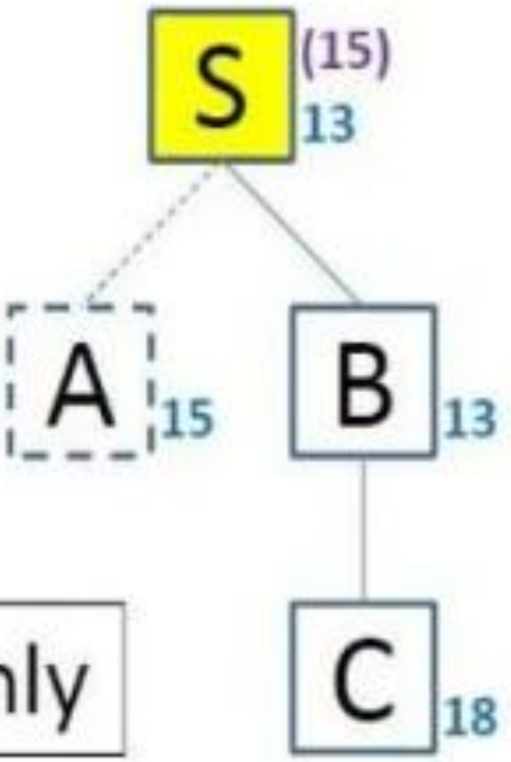
SMA* Algorithm

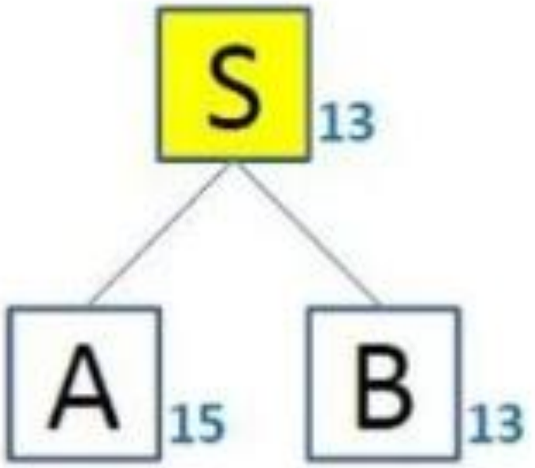
- Optimizes A* to work within reduced memory
- ***Key Idea:***
  - **IF** memory **full** for **extra node (C)**
  - **Remove highest f-value leaf (A)**
  - **Remember best-forgotten child** in <u>*each*</u> parent node **(15 in S)**



E.g. Memory of 3 nodes only

SMA* Algorithm

- **Generate Children 1 by 1**
  - **Expanding**: add <u>1 child at the time</u> to QUEUE
  - Avoids **memory overflow**
  - **Allows monitoring** if nodes need deletion

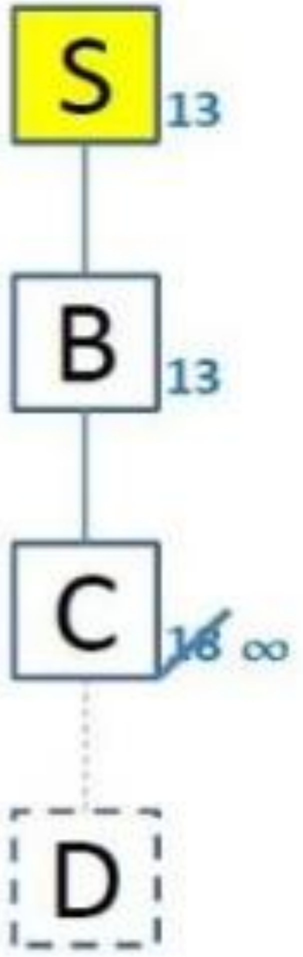


First add A later B

SMA* Algorithm

- **Too long paths: Give up**
  - **Extending** path **cannot fit** in **memory**
    - **give up (C)**
  - Set **f-value** node **(C)** to ∞
    - **Remembers:**
      path cannot be found here

