i have a lot of comments in my implementation so i will describe sortly what i do and some basic parts of the code:

QUESTION 1 TO 4 :ARE SIMILAR SEARHING ALGORITMHS

USING A FRONTIER AND POPPING POINTS BASED ON A DIFFERENT WAY OF QUENING

Q1:line 79 -81:getting the initial state of the problem and checking if its the goal state

line 85-87:because we are going to perform a dfs we will need a LIFO frontier,we will use a stack we will push the fisrt node(initial_state,initial_path) in it ,we aslo initialize a set which will keep the already explored states

line 93-105: #while the frontier has nodes keep popping nodes and check if the state is expolred

and also if the state is the goal state

lastly,for each state (current_state) i take its succesors push new nodes to the frontier

#as the formed ((next_state,path_for_the_next_state))

where path_for_the_next_state=path+a Direction

Q2:exactly the same approach with question 1 only the frontier is different(LIFO)

Q3:we prioratize the node with the least total cost

the quening is based on this principle .

for each succesor we calculate the totalcost and based on that and on the fact that he may be unvisited we update the frontier and the explored list

Q4:a star is a compination of USC and BEST fisrt

the node with the highest priority is the one with the least f(n)

where f(n)=h(n)+g(n)

where h(n) the cost of the most cheap path from state n to a goal-states

and g(n)=the minimum cost to state n

Q6 and Q7 is the implementation of heuristics

Q6: first we find the unvisited corners!

finding each time the nearest unvisited corner using mangattanDistance

and adds it to the heuristic the remove the corner from the unvisited and

and update the position

do that until there is no more unvisitedCorners

Q7: takes time to run but expands below 7000

we use mazeDistance in order to define the heuristic

for each dot of food left, calculate the distance between pacman and dot of food

finds to the distance to the farthest dot

Q8:The problem indicates that even a good heuristic

would fail to find the optimal path in a short time.

AnyFoodSearchProblem can also be done by the

already defined gameState.getFood() function, and

the function of findPathToClosestDot can be simply

constructed by search algorithms (e.g. BFS, UCS,

A*) as we have implemented . I use BFS!