Artificial Intelligence
Homework no. 1

Alireza Rostami
Student Number: 9832090

Find the LaTeX code of this masterpiece at my Github @ github.com/WellOfSorrows.

Question 1

1.1 ***Tennis Against the Wall***

Specifying PEAS:

Performance Measure	**Environment**	**Actuators**	**Sensors**
hit speed, hit accuracy	*playground, racket, ball, wall*	*ball, racket, joint arm*	*ball locator, camera, racket sensor*

Environment Types:

Observable?	**Deterministic/Stochastic**	**Episodic/Sequential**	**Static/Dynamic**	**Discrete/Continuous**	**Single/Multi-agent**
Yes	*Partly*	*Sequential*	*Partly*	*Continuous*	*Single*

1.2 ***Digikala Site***

Specifying PEAS:

Performance Measure	**Environment**	**Actuators**	**Sensors**
price, quality seller, product review	*web, vendors, shippers*	*fill-in form, follow URL, display to user*	*HTML*

Environment Types:

Observable?	**Deterministic/Stochastic**	**Episodic/Sequential**	**Static/Dynamic**	**Discrete/Continuous**	**Single/Multi-agent**
No	*Partly*	*Sequential*	*Partly*	*Discrete*	*Multi*

Question 2

2.1 Playing soccer agent

Goal-based. Since the agent only cares about scoring more goals in the game; it will sense the environment and perform actions needed to score more.

2.2 Exploring the subsurface ocean of Titan agent

Utility-based. Because of the uncertainty in the world, achieving the desired goal is not enough. We may look for quicker, safer, cheaper ways to achieve our goal. So our agent must choose the action that maximizes the expected utility. Therefore, it is a utility-based agent.

2.3 Bidding on an item at an auction agent

Simple reflex agent. Since the environment if fully observable and can only perform one action, that being bidding an higher amount than the amount bid by the previous agent. So our agent only cares about the current precept, ignoring the rest of the precept history. Therefore, it is a simple reflex agent.

Question 3

3.1 BFS

Yes. BFS is complete and is guaranteed to give us an answer. Since BFS probes node level by level, we can also ascertain which password it finds first. The breadth-first search for this specific problem will find `"CBAC"` since it is the shortest password among other candidate passwords.

3.2 DFS

No, in the general sense. DFS is not complete and not guaranteed to give an answer. If we call the successor function recursively without keeping in mind that the password is at most 10 characters long, then the tree would be infinite and DFS will be producing `"AAAAAA···"`. But, if we set a limit of 10 for DFS, then yes. If given a bound equal to 10, DFS would return the password `"AAACCC"` since ties are resolved alphabetically and the string `"AAACCC"` has the most initial A's among all other passwords, hence bounded DFS finding `"AAACCC"` as its solution.

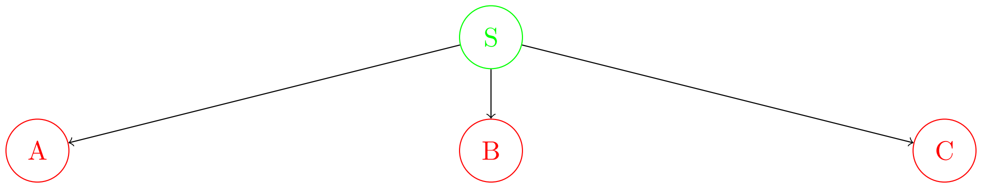
3.3 UCS

Yes. UCS is complete and gives the optimal solution. Given the model and the cost function specified in the question, the password returned by the UCS is `"BABAB"` with cost of 8, since it has 3 B's and 2 A's. The costs of all passwords would be (with respect to the specified order in the question): $\{12, 11, 8, 14, 9, 11\}$.

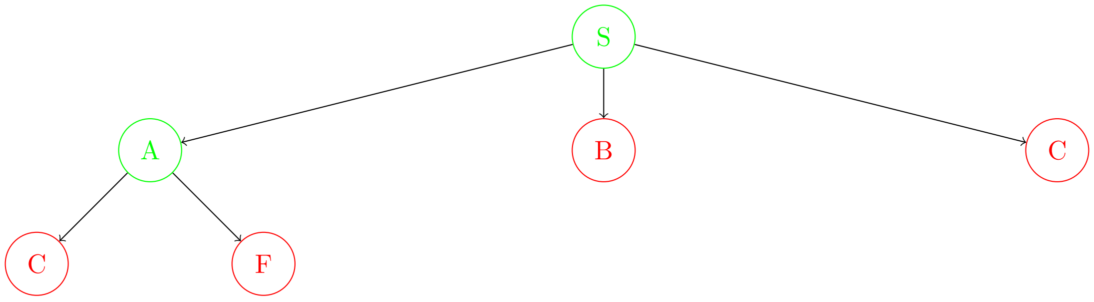
Question 4

4.1 Breadth-first Search

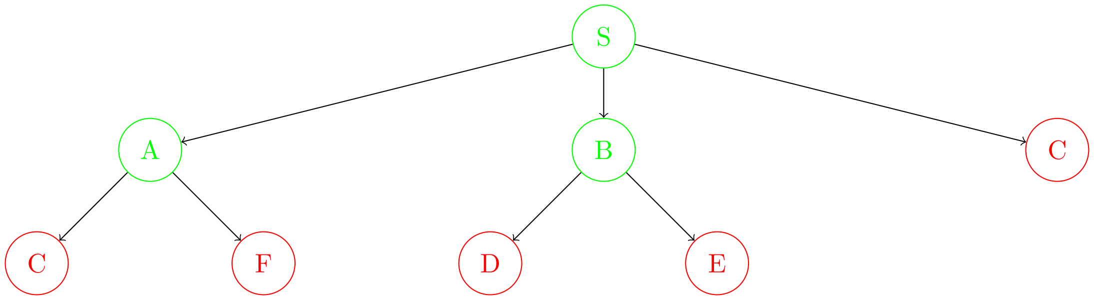
Step 1. We start expanding from the node S.



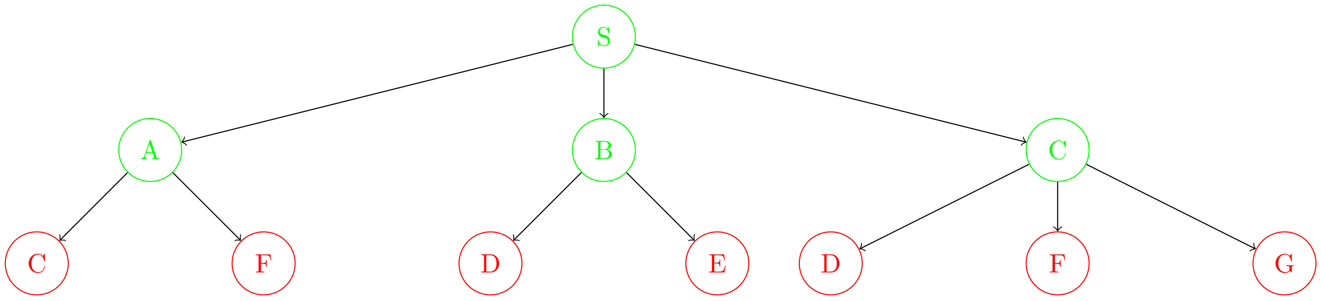
Step 2. We expand the node A, since it comes alphabetically before B and C. Since the node S is already expanded, we will not write it again.



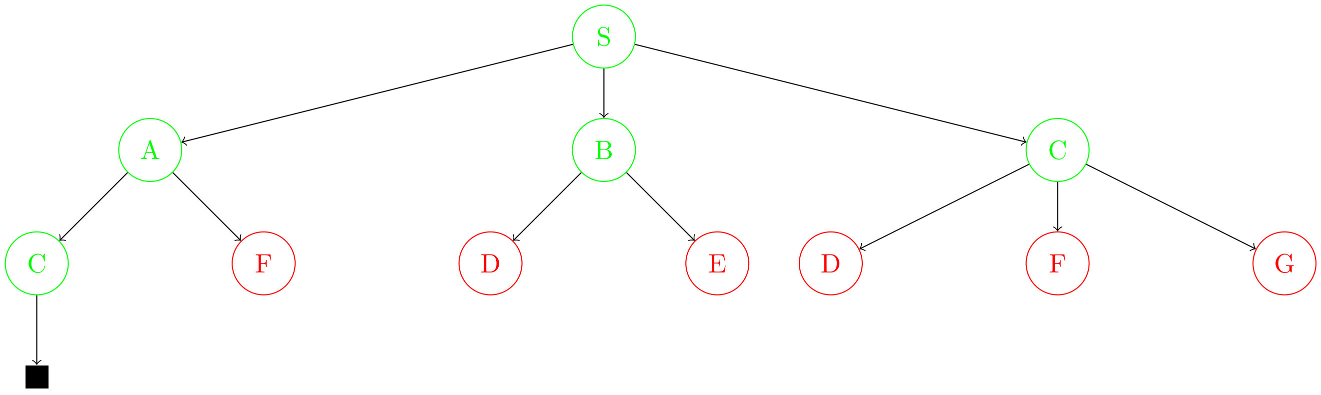
Step 3. Now we expand the node B.