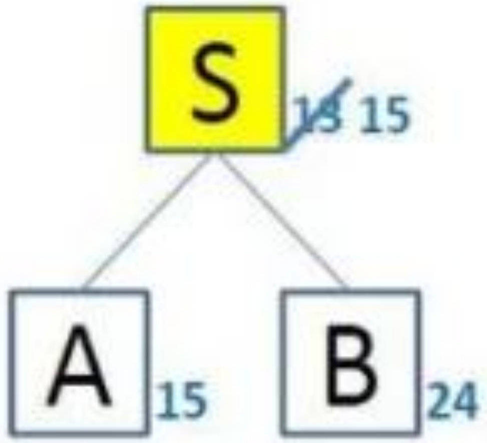
E.g. Memory of 3 nodes only



Handling Long Paths i.e. Too Many Nodes In The Memory

SMA* Algorithm

- **Adjust f-values**
  - **IF** all children M_i of node N have been explored
  - **AND** $\forall i$: $f(S \ldots M_i) > f(S \ldots N)$
  - **THEN reset** (through N $\Longrightarrow$ through children)
    - $f(S \ldots N) = \min\{f(S \ldots M_i) \mid M_i \text{ child of } N\}$



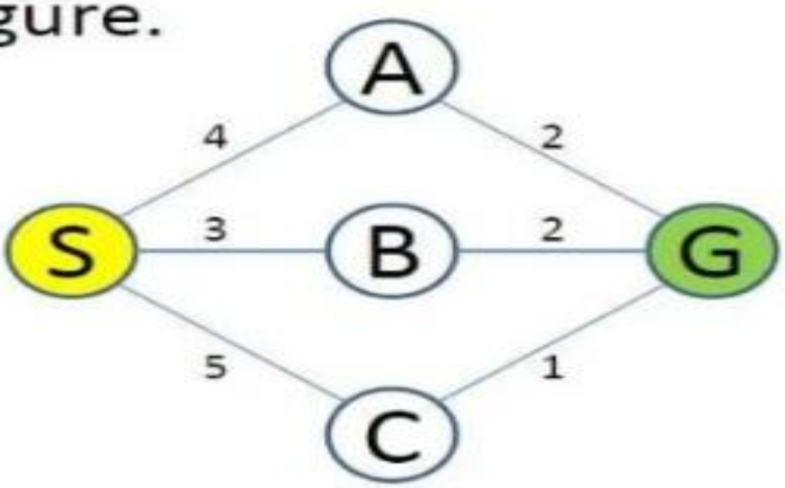
Better estimate for f(S)

Adjusting The f Value

Problem

SMA* by Example

- Perform SMA* (memory: 3 nodes) on the following figure.

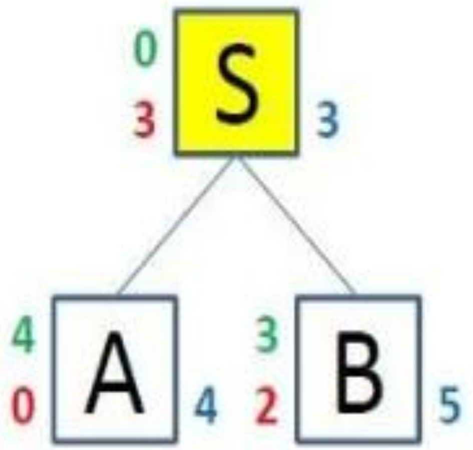


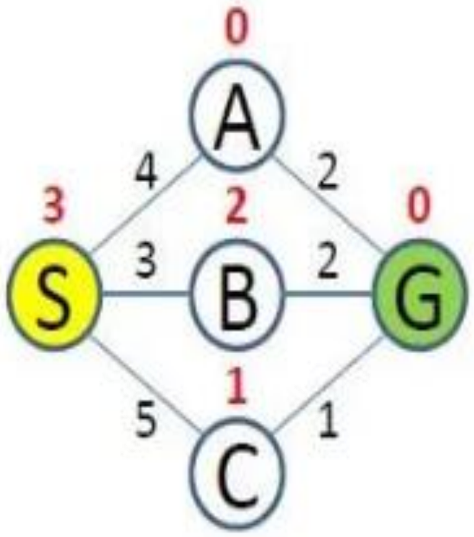
	S	A	B	C	G
heuristic	3	0	2	1	0

[illegible] we have the problem we are about to solve. Our requirement is to find the shortest path starting from node "S" to node "G". We can see the graph and how the nodes are connected. Also, we can see the table, containing the heuristic values for each of the nodes.

The Problem of Simplified Memory Bounded A*

SMA* by Example





Generate children
(One by one)

Image 6: Evaluating The Children

On *Image 6* we can see how the algorithm evaluates the children. Firstly it added the "A" and "B" nodes (starting alphabetically). On the right of the node in blue color is the f value, on the left is the g value or the cost to reach that node(in green color), and the h value or the heuristic (in red color).

