In this project we implemented the following:

1. <u>Depth FIrst Search</u>

Implementation:
Using stack as data structure and a list of visited nodes.

State pushed onto stack:
Current Position and action list to reach that Position(list of steps North,East,West,South)

Observations:

Maze Layout	Agent type	Path Cost	Nodes expanded
Tiny Maze	Search Agent	10	15
Medium Maze	Search Agent	130	146
Big Maze	Search Agent	210	390
Open Maze	Search Agent	298	806

- The order of the exploration was the way i expected as the search goes down along the depth before checking other paths on the same depth.

- Pacman on its way to goal, does not go to all the explored squares. Rather a path without any dead ends was followed.

- It was clearly observed in Medium and Open Maze that instead of going through path with lesser cost, Depth FIrst Search encountered a much longer path. Tiny Maze also follows the same, but has lesser effect than Medium and Open Maze.

  Thus DFS is not optimal.

2. <u>Breadth First Search</u>

Implementation:
Using queue as data structure and a list of visited nodes.

State pushed onto queue:
Current Position and action list to reach that Position(list of steps North,East,West,South)

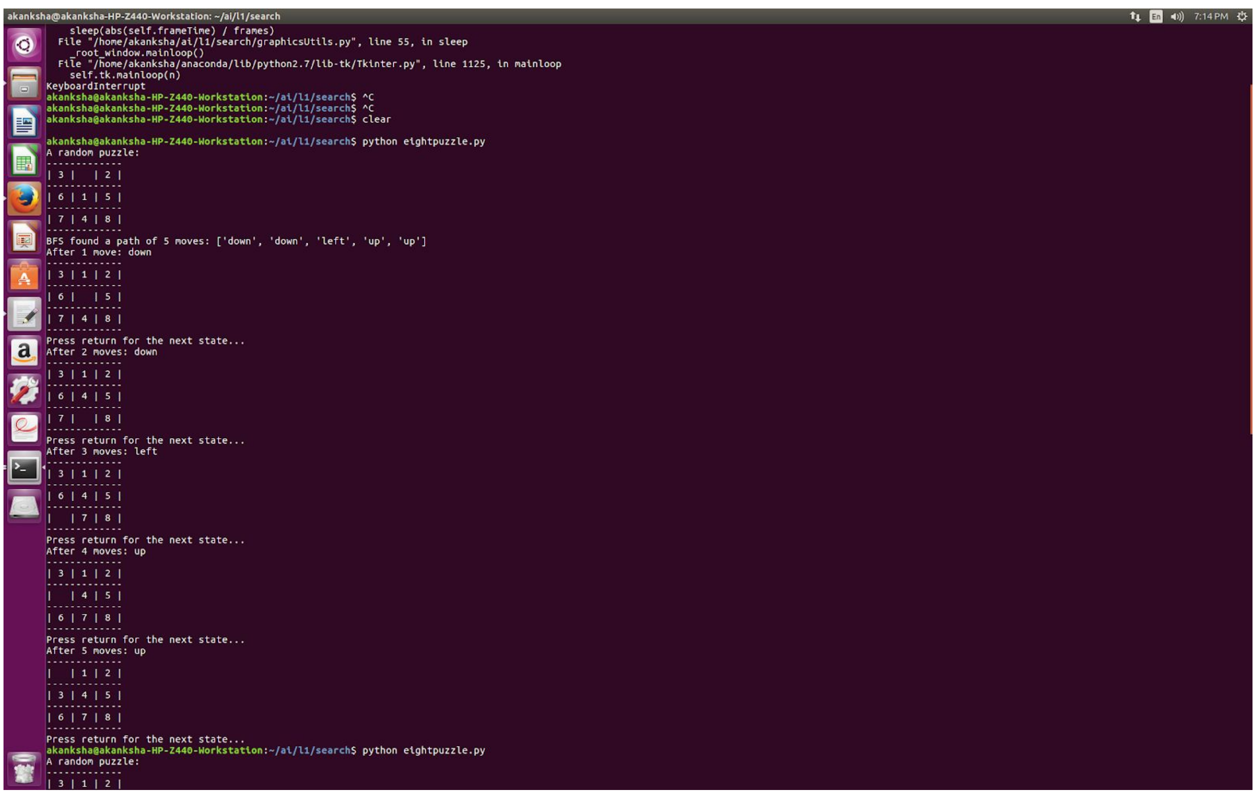
Observations:

Maze Layout	Agent type	Path Cost	Nodes expanded
Tiny Maze	Search Agent	8	15
Medium Maze	Search Agent	68	269
Big Maze	Search Agent	210	620
Open Maze	Search Agent	54	682

- Breadth First Search found path costs for Tiny, Medium and Open Maze less than Depth First Search.

