SDL2 Course Day-8
AI & Pathfinding

What Is AI?

Artificial Intelligence is a very very broad term that can mean a lot of things.
When we say a particular object is Artificially Intelligent, in a very broad way, it means that the object can do stuff without us explicitly telling it to.

An AI **Agent** in some **Environment** performs the following actions:
> Observes the **State** of the Environment.
> Processes and chooses some **Action** to do on the Environment.

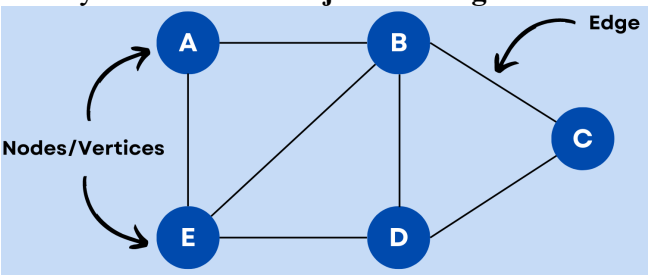
The Action that the Agent performs **changes the state** of the Environment.
Then the Agent observes the new state and the loop repeats.

Graphs:

It's a data structure which is extremely convenient in representing city maps or a lot of other problems.
A graph consists of two elements:
Nodes (Circles) & **Edges** (Lines connecting two circles).

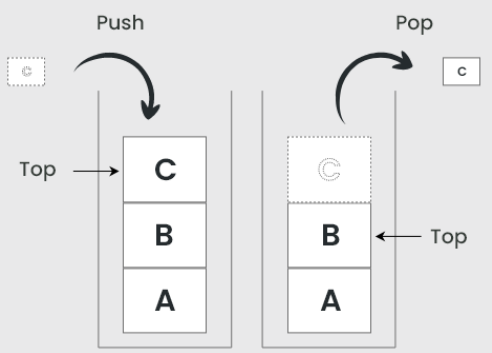
In the context of a city map:
> cities represent nodes.
> roads represent edges. (A road / highway connects two cities)

If there is a number / weight on each edge, it's called as **weighted graph**.
Else if there are **no** numbers / weights, then it's called as **unweighted graph.**
We say two nodes are **adjacent / neighbours** if an edge is connecting them**.**

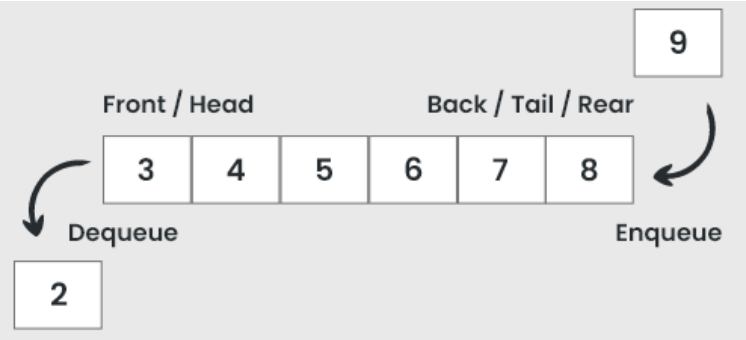


Stacks & Queues:

Stack:



Queue:



To convert a list to a stack or a queue:
If we insert and delete from the same side of the list – stack,
If we insert and delete from different sides of the list – queue.

Depth First Search & Breadth First Search & Greedy Best First Search:

1. Initialize **Frontier** and **Explored** as two lists (Explored can be a set also).
2. add starting node to Frontier
3. while (Frontier is not empty) {
    1. remove a node from frontier **(Difference between BFS, DFS & GBFS)**
    2. add node to explored
    3. check if node is goal or not
    4. if goal => get path from explored.
    5. if not goal => for each neighbour in the node's neighbours {
        1. if the neighbour is not in explored and not in frontier, add it to frontier
        2. set parent of neighbour as node **only if** it is added to the frontier.
    }
}

DFS: Frontier is stack.
BFS: Frontier is queue.
GBFS: Frontier is sorted based on heuristic (node with smallest heuristic value is removed).

Heuristic Function:

Heuristic is a fancy word for an **estimation / prediction**.
When we try to estimate how far a node is from the goal node, we call this distance as the heuristic value of the node.

Manhattan's Heuristic:

$$h(node) = |node.x - goal.x| + |node.y - goal.y|$$

Euler's Heuristic:

$$h(node) = \sqrt{(node.x - goal.x)^2 + (node.y - goal.y)^2}$$

Cost Function:

Cost function is the total cost we have currently spent to get to the current node.
This is the **Actual Distance** we have traversed so far to get to the current node since the start of our journey.
For a node who is discovered by a parent:

$$c(node) = edge\_weight(node, parent) + c(parent)$$

Note that the cost of the starting node or $c(start) = 0$.
The cost of the nodes discovered by the starting node (node's adjacent to the start node) are the edge weights between the start node and its neighbour + 0.

A*:

1. Initialize **Frontier** and **Explored** as two lists (Explored can be a set also).
2. add starting node to Frontier.
3. while (Frontier is not empty) {
    1. remove node with smallest f value from frontier
    2. add the node to explored
    3. check if node is goal or not
    4. if goal => get path from explored
    5. if not goal => for each neighbour in neighbours {
        1. calculate f value of neighbour.
        2. if it is not in explored and not in frontier, add it to frontier
        3. if it is in explored or frontier and if its f value is lesser than what is present, we remove it from wherever it is and add it to frontier
        4. set parent of neighbour as node **only if** it is added to the frontier.
    }
}

$$f(node) = h(node) + c(node)$$
$$c(node) = edge\_weight(node, parent) + c(parent)$$

Pros & Cons:

> BFS:

- **Always** finds the shortest path in an **Unweighted Graph** (No edge weights).
- Explores a lot (huge number of nodes), hence takes **longer time**.

> DFS:

- Explores lesser number of nodes (compared to BFS), hence takes **shorter time**.
- **May or May not** find the shortest path.

> GBFS:

- Explores nodes **which it thinks are closer** to the goal.
- **Very unlikely** to find shortest path in complex graphs.

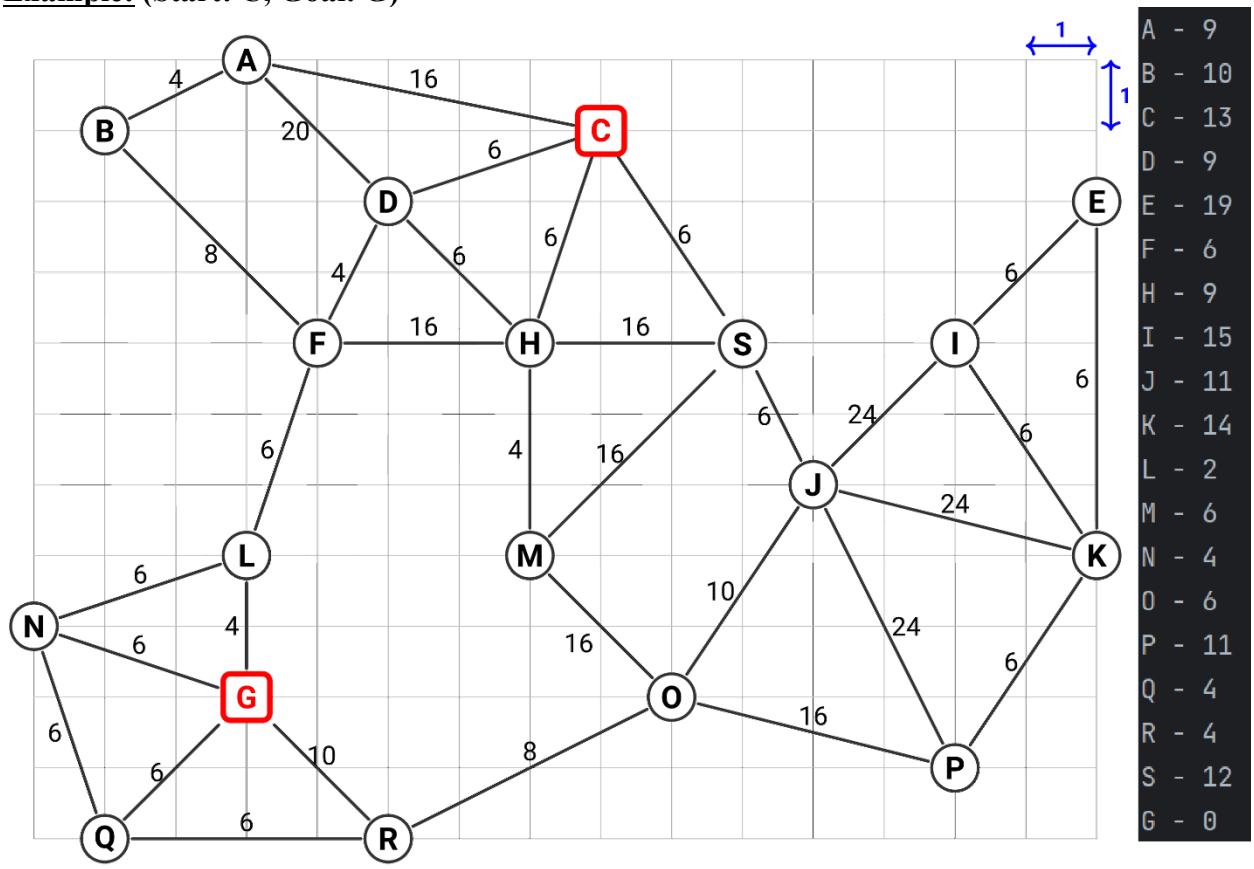
> A*:

- Explores nodes which it thinks are closer while also **considering the current cost so far.**
- **Very likely** to find shortest path; it all depends on how good the heuristic function is.

References:

Stack: https://www.geeksforgeeks.org/stack-data-structure/
Queue: https://www.geeksforgeeks.org/queue-data-structure/
Graph: https://www.geeksforgeeks.org/introduction-to-graphs-data-structure-and-algorithm-tutorials

Example: **(Start: C, Goal: G)**



> BFS:

```
Iteration 1
Frontier: {C}
Explored: {}
Node removed: C
Neighbours: {(A, not found), (D, not found), (H, not found), (S, not found)}

Iteration 2
Frontier: {A, D, H, S}
Explored: {(C, -)}
Node removed: A
Neighbours: {(B, not found), (C, found), (D, found)}

Iteration 3
Frontier: {D, H, S, B}
Explored: {(A, C), (C, -)}
Node removed: D
Neighbours: {(A, found), (C, found), (F, not found), (H, found)}
```

```
Iteration 4
Frontier: {H, S, B, F}
Explored: {(A, C), (C, -), (D, C)}
Node removed: H
Neighbours: {(C, found), (D, found), (F, found), (M, not found), (S, found)}

Iteration 5
Frontier: {S, B, F, M}
Explored: {(A, C), (C, -), (D, C), (H, C)}
Node removed: S
Neighbours: {(C, found), (H, found), (J, not found), (M, found)}

Iteration 6
Frontier: {B, F, M, J}
Explored: {(A, C), (C, -), (D, C), (H, C), (S, C)}
Node removed: B
Neighbours: {(A, found), (F, found)}
```

```
Iteration 7
Frontier: {F, M, J}
Explored: {(A, C), (B, A), (C, -), (D, C), (H, C), (S, C)}
Node removed: F
Neighbours: {(B, found), (D, found), (H, found), (L, not found)}

Iteration 8
Frontier: {M, J, L}
Explored: {(A, C), (B, A), (C, -), (D, C), (F, D), (H, C), (S, C)}
Node removed: M
Neighbours: {(H, found), (O, not found), (S, found)}

Iteration 9
Frontier: {J, L, O}
Explored: {(A, C), (B, A), (C, -), (D, C), (F, D), (H, C), (M, H), (S, C)}
Node removed: J
Neighbours: {(I, not found), (K, not found), (O, found), (P, not found), (S, found)}

Iteration 10
Frontier: {L, O, I, K, P}
Explored: {(A, C), (B, A), (C, -), (D, C), (F, D), (H, C), (J, S), (M, H), (S, C)}
Node removed: L
Neighbours: {(F, found), (G, not found), (N, not found)}
```

```
Iteration 11
Frontier: {O, I, K, P, G, N}
Explored: {(A, C), (B, A), (C, -), (D, C), (F, D), (H, C), (J, S), (L, F), (M, H), (S, C)}
Node removed: O
Neighbours: {(J, found), (M, found), (P, found), (R, not found)}
```

```
Iteration 12
Frontier: {I, K, P, G, N, R}
Explored: {(A, C), (B, A), (C, -), (D, C), (F, D), (H, C), (J, S), (L, F), (M, H), (O, M), (S, C)}
Node removed: I
Neighbours: {(E, not found), (J, found), (K, found)}

Iteration 13
Frontier: {K, P, G, N, R, E}
Explored: {(A, C), (B, A), (C, -), (D, C), (F, D), (H, C), (I, J), (J, S), (L, F), (M, H), (O, M), (S, C)}
Node removed: K
Neighbours: {(E, found), (I, found), (J, found), (P, found)}

Iteration 14
Frontier: {P, G, N, R, E}
Explored: {(A, C), (B, A), (C, -), (D, C), (F, D), (H, C), (I, J), (J, S), (K, J), (L, F), (M, H), (O, M), (S, C)}
Node removed: P
Neighbours: {(J, found), (K, found), (O, found)}

Iteration 15
Frontier: {G, N, R, E}
Explored: {(A, C), (B, A), (C, -), (D, C), (F, D), (H, C), (I, J), (J, S), (K, J), (L, F), (M, H), (O, M), (P, J), (S, C)}
Node removed: G

Goal found!
Path : {C, D, F, L, G}
```

Final path cost = 6 + 4 + 6 + 4 = 20;
Number of Explored nodes = 15

> DFS:

```
Iteration 1
Frontier: {C}
Explored: {}
Node removed: C
Neighbours: {(A, not found), (D, not found), (H, not found), (S, not found)}

Iteration 2
Frontier: {A, D, H, S}
Explored: {(C, -)}
Node removed: S
Neighbours: {(C, found), (H, found), (J, not found), (M, not found)}
```

```
Iteration 3
Frontier: {A, D, H, J, M}
Explored: {(C, -), (S, C)}
Node removed: M
Neighbours: {(H, found), (O, not found), (S, found)}

Iteration 4
Frontier: {A, D, H, J, O}
Explored: {(C, -), (M, S), (S, C)}
Node removed: O
Neighbours: {(J, found), (M, found), (P, not found), (R, not found)}

Iteration 5
Frontier: {A, D, H, J, P, R}
Explored: {(C, -), (M, S), (O, M), (S, C)}
Node removed: R
Neighbours: {(G, not found), (O, found), (Q, not found)}

Iteration 6
Frontier: {A, D, H, J, P, G, Q}
Explored: {(C, -), (M, S), (O, M), (R, O), (S, C)}
Node removed: Q
Neighbours: {(G, found), (N, not found), (R, found)}
```

```
Iteration 7
Frontier: {A, D, H, J, P, G, N}
Explored: {(C, -), (M, S), (O, M), (Q, R), (R, O), (S, C)}
Node removed: N
Neighbours: {(G, found), (L, not found), (Q, found)}

Iteration 8
Frontier: {A, D, H, J, P, G, L}
Explored: {(C, -), (M, S), (N, Q), (O, M), (Q, R), (R, O), (S, C)}
Node removed: L
Neighbours: {(F, not found), (G, found), (N, found)}

Iteration 9
Frontier: {A, D, H, J, P, G, F}
Explored: {(C, -), (L, N), (M, S), (N, Q), (O, M), (Q, R), (R, O), (S, C)}
Node removed: F
Neighbours: {(B, not found), (D, found), (H, found), (L, found)}
```

```
Iteration 10
Frontier: {A, D, H, J, P, G, B}
Explored: {(C, -), (F, L), (L, N), (M, S), (N, Q), (O, M), (Q, R), (R, O), (S, C)}
Node removed: B
Neighbours: {(A, found), (F, found)}

Iteration 11
Frontier: {A, D, H, J, P, G}
Explored: {(B, F), (C, -), (F, L), (L, N), (M, S), (N, Q), (O, M), (Q, R), (R, O), (S, C)}
Node removed: G

Goal found!
Path : {C, S, M, O, R, G}
```

