ARTIFICIAL INTELLIGENCE (A.I.)

5. Iterative Deepening A*

IDA* is a combination of the depth first iterative deepening and A* algorithm.

ALGORITHM :

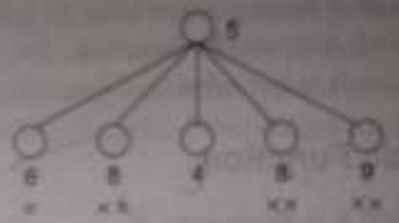
1. For each iteration perform a DFS pruning off a branch when its total cost (g+h) exceeds a given threshold.

2. The initial threshold starts at the estimate cost of the start state and increases for each iteration of the algorithm.

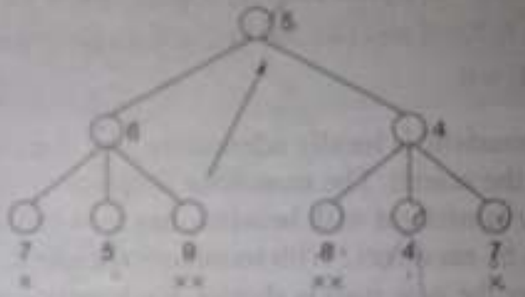
3. The threshold used for the next iteration is the minimum cost of all values exceeded the current threshold.

4. These steps are repeated till we find a goal state.

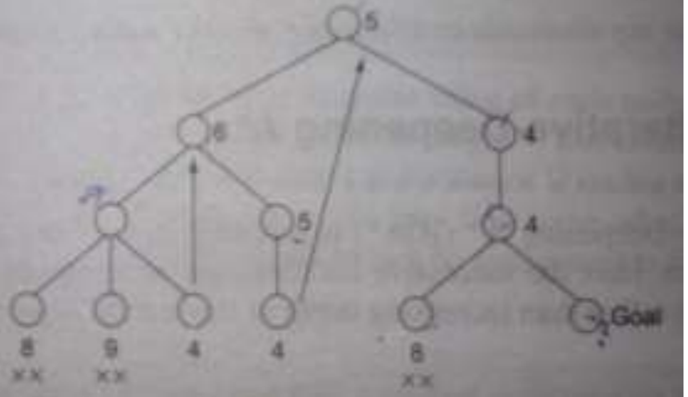
1st iteration (Threshhold = 5)



2nd iteration (Threshold = 6)



3rd iteration (Threshold = 7)



7. SMA* (Simplified Memory Bounded A*): is a shortest path algorithm that is based on the A* algorithm. The difference between SMA* and A* is that SMA* uses a bounded memory, while the A* algorithm might need exponential memory.

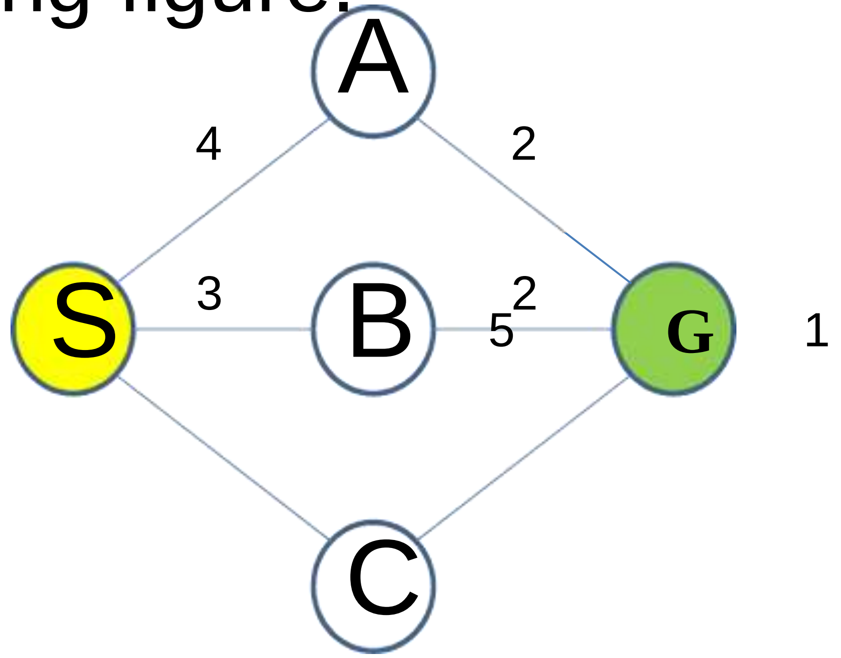
Like the A*, it expands the most promising branches according to the heuristic and evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal: **f(n) = g(n) + h(n)**

