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*
DFS

(S)

closed_list = { }
DFS

(S)

closed_list = {S}
DFS

(S)

(S,A)  (S,B)  (S,C)

closed_list = {S}
(S, A) is chosen to expand because it alphabetically precedes (S, B) and (S, C)
DFS

closed_list = \{S, A\}

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 \}

(S)

(S, A)

(S, A, D)

(S, A, D, B)

(S, A, D, B, F)

(S, B)

(S, C)
DFS

closed_list = {S, A, D, B}

(S) → (S, A) → (S, A, D) → (S, A, D, B)

(S, B) → (S, C)

(S, A, D, F)
DFS

closed_list = {S, A, D, B }

(S)
closed_list = {S, A, D, B, G}
DFS

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

(S)

(S,A)

(S,A,D)

(S,A,D,B)

(S,A,D,B,E)

(S,A,D,B,E,C)

(S,A,D,B,E,F)

(S,A,D,B,E,F,H)

(S,B)

(S,C)
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.
closed_list = \{ S, A, D, B, G, C, F \}
DFS

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

(S)

(S,A)

(S,A,D)

(S,A,D,B)

(S,A,D,B,E)

(S,A,D,B,E,C)

(S,A,D,B,E,F)

(S,A,D,B,E,F,G)

(S,B)

(S,C)

(S,A,D)

(S,A,D,F)

(S,A,D,B,G)

(S,A,D,B,G)

(S,A,D,B,E,H)
DFS

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

(S)

(S,A)

(S,A,D)

(S,A,D,B)

(S,A,D,B,E)

(S,A,D,B,E,C)

(S,A,D,B,E,F)

(S,A,D,B,E,F,G)

Declare success!!!

Nodes expanded: 8
Path returned: (S,A,D,B,E,F,G)
BFS

(S)  closed_list = {}
BFS

(S)  closed_list = {S}
BFS

closed_list = {S }
BFS

closed_list = \{ S, A \}

(S)  
|   
(S, B)  
|   
(S, A)  

(S, C)
BFS

closed_list = \{S,A\}
BFS

closed_list = \{ S, A, B \}
BFS

closed_list = {S,A,B }

(S)
  /  
(S,A)  (S,B)
  |    /  
(S,A,D)  (S,B,D)  (S,B,E)
        |    /  
        (S,B,G)  

BFS

closed_list = {S,A,B,C }

(S) → (S,A) → (S,A,D)

(S,B) → (S,B,D) → (S,B,E)

(S,C)

(S,B,G)
BFS

closed_list = \{ S, A, B, C \}
BFS

closed_list = \{ S, A, B, C, D \}
BFS

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

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

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

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

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

Nodes expanded: 7
Path returned: (S, B, G)
(S), 0   closed_list = {}
UCS

(S), 0  
closed_list = {S }
closed_list = {S}
closed_list = \{ S, C \}
closed_list = {S, C}
closed_list = {S,C,A}
closed_list = {S, C, A}
closed_list = {S, C, A, E}
closed_list = {S,C,A,E}
closed_list = {S, C, A, E, B }
closed_list = \{S,C,A,E,B\}
closed_list = {S,C,A,E,B,D}
closed_list = \{ S, C, A, E, B, D \}
closed_list = {S,C,A,E,B,D }

Skip it!!!
closed_list = {S, C, A, E, B, D, H }

UCS

(S), 0

(S, A), 3

(S, B), 6

(S, C), 2

(S, C, E), 3

(S, A, D), 6

(S, C, E, B), 5

(S, C, E, B, D), 9

(S, C, E, B, G), 14

(S, C, E, F), 9

(S, C, E, H), 8

(S, A, D, F), 11
closed_list = {S,C,A,E,B,D,H }

UCS

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

\text{closed\_list = \{S,C,A,E,B,D,H,F \}}

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

Nodes expanded: 9
Path returned: (S,C,E,B,G)
Greedy

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

closed_list = {}

(S), 12
Greedy

closed_list = \{S\}

(S), 12
Greedy

closed_list = {S}

(S), 12

(S,A), 11  (S,B), 9  (S, C), 5
closed_list = \{S, C\}

Greedy

(S), 12

(S,A), 11 (S,B), 9 (S, C), 5
Greedy

closed_list = \{S, C\}

(S), 12

(S, A), 11
(S, B), 9
(S, C), 5

(S, C, E), 3
closed_list = {S, C, E}

Greedy

(S), 12

(S, A), 11  (S, B), 9  (S, C), 5

(S, C, E), 3
Greedy

`closed_list = {S,C,E}`

```
(S), 12

(S,A), 11  (S,B), 9  (S, C), 5

(S, C, E), 3

(S, C, E, B), 8  (S, C, E, F), 5  (S, C, E, H), 6
```
Greedy

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

(S), 12

(S,A), 11 (S,B), 9 (S, C), 5

(S, C, E), 3

(S, C, E, B), 8 (S, C, E, F), 5 (S, C, E, H), 6
closed_list = \{S,C,E,F\}

Greedy
Greedy

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

closed_list = {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
closed_list = {S}

A*

(S, A), 3+11 → (S), 0+12 → (S, B), 6+9 → (S, C), 2+5
A*

closed_list = \{S,C\}

(S, A), 3+11

(S, B), 6+9

(S), 0+12

(S, C), 2+5
A*

closed_list = {S, C}

(S, A), 3+11

(S, B), 6+9

(S, C), 2+5

(S, C, E), 3+3
A*

closed_list = {S, C, E}

(S, A), 3+11

(S), 0+12

(S, B), 6+9

(S, C), 2+5

(S, C, E), 3+3
A*  

\[
\text{closed_list} = \{S, C, E\}
\]
closed_list = {S,C,E,A}

A*

(S, A), 3+11
(S, B), 6+9
(S, C), 2+5
(S, C, E), 3+3
(S, C, E, B), 5+9
(S, C, E, H), 8+6
(S, C, E, F), 9+5
(S), 0+12
closed_list = \{S, C, E, A\}

A*

(S, A), 3+11
(S, A, D), 6+8
(S, A, D), 6+8

(S), 0+12
(S), 0+12

(S, B), 6+9
(S, B), 6+9

(S, C), 2+5
(S, C), 2+5

(S, C, E), 3+3
(S, C, E), 3+3

(S, C, E, B), 5+9
(S, C, E, B), 5+9

(S, C, E, H), 8+6
(S, C, E, H), 8+6

(S, C, E, F), 9+5
(S, C, E, F), 9+5
closed_list = \{S,C,E,A,D\}

A*

(S), 0+12

(S, A), 3+11

(S, A, D), 6+8

(S, B), 6+9

(S, C), 2+5

(S, C, E), 3+3

(S, C, E, B), 5+9

(S, C, E, H), 8+6

(S, C, E, F), 9+5
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}

(S), 0+12

(S, A), 3+11

(S, A, D), 6+8

(S, A, D, B), 10+9

(S, A, D, F), 11+5

(S, B), 6+9

(S, C), 2+5

(S, C, E), 3+3

(S, C, E, B), 5+9

(S, C, E, B, S), 11+12

(S, C, E, B, D), 9+8

(S, C, E, B, G), 14+0

(S, C, E, H), 8+6

(S, C, E, F), 9+5
Node expanded: 7
Path returned: (S, C, E, B, G)