- For Breadth FIrst search, the solution found is **optimal** as it checks all the nodes present at a particular depth and then increasingly goes to next depth. So, all the mazes have optimal path costs in above table.

- Below shows the execution of Eight Puzzle problem:



3. <u>Varying Cost Function</u>

Implementation:
Using priority queue as data structure and a list of visited nodes.

State pushed onto queue:

- Current Position and action list to reach that Position(list of steps North, East, West, South)
- Total cost of actions to reach current position

Observations:

Maze Layout	Agent type	Path Cost	Nodes expanded
Tiny Maze	Search Agent	8	15
Medium Maze	Search Agent	68	269
Big Maze	Search Agent	210	620
Open Maze	Search Agent	54	682
Medium Dotted Maze	Stay East Search Agent	1	186
Medium Scary Maze	Stay West Search Agent	68719479864	108

- We could clearly see that Tiny, Medium, Big and Open Maze reach to the goal state with same path cost as BFS. This is because to reach every adjacent position, the cost is 1. Thus, these uniform costs make **Uniform Cost Search reduced to BFS**. Hence, observing same path costs and expansion of nodes.

- Also, path cost obtained for **UCS is optimal**.

- Very less cost optimal path is observed for Medium Dotted Maze with Stay East Search Agent. While very large cost optimal path is observed for Medium Scary Maze with Stay West Search Agent.
  This is due to their exponential cost functions.

4. <u>Astar Search</u>

Implementation:
Using priority queue as data structure and a list of visited nodes.

State pushed onto queue:

- Current Position and action list to reach that Position(list of steps North, East, West, South)
- Total cost=Heuristic value estimated to reach goal state plus path cost to reach current position

Observations:

a) If no heuristic is passed in command line argument:

Maze Layout	Agent type	Path Cost	Nodes expanded
Tiny Maze	Search Agent	8	15
Medium Maze	Search Agent	68	269
Big Maze	Search Agent	210	620
Open Maze	Search Agent	54	682

- If no heuristic is passed to Astar search in command line, it takes the null heuristic and performs in a similar manner as UCS.
  This is because the total cost estimated becomes equal to actual path cost to reach current position.

b) With Manhattan Heuristic:

Maze Layout	Agent type	Path Cost	Nodes expanded
Tiny Maze	Search Agent	8	14
Medium Maze	Search Agent	68	221
Big Maze	Search Agent	210	549
Open Maze	Search Agent	54	535

- It could be clearly seen that with a heuristic, the search is able to expand less number of nodes, saving time and space.
  Thus, performs better than both Uniform Cost Search and Breadth First Search.

Comparison of strategies on Open Maze:

- With Open Maze all BFS, UCS,Astar finds the optimal path to goal of cost 54. While they differ in the number of nodes expanded and Astar expands 535 nodes, least among the three.
- While, DFS also reaches the goal state with a path cost of 298 expanding about 806 nodes.

  Thus, it is quite clear from here, that **DFS does not find an optimal solution**. While **BFS,UCS, Astar get the optimal solution with number of nodes expanded least in Astar**.

5. <u>Corners Problem</u>

Implementation:

- **Defining state for the problem:**
  State consists of current position and a list of corners visited

- **Defining a new start state:**
  Start state is starting position of the problem and an empty list as initially, no corners are visited.

- **Goal state of the problem:**
  When the visited list contains all the four corners we reach the goal state.

- **Getting successors for node:**
  Get the next position. For next position, check for walls. If no walls present and its a corner that is not visited, add it to visited list, update with next state with new list. Else, the visited list remains same and only the next position is updated.

Observations:

Maze Layout	Search	Path Cost	Nodes expanded
Tiny Corners	BFS	28	435
Medium Corners	BFS	106	2448
Tiny Corners	Astar (Null Heuristic)	28	435
Medium Corners	Astar (Null Heuristic)	106	2448

- BFS and Astar(without any heuristics) perform the same in path cost and number of nodes expanded.
  The number of nodes expanded although could be reduced by implementing a heuristic that is admissible and consistent for the corner problem.

6. <u>Corner Problem Heuristic</u>

Implementation:

- **Defining a heuristic:**
  In my implementation, **"***heuristic is the manhattan distance from current position to the nearest corner plus the minimum path to cover all the corners from this nearest corner***"**.

  Admissibility of heuristic:
  The heuristic chosen is admissible as the manhattan distance gives the distance without considering walls in the problem. Thus, the heuristic distance would definitely be a lower bound to the actual distance.

  Consistency of heuristic:
  It is also consistent as the manhattan distance remains same between the four corners.

Observations:

Maze Layout	Search	Path Cost	Nodes expanded
Tiny Corners	Astar (Corner Heuristic)	28	217
Medium Corners	Astar (Corner Heuristic)	106	901

- We observe that the optimal path for Astar with corner heuristic remains the same as Astar without heuristic.
  Though number of nodes expanded with heuristic, decreases almost to half the number of nodes expanded without heuristic. This, also reduces the amount of time taken to execute.

  Thus, defining admissible and consistent corner heuristic we are able to increase the performance of Pacman to reach the goal state.

7. <u>Eating all the dots</u>

Implementation:

- Different admissible and consistent heuristics considered:

    1. Closest food from current position:
        - Manhattan Distance heuristic
        - Maze Distance heuristic

    2. Farthest food from the current position:
        - Manhattan Distance
        - Maze Distance

Observations:

- All of these observations are taken with Astar search with different food heuristic defined

Maze Layout	Heuristic	Path Cost	Nodes expanded
Tricky Search	No heuristic	60	16688
Tricky Search	Closest food(Manhattan distance)	60	13898
Tricky Search	Closest food(Maze distance)	60	12372
Tricky Search	Farthest food(Manhattan distance)	60	9551
Tricky Search	Farthest food(Maze distance)	60	4137

★ All the heuristics give optimal path costs but differ in the number of nodes expanded.

★ Without heuristics Astar performs the worst, expanding close to 16,700 nodes.

★ As the farthest food from current position with maze distance heuristic decreases the number of nodes expanded to minimum, we consider this heuristic for the next set of observations.

★ Though the number of nodes expanded by farthest food with maze distance heuristic is minimum but the time taken is maximum out of the other heuristics close to 41.9 seconds.

- Observations with different mazes considering Farthest food heuristic with maze distance

Maze Layout	Search	Path Cost	Nodes expanded
Test Search	Uniform Cost Search	7	14
Test Search	Astar (with heuristic)	7	10

Maze Layout	Search	Path Cost	Nodes expanded
Tiny Search	Uniform Cost Search	27	5057
Tiny Search	Astar (with heuristic)	27	2372

Maze Layout	Search	Path Cost	Nodes expanded
Tricky Search	Uniform Cost Search	60	16688
Tricky Search	Astar (with heuristic)	60	4137

  ★ In all the cases above, Astar outperforms than UCS by expanding less number of nodes with same path costs.

8. <u>Suboptimal Search</u>

Implementation:

- **Finding path to Closest Food:**
  Calling either UCS or BFS to find the path to closes food from current game state
- **Defining goal state for any food search problem:**
  If the current position has food, it is a goal state

Observations:

Maze Layout	Search	Path Cost
Test Search	BFS/UCS	7

Tiny Search	BFS/UCS	31
Tricky Search	BFS/UCS	68
Big Search	BFS/UCS	350

- Using BFS or UCS gave the same results as costs are uniform over here.

- The Closest Dot Search Agent may or may not find the shortest path through the maze. This is because at a time there might be two dots present at same distance, so any one dot will be chosen, leaving the other dot to be chosen after the other dots.

  This could be clearly seen in Big Search problem. Thus, the path cost returned here is not the optimal path cost.