Algorithm

1. At each stage, one successor is added to the deepest lowest-f-cost node that has some successors not currently in the tree. The left child B is added to the root A.
2. Now $f(A)$ is still 12, so we add the right child G ($f = 13$). Now that we have seen all the children of A, we can update its f-cost to the minimum of its children, that is, 13. The memory is now full.
3. G is now designated for expansion, but we must first drop a node to make room. We drop the shallowest highest-f-cost leaf, that is, B. When we have done this, we note that A's best forgotten descendant has $f = 15$, as shown in parentheses. We then add H, with $f(H) = 18$. Unfortunately, H is not a goal node, but the path to H uses up all the available memory. Hence, there is no way to find a solution through H, so we set $f'(H) = \infty$.
4. G is expanded again. We drop H, and add I, with $f(I) = 24$. Now we have seen both successors of G, with values of oo and 24, so $f(G)$ becomes 24. $f(A)$ becomes 15, the minimum of 15 (forgotten successor value) and 24. Notice that I is a goal node, but it might not be the best solution because A's f-cost is only 15.
5. A is once again the most promising node, so B is generated for the second time. We have found that the path through G was not so great after all.
6. C, the first successor of B, is a nongoal node at the maximum depth, so $f(C)$ = oo.
7. To look at the second successor, D, we first drop C. Then $f(D) = 20$, and this value is inherited by B and A.
8. Now the deepest, lowest-f-cost node is D. D is therefore selected, and because it is a goal node, the search terminates.

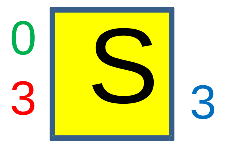
SMA* by Example

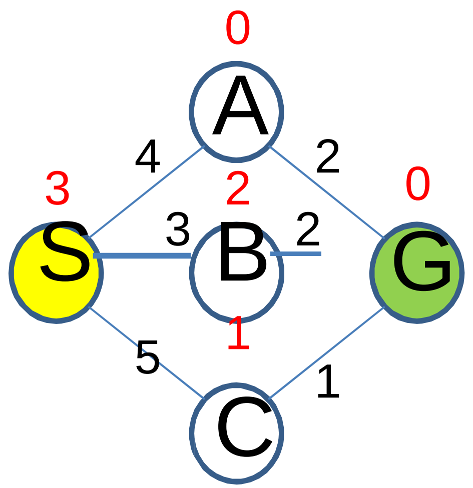
- Perform SMA* (memory: 3 nodes) on the following figure.



