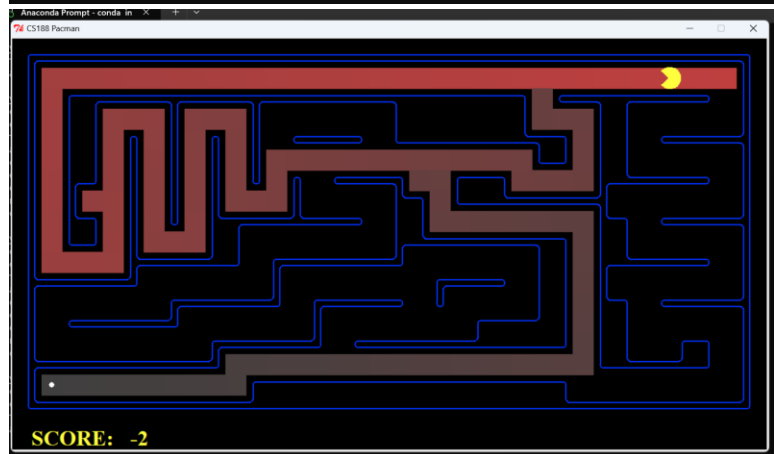


## 1. Depth-First Search (DFS)

After implementing the depth-first search, and running the command in the terminal, this is the solution it found:

- The exploration order is not what I had expected.
- Pacman doesn't go through all the squares.

```
found with total cost of 7 in 0.1 seconds
nodes expanded: 7
emerges victorious! Score: 500
C:\Users\user\Downloads\search>python pacman.py -l trickySearch -p AStarFoodSearchAgent
found with total cost of 12 in 0.2 seconds
C:\Users\user\Downloads\search>python pacman.py -l tinyMaze -p SearchAgent
SearchAgent
onSearchProblem
found with total cost of 10 in 0.0 seconds
nodes expanded: 15
emerges victorious! Score: 500
C:\Users\user\Downloads\search>python pacman.py -l tinyMaze -p SearchAgent
SearchAgent
using function depthFirstSearch
SearchAgent
using problem type PositionSearchProblem
found with total cost of 10 in 0.0 seconds
nodes expanded: 15
```



```
prompt - conda in X + v
t] using function depthFirstSearch
t] using problem type PositionSearchProblem
with total cost of 10 in 0.0 seconds
s expanded: 15
ges victorious! Score: 500
re: 500.0
500
1/1 (1.00)
Win

sers\user\Downloads\search\search>python pac
t] using function depthFirstSearch
t] using problem type PositionSearchProblem
with total cost of 130 in 0.0 seconds
s expanded: 146
ges victorious! Score: 380
re: 380.0
380
1/1 (1.00)
Win

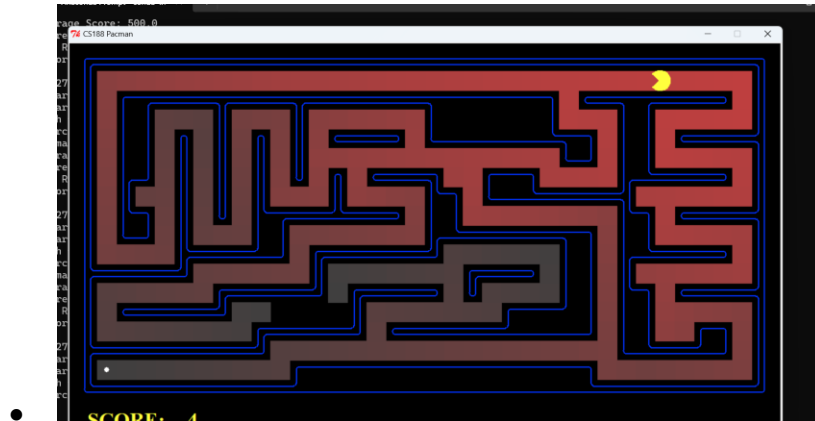
sers\user\Downloads\search\search>python pac
t] using function depthFirstSearch
t] using problem type PositionSearchProblem
with total cost of 210 in 0.0 seconds
s expanded: 390

sers\user\Downloads\search\search>python pac
t] using function depthFirstSearch
t] using problem type PositionSearchProblem
with total cost of 210 in 0.0 seconds
s expanded: 390
```

