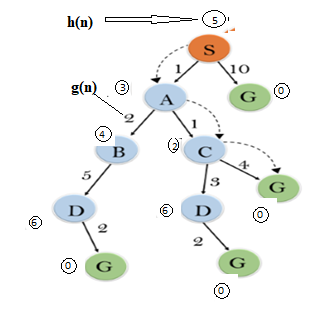
**Step-7:**

* Go back to Step-2.

**Example 1: -**



**Solution:**

****

## **Solution-**

**Step-01:**

* We start with node s.
* Node A and Node G can be reached from node s.

A\* Algorithm calculates f(B) and f(G).

* f(A) = 1 + 3 = 4
* f(G) = 10 + 0 = 10

Since f(A) < f(G), so it decides to go to node A.

Path- S→ A

**Step-02:**

Node B and Node C can be reached from node A.

A\* Algorithm calculates f(B) and f(C).

* f(B) = (2+1) + 4 = 7
* f(C) = (1+1) + 2 = 4

Since f(C) < f(B), so it decides to go to node C.

Path- S→ A → C

**Step-03:**

Node D and Node G can be reached from node C.

A\* Algorithm calculates f(D) and f(G).

* f(D) = (1+1+3) + 6 = 11
* f(G) = 1+1+4 + 0 = 6

Since f(G) < f(D), so it decides to go to node G.

Path- S → A → C → G

This is the required shortest path from node S to node G.