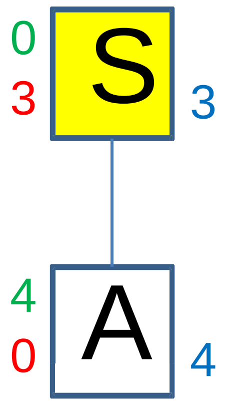
	S	A	B	C	G
heuristic	3	0	2	1	0

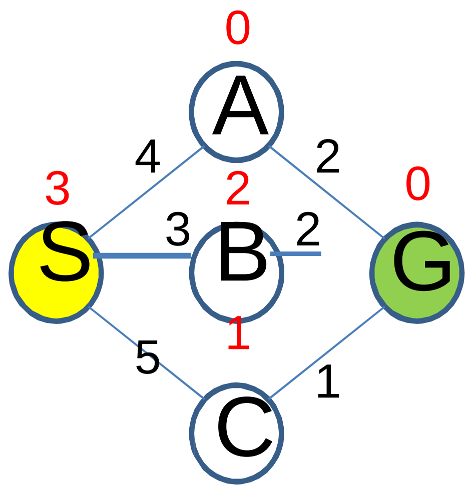
SMA* by Example





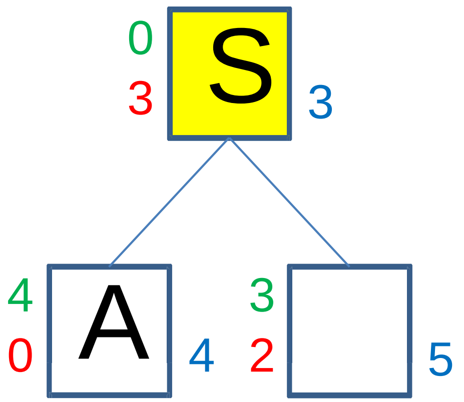
SMA* by Example

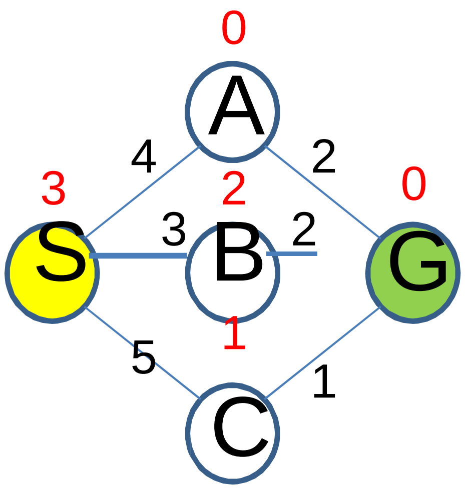




Generate children
(One by one)

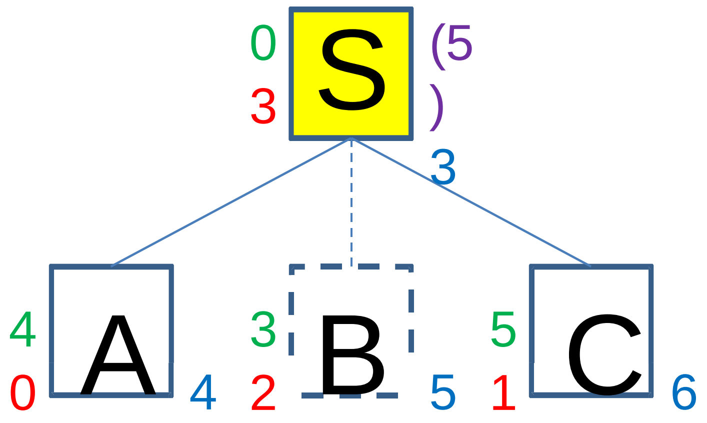
SMA* by Example

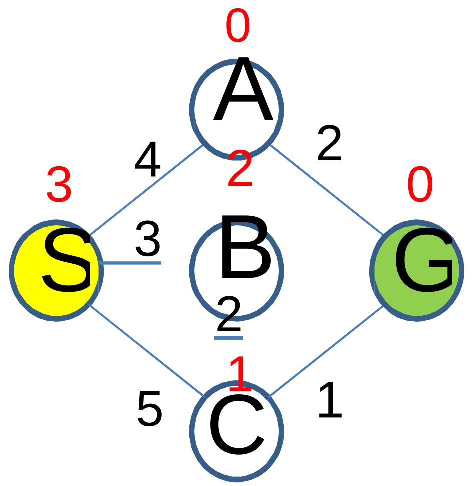




Generate children
(One by one)

