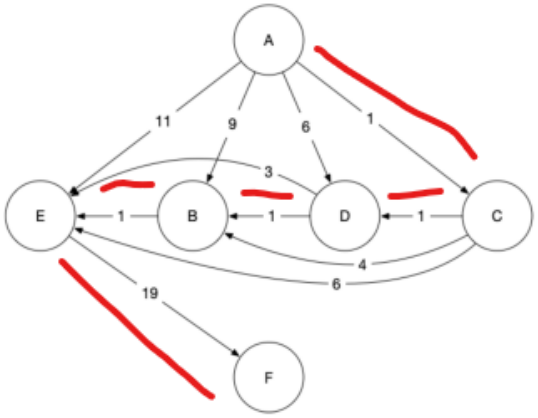
Question 1: Find a path from the start (A) to the goal (F) using DFS of Generic Search Algorithm

a) Show open/closed data structures for each iteration
b) Show final search tree after goal state is reached
c) Show final result (path)

a)
Iteration 1: Open = [**C(1), D(6), B(9), E(11)**] Closed = [A]
Iteration 2: Open = [**D(1), B(4), E(6),** D(6), B(9), E(11)] Closed = [A, C]
Iteration 3: Open = [**B(1), E(3),** B(4), E(6), D(6), B(9), E(11)] Closed = [A, C, D]
Iteration 4: Open = [**E(1)**, E(3), B(4), E(6), D(6), B(9), E(11)] Closed = [A,C, D, B]
Iteration 5: Open = [**F(19)**, E(3), B(4), E(6), D(6), B(9), E(11)] Closed = [A, C, D, B, E]
Iteration 6: Open = [E(3), B(4), E(6), D(6), B(9), E(11)] Closed = [A, C, D, B, E, F]

b)



c)
Final Path: [A, C, D, B, E, F]

Question 2: Use uniform cost search

UCS Steps
1.
A(0)
Open: [**C(1), D(6), B(9), E(11)**]
Closed: [A]

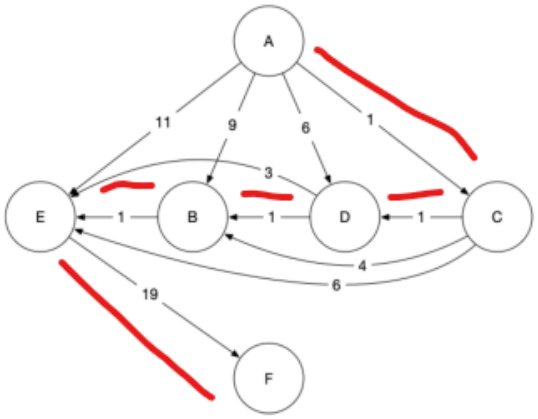
2.
C(1)
[**D(2), B(5),** D(6), **E(7),** B(9), E(11)]
[A, C]

3.
D(2)
[**B(3), E(5),** B(5), D(6), E(7), B(9), E(11)]
[A, C, D]

4.
B(3)
[**E(4),** E(5), B(5), D(6), E(7), B(9), E(11)]
[A, C, D, B]

5.
E(4)
[E(5), B(5), D(6), E(7), B(9), E(11), **F(23)**]
[A, C, D, B, E]

Go through each node → until it's **F(23)** turn then exit!



Final Path: [A C D B E F]

Question 3:

1.
A(23)
Open: [**E(11), B(12), D(13), C(14)**]
Closed: [A]

2.
E(11)
[B(12), D(13), C(14), **F(30)**]
[A, E]

3.
B(12)
[**E(10)**, D(13), C(14), F(30)]
[A, E, B]

4.
E(10)
[D(13), C(14), **F(29)**, F(30)]
[A, E, B, E]

5.
D(13)

[**B(7), E(9)**, C(14), F(29), F(30)]
[A, E, B, E, D]