SMA* by Example

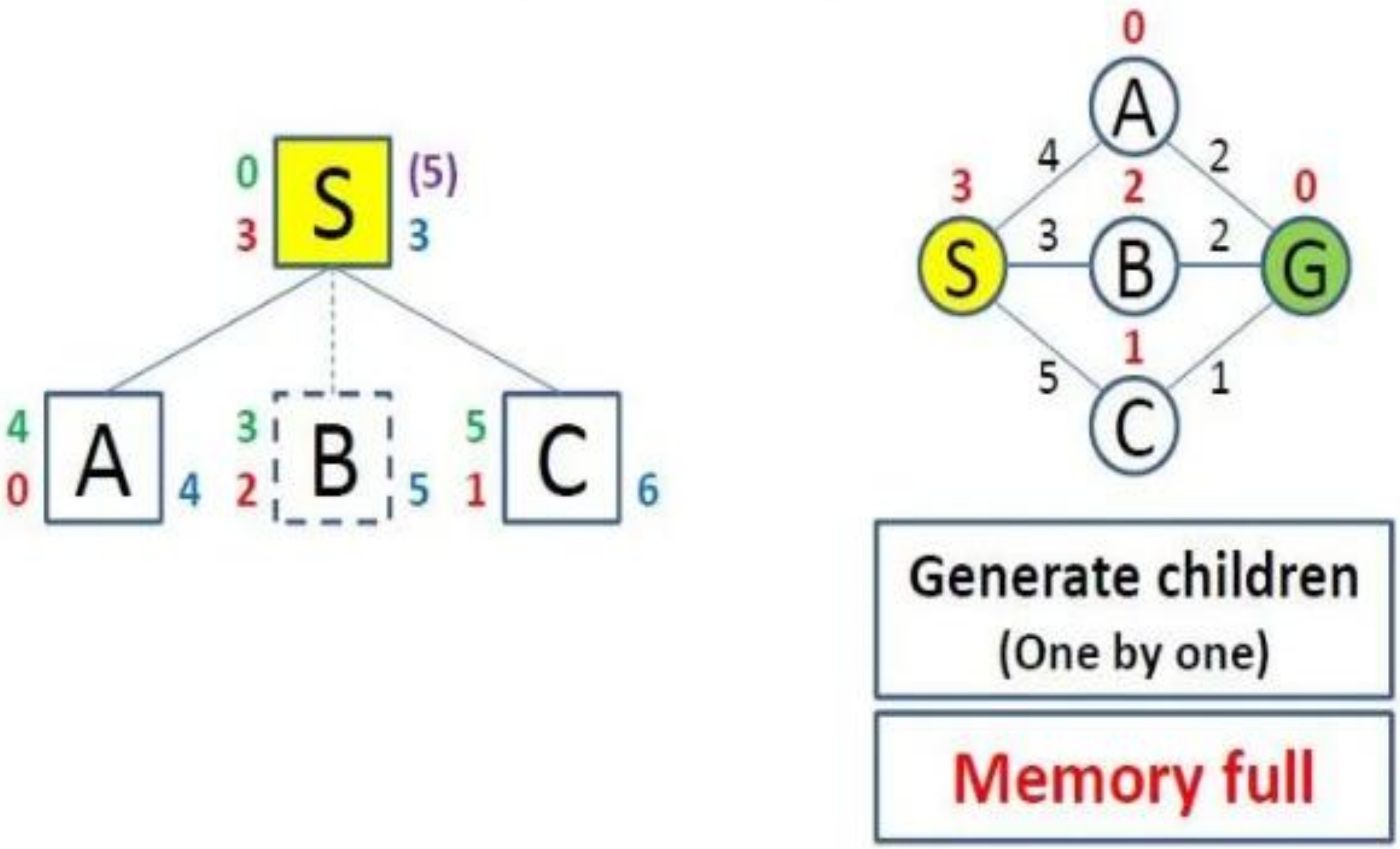
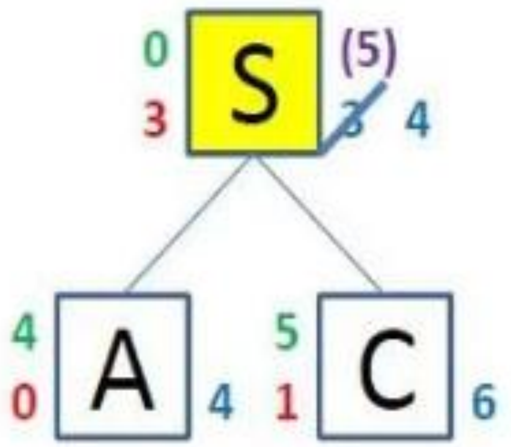
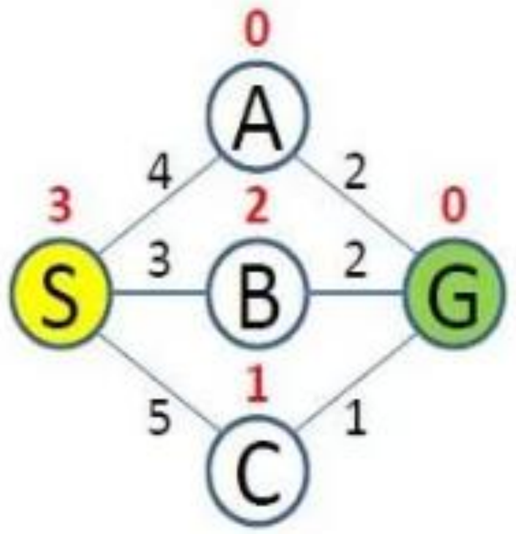


Image 7: Full Memory While Evaluation

On *Image 7* we can see what happens when you want to evaluate more nodes than what the memory allows. The already evaluated children with the highest f value get removed, but its value is remembered in its parent(the number in purple color on the right of the “S” node).

SMA* by Example





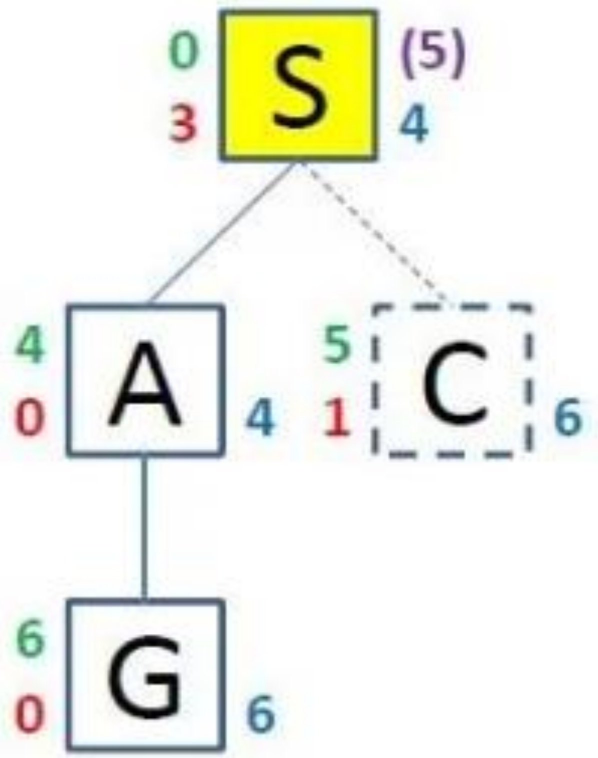
All children are explored

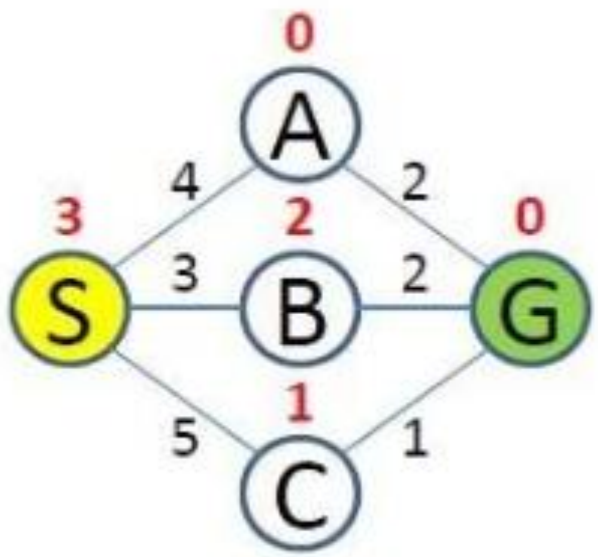
Adjust f-values

Image 8: Adjusting The f Value

On *Image 8* we can see that if all accessible children nodes are visited or explored we adjust the parent's f value to the value of the children with the lowest f value.