SMA* by Example

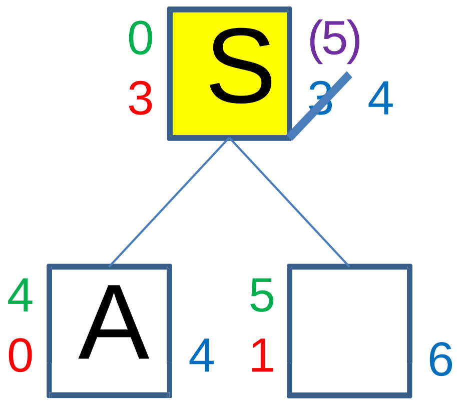


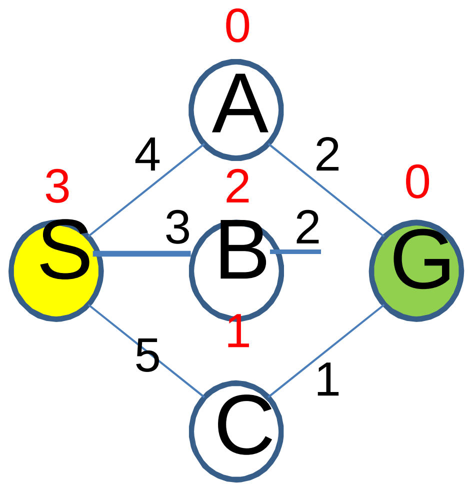


Generate children
(One by one)

Memory full

SMA* by Example

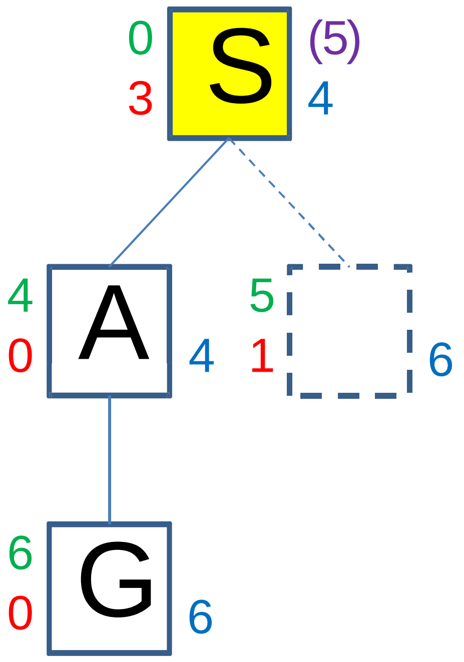


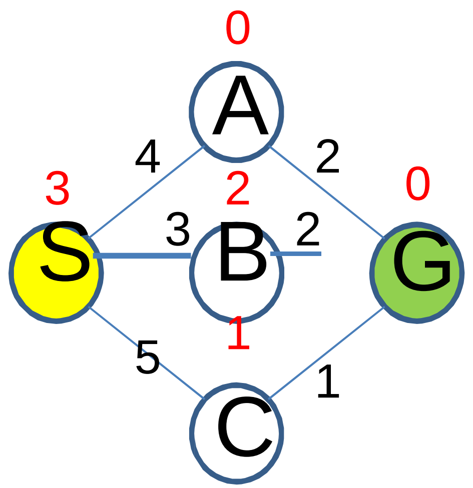


All children are explored

Adjust f-values

SMA* by Example

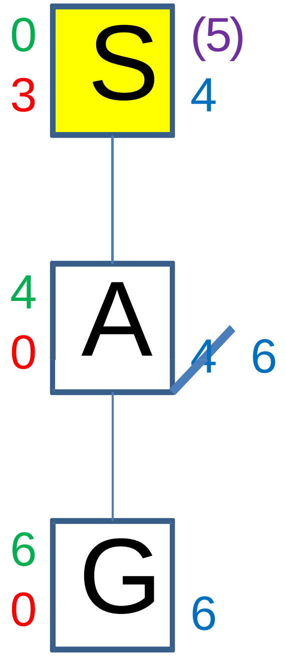


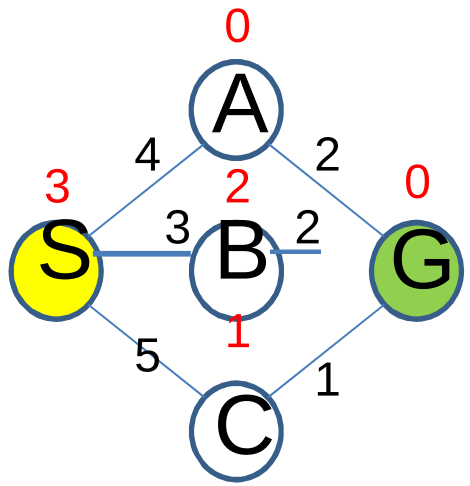


Generate children
(One by one)

Memory full

SMA* by Example

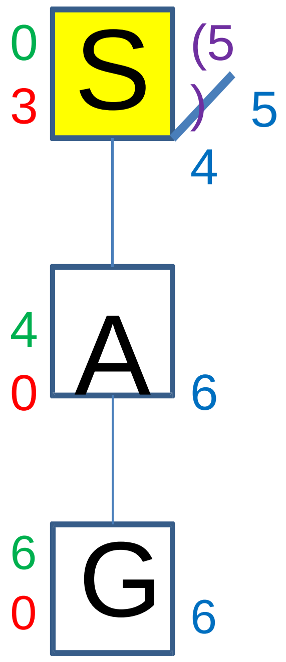


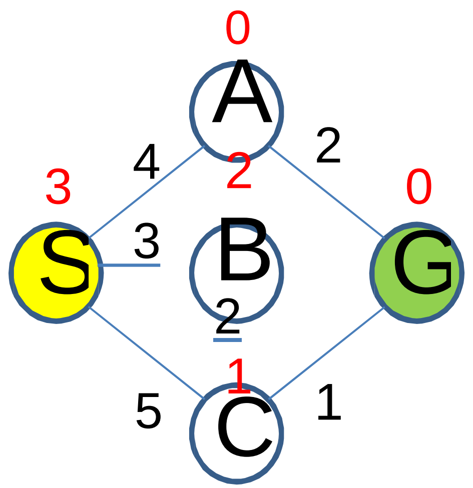


All children are explored

Adjust f-values

SMA* by Example

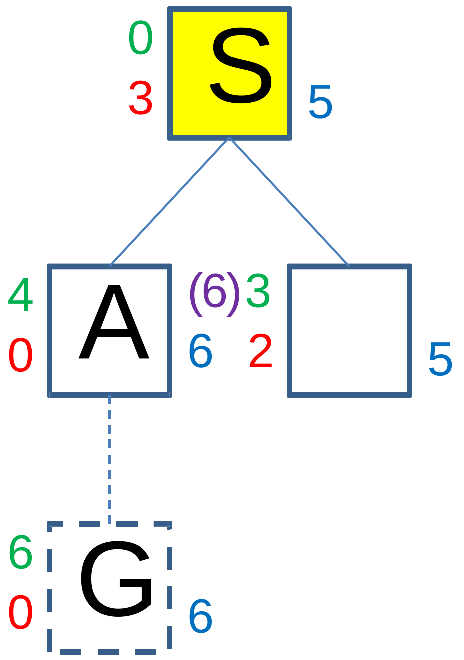


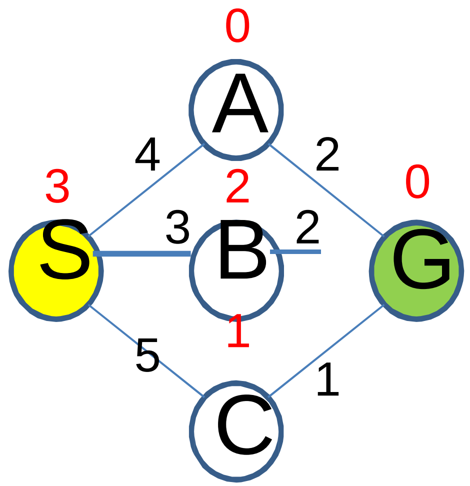


All children are explored (update)

Adjust f-values

SMA* by Example

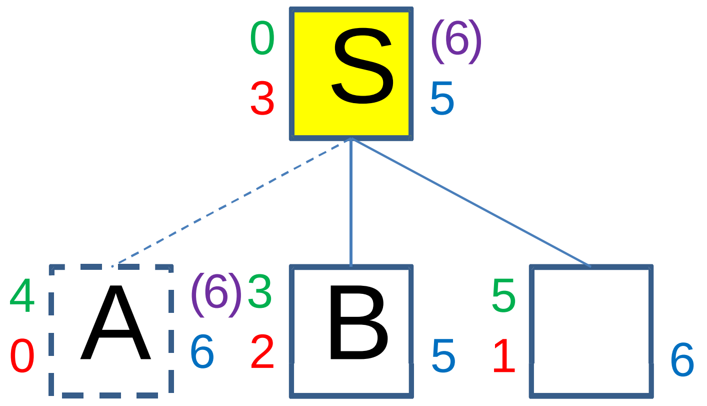


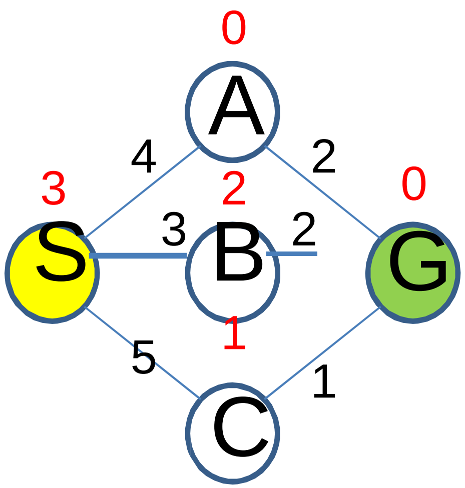


Generate children
(One by one)

Memory full

SMA* by Example

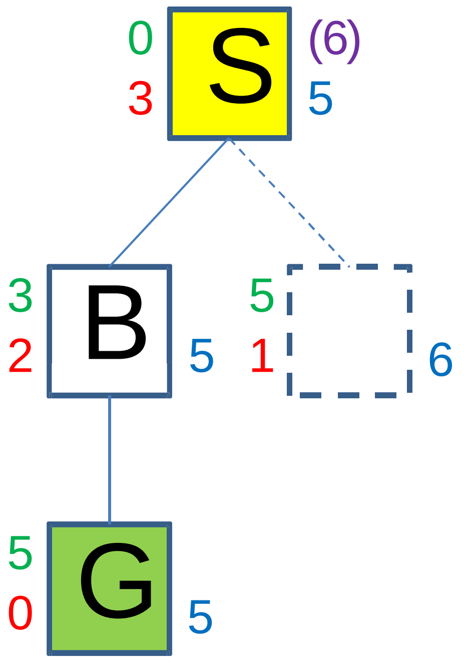


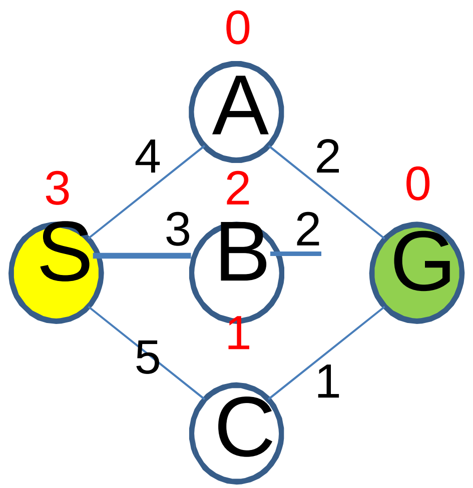


Generate children
(One by one)

Memory full

SMA* by Example





Generate children
(One by one)

Memory full