Final path cost = 6 + 16 + 16 + 8 + 10 = 56;
Number of Explored nodes = 11

> GBFS:

```
Iteration 1
Frontier: {(C, 13)}
Explored: {}
Node removed: C
Neighbours: {(A, 9, not found), (D, 9, not found), (H, 9, not found), (S, 12, not found)}

Iteration 2
Frontier: {(A, 9), (D, 9), (H, 9), (S, 12)}
Explored: {(C, -)}
Node removed: A
Neighbours: {(B, 10, not found), (C, 13, found), (D, 9, found)}

Iteration 3
Frontier: {(D, 9), (H, 9), (S, 12), (B, 10)}
Explored: {(A, C), (C, -)}
Node removed: D
Neighbours: {(A, 9, found), (C, 13, found), (F, 6, not found), (H, 9, found)}

Iteration 4
Frontier: {(H, 9), (S, 12), (B, 10), (F, 6)}
Explored: {(A, C), (C, -), (D, C)}
Node removed: F
Neighbours: {(B, 10, found), (D, 9, found), (H, 9, found), (L, 2, not found)}
```

```
Iteration 5
Frontier: {(H, 9), (S, 12), (B, 10), (L, 2)}
Explored: {(A, C), (C, -), (D, C), (F, D)}
Node removed: L
Neighbours: {(F, 6, found), (G, 0, not found), (N, 4, not found)}

Iteration 6
Frontier: {(H, 9), (S, 12), (B, 10), (G, 0), (N, 4)}
Explored: {(A, C), (C, -), (D, C), (F, D), (L, F)}
Node removed: G

Goal found!
Path : {C, D, F, L, G}
```

Final path cost = 6 + 4 + 6 + 4 = 20;
Number of Explored nodes = 6

A*:

```
Iteration 1
Frontier: {(C, 13)}
Explored: {}
Node removed: C
Neighbours: {(A, 9 + 16, not found), (D, 9 + 6, not found), (H, 9 + 6, not found), (S, 12 + 6, not found)}

Iteration 2
Frontier: {(A, 25), (D, 15), (H, 15), (S, 18)}
Explored: {(C, -)}
Node removed: D
Neighbours: {(A, 9 + 26, found, 25), (C, 13 + 12, found, 13), (F, 6 + 10, not found), (H, 9 + 12, found, 15)}

Iteration 3
Frontier: {(A, 25), (H, 15), (S, 18), (F, 16)}
Explored: {(C, -), (D, C)}
Node removed: H
Neighbours: {(C, 13 + 12, found, 13), (D, 9 + 12, found, 15), (F, 6 + 22, found, 16),
             (M, 6 + 10, not found), (S, 12 + 22, found, 18)}

Iteration 4
Frontier: {(A, 25), (S, 18), (F, 16), (M, 16)}
Explored: {(C, -), (D, C), (H, C)}
Node removed: F
Neighbours: {(B, 10 + 18, not found), (D, 9 + 14, found, 15), (H, 9 + 26, found, 15), (L, 2 + 16, not found)}

Iteration 5
Frontier: {(A, 25), (S, 18), (M, 16), (B, 28), (L, 18)}
Explored: {(C, -), (D, C), (F, D), (H, C)}
Node removed: M
Neighbours: {(H, 9 + 14, found, 15), (O, 6 + 26, not found), (S, 12 + 26, found, 18)}
```

```
Iteration 6
Frontier: {(A, 25), (S, 18), (B, 28), (L, 18), (O, 32)}
Explored: {(C, -), (D, C), (F, D), (H, C), (M, H)}
Node removed: S
Neighbours: {(C, 13 + 12, found, 13), (H, 9 + 22, found, 15), (J, 11 + 12, not found), (M, 6 + 22, found, 16)}

Iteration 7
Frontier: {(A, 25), (B, 28), (L, 18), (O, 32), (J, 23)}
Explored: {(C, -), (D, C), (F, D), (H, C), (M, H), (S, C)}
Node removed: L
Neighbours: {(F, 6 + 22, found, 16), (G, 0 + 20, not found), (N, 4 + 22, not found)}

Iteration 8
Frontier: {(A, 25), (B, 28), (O, 32), (J, 23), (G, 20), (N, 26)}
Explored: {(C, -), (D, C), (F, D), (H, C), (L, F), (M, H), (S, C)}
Node removed: G

Goal found!
Path : {C, D, F, L, G}
```

Final path cost = 6 + 4 + 6 + 4 = 20;
Number of Explored nodes = 8

For start node as ‘S’:

BFS:

```
Path : {S, H, F, L, G}
```

Final path cost = 16 + 16 + 6 + 4 = 42;
Number of Explored nodes = 17

DFS:

```
Path : {S, M, O, R, G}
```

Final path cost = 16 + 16 + 8 + 10 = 50;
Number of Explored nodes = 12

GBFS:

```
Path : {S, M, O, R, G}
```

Final path cost = 16 + 16 + 8 + 10 = 50;
Number of Explored nodes = 5

A*:

```
Path : {S, C, D, F, L, G}
```

Final path cost = 6 + 6 + 4 + 6 + 4 = 26;
Number of Explored nodes = 10