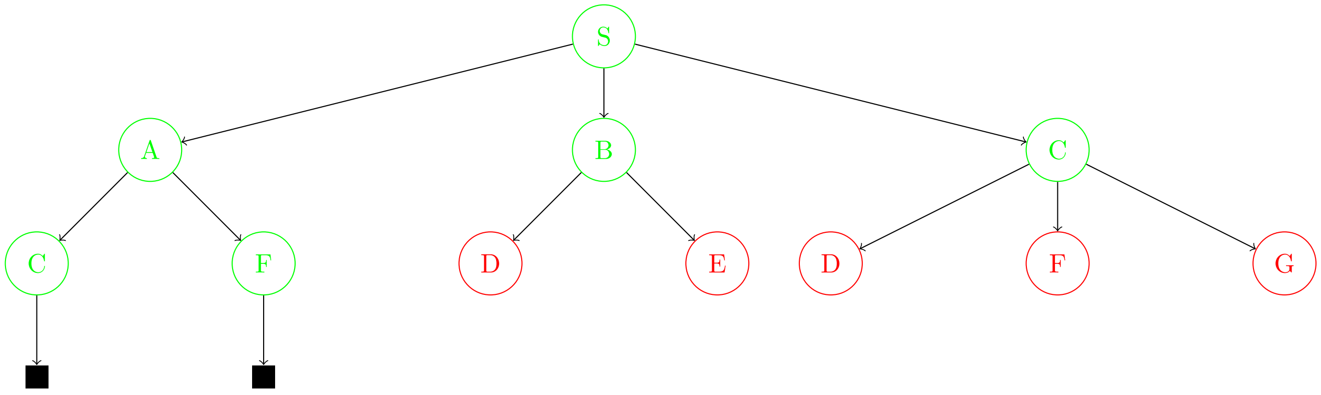
Step 4. We expand the node C.



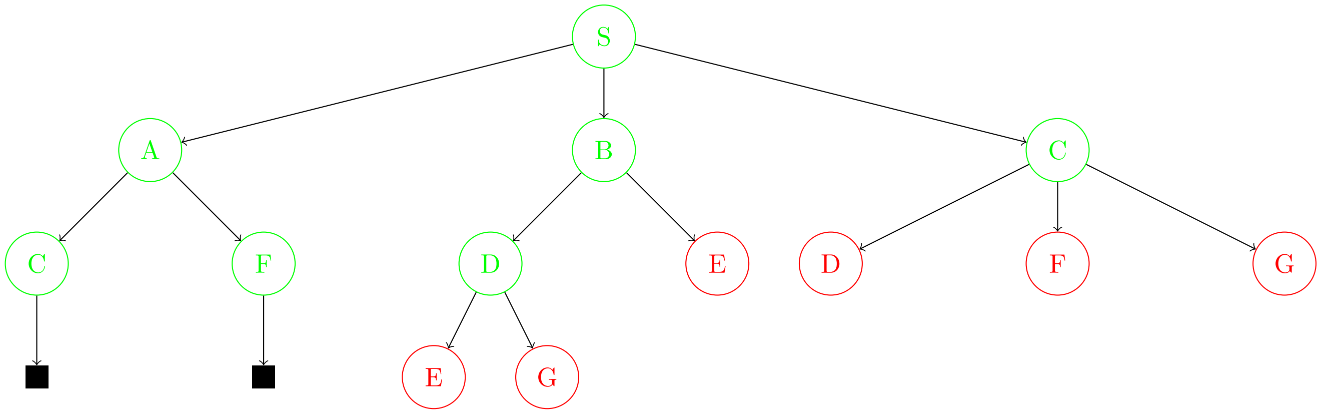
Step 5. Now we try to expand the node C. Since C is already expanded, we will not expand it again and we reach the end of a branch.



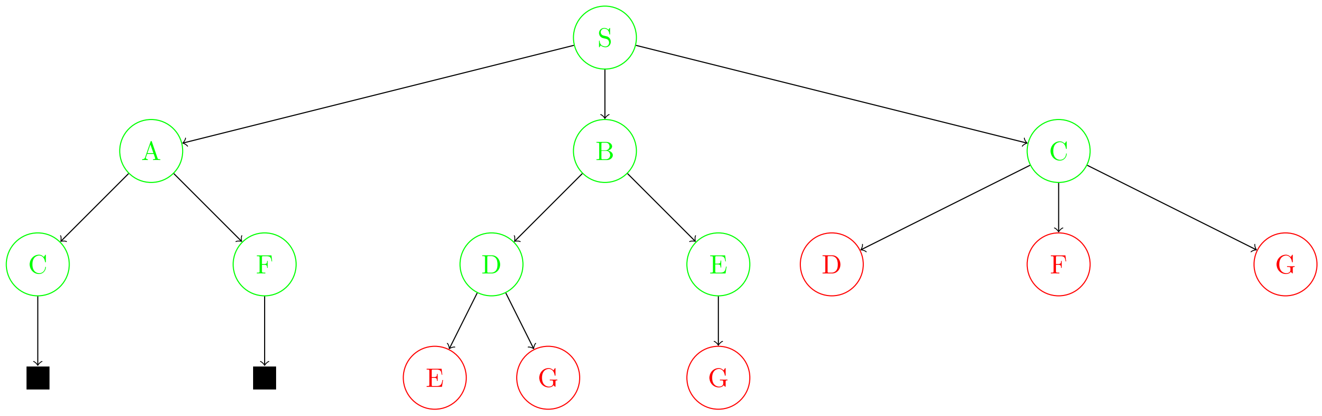
Step 6. We expand the node F. Since the children of F are the nodes A and C, and they both have been visited, we will not write them again. So F also results in an end of a branch.



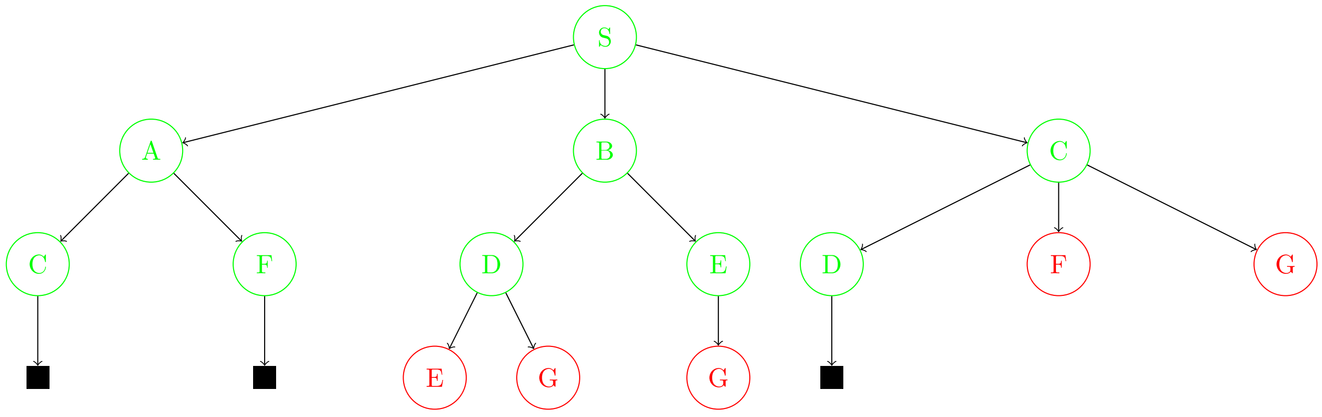
Step 7. We expand the node D. Two of the children of D, namely C and B, have been visited, we will not write them again.



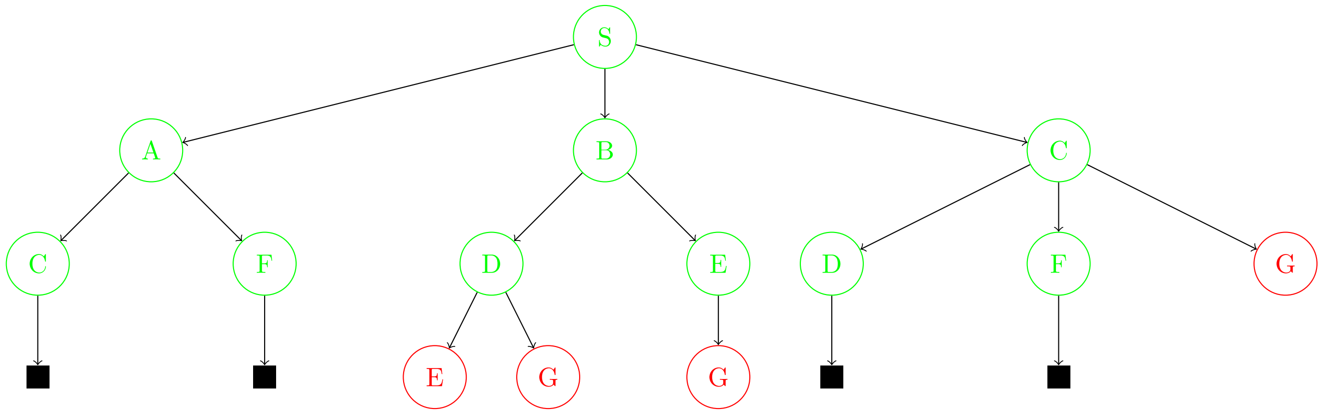
Step 8. We expand the node E. Two of the children of E, namely B and D, have been visited, we will not write them again.



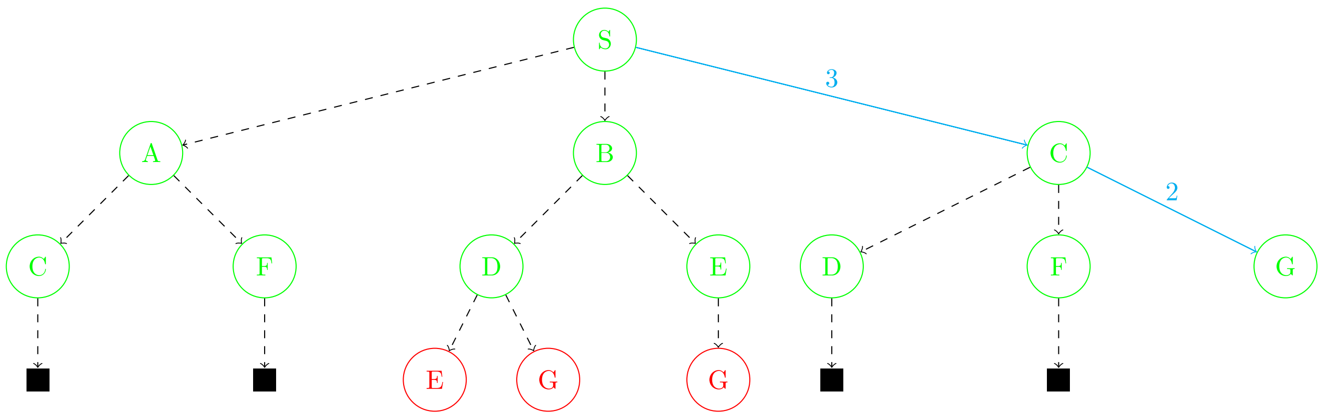
Step 9. We try to expand the node D. It has been expanded before, so it results in an end.



Step 10. We try to expand the node F. It has been expanded before, so no need to expand it again. So it results in an end.



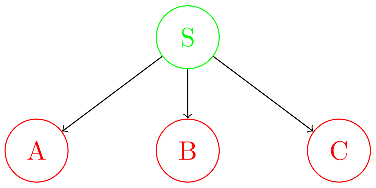
Step 11. We try to expand the node G. Since G is the end goal, the search is over. The path obtained from breadth-first search is show by the cyan line.



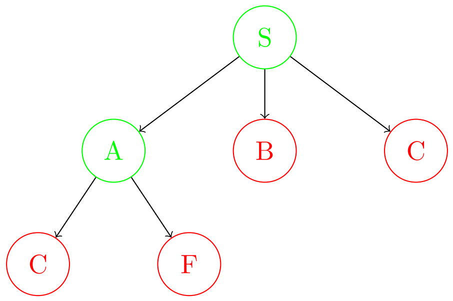
$$\Rightarrow Cost = 5$$

4.2 Depth-first Search

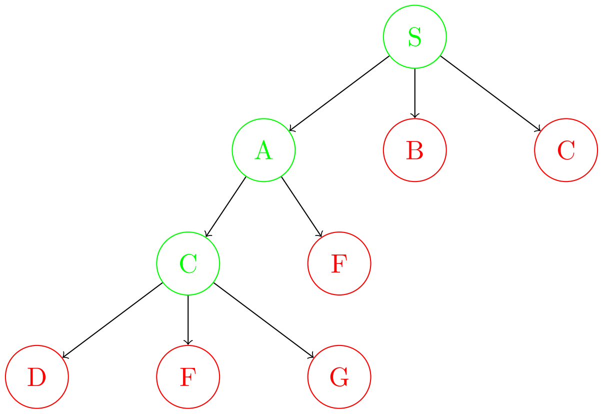
Step 1. We start expanding from the node S.



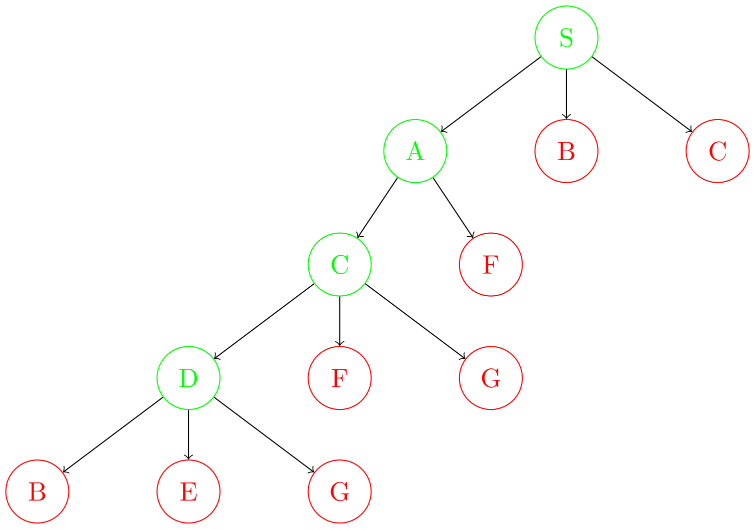
Step 2. We expand the node A, since it comes alphabetically before B and C. Since the node S is already expanded, we will not write it again.



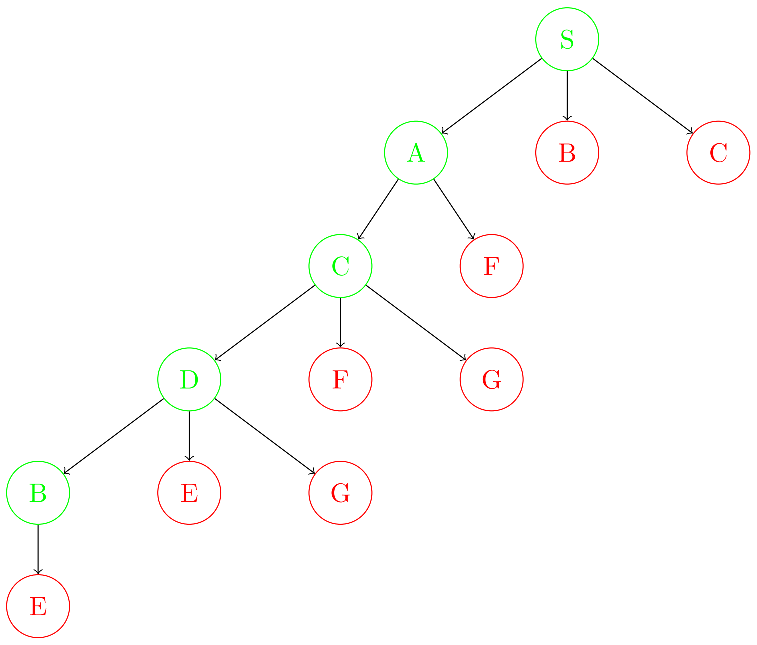
Step 3. We now expand the node C, since it comes alphabetically before node F.