6.
B(7)
[**E(8)**, E(9), C(14), F(29), F(30)]
[A, E, B, E, D, B]

7.
E(8)
[E(9), C(14), **F(27),** F(29), F(30)]
[A, E, B, E, D, B, E]

8.
E(9)
[C(14), F(27), **F(28)**, F(29), F(30)]
[A, E, B, E, D, B, E, E]

9.
C(14)
[**E(7), B(8), D(9),** F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C]

10.
E(7)
[B(8), D(9)**, F(26)**, F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E]

11.
B(8)
[**E(6),** D(9), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B]

12.
E(6)
[D(9), **F(25),** F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E]

13.
D(9)
[**E(5), B(6),** F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D]

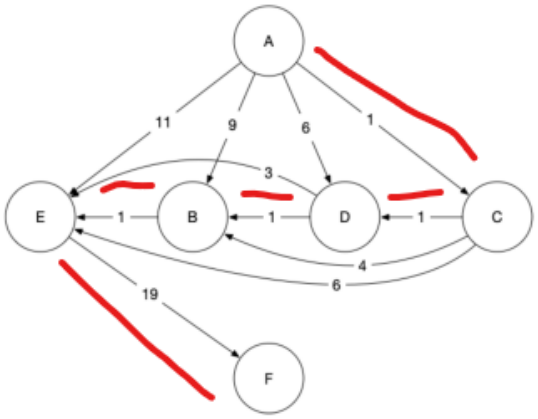
14.
E(5)
[B(6), **F(24),** F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D, E]

15.
B(6)

[**E(4),** F(24), F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D, E, B]

16.
E(4)
[**F(23)**, F(24), F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D, E, B, E]

17. **(only goal node is left in the array, pick lowest cost and end the pathing)**
F(23)
[F(24), F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D, E, B, E, **F**]



Final Path [A, C, D, B, E, F]

Question 4:

This heuristic is admissible because **every** path from any given node **n** on the graph to the goal exceeds the heuristic value of node **n**. The closest values are the heuristic value of A vs real cost to goal which is 23 (passes the admissible check 23<=23) as well as the goal F (passes the check as well 0<=0).

Question 5: It's the exact same search path as the A* search algorithm because a consistent heuristic is also admissible and will traverse a graph in the same way

1.
A(23)
Open: [**E(11), B(12), D(13), C(14)**]
Closed: [A]

2.
E(11)
[B(12), D(13), C(14), **F(30)**]
[A, E]

3.
B(12)

[**E(10)**, D(13), C(14), F(30)]
[A, E, B]

4.
E(10)
[D(13), C(14), **F(29)**, F(30)]
[A, E, B, E]

5.
D(13)
[**B(7), E(9)**, C(14), F(29), F(30)]
[A, E, B, E, D]

6.
B(7)
[**E(8)**, E(9), C(14), F(29), F(30)]
[A, E, B, E, D, B]

7.
E(8)
[E(9), C(14), **F(27),** F(29), F(30)]
[A, E, B, E, D, B, E]

8.
E(9)
[C(14), F(27), **F(28)**, F(29), F(30)]
[A, E, B, E, D, B, E, E]

9.
C(14)
[**E(7), B(8), D(9),** F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C]

10.
E(7)
[B(8), D(9)**, F(26)**, F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E]

11.
B(8)
[**E(6),** D(9), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B]

12.
E(6)
[D(9), **F(25),** F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E]

13.
D(9)

[**E(5), B(6),** F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D]

14.
E(5)
[B(6), **F(24),** F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D, E]

15.
B(6)
[**E(4),** F(24), F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D, E, B]

16.
E(4)
[**F(23)**, F(24), F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D, E, B, E]

17. **(only goal node is left in the array, pick lowest cost and end the pathing)**
F(23)
[F(24), F(25), F(26), F(27), F(28), F(29), F(30)]
[A, E, B, E, D, B, E, E, C, E, B, E, D, E, B, E, **F**]

Question 6:

Components to solve this search problem:
-cost to move rooms
-how many moves before it loses power
-what is the most amount of energy it can regain (in moves) when it comes into contact with the charger
-what are the conditions to charging to full battery (time)

Question 7:

Uniform Cost Search would provide us with the fastest and cheapest route to navigate/clean all 9 spaces with diligence. I'd also move the charger location to the center of the grid so the roomba will be, at most, 2 moves away from the charger at any given room.

If the robot had to recharge after cleaning a room, the primary question would be how long would it take before the robot stopped moving. Better yet, does neglecting charging impact the robot's ability to move around the grid or does it simply lose its cleaning functionality?