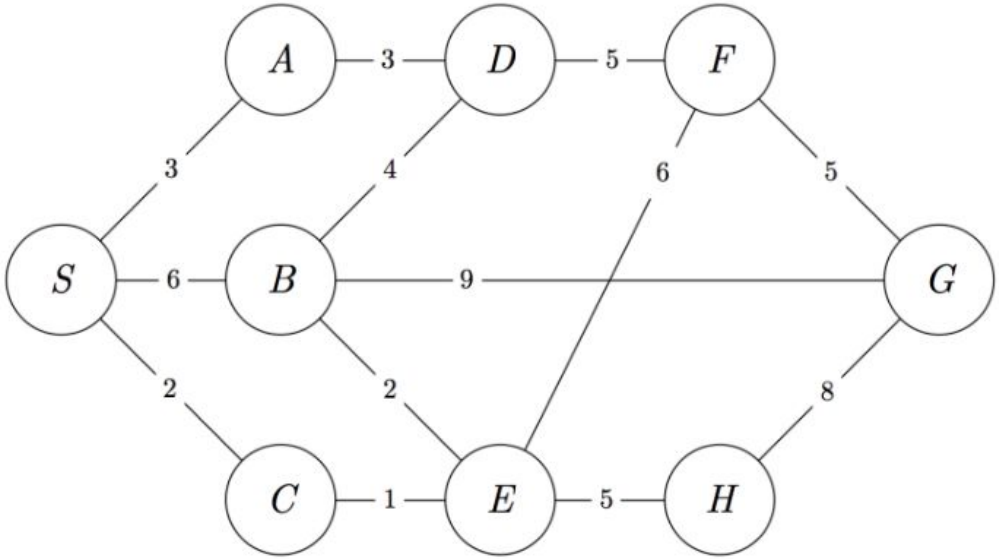


For each of the following search algorithms, find the nodes expanded and the path returned (break ties alphabetically, e.g. S->A->D precedes S->C->E->B): Edges are bi-directional. Use graph search.

- DFS
- BFS
- UCS
- Greedy
- A*

State	Heuristic
S	12
A	11
B	9
C	5
D	8
E	3
F	5
H	6
G	0

DFS

(S)

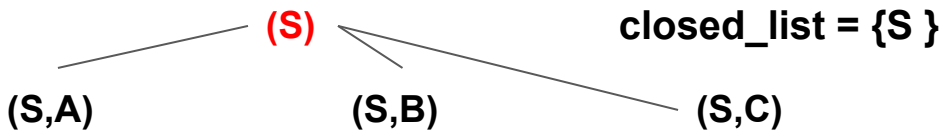
closed_list = { }

DFS

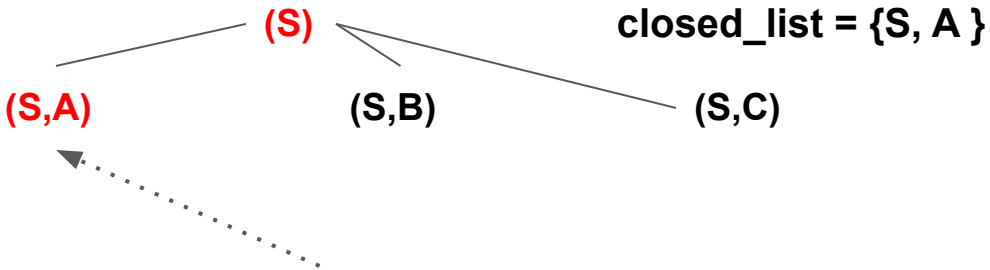
(S)

closed_list = {S }

DFS

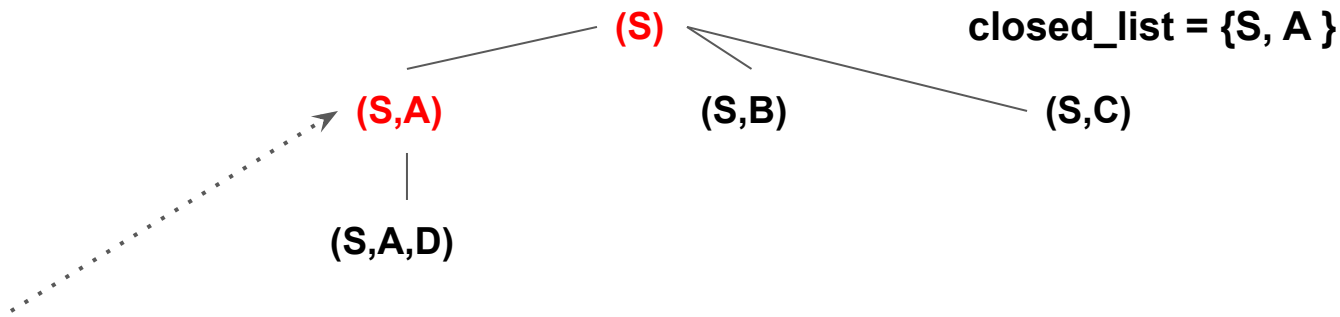


DFS



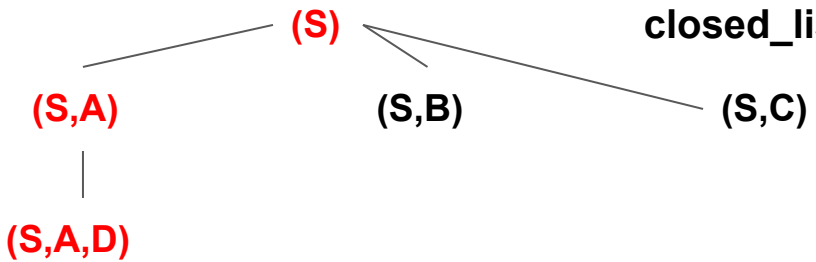
(S, A) is chosen to expand because it alphabetically precedes (S,B) and (S,C)

DFS



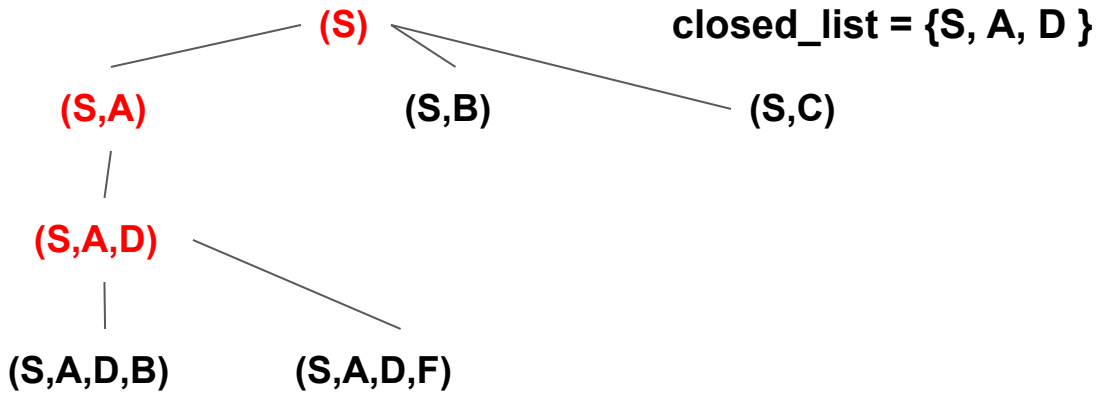
Note (S,A,S) is also considered but is not added because S is already in the closed list.
We don’t add nodes that lead to visited states.

DFS



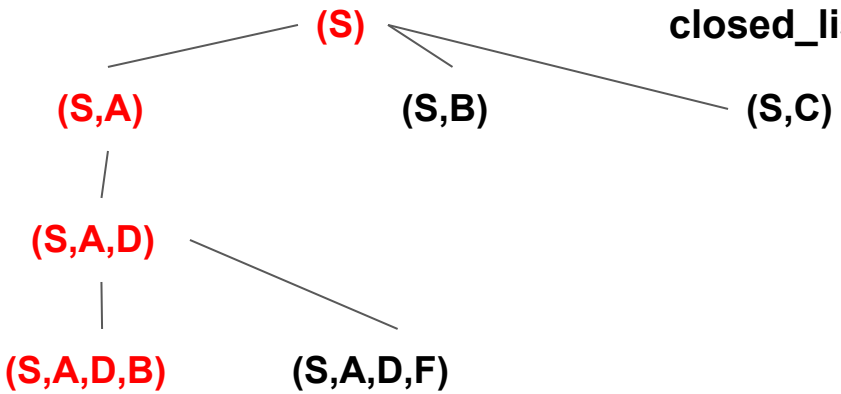
closed_list = {S, A, D }

DFS



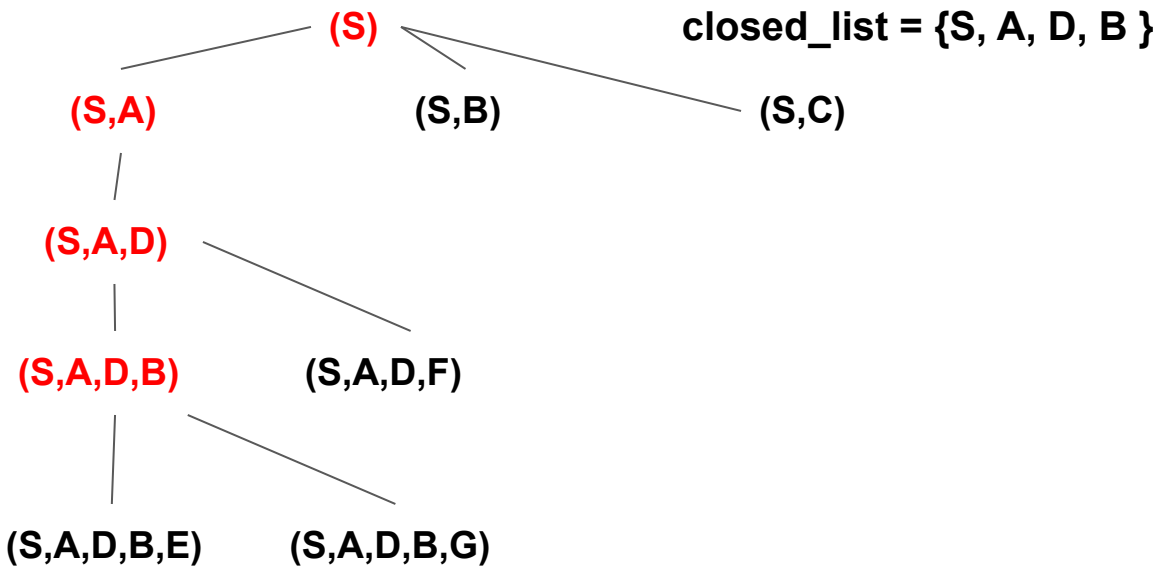
closed_list = {S, A, D }

DFS

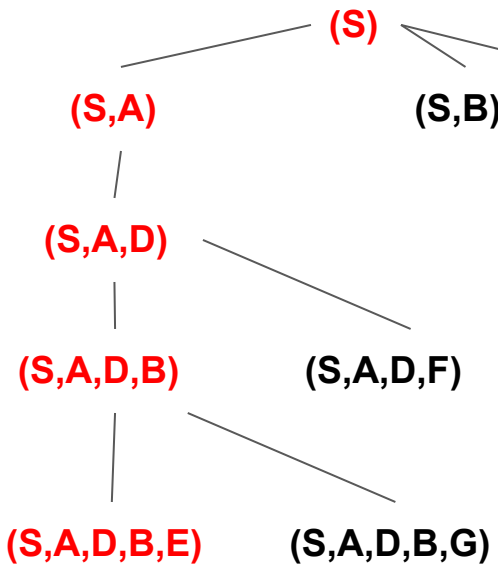


closed_list = {S, A, D, B }

DFS

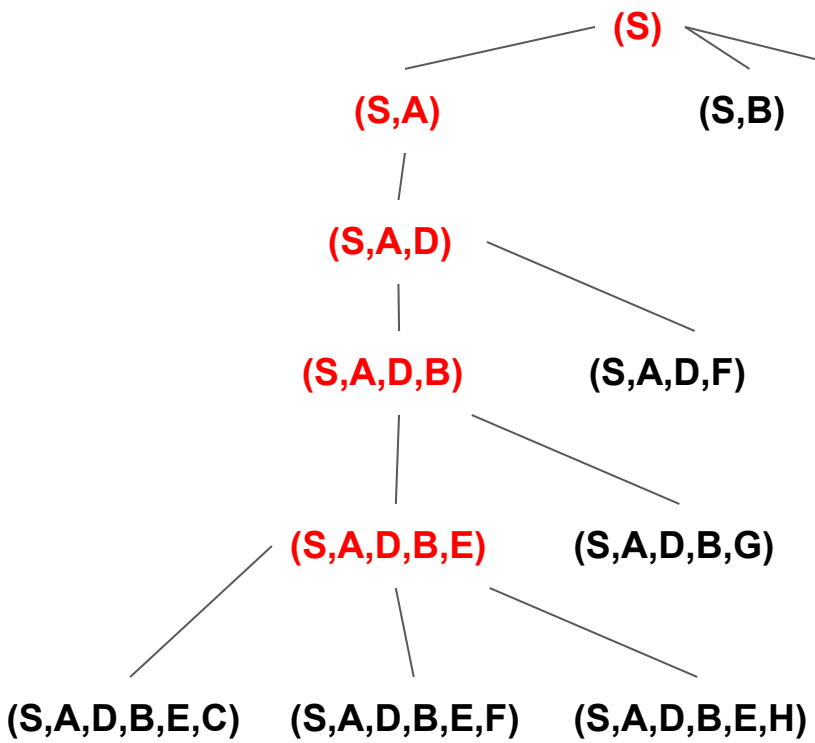


DFS



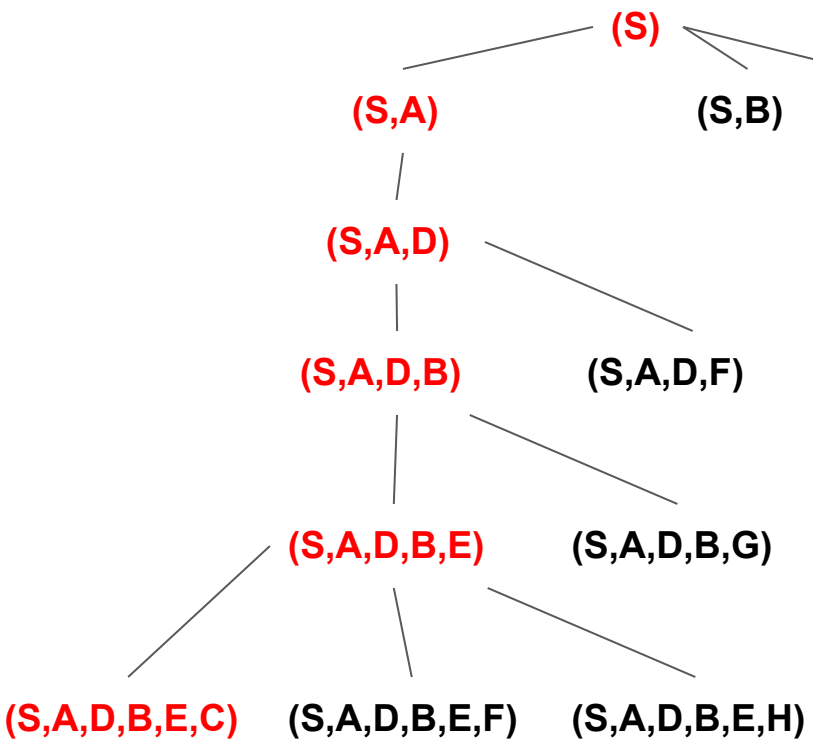
closed_list = {S, A, D, B, G }

DFS



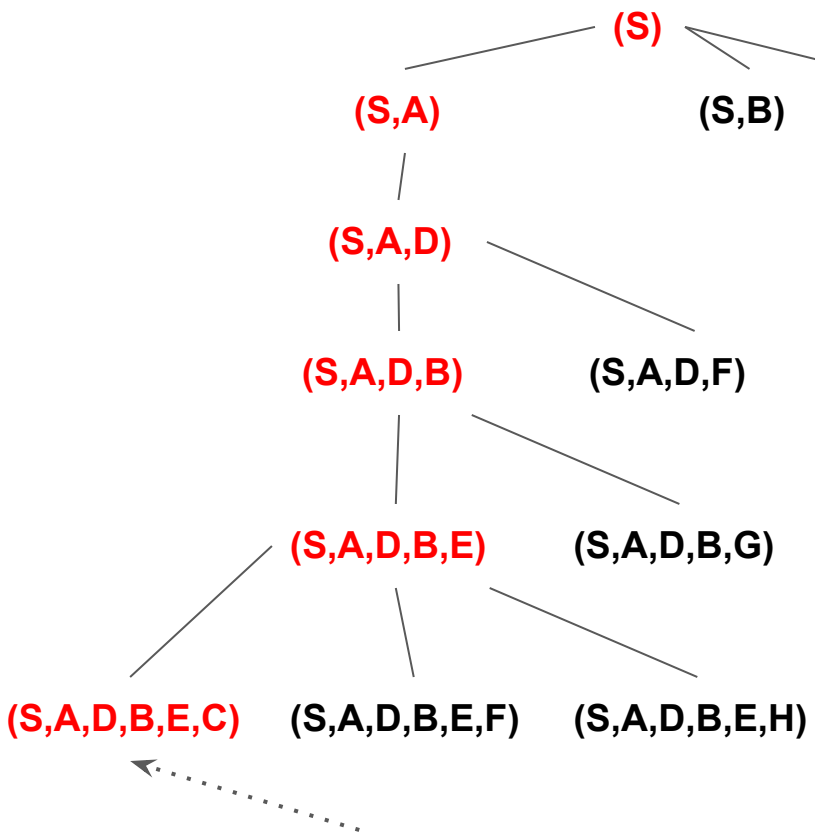
closed_list = {S, A, D, B, G }

DFS



closed_list = {S, A, D, B, G, C }

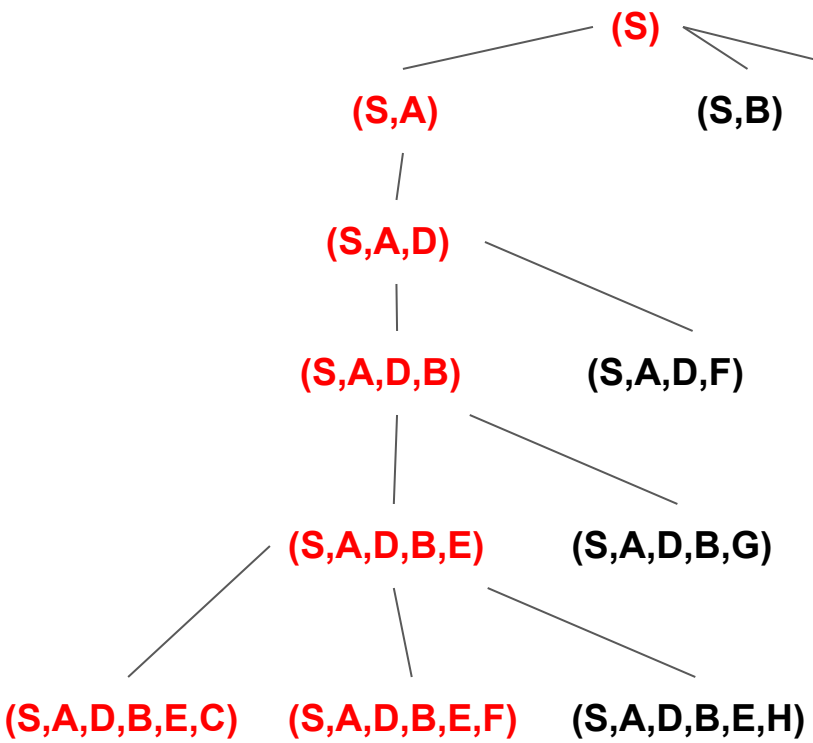
DFS



closed_list = {S, A, D, B, G, C }

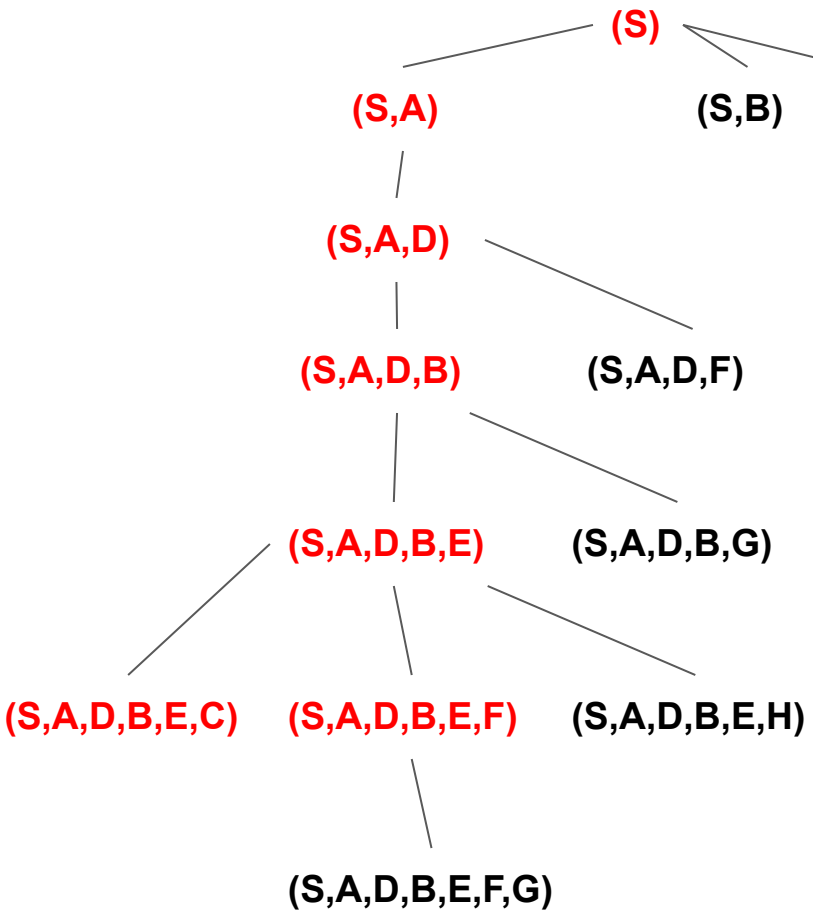
No unvisited neighbouring state from C to put into fringe. Proceed to expand the next node from fringe.

DFS



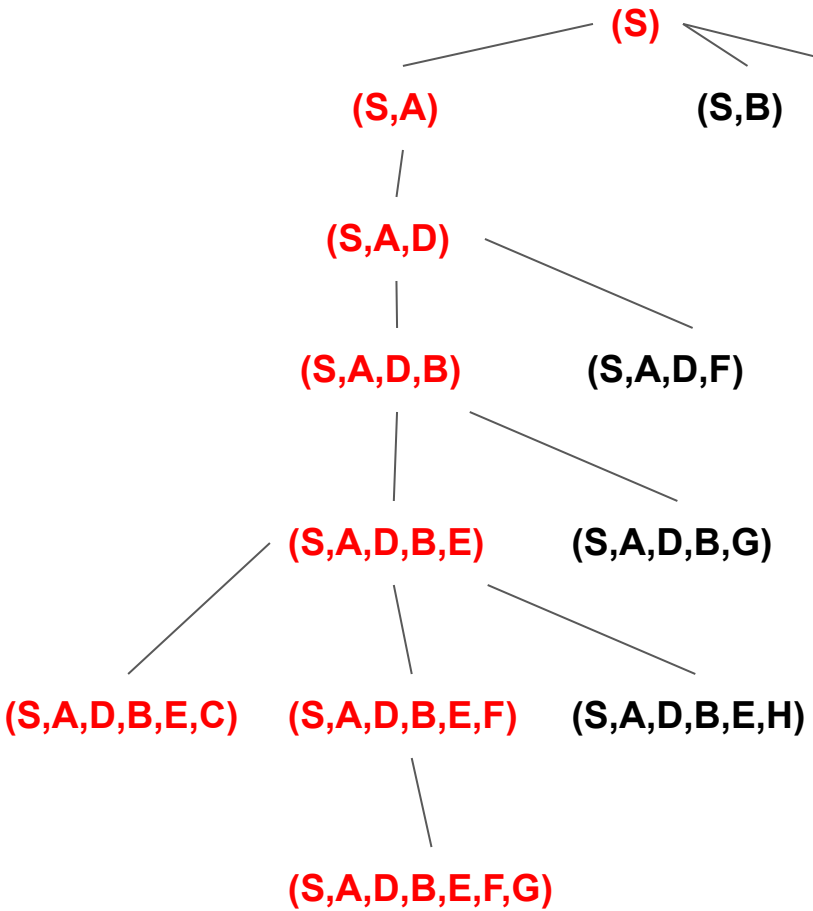
closed_list = {S, A, D, B, G, C, F }

DFS



closed_list = {S, A, D, B, G, C, F }

DFS



closed_list = {S, A, D, B, G, C, F, G }

Nodes expanded: 8

Path returned: (S,A,D,B,E,F,G)

Declare success!!!

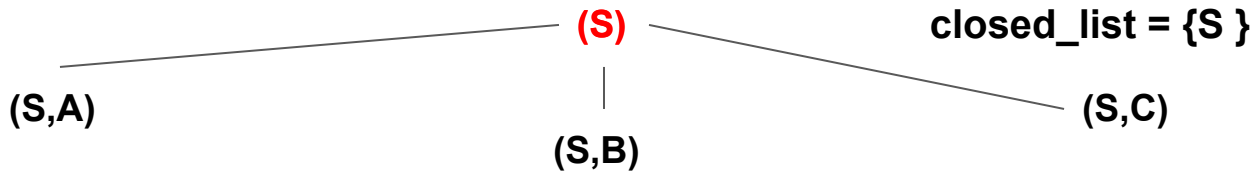
BFS

(S) closed_list = {}

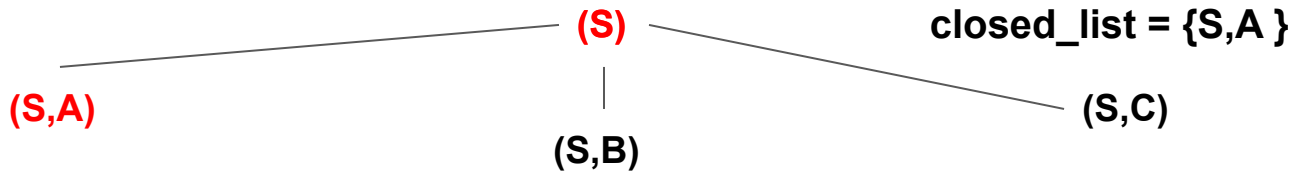
BFS

(S) closed_list = {S}

BFS

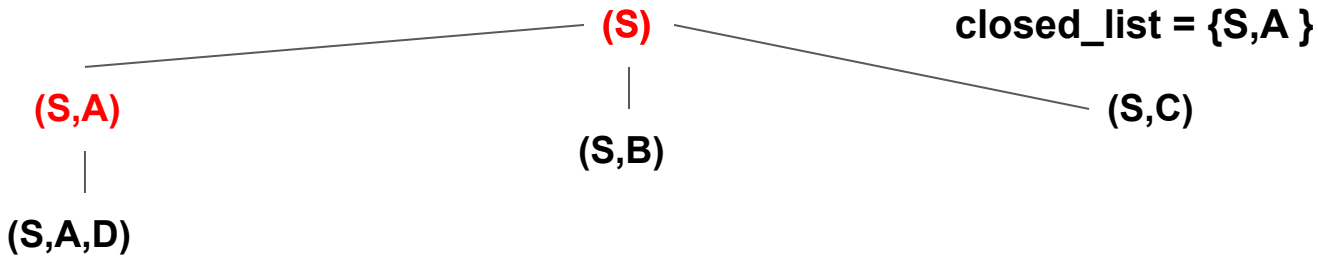


BFS

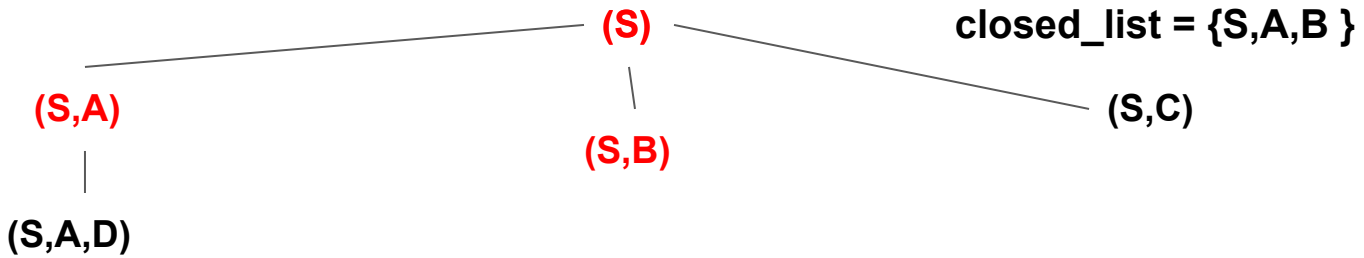


closed_list = {S,A }

BFS

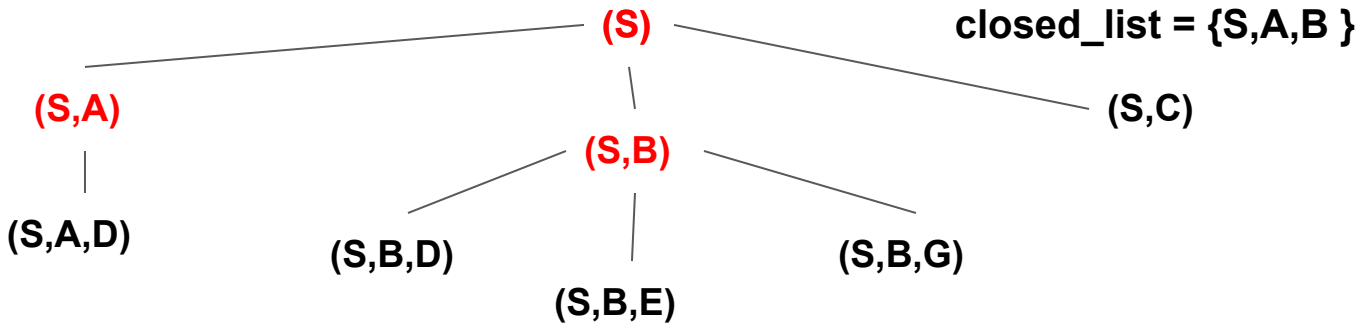


BFS

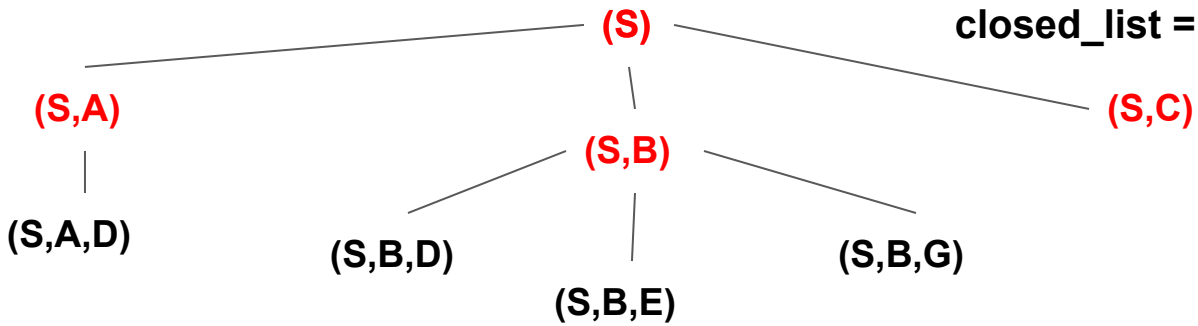


closed_list = {S,A,B }

BFS

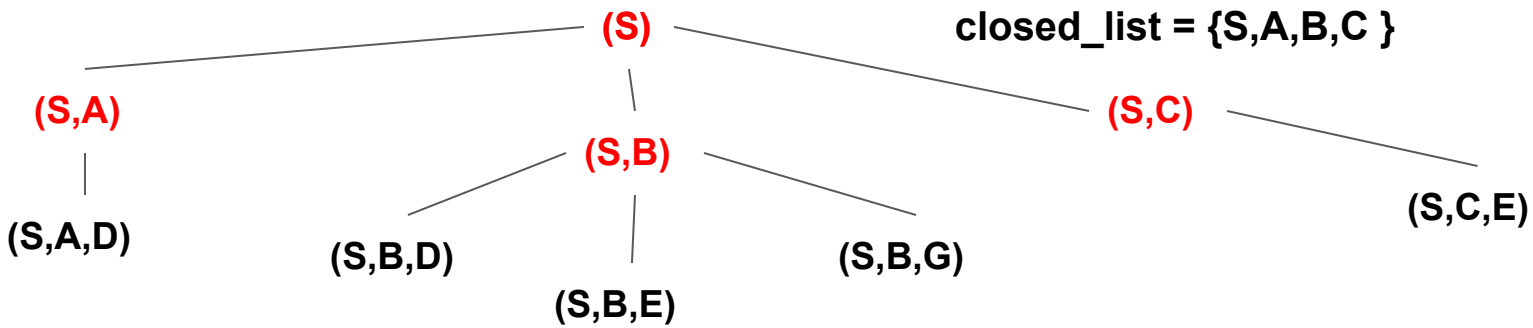


BFS

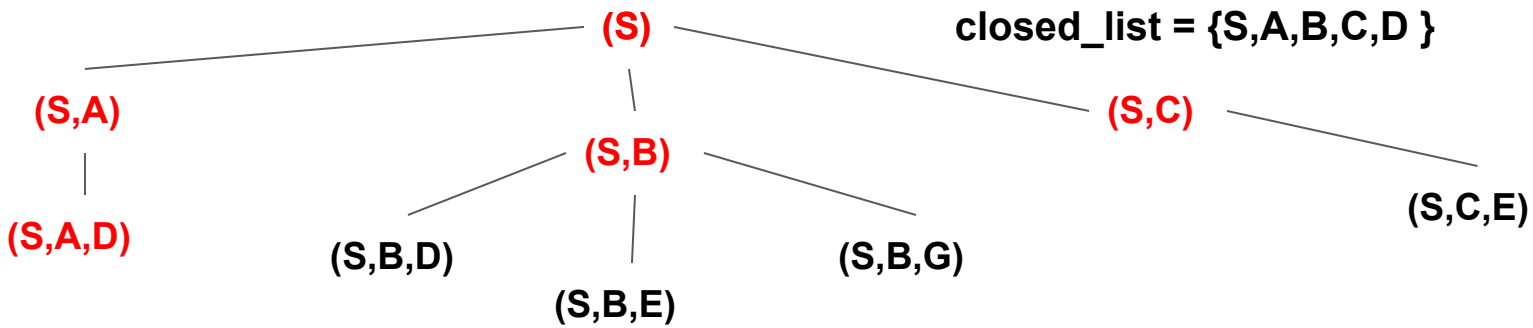


closed_list = {S,A,B,C }

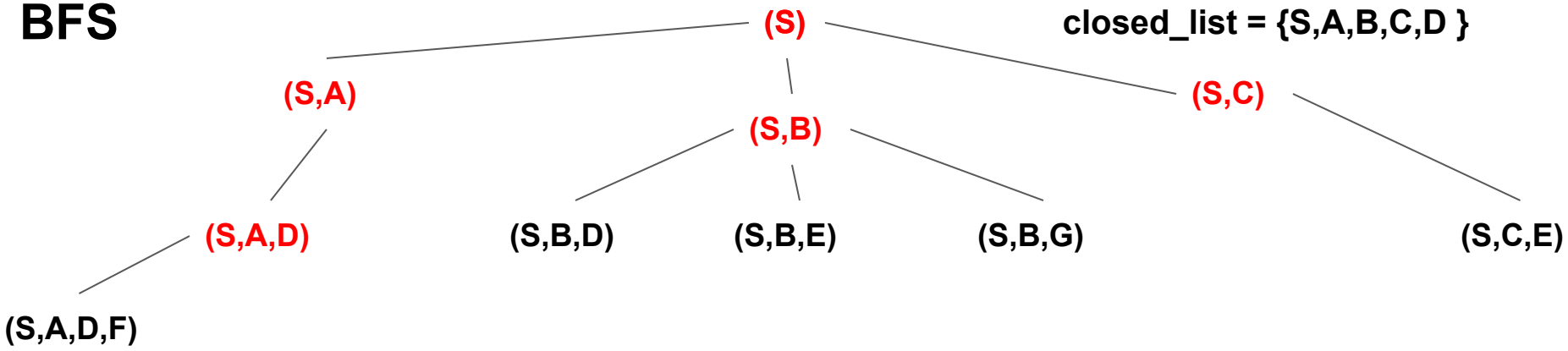
BFS



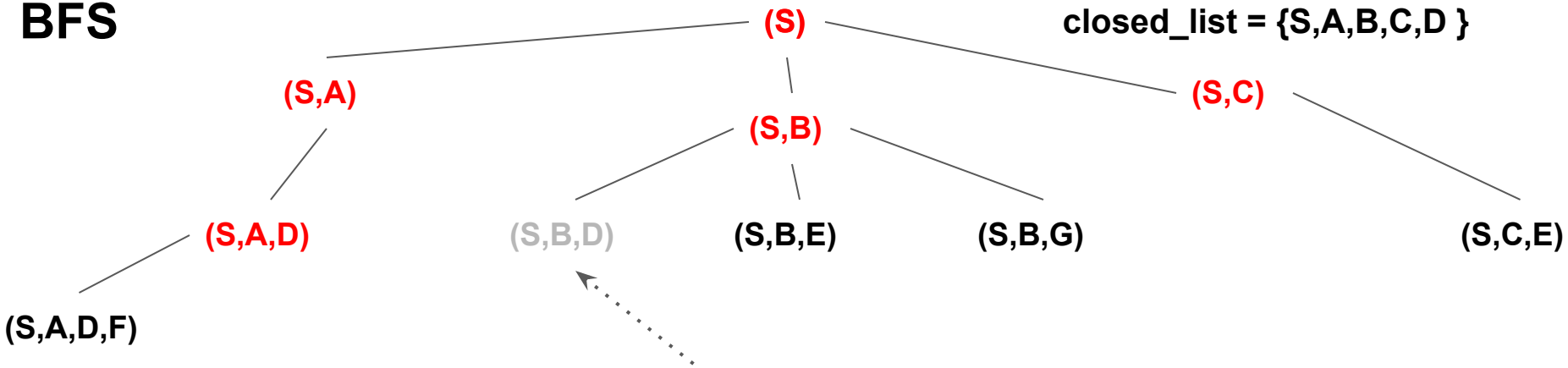
BFS



BFS

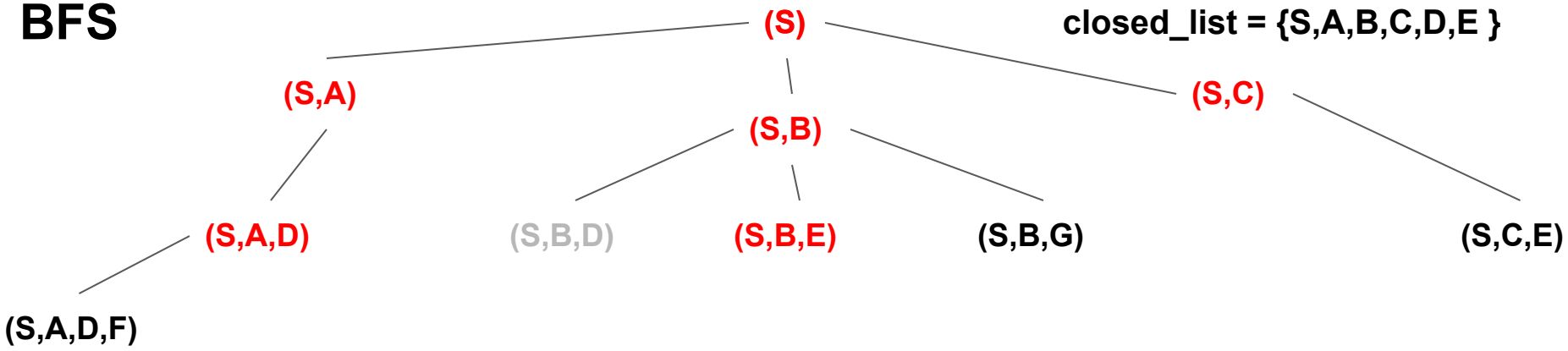


BFS

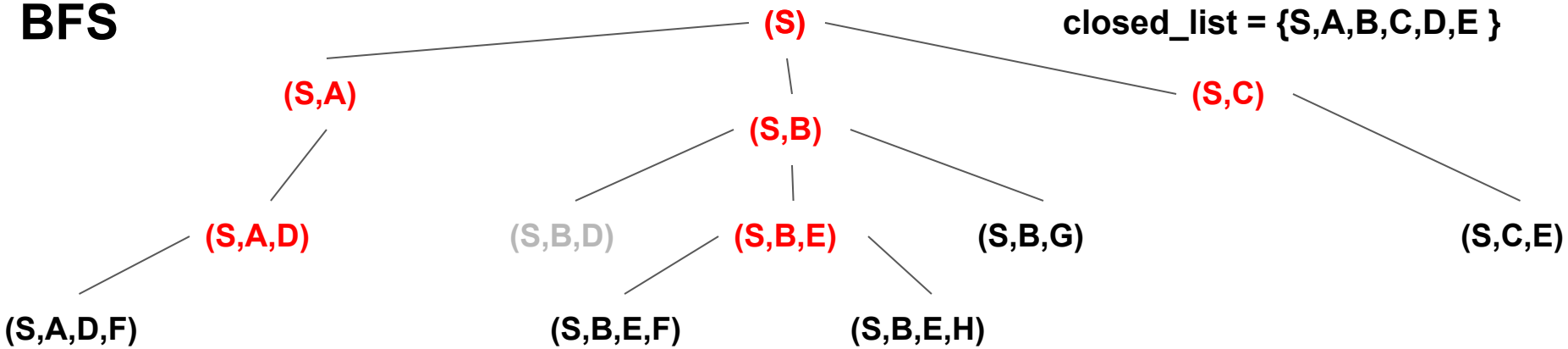


We plan to expand (S,B,D) but realize state D has already been visited before (using closed_list) so skip it!

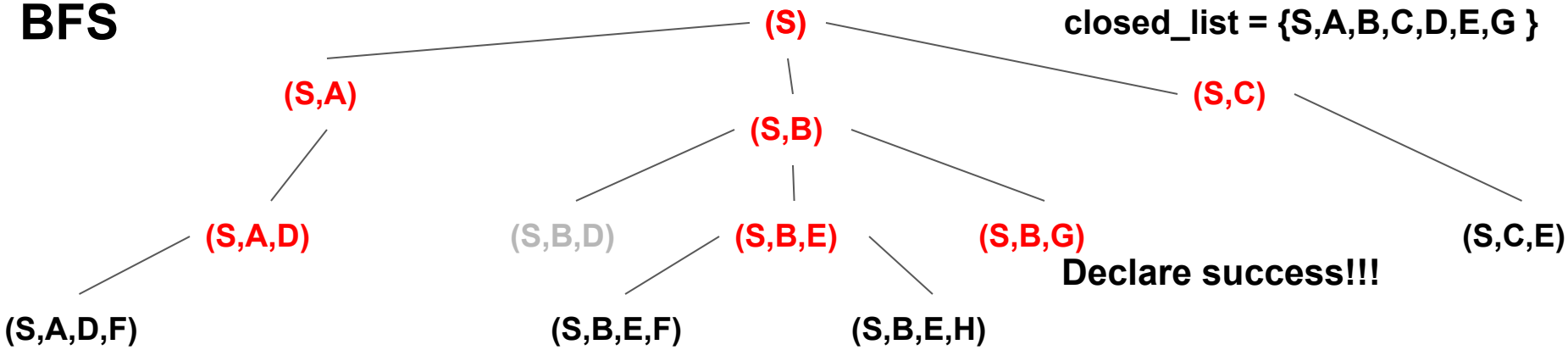
BFS



BFS


(S)
closed_list = {S,A,B,C,D,E }
(S,A)
(S,B)
(S,C)
(S,A,D)
(S,B,D)
(S,B,E)
(S,B,G)
(S,C,E)
(S,A,D,F)
(S,B,E,F)
(S,B,E,H)

BFS



Nodes expanded: 7
Path returned: (S,B,G)

UCS

(S),0

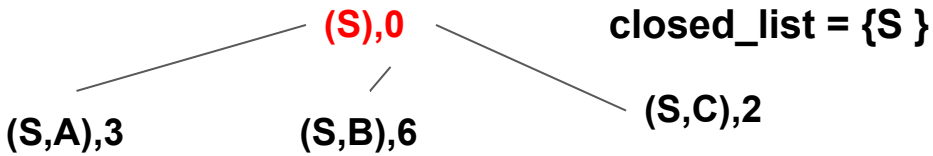
closed_list = { }

UCS

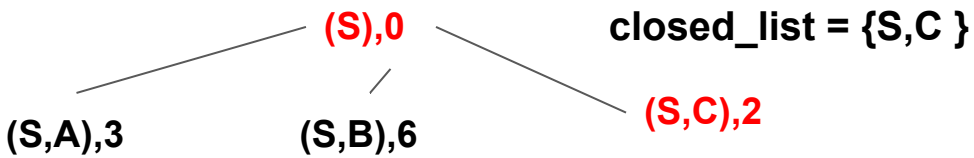
(S),0

closed_list = {S }

UCS

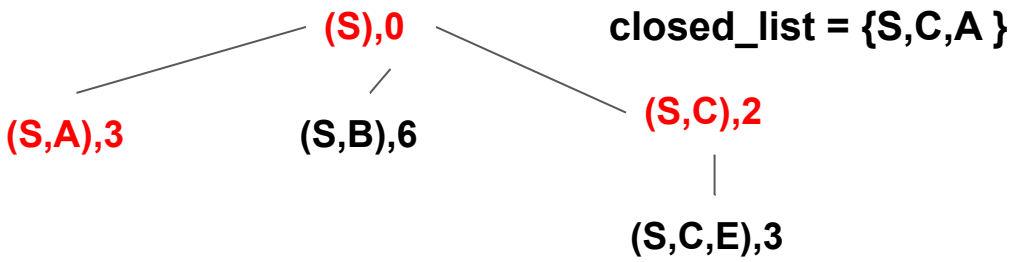


UCS

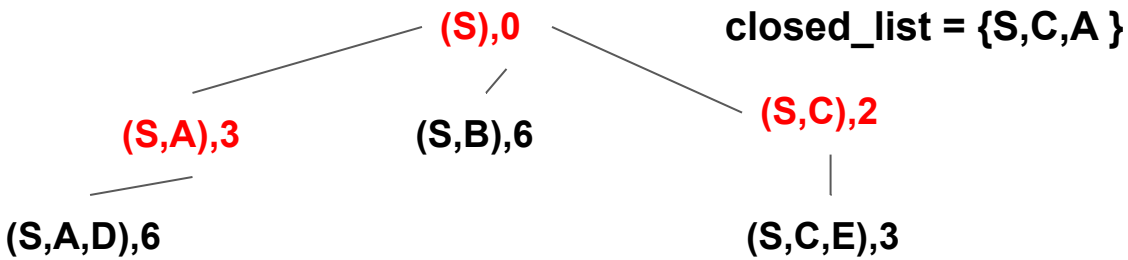


UCS

UCS

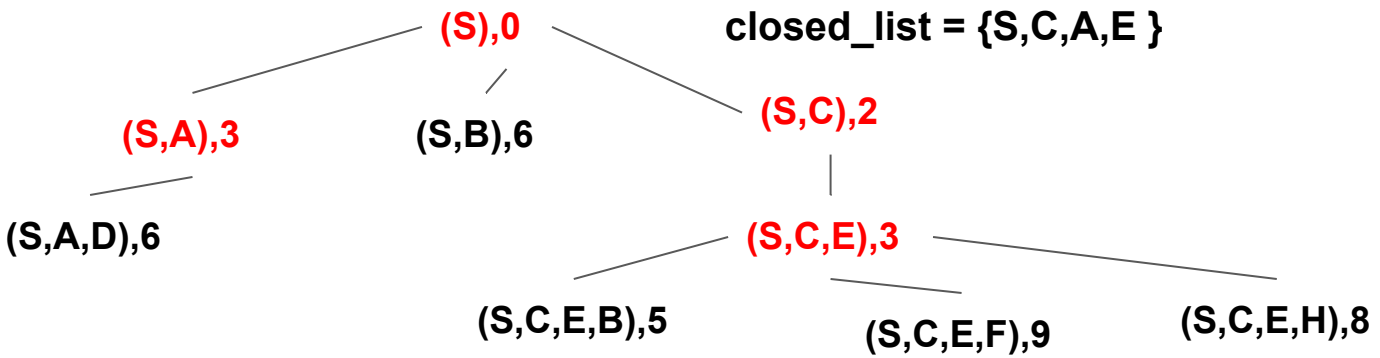


UCS

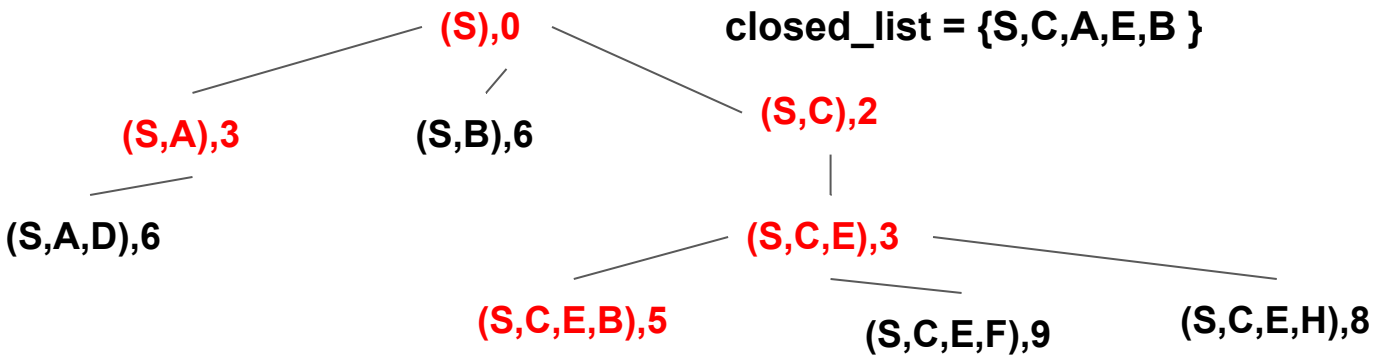


UCS

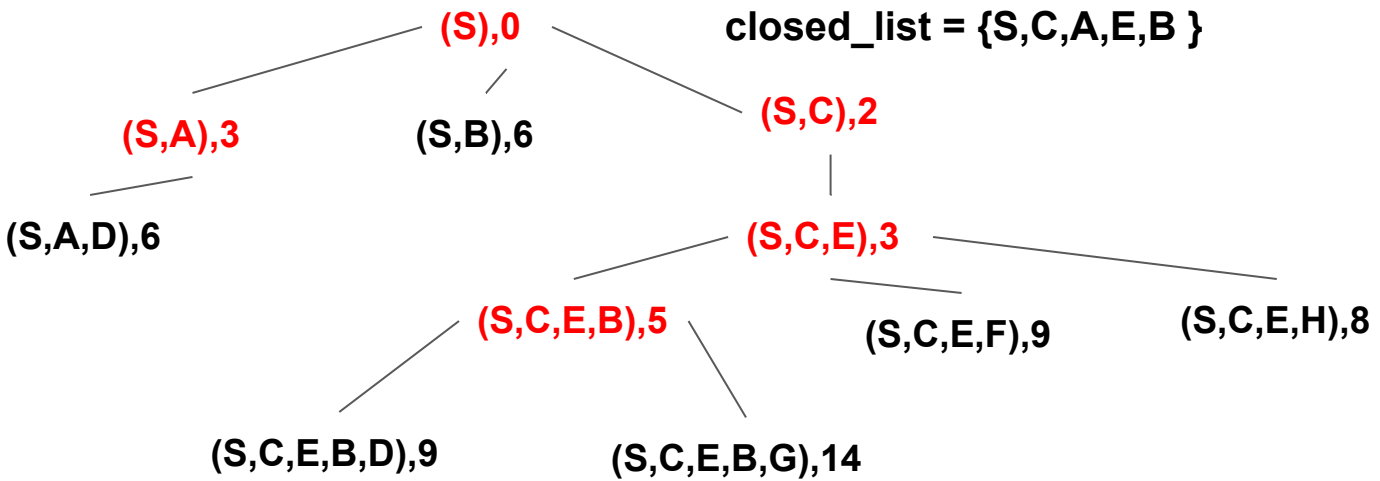
UCS



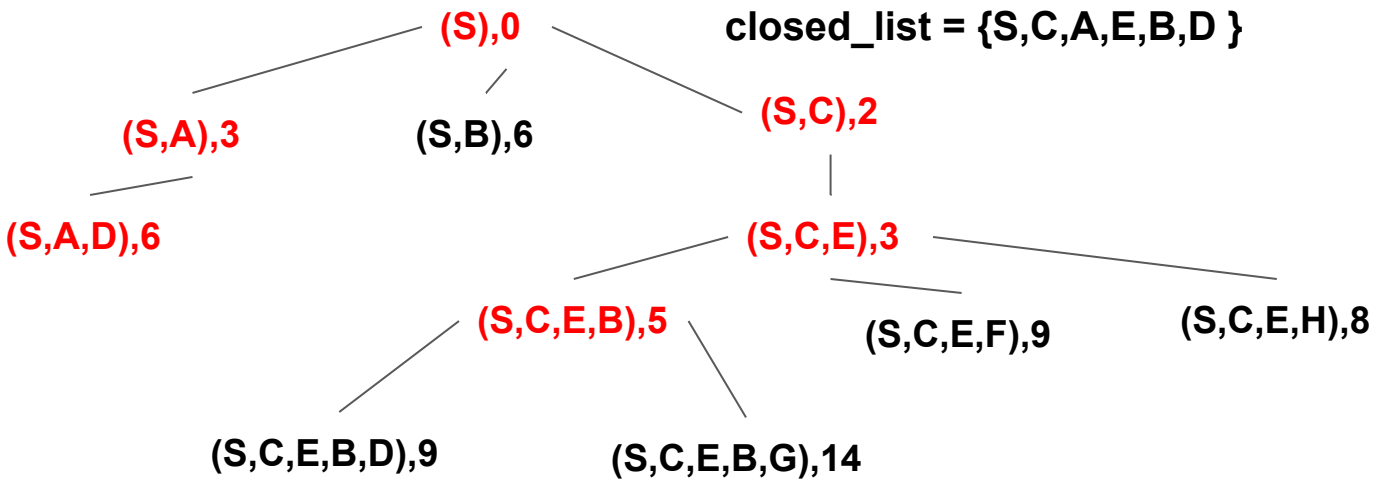
UCS



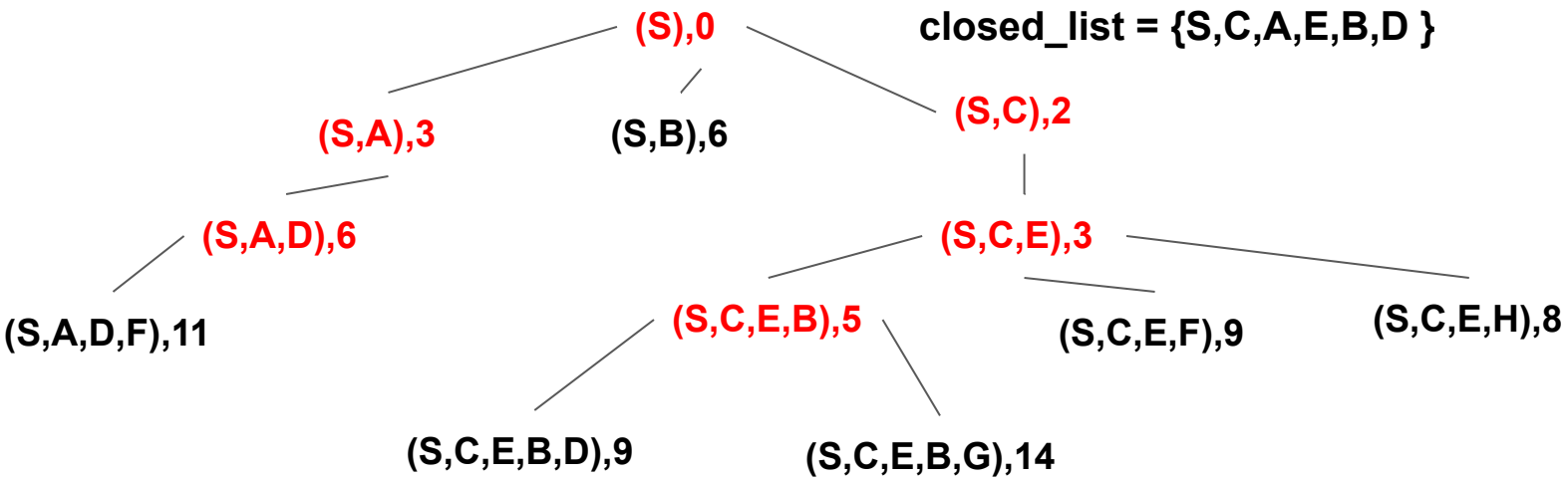
UCS


(S),0
closed_list = {S,C,A,E,B }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14

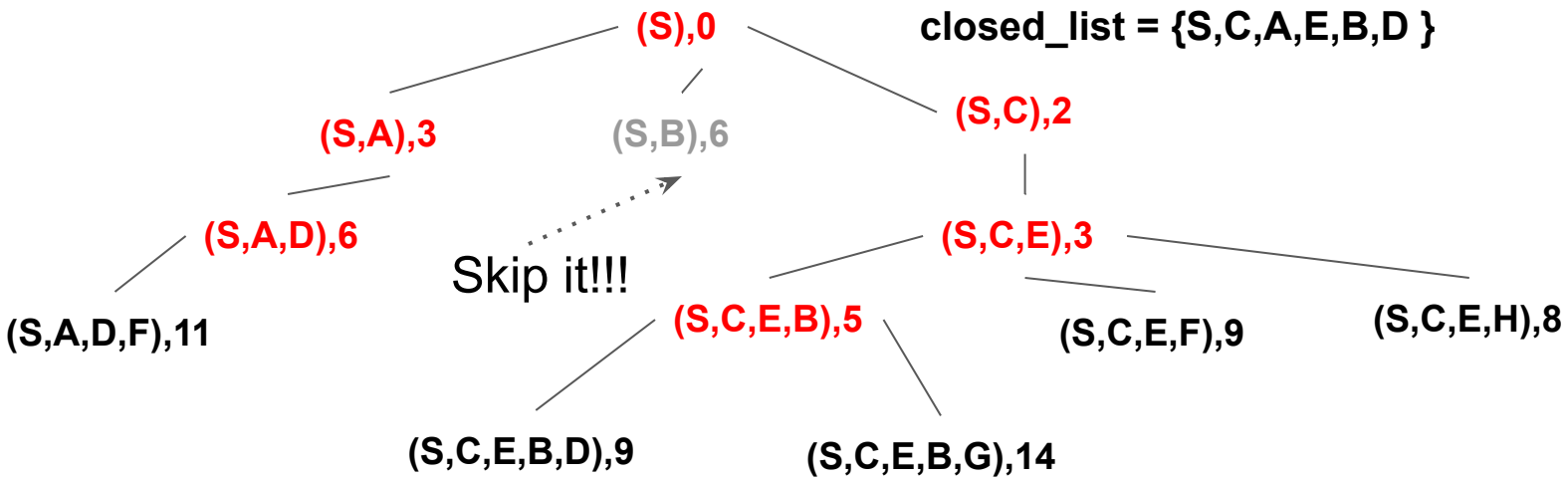
UCS


(S),0
closed_list = {S,C,A,E,B,D }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14

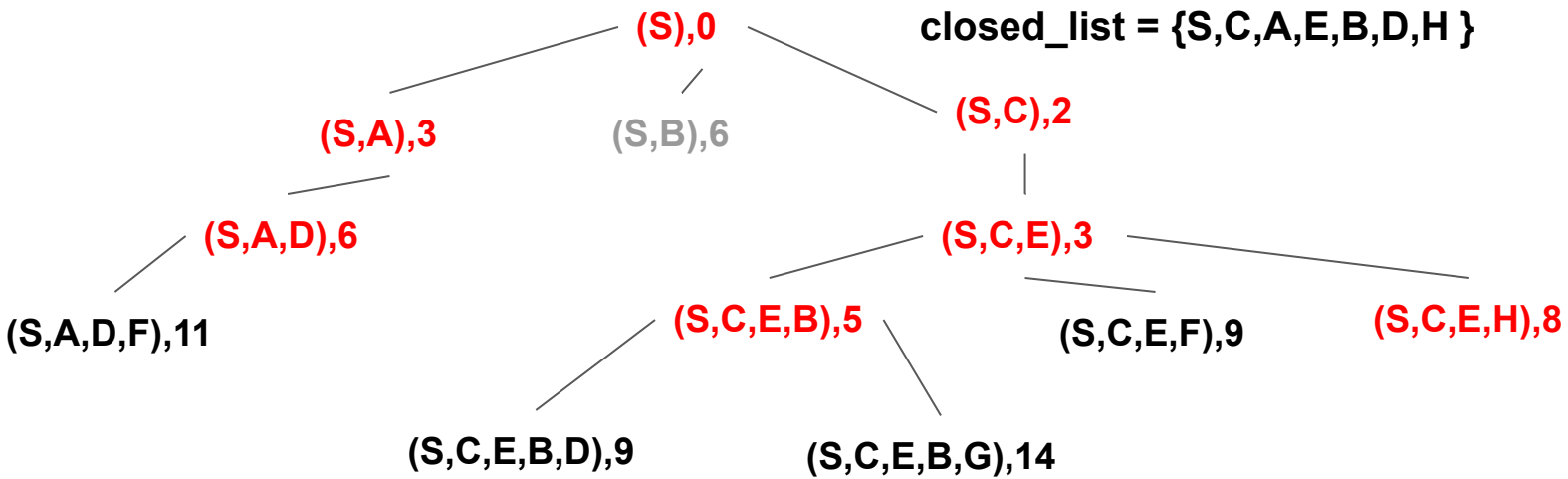
UCS


(S),0
closed_list = {S,C,A,E,B,D }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,A,D,F),11
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14

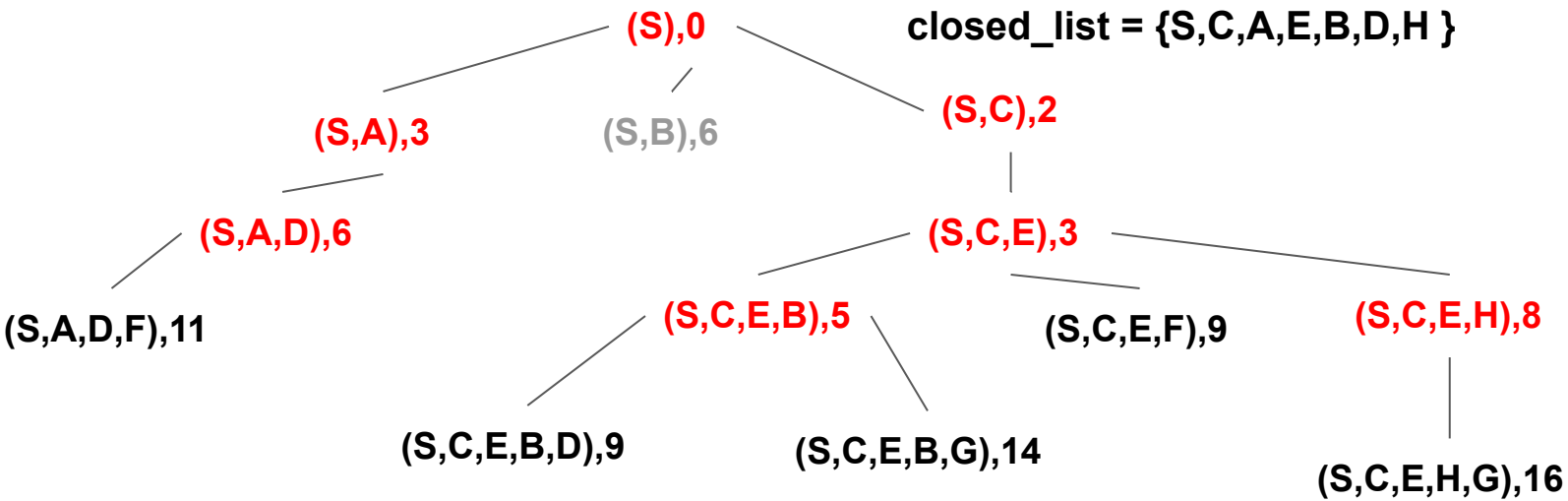
UCS


(S),0
closed_list = {S,C,A,E,B,D }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
Skip it!!!
(S,C,E),3
(S,A,D,F),11
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14

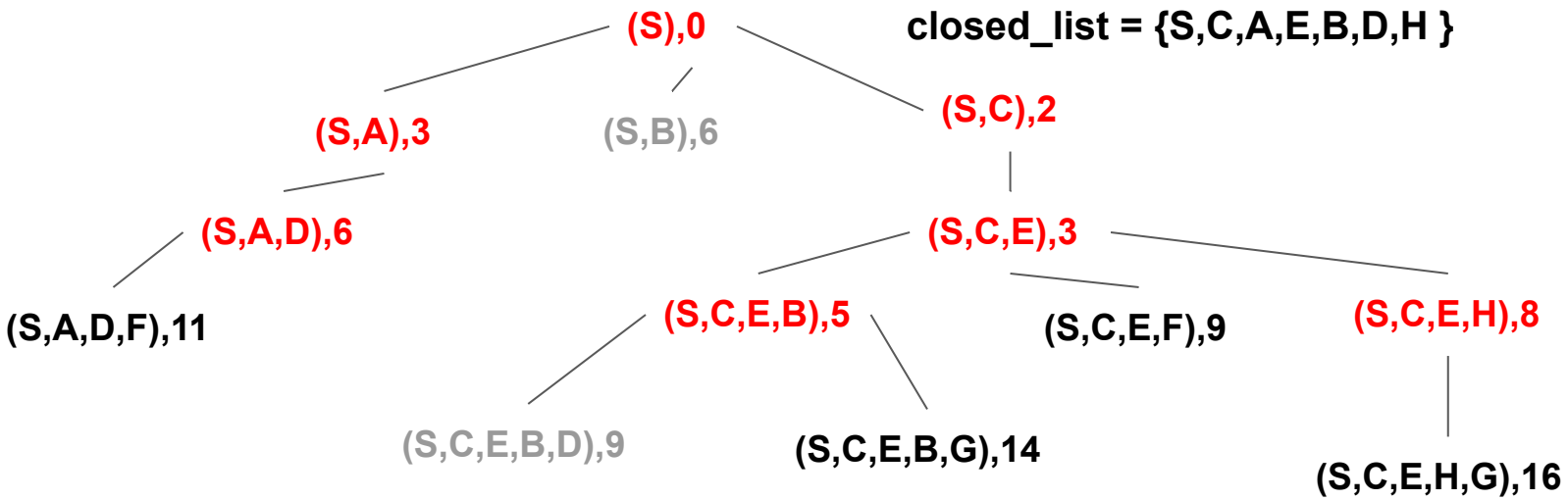
UCS


(S),0
closed_list = {S,C,A,E,B,D,H }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,A,D,F),11
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14

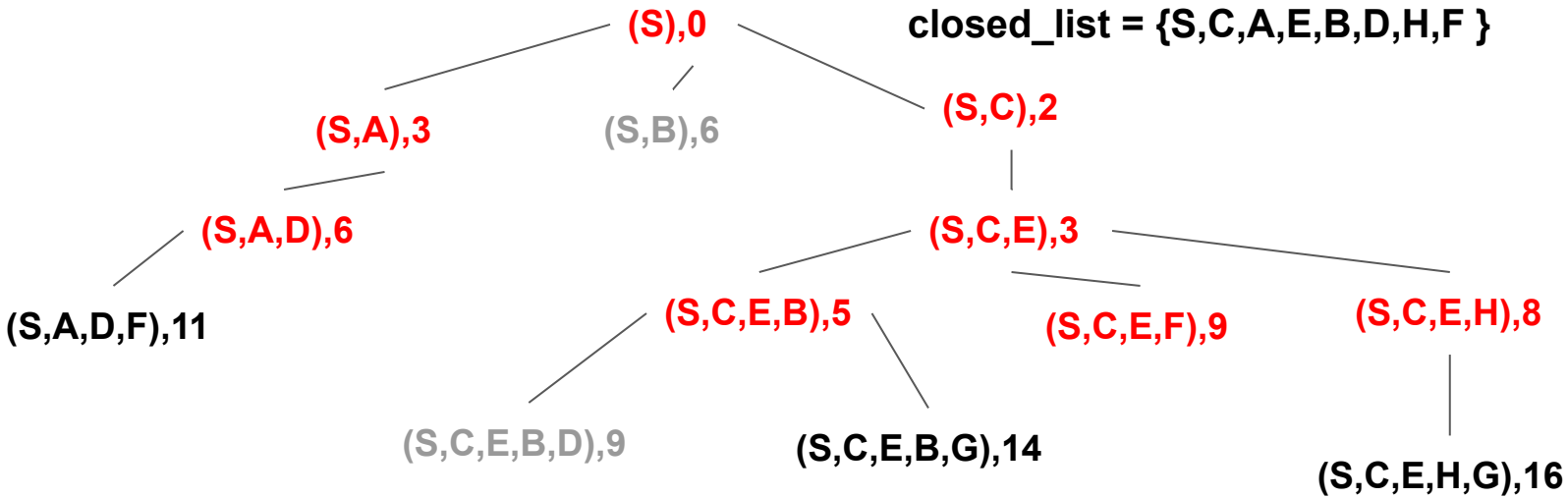
UCS


(S),0
closed_list = {S,C,A,E,B,D,H }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,A,D,F),11
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14
(S,C,E,H,G),16

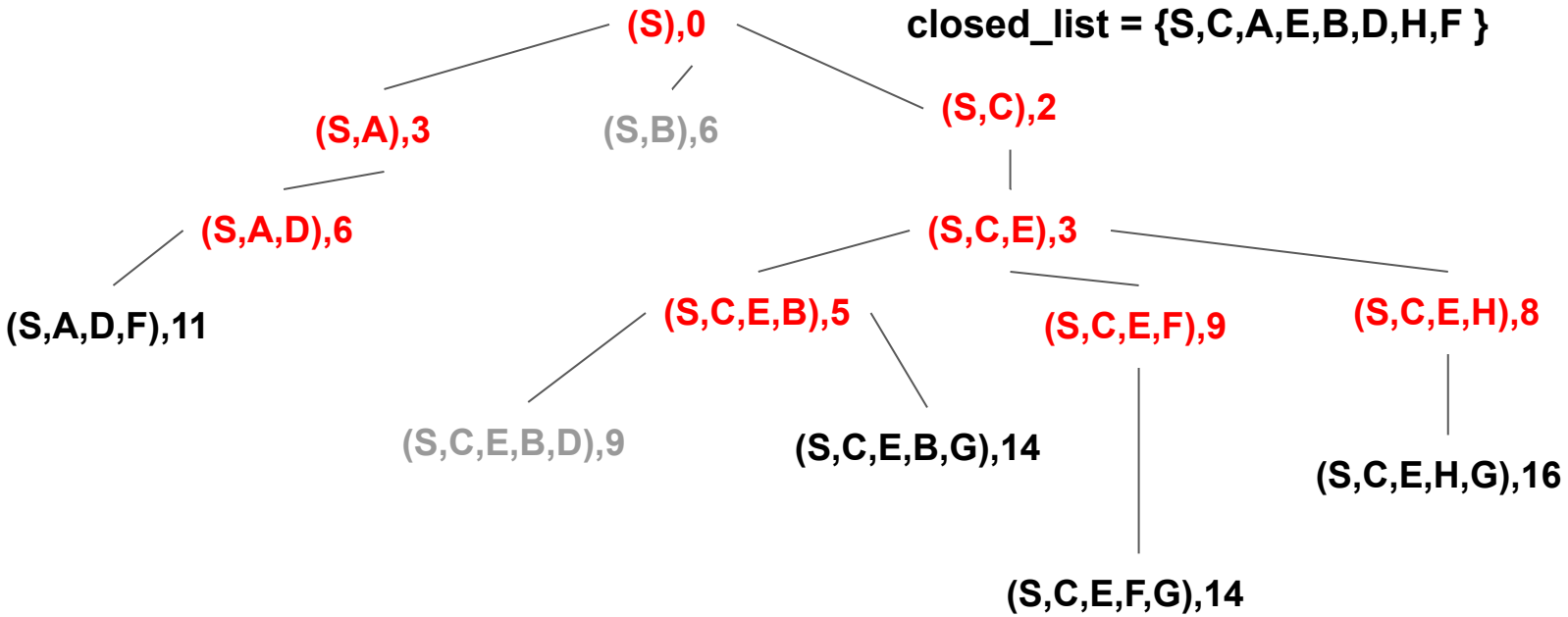
UCS


(S),0
closed_list = {S,C,A,E,B,D,H }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,A,D,F),11
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14
(S,C,E,H,G),16

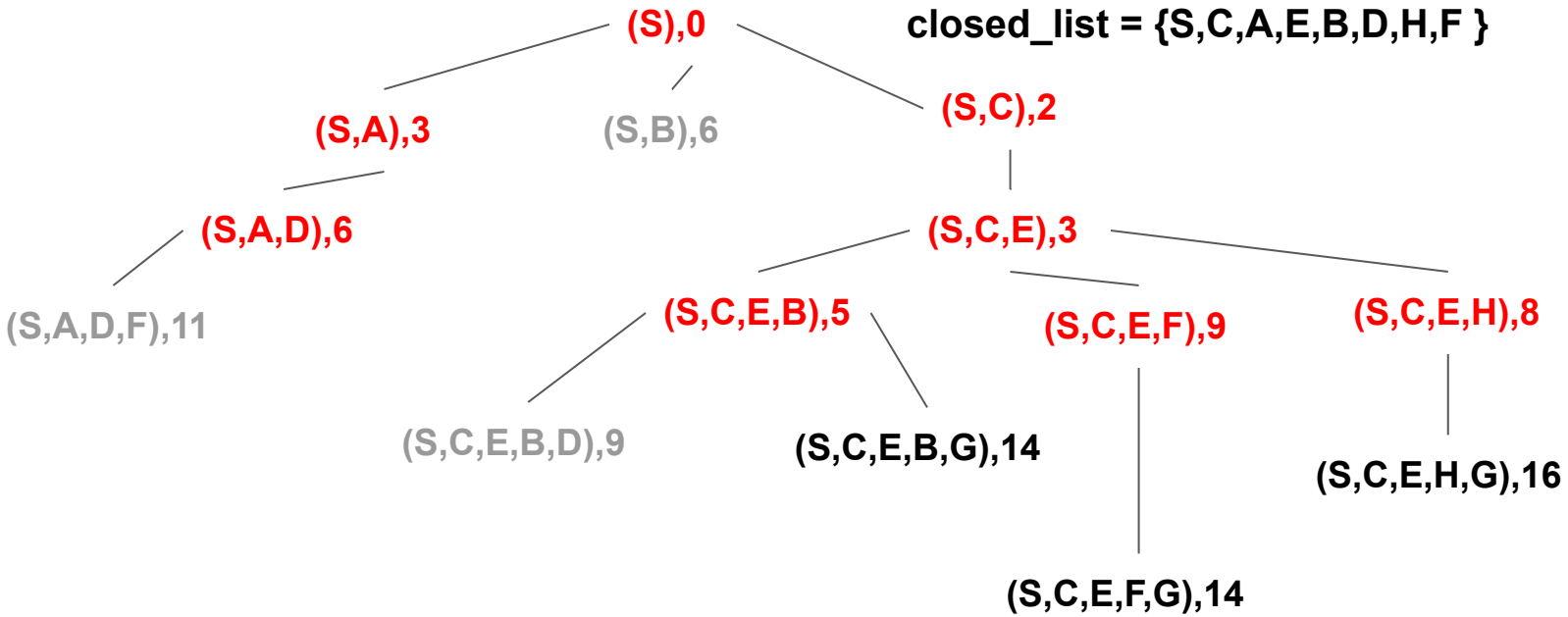
UCS


(S),0
closed_list = {S,C,A,E,B,D,H,F }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,A,D,F),11
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14
(S,C,E,H,G),16

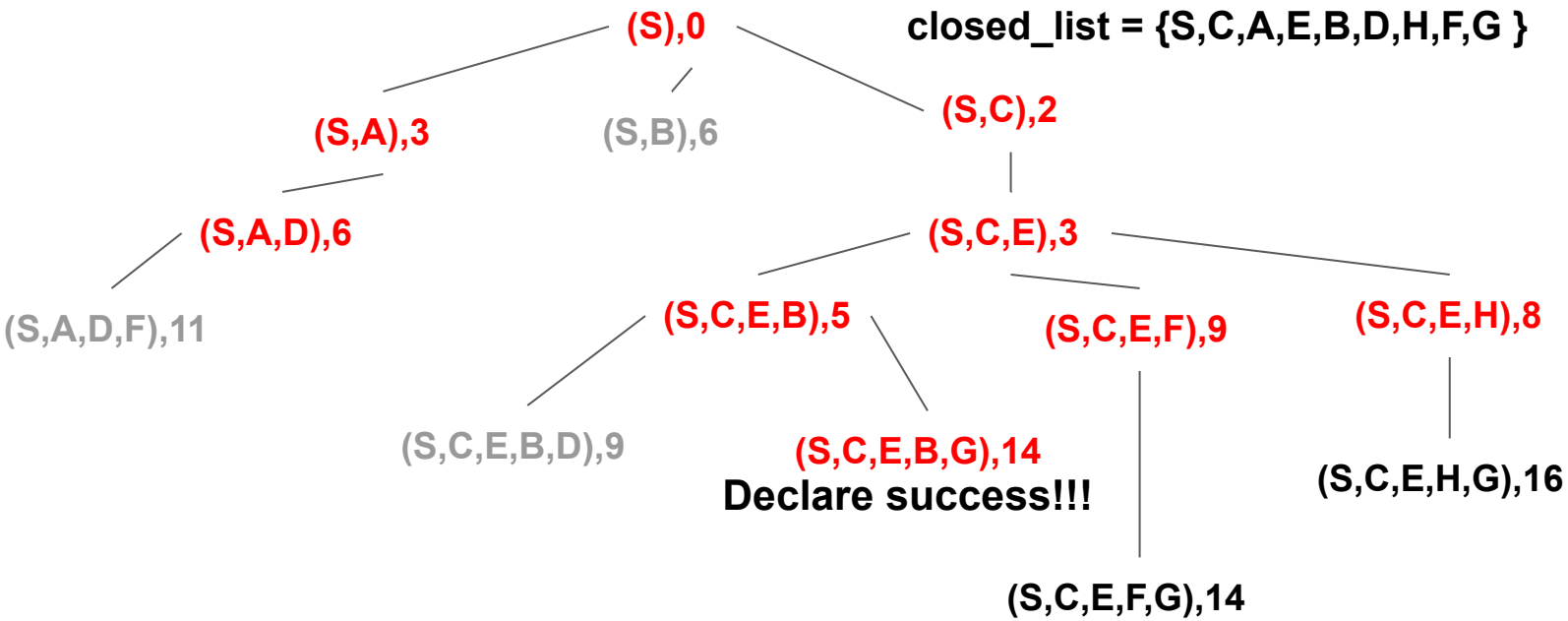
UCS


(S),0
closed_list = {S,C,A,E,B,D,H,F }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,A,D,F),11
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14
(S,C,E,H,G),16
(S,C,E,F,G),14

UCS


(S),0
closed_list = {S,C,A,E,B,D,H,F }
(S,A),3
(S,B),6
(S,C),2
(S,A,D),6
(S,C,E),3
(S,A,D,F),11
(S,C,E,B),5
(S,C,E,F),9
(S,C,E,H),8
(S,C,E,B,D),9
(S,C,E,B,G),14
(S,C,E,H,G),16
(S,C,E,F,G),14

UCS



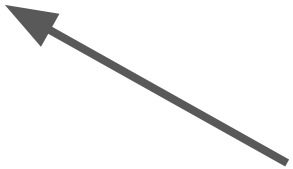
Nodes expanded: 9

Path returned: (S,C,E,B,G)

Greedy

closed_list = { }

(S), 12



Each node is a tuple of (path to state s, heuristic of s)

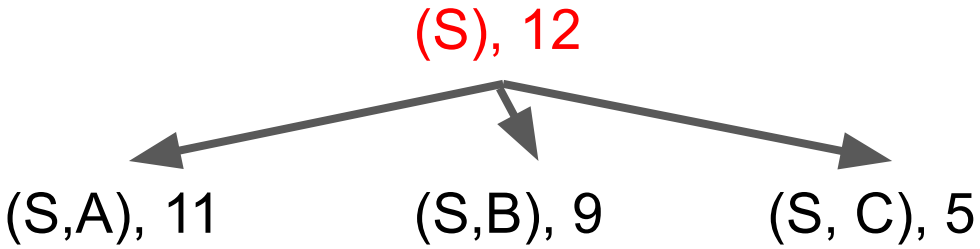
Greedy

closed_list = {S}

(S), 12

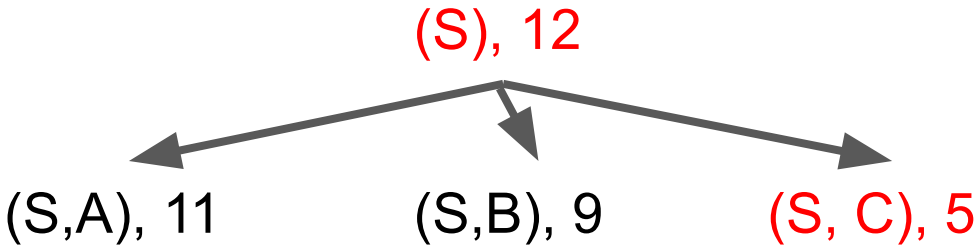
Greedy

closed_list = {S }



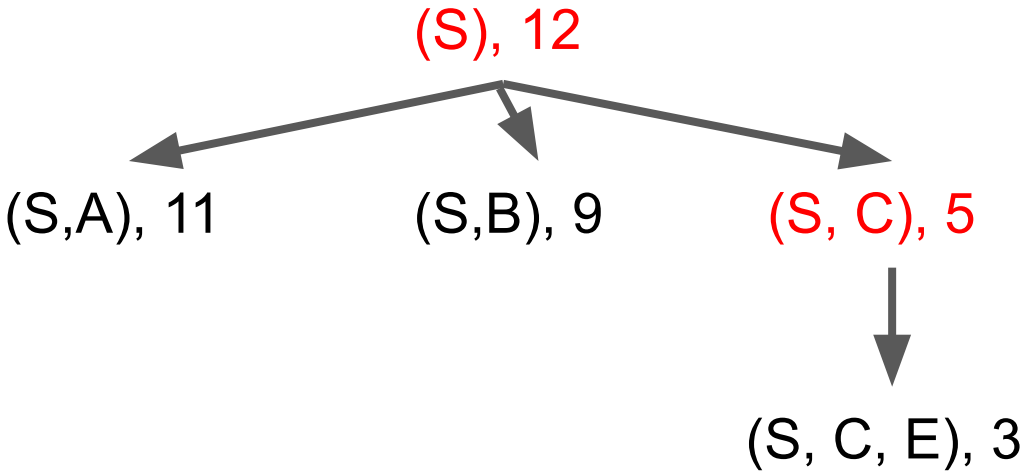
Greedy

closed_list = {S,C }



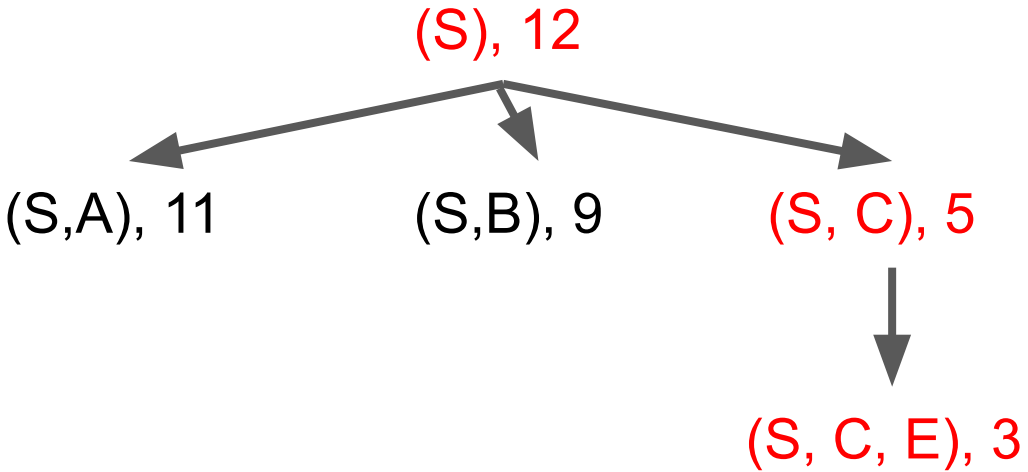
Greedy

closed_list = {S,C }



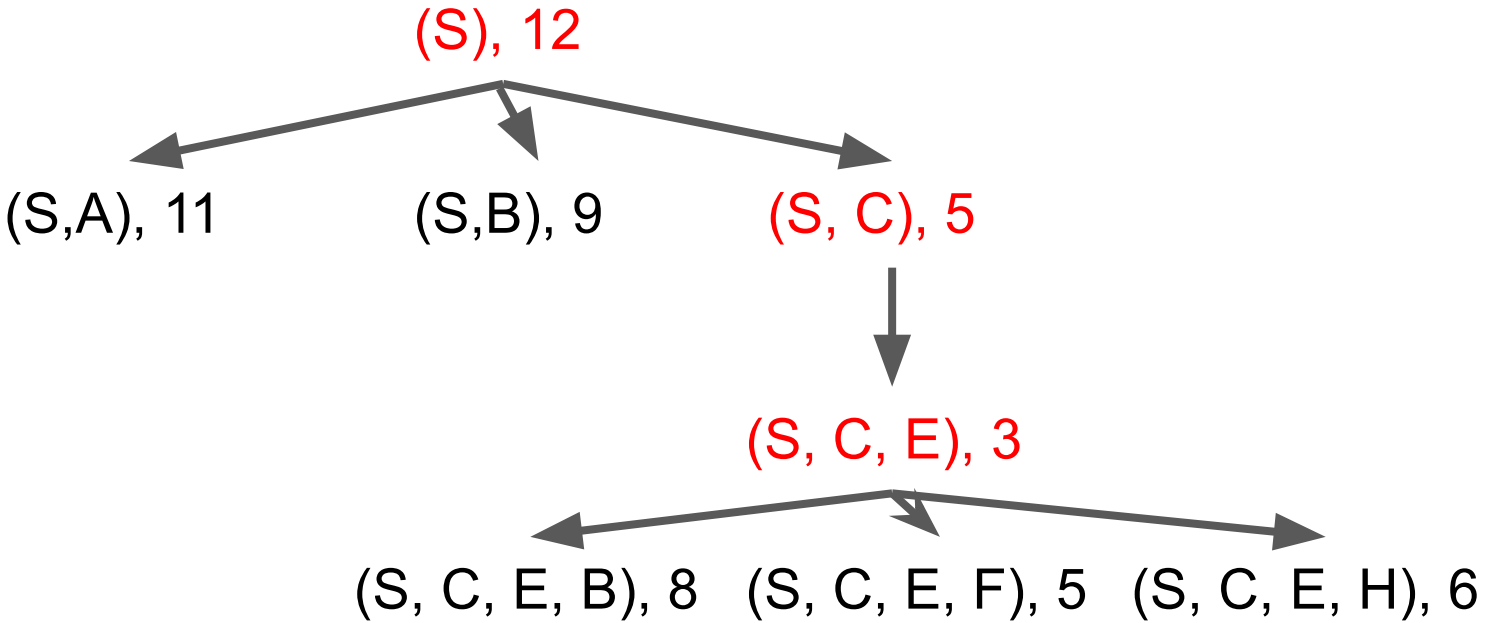
Greedy

closed_list = {S,C,E }



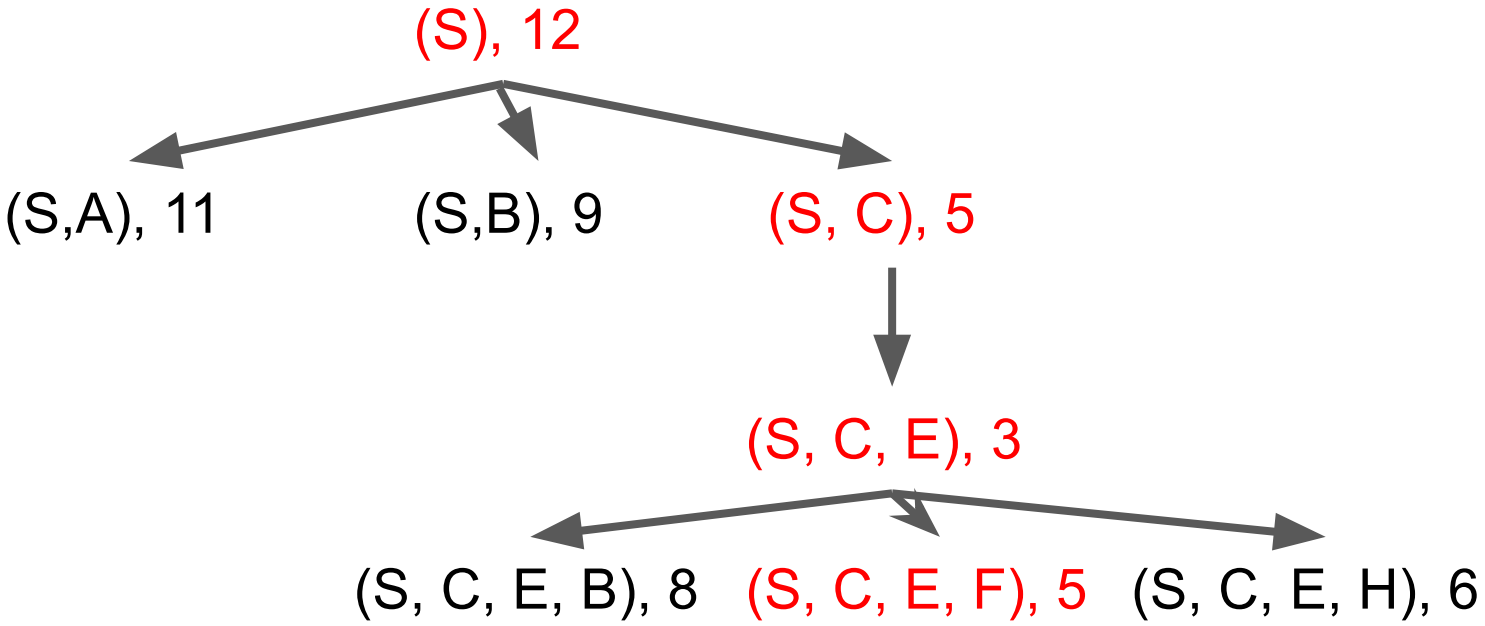
Greedy

closed_list = {S,C,E }



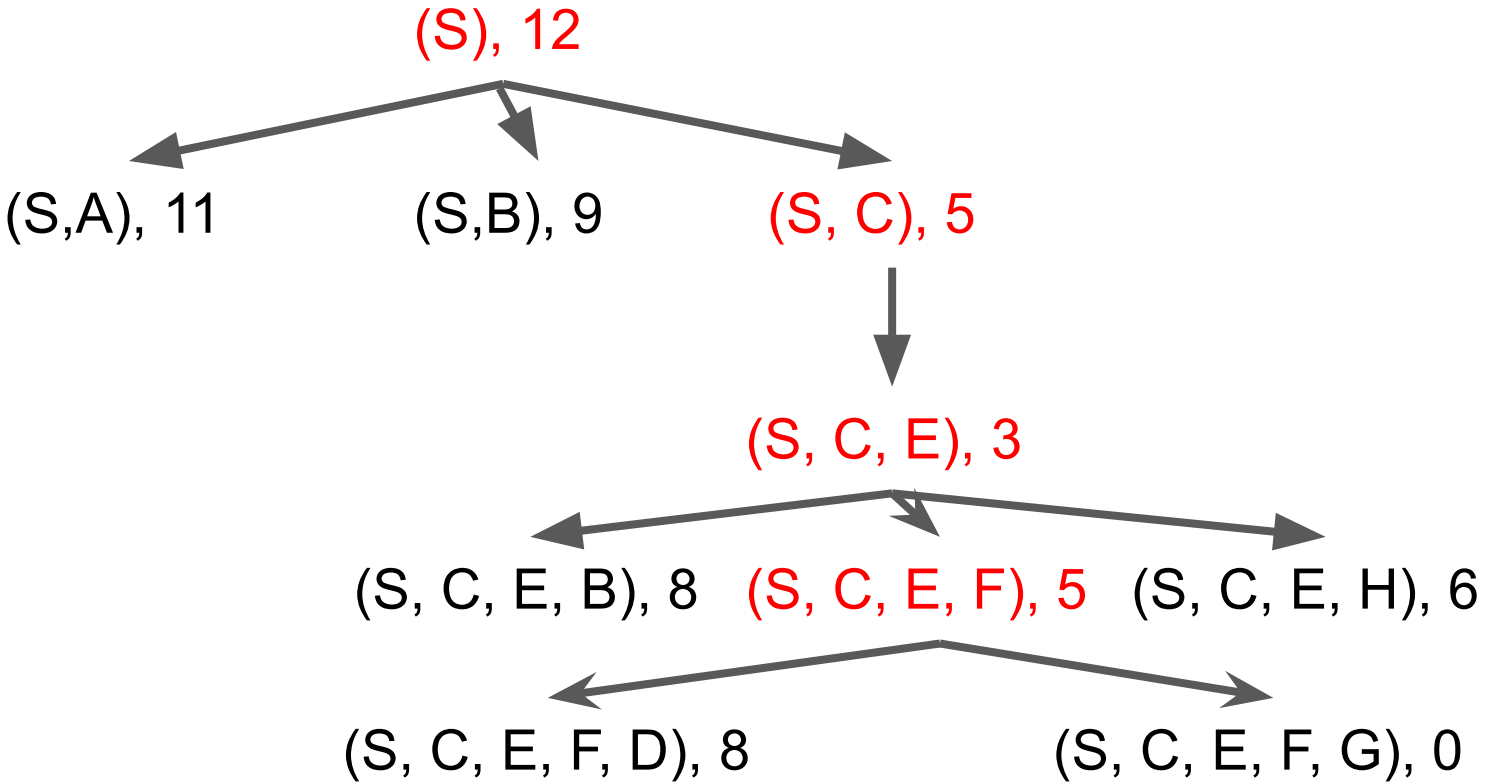
Greedy

closed_list = {S,C,E,F }



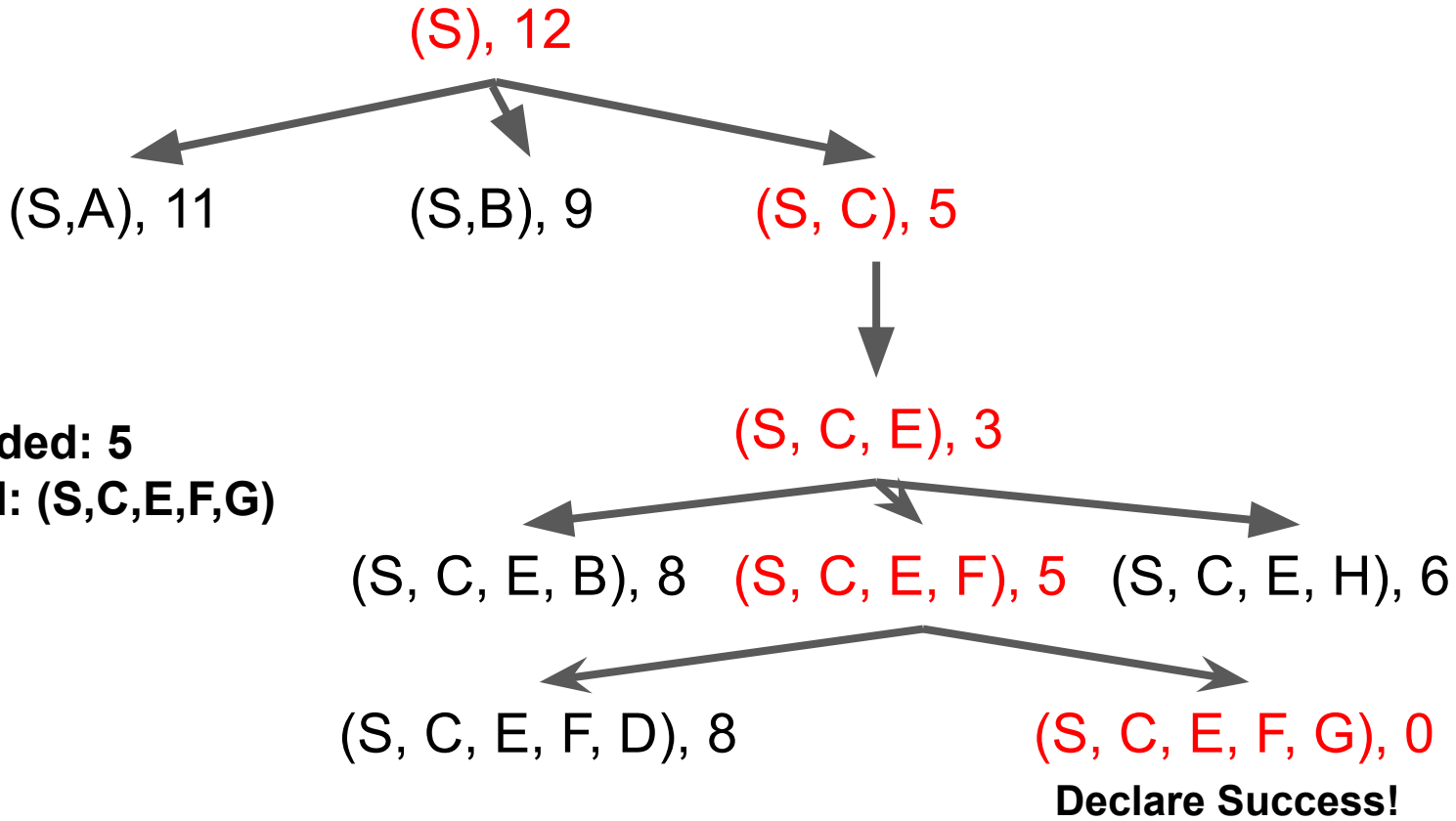
Greedy

closed_list = {S,C,E,F }



Greedy

closed_list = {S,C,E,F,G }



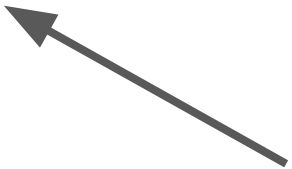
Nodes expanded: 5
Path returned: (S,C,E,F,G)

Declare Success!

A*

closed_list = {}

(S), 0+12



Each node is a tuple of
(path to state s, cumulative cost to s + heuristic of s)

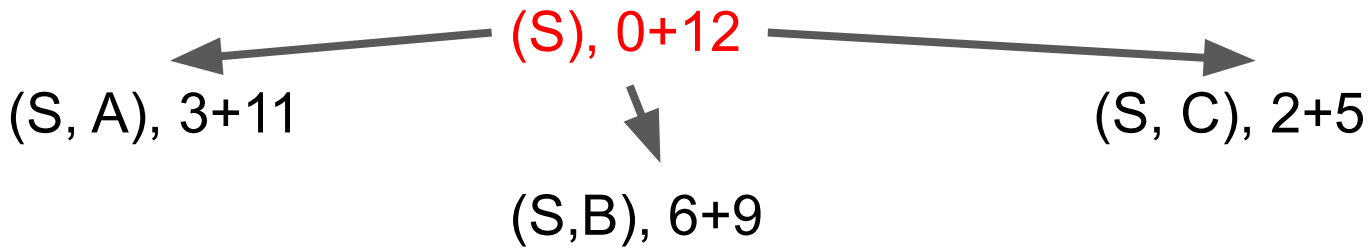
A*

closed_list = {S}

(S), 0+12

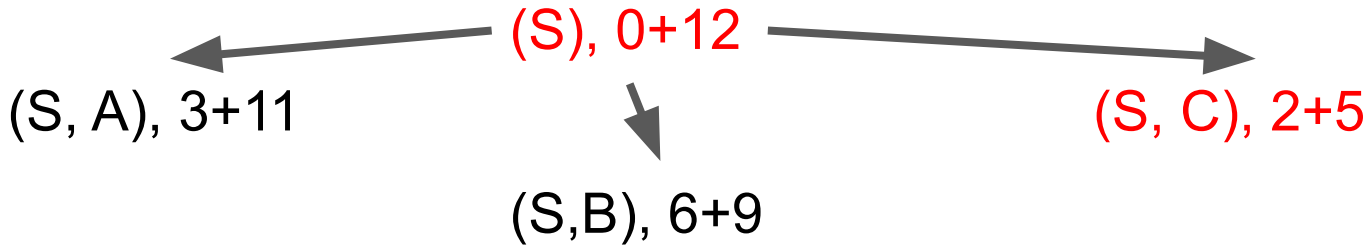
A*

closed_list = {S}



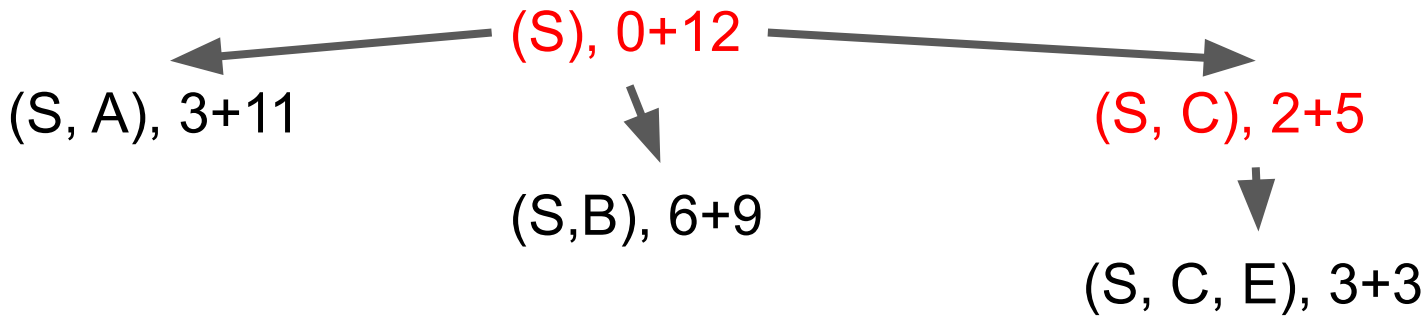
A*

closed_list = {S,C}



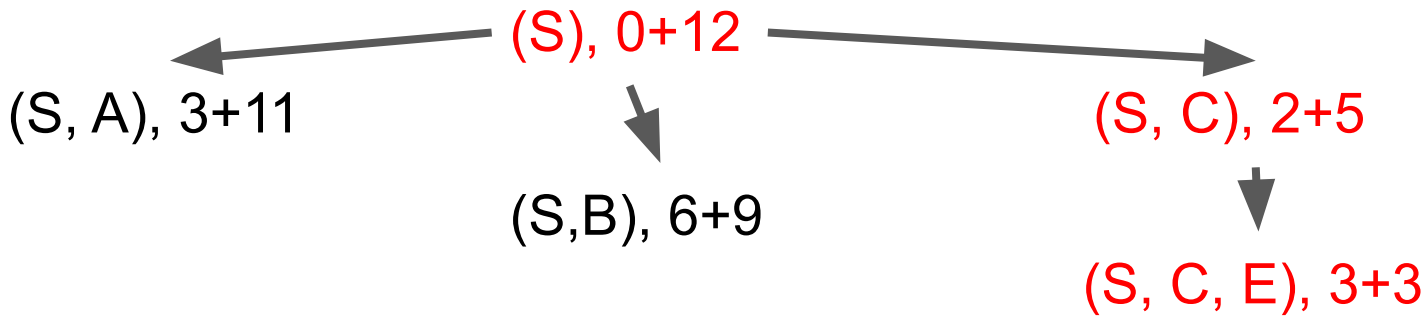
A*

closed_list = {S,C}



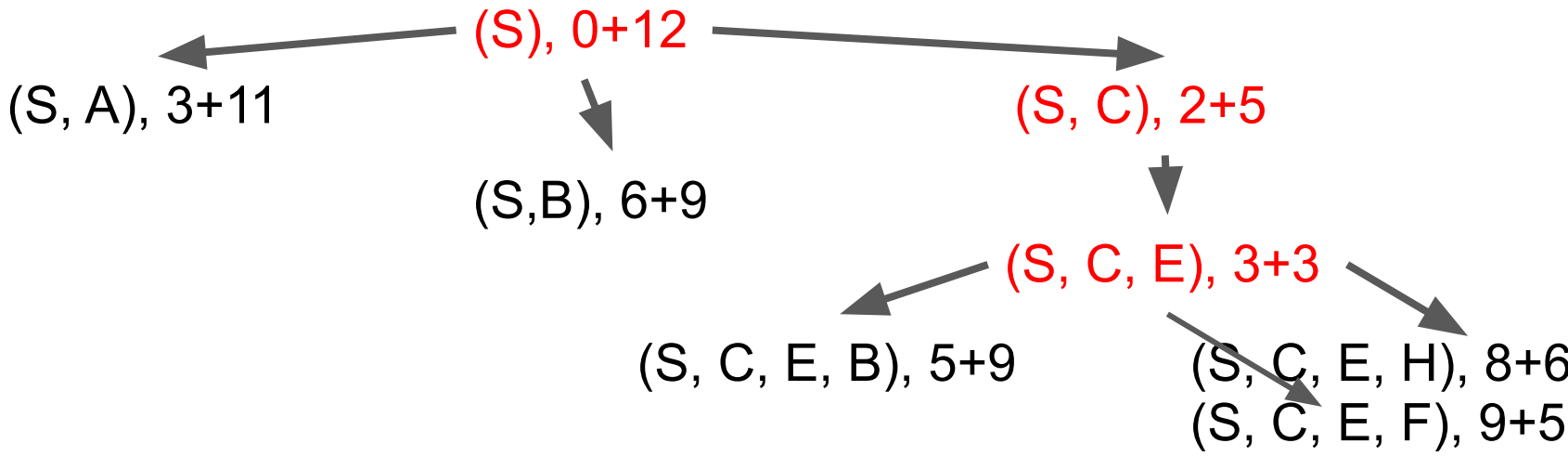
A*

closed_list = {S,C,E}



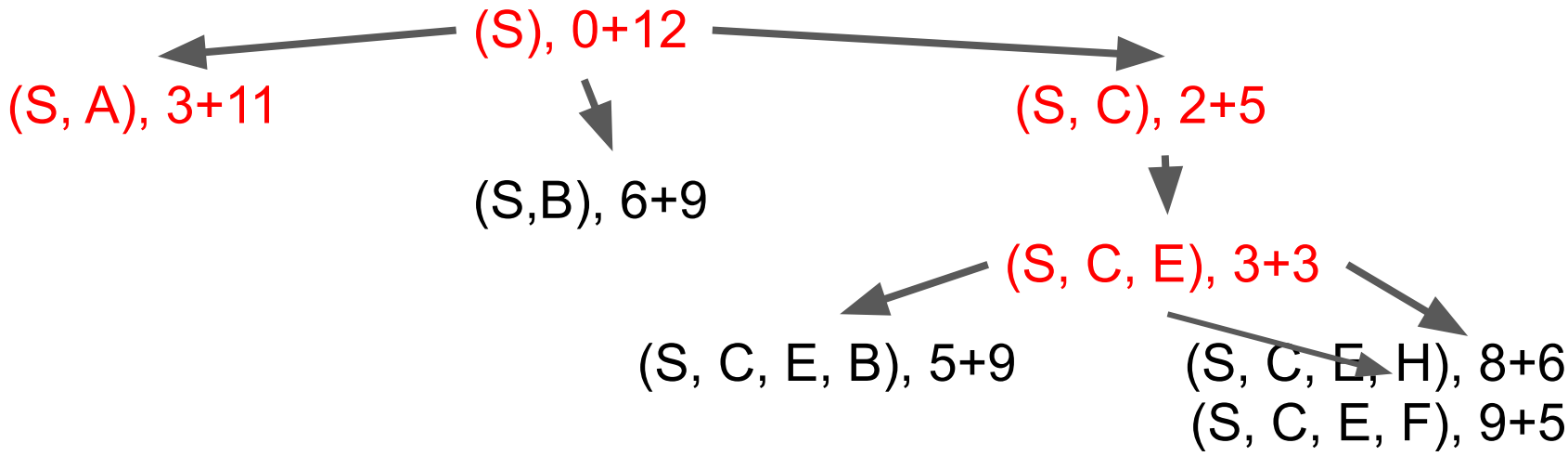
A*

closed_list = {S,C,E}



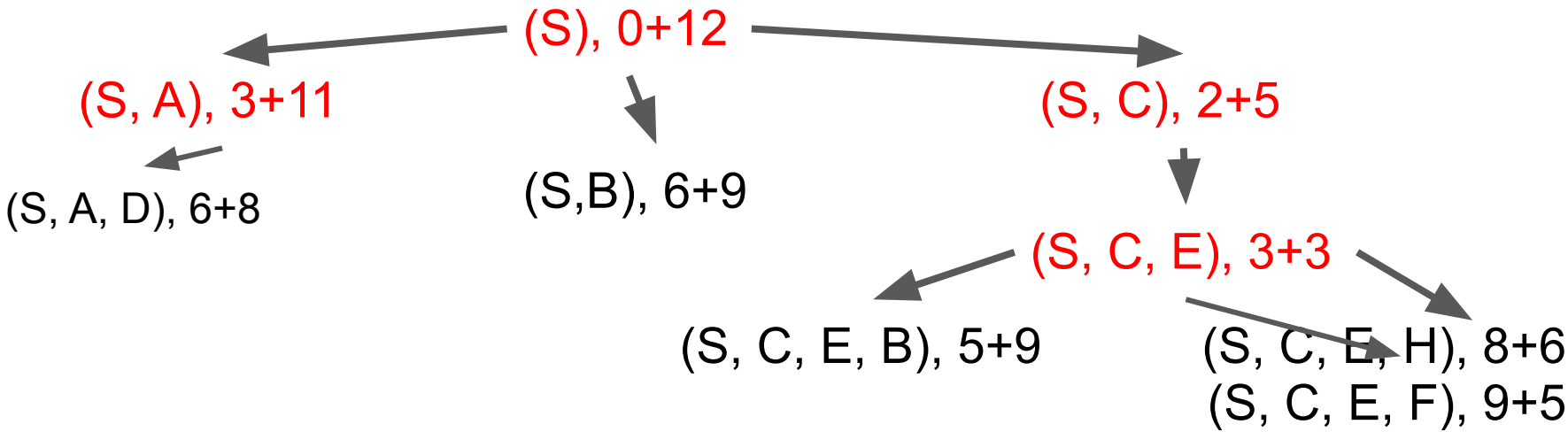
A*

closed_list = {S,C,E,A}



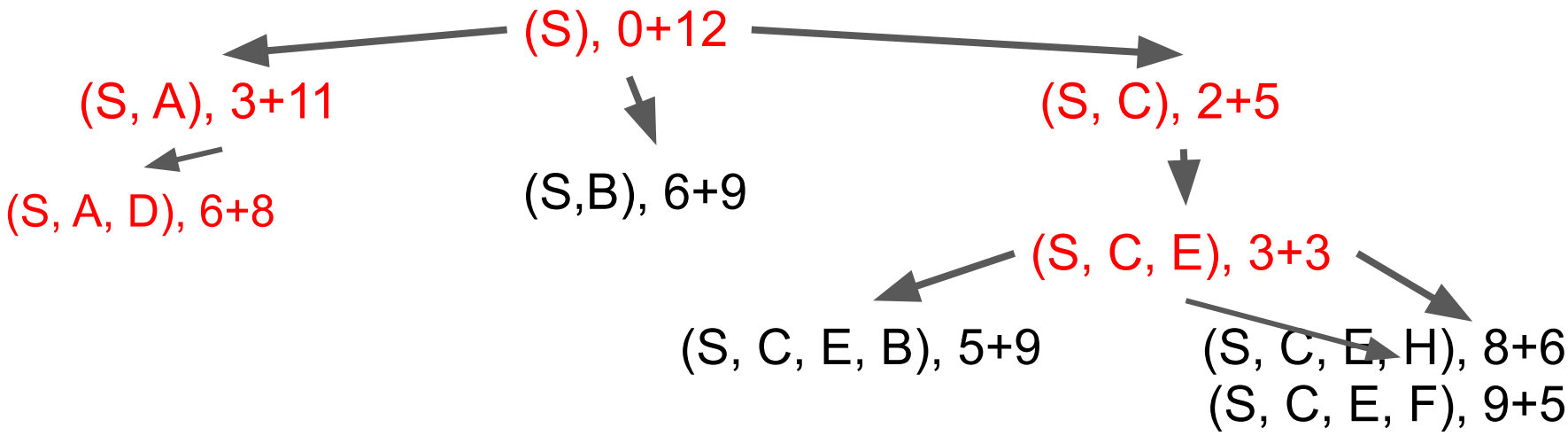
A*

closed_list = {S,C,E,A}



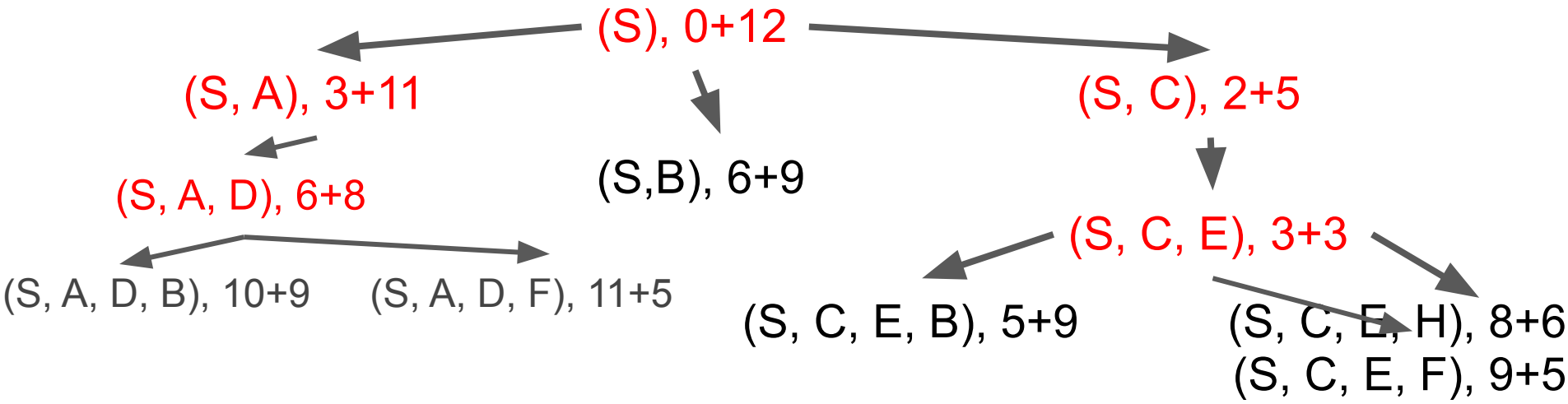
A*

closed_list = {S,C,E,A,D}



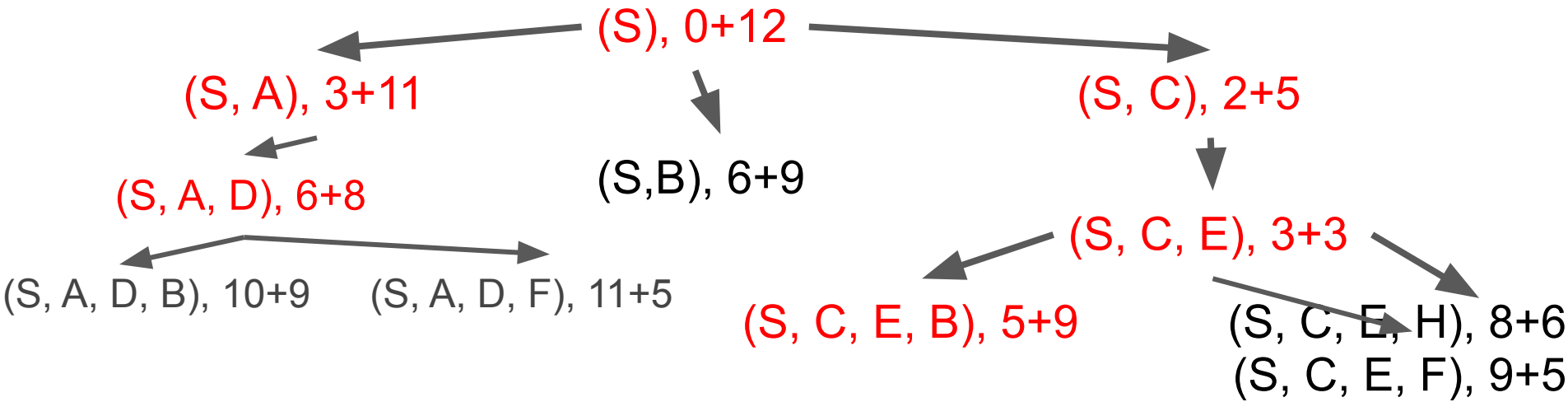
A*

closed_list = {S,C,E,A,D}



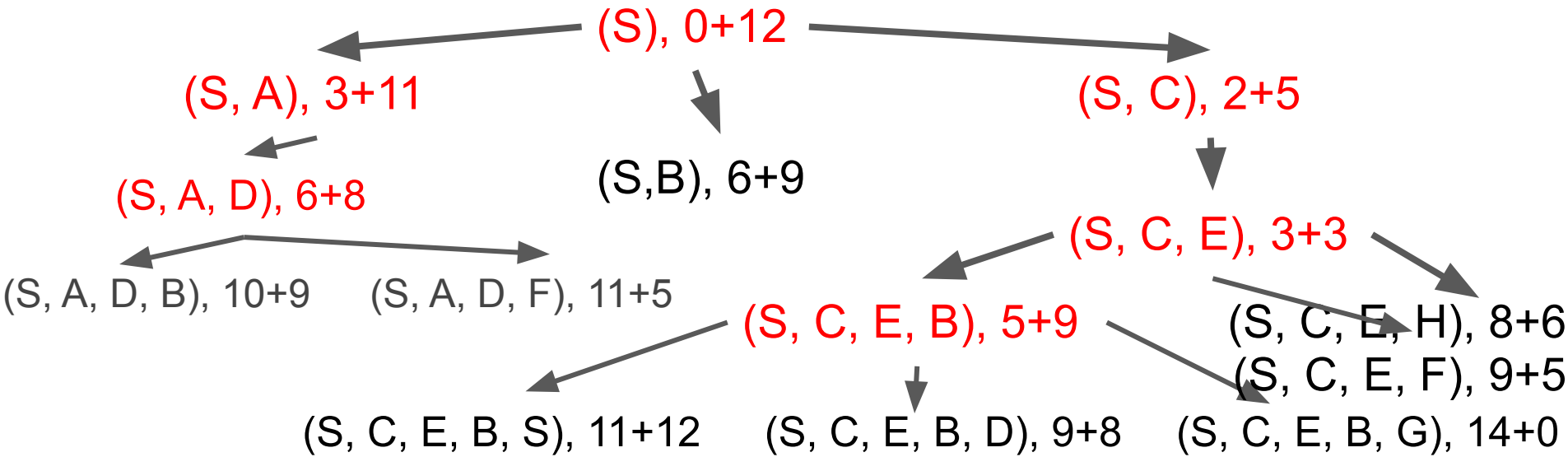
A*

closed_list = {S,C,E,A,D,B}



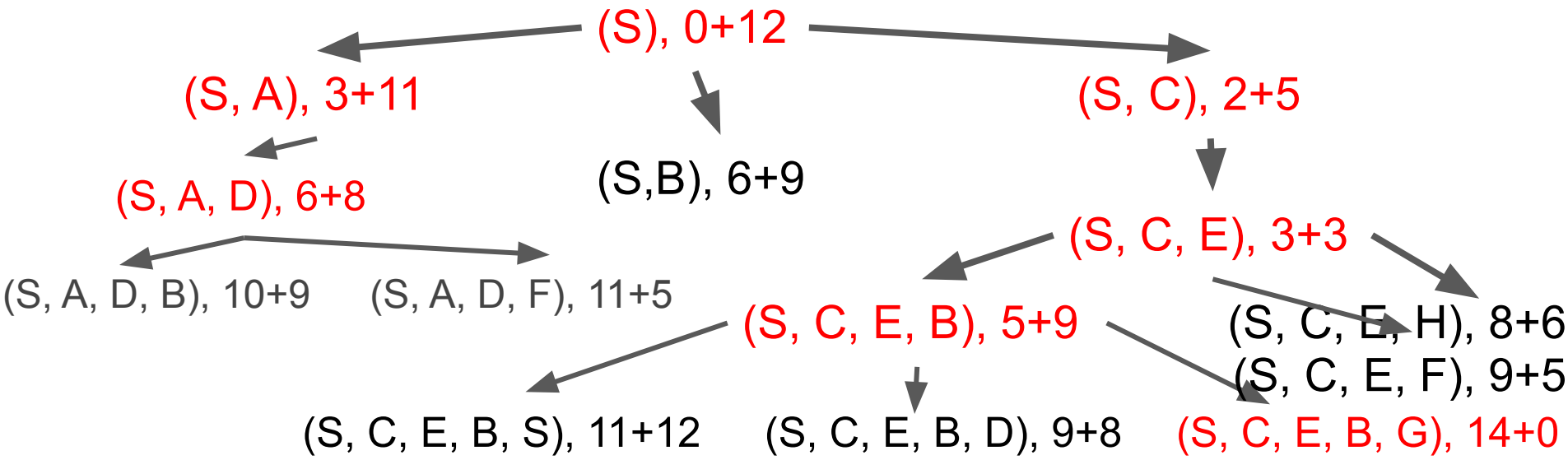
A*

closed_list = {S,C,E,A,D,B}



A*

closed_list = {S,C,E,A,D,B,G}



Declare Success!!!

Node expanded: 7

Path returned: (S, C, E, B, G)