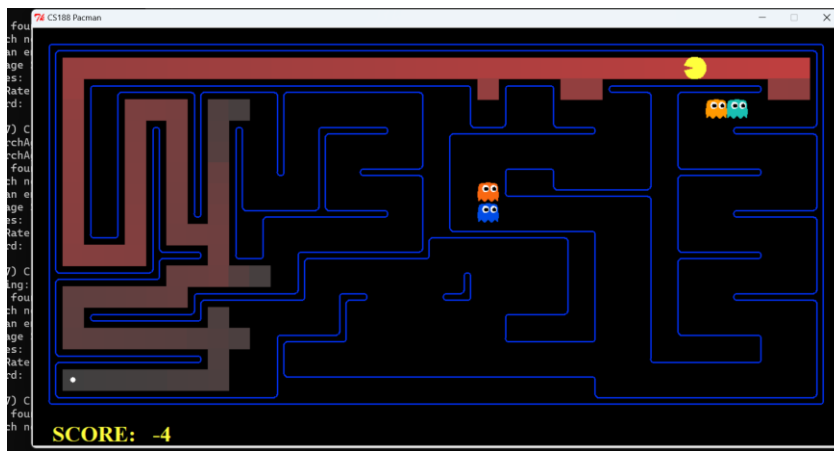
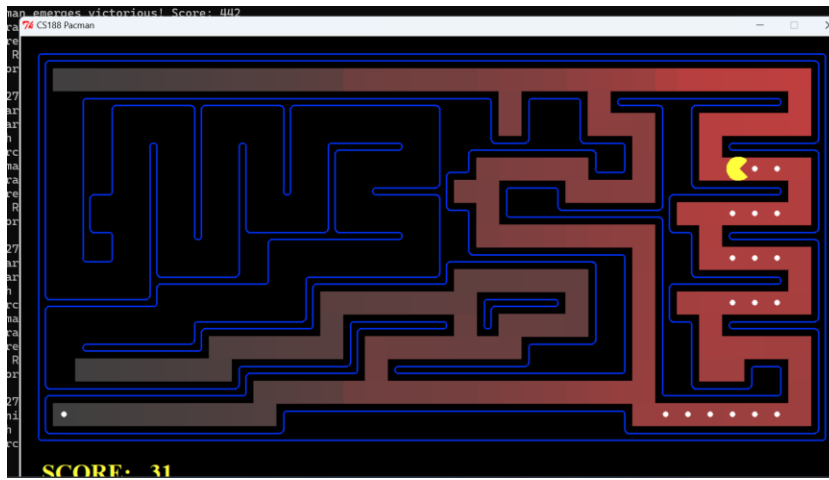
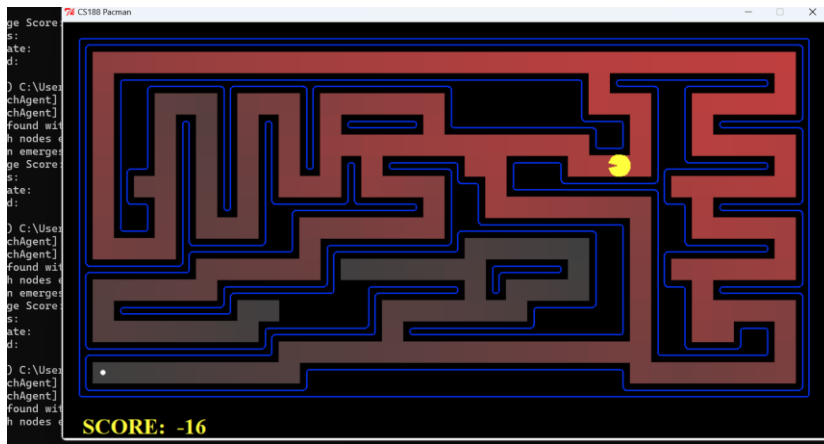


## 2. Breadth-First Search(BFS)

- It indeed does get a least cost solution.



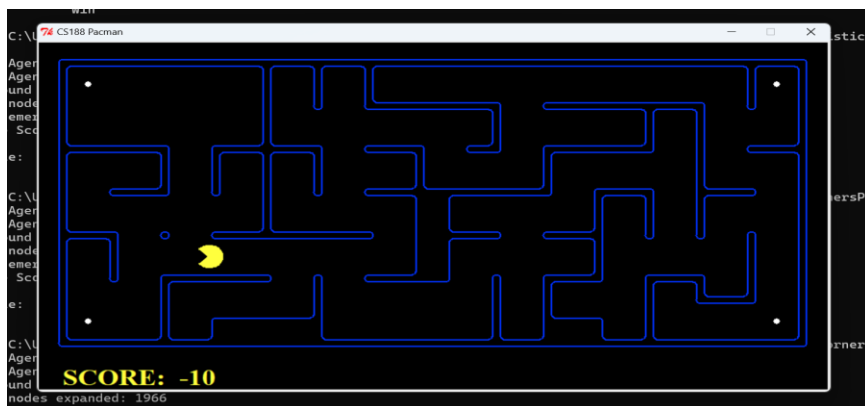
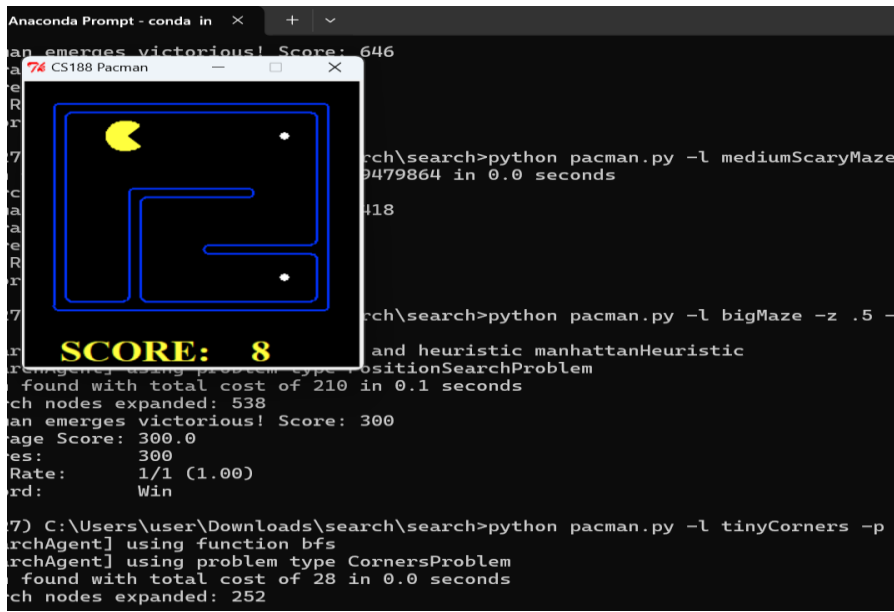
## 3. Implement the uniform-cost graph search algorithm