SMA* by Example





Generate children
(One by one)

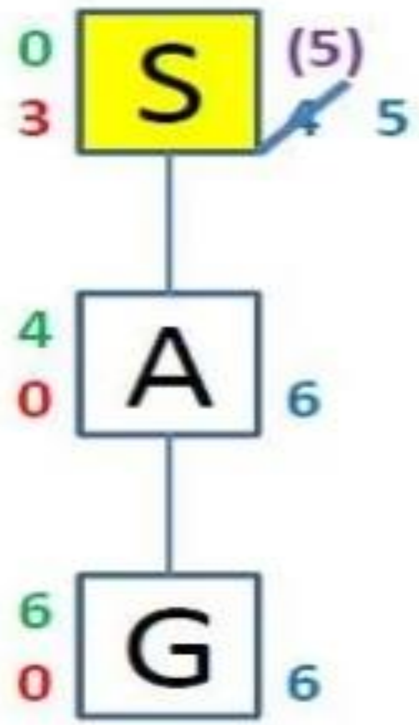
Memory full

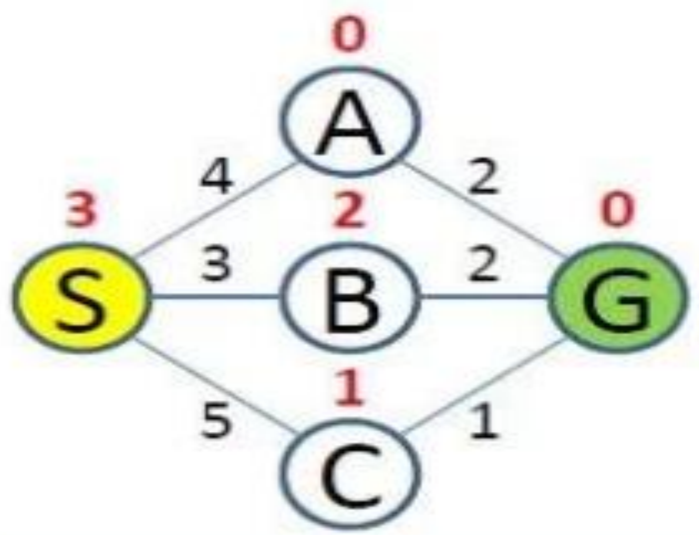
Image 9: Memory Full Evaluate Through The Lowest f Value Child

On *Image 9* we can see how the algorithm evaluates the children of the “A” node. The only children is the “G” node which is the goal, which means there is now a way for exploring further.

Now we remove the “C” node, since it has the highest f value but we don’t remember it, since the “B” node has a lower value. Now we update the f value of the “A” node to 6 because all the children accessible are explored.

SMA* by Example





All children are explored (update)

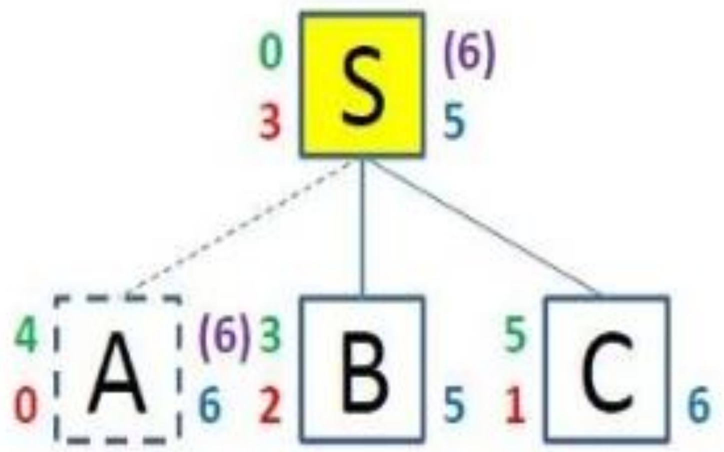
Adjust f-values

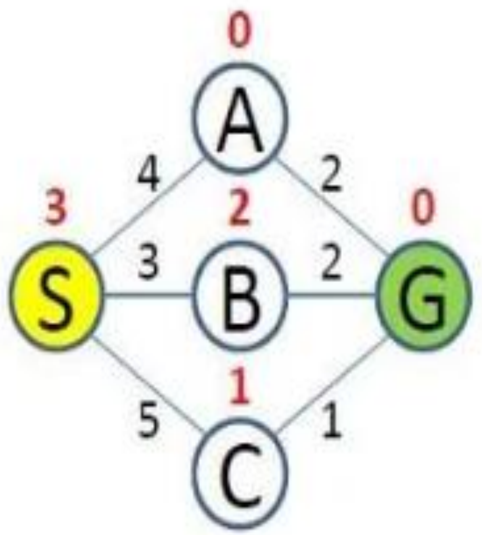
Image 10: Update The Root Value ("S" node) To The Remembered Value

Just because we've reached the goal node, doesn't mean that the algorithms are finished. As we can see the remembered f value of the "B" node is lower than the f value of the "G" node explored through the "A" node.

This gives hope that we might reach the goal node again for a lower f value or lower total cost. We also update the f value of the "S" node to 5 since there is no more child node with value 4.

SMA* by Example





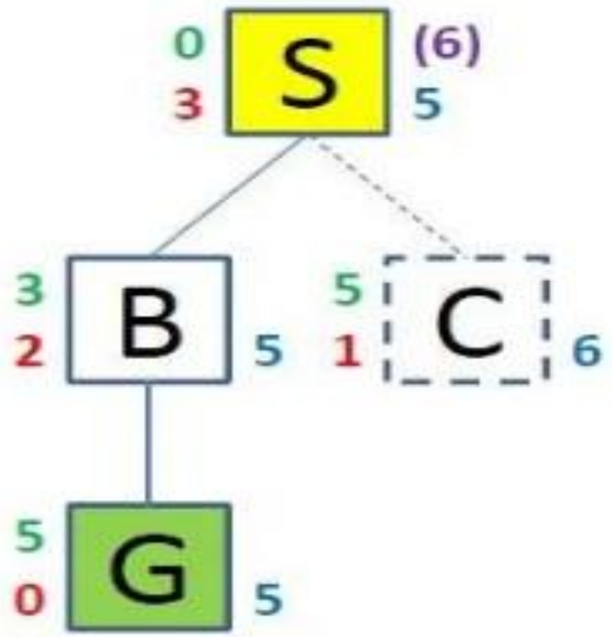
Generate children
(One by one)

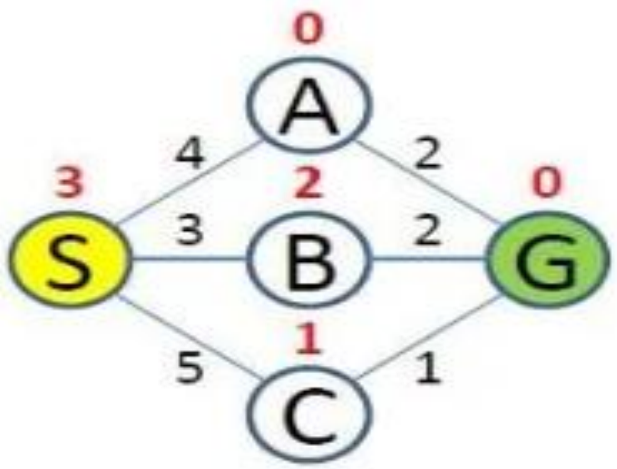
Memory full

Image 11: Remove The Goal Leading Node

As we can see on *Image 11* we will remove the "A" node since it has already discovered the goal and has the same f value as the "C" node so we will remember this node in the parent "S" node.

SMA* by Example





Generate children
(One by one)

Memory full

Image 12: Reaching The Goal For The Lowest Total Cost

On *Image 12* we can see that we've reached the goal again, this time through the "B" node. This time cost is lower than through the "A" node.

The memory is full, which means we need to remove the "C" node, because it has the highest f value, and it means that if we eventually reach the goal from this node the total cost at best case scenario will be equal to 6, which is worse than we already have.

At this stage, the algorithm terminates, and we have found the shortest path from the "S" node to the "G" node.

ADVANTAGES OF SMA*

-It will utilize whatever memory is made available to it.

-It avoids repeated states as far as its memory allows.

-It is complete if the available memory sufficient to store the shallowest solution path.

-It is optimal if enough memory is available to store the shallowest optimal solution path. Otherwise, it returns the best solution that can be reached , with the available memory.

-It is a very popular graph pathfinding algorithm, mainly because it is based on the A* algorithm, which already does a very good job of exploring the shortest path in graphs. This algorithm is better than A* because it is memory-optimized.