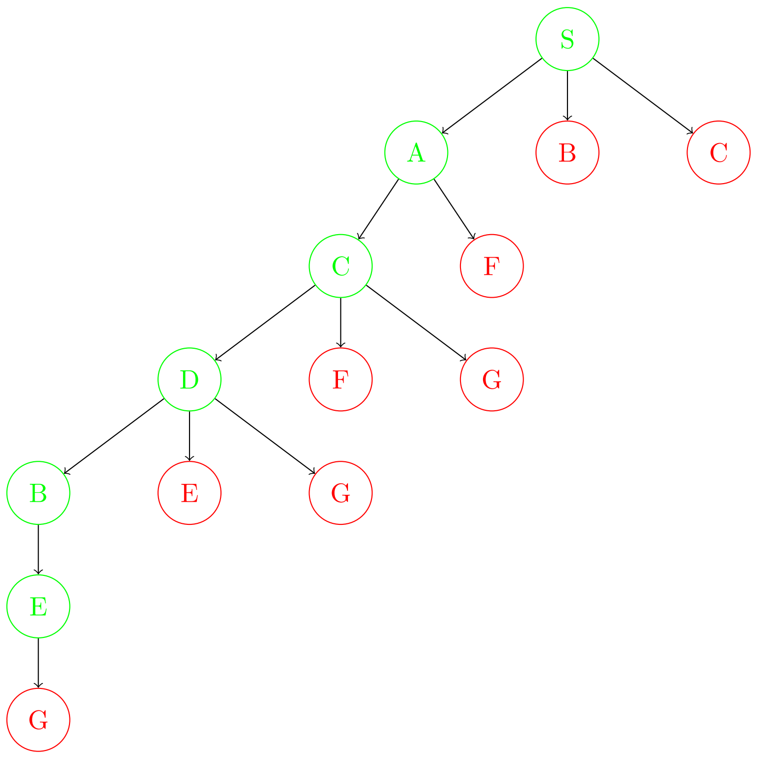
Step 4. We expand the node D, since it comes alphabetically before node F and node G.



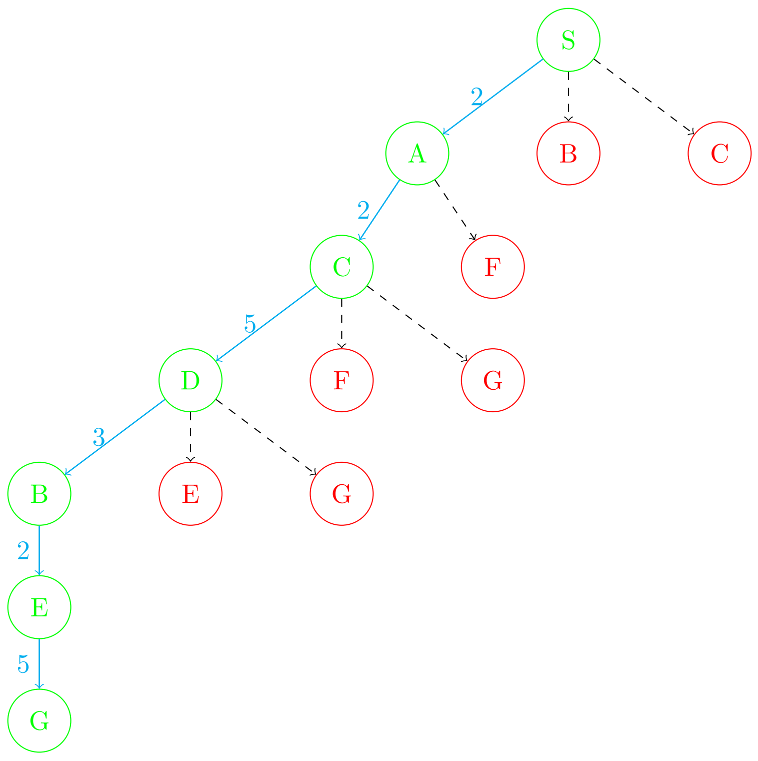
Step 5. We expand the node B, since it comes alphabetically before node E and node G.



Step 6. We expand the node E.



Step 7. We have reached the end goal G, thus the search is over. The path returned by the depth-first search is shown by the cyan line.



$$\Rightarrow Cost = 19$$

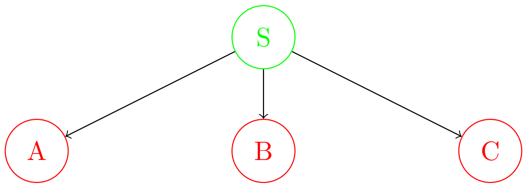
4.3 Iterative Deepening Search

Depth Bound = 0

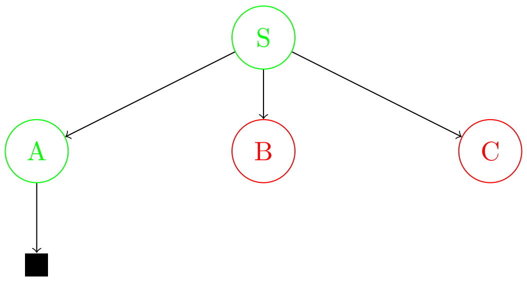
Since the node S is the starting node and it is not our goal, thus we need to increment the depth bound.

Depth Bound = 1

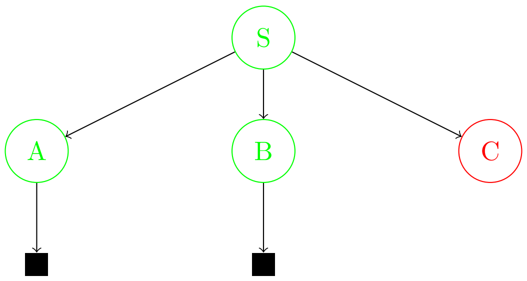
Step 1. We expand the node S.



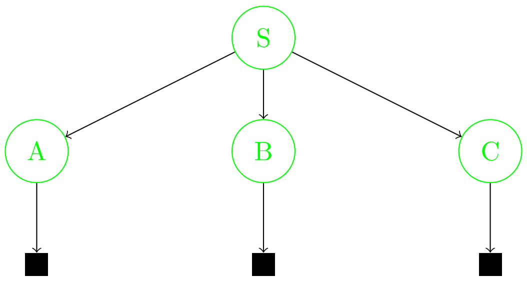
Step 2. We now try to expand the node A, since it comes alphabetically before nodes B and C. Since our maximum depth bound is 1, so we cannot expand A.



Step 3. We now try to expand the node B. Since our maximum depth bound is 1, so we cannot expand node B neither.

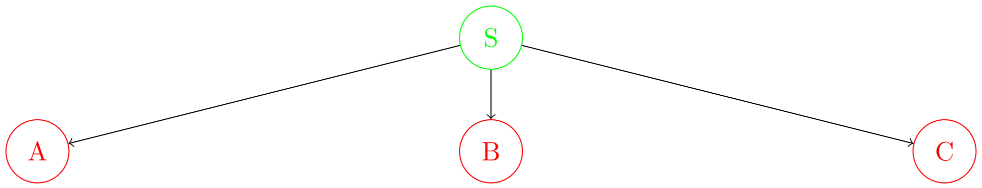


Step 4. We now try to expand the node C. Since our maximum depth bound is 1, so we cannot expand node C neither. We did not reach the goal node, so the depth must be incremented.

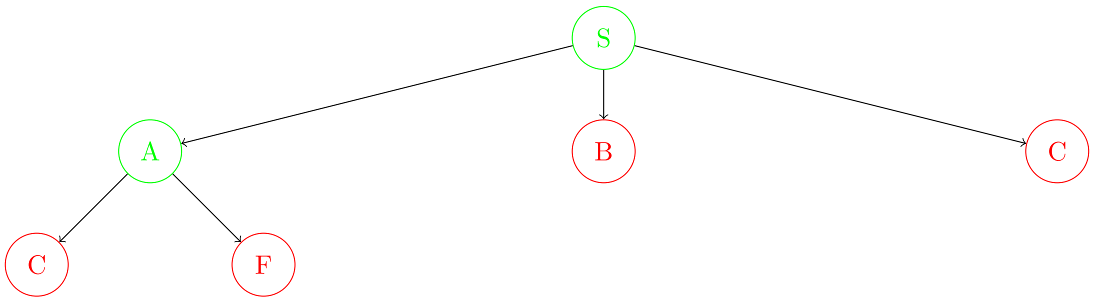


Depth Bound = 2

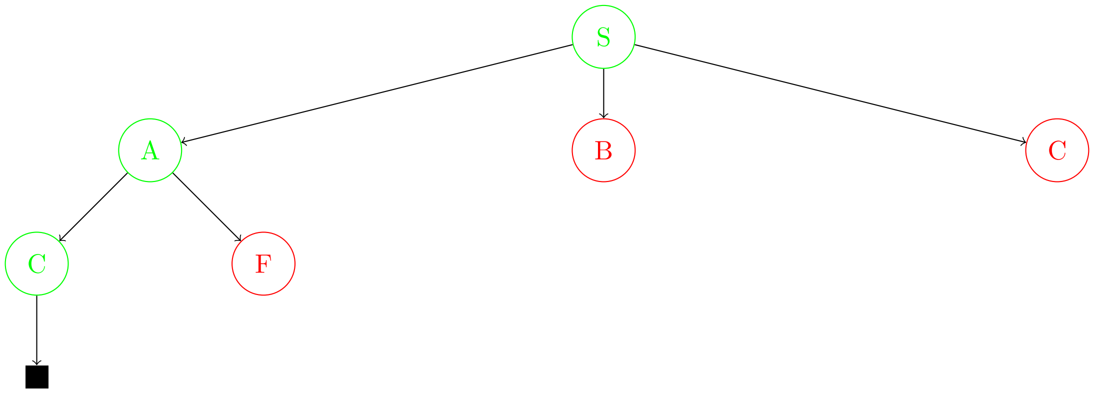
Step 1. We expand the node S.



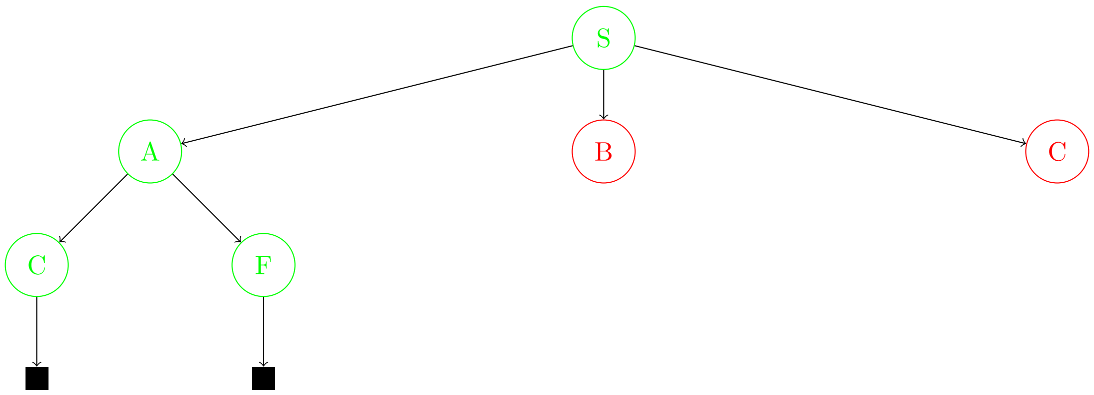
Step 2. We now try to expand the node A, since it comes alphabetically before nodes B and C. Node S has already been expanded so no need to write it again.



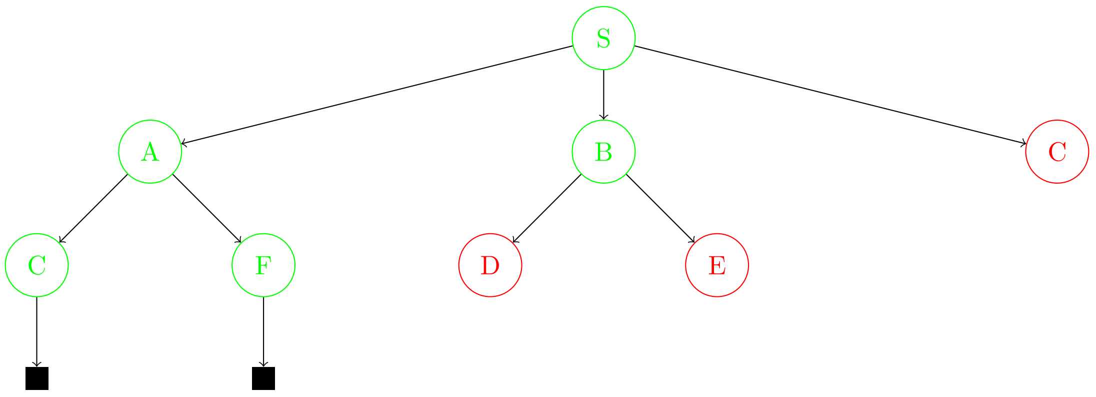
Step 3. We now try to expand the node C, since it comes alphabetically before node F. We have reached the maximum depth, and C is not our goal, so we fail to expand C and we reach end of the branch.



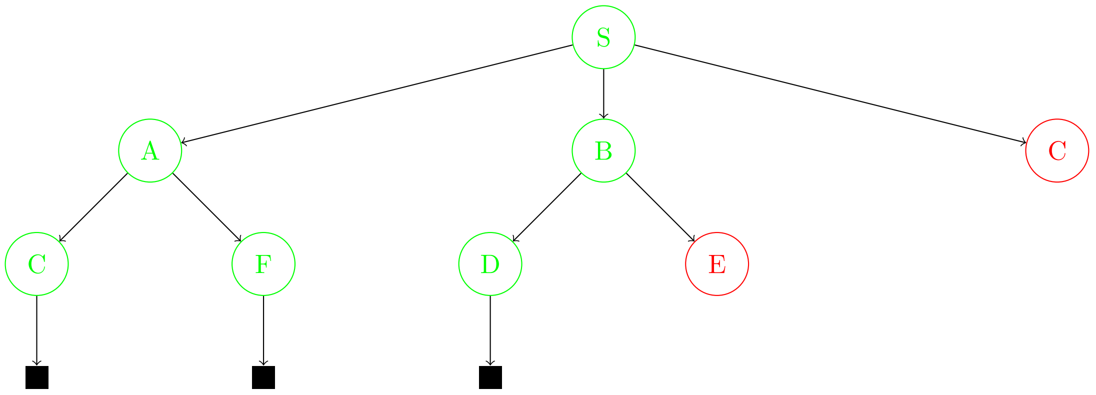
Step 4. We now try to expand the node F. We have reached the maximum depth, and F is not our goal, so we fail to expand F and we reach end of the branch.



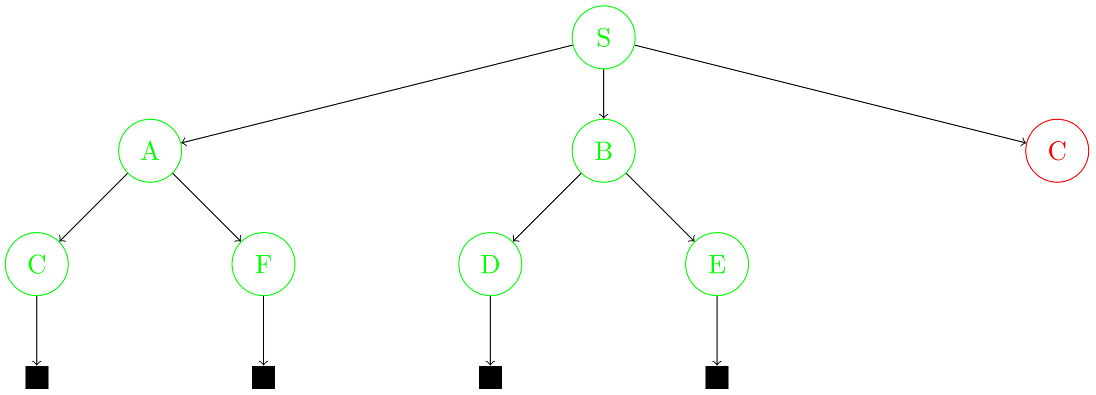
Step 5. We now expand the node B.



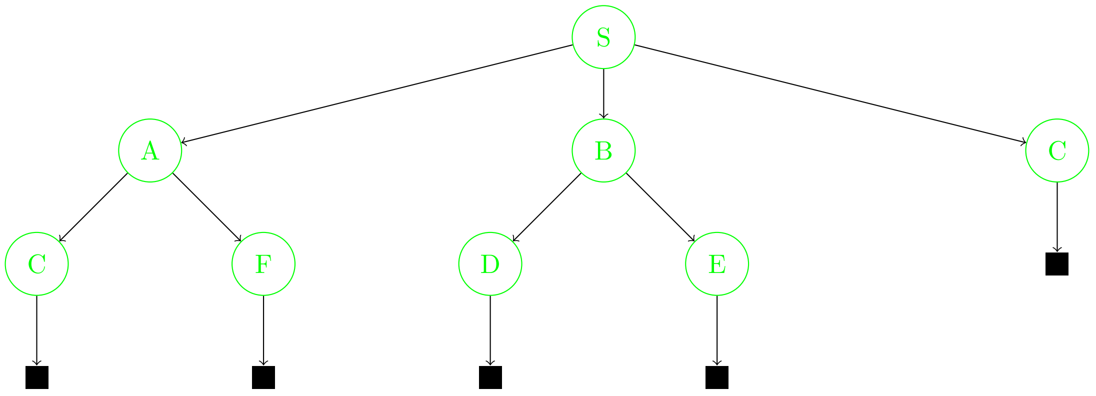
Step 6. We try to expand D. We have reached the maximum depth, and D is not our goal, so we fail to expand D and we reach end of the branch.



Step 7. We try to expand E. We have reached the maximum depth, and E is not our goal, so we fail to expand E and we reach end of the branch.

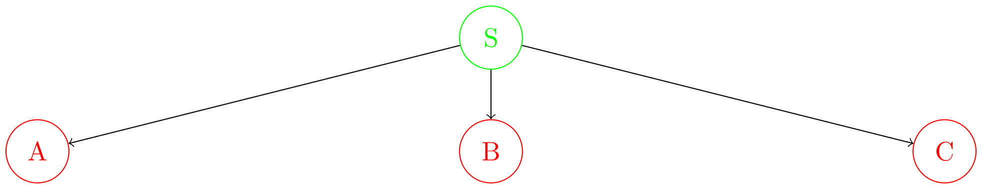


Step 8. We try to expand C. Since C was already expanded, we cannot expand it again.

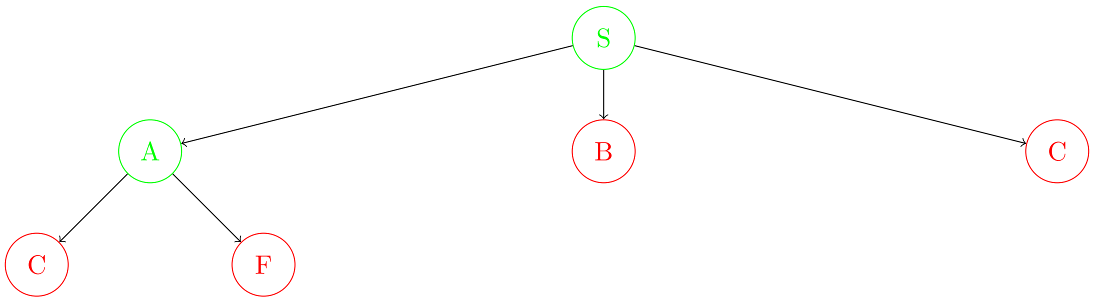


Depth Bound = 3

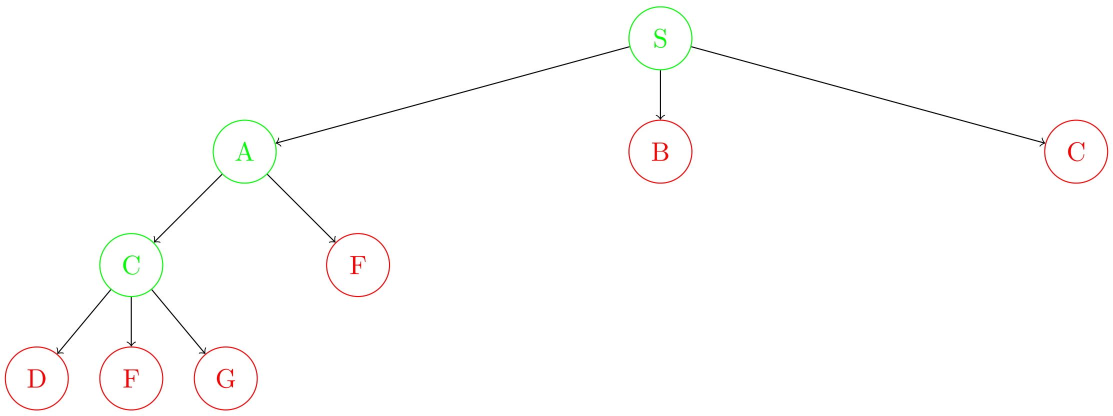
Step 1. We expand the node S.



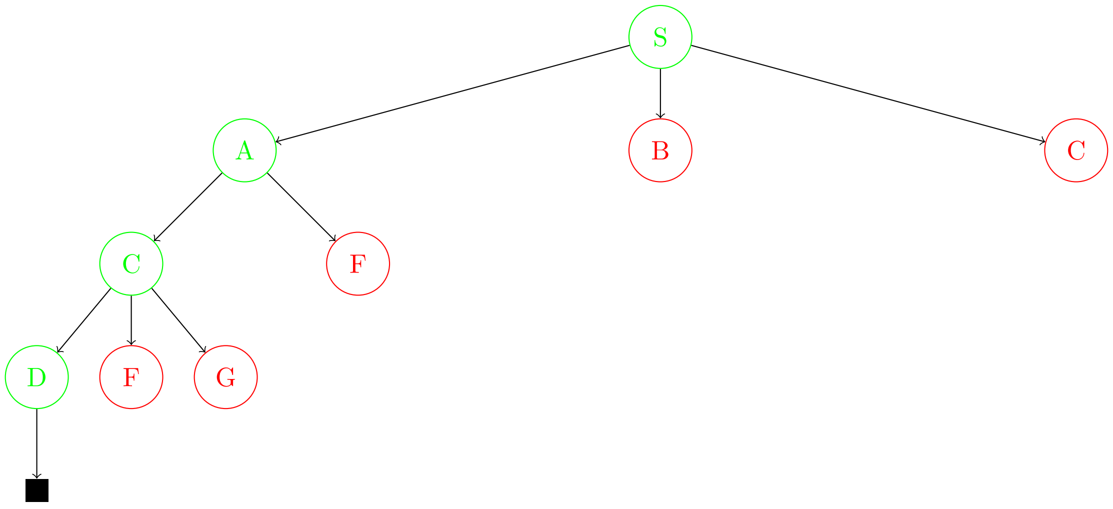
Step 2. We now try to expand the node A, since it comes alphabetically before nodes B and C. Node S has already been expanded so no need to write it again.



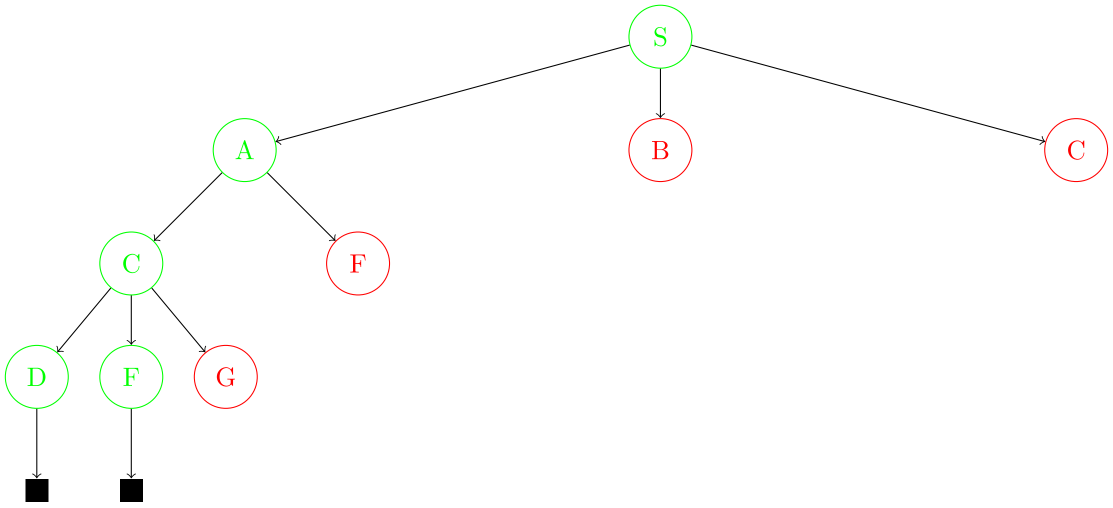
Step 3. We now try to expand the node C, since it comes alphabetically before node F.



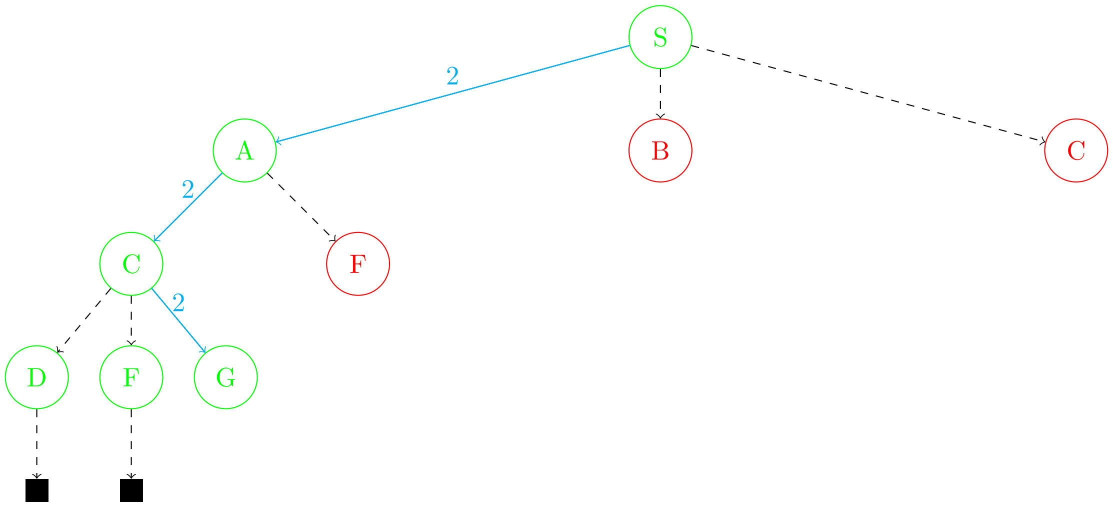
Step 4. We now try to expand the node D, since it comes alphabetically before nodes F and G. We have reached the maximum depth, so we reach the end of branch.



Step 5. We now try to expand the node F. We have reached the maximum depth, so we reach the end of branch.



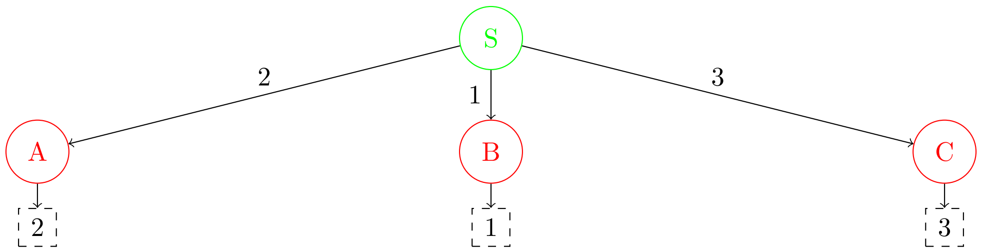
Step 6. We now try to expand the node G. Since G is our goal, thus the search is over. The path obtained from iterative deepening search is shown by the cyan line.



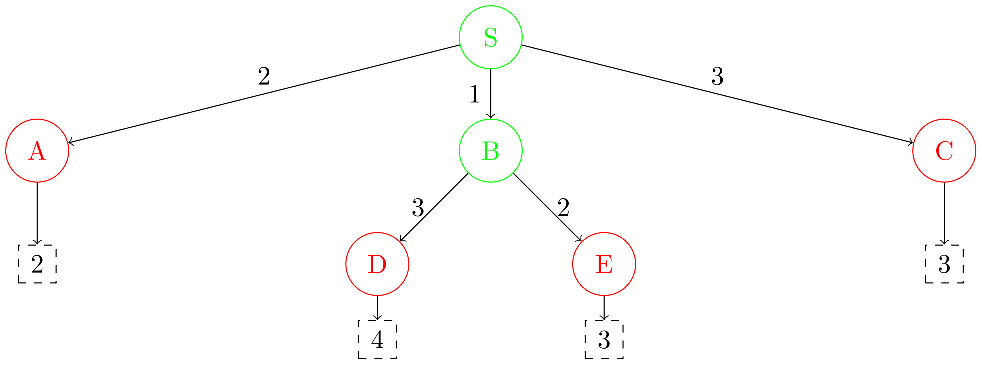
$$\Rightarrow Cost = 6$$

4.4 Uniform Cost Search

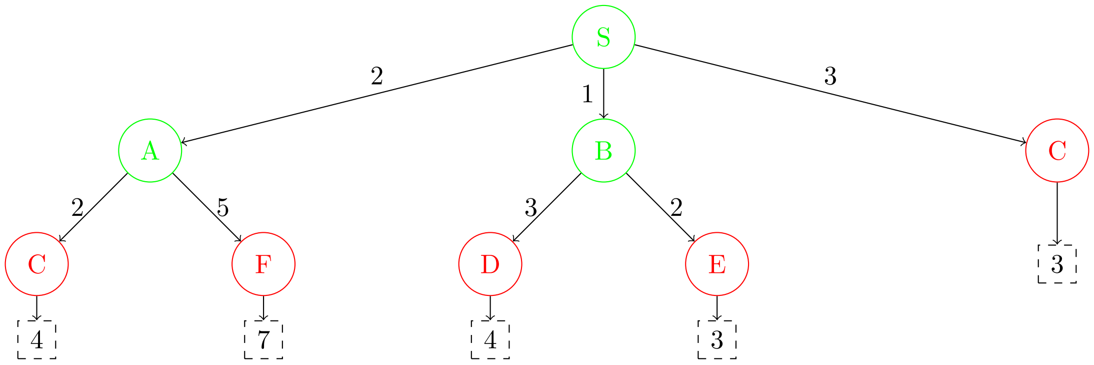
Step 1. We start expanding from the node S.



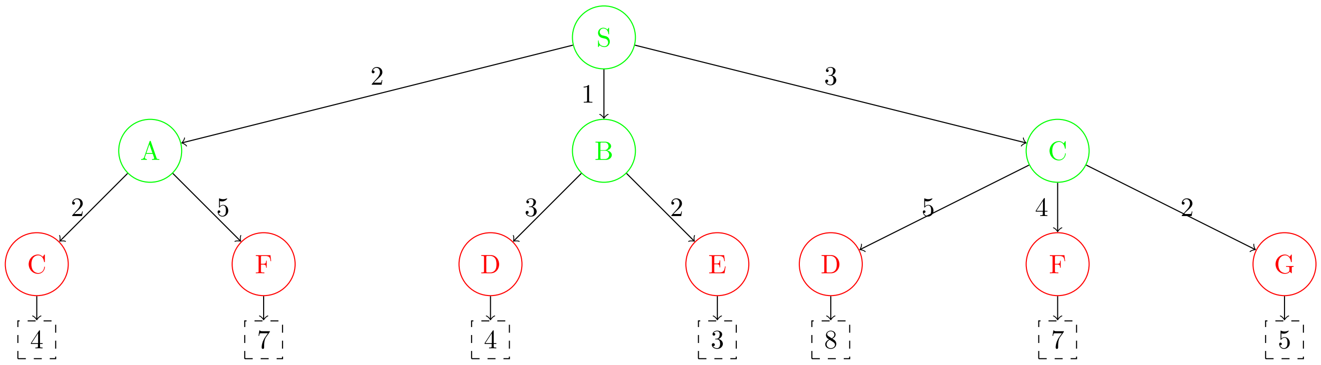
Step 2. We now expand node B because it has the minimum cost.



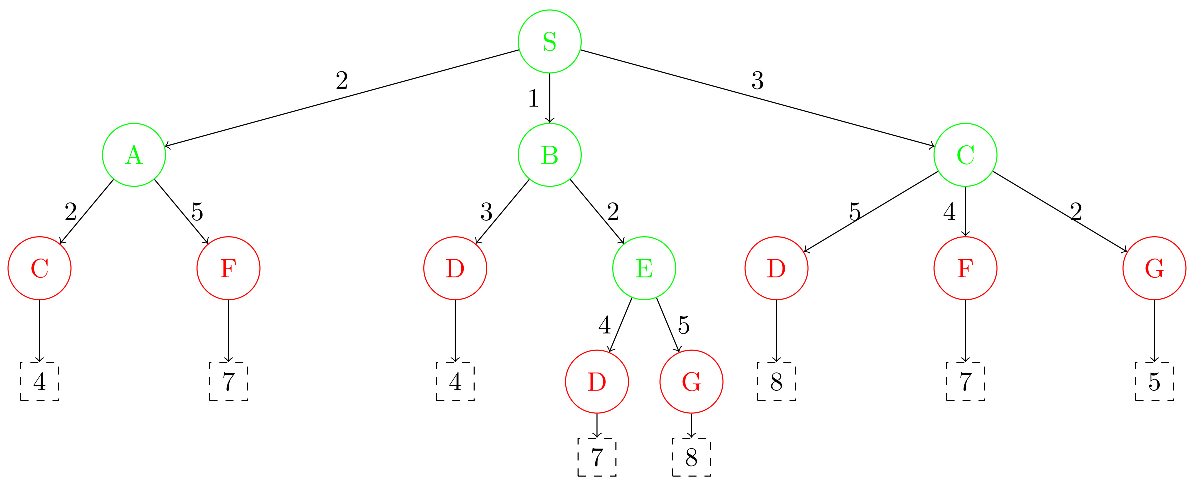
Step 3. We now expand node A because it is the cheapest.



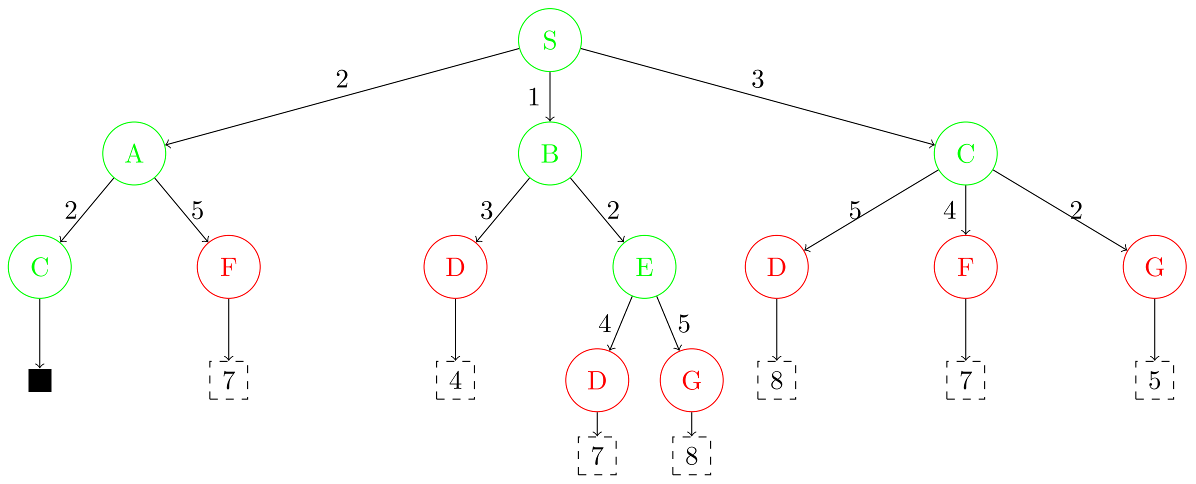
Step 4. We now expand node C since it's the cheapest and it comes alphabetically before node E.



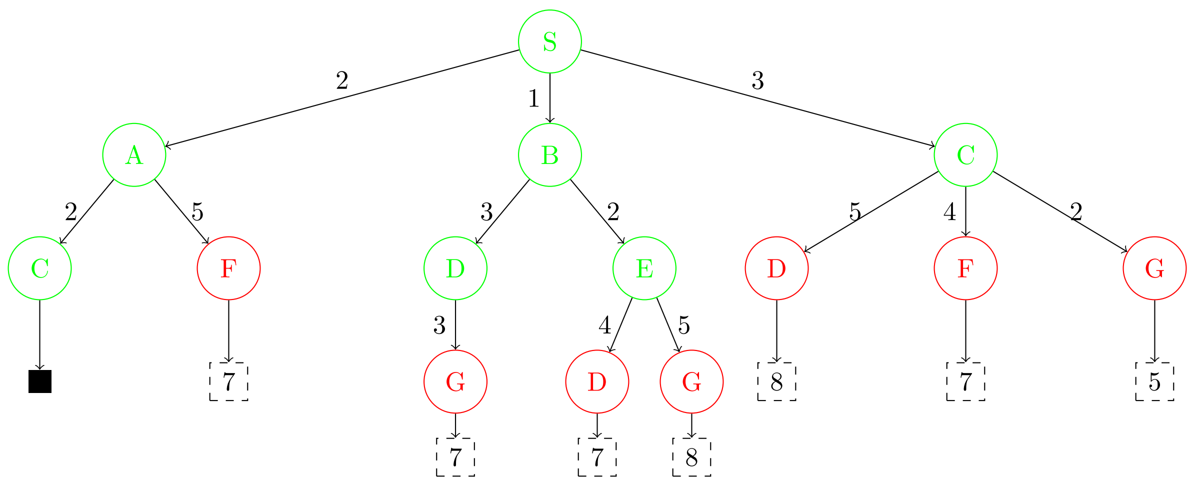
Step 5. We now expand node E since it is the cheapest.



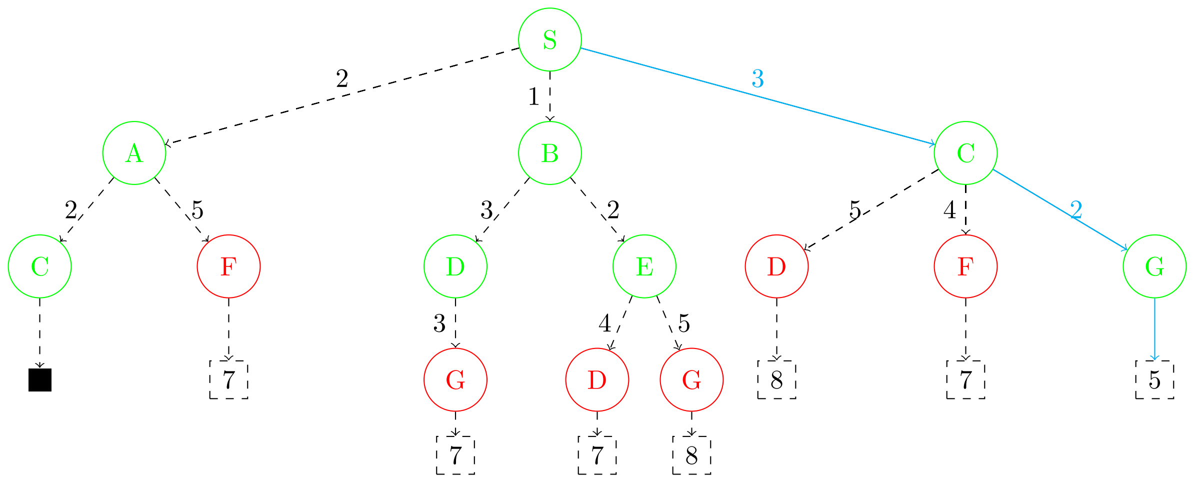
Step 6. We now try to expand node C since it is the cheapest and comes before node D. But we have already expanded C. So we reach the end of the branch



Step 7. We now try to expand node D since it is the cheapest. We have already expanded three of the children of D namely B, C and E. So only G remains.



Step 8. We now try to expand node G since it is the cheapest. Since G is our goal node, the search is over. The path returned by the uniform cost search is shown by the cyan line.



$$\Rightarrow Cost = 5$$

To summarize, the paths and the costs returned by different search methods are listed below.

Method	Path	Cost
BFS	$S \to C \to G$	5
DFS	$S \to A \to C \to D \to B \to E \to G$	19
IDS	$S \to A \to C \to G$	6
UCS	$S \to C \to G$	5