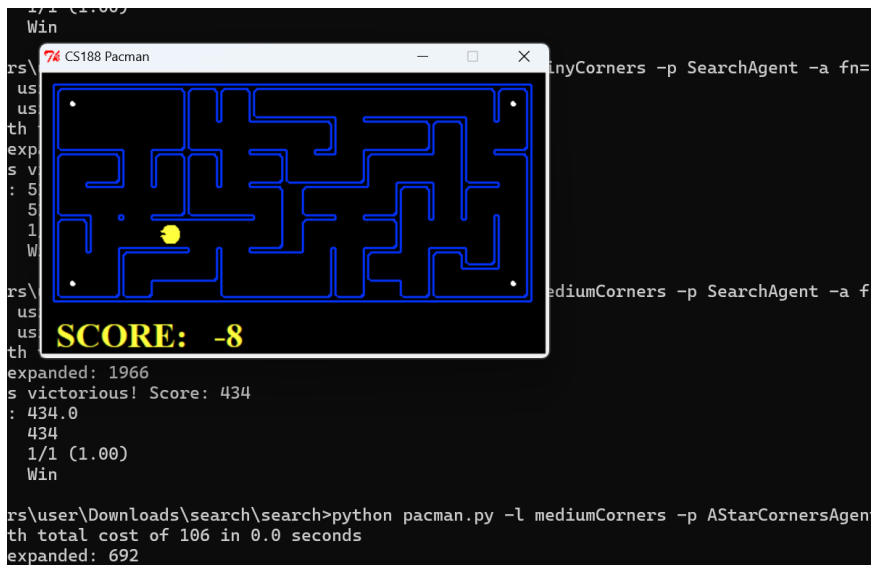
#### 4. A\* Search



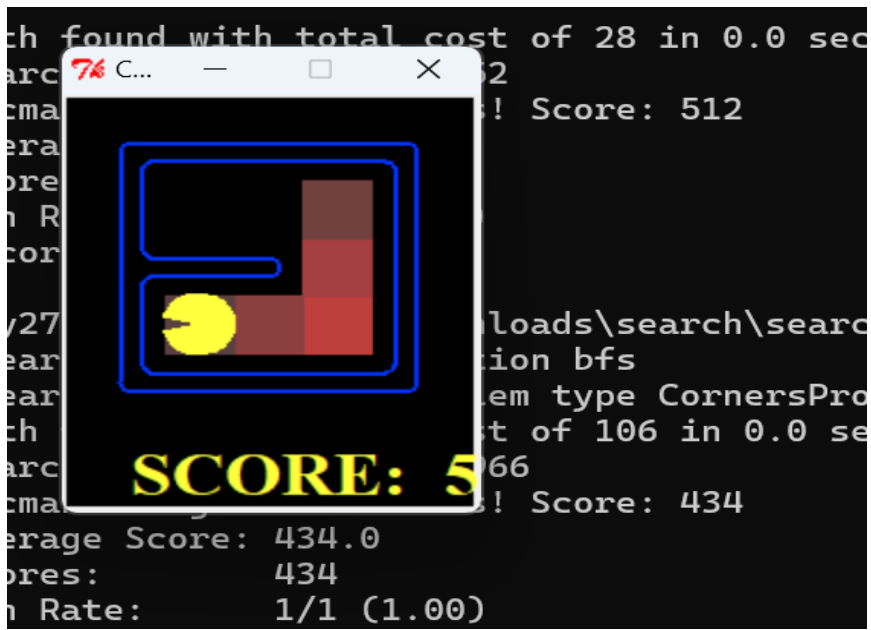
## 5. Finding All the Corners



## 6. Implementing a non-trivial



## 7. Eating all the dots



## 8. Filling in foodHeuristic

