INFO8006: Project 1 - Report
Search Agent Based on Algorithms(BFS,DFS and A*)

SAFFO NGUOANDJO Borel- s204863
DANDJI AYAWO DESIRE - s197206

October 11, 2020

1 Problem statement

a. **The set of possible states :** {(a,b), all booleans position PACMAN can take }.
the initial state is the instance named s of gamestate passed as argument in the get-action() function; it is also called starting point, the first state that pacman take in the game.

b. **Set of Legal action:** action(A) = {West, East, North, South }. The pacman agent can:
.Either go West (or Right)
.Either go East (or Left)
.Go North (or UP)
.Go South (or Down)

c. **Transition model**:is the list of pairs of successor states and action returned by generatePacmanState() if current state is given.

d. **The goal test** : is the state of the dot (goal state), in this case it returns true if all the food or dots in the maze have been eaten and false it if not. we can verify this by applying .isWin() method on the current state.

e. **The step cost** : is the cost of moving from state s to state s′ depending on the action taken It's defined by the the cost function c(s,a,s′) where s is the current state , a is the action taken and s′ the state we want to reach.

2 Implementation

a. Error of DFS:

In the DFS, we have a function which defines the position of the pacman ″ GetPosition″ but we de not have the function ″getFood ″ which allows it to take food. To fix this, we just added the getFood function. The algorithm does not specify if a node is already visited or not. Added to the boolean grid containing the food at the key also corrected this error.

b. *implemetation .*

c. Description of our cost function
the cost function(g(n)) is the function use to calculate a cost of path to n from the start node, The heuristic function(h(n)) allows A* to estimate the minimum cost from any vertex n to the goal and

also used to control A* behavior. In our case the h(n) guaranties the optimality of A* because to move from any vertex to the goal, h(n) is always inferior 0, it can't be equal to 0 or superior 0 in our case from where h(n) guaranties the opimality of A*.

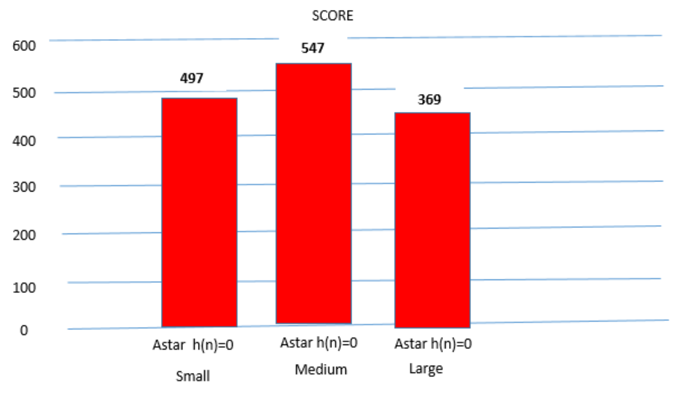
d. In case of h(n) =0, then only g(n) play the rôle of f(n) ; (f(n) = g(n)), and then A* becomes Dijkstra's Algorithm and guaranteed to find a shortest path.

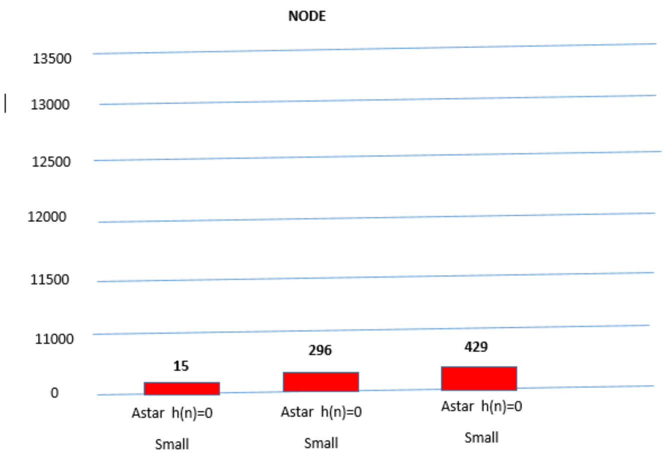
e. *implemetation .*

f. Step cost and heuristic functions in BFS :
we don't need g(n) and h(n) to implement BFS because BFS is based on Queue which is FIFO data structure.

3 Experiment 1

a. Report the results as a bar plots in term of Score





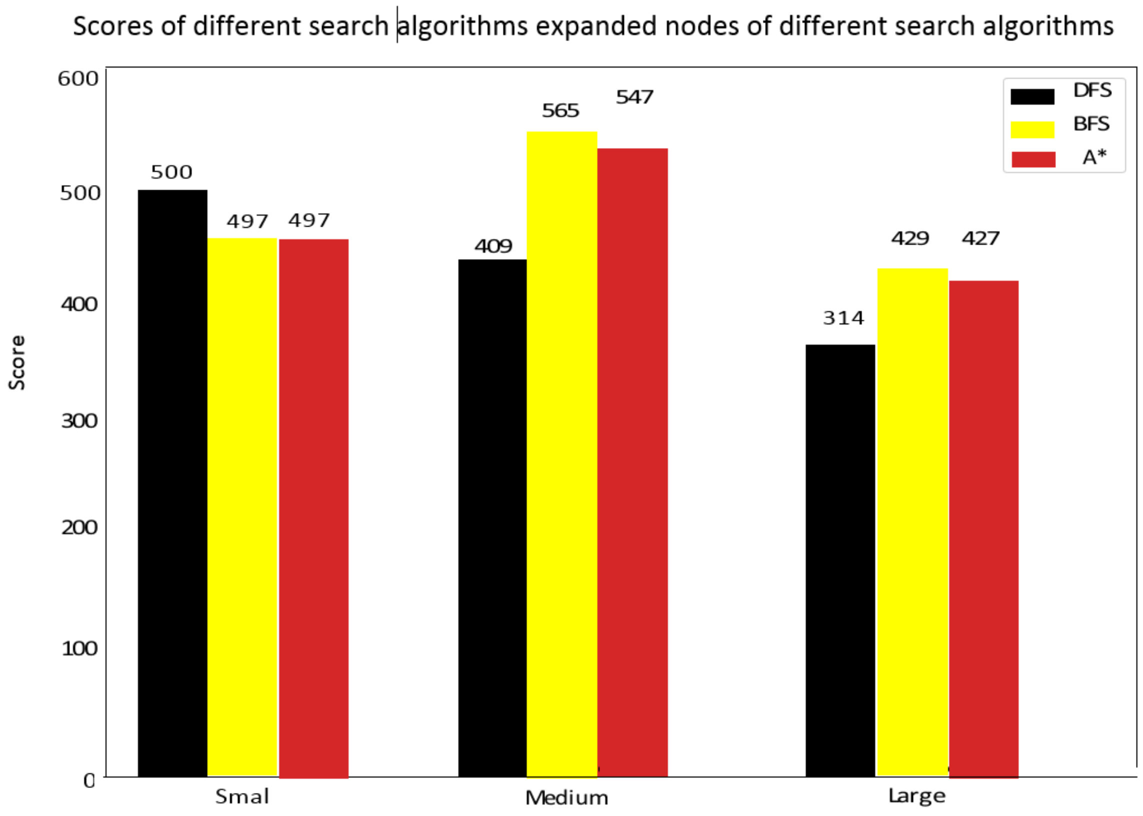
b. Report the results as a bar plots in term o Number of expanded nodes:

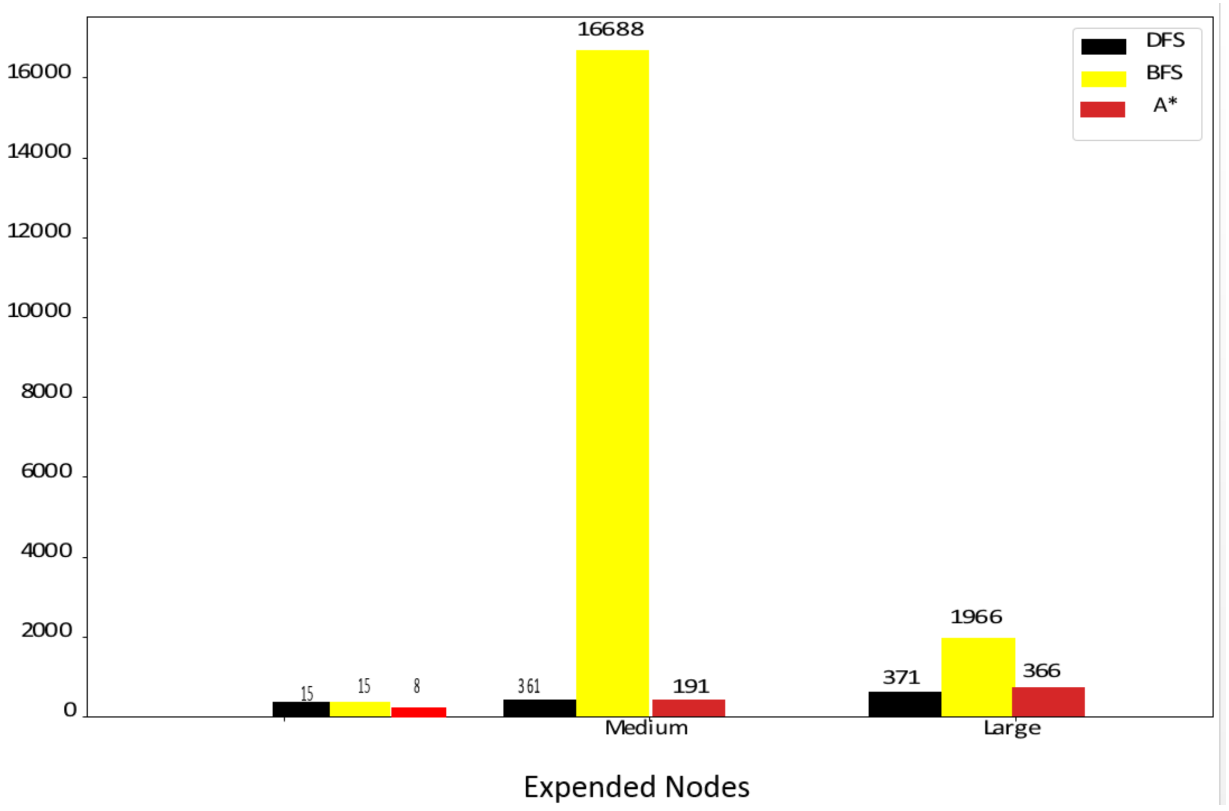
c. The difference is only that, the A* with g(n) and h(n) tries to check a better path for using the heuristic function while the A* with g(n) and h(n) = 0 just explore all possibl path. In conclusion the first one is better than the second .

d. if h(n) = 0 ; A* corresponds to Uniform cost Search Algorithm (who is a Dijkstra's variant) .

4 Experiment 2

a. Bar plots :





b. Differences between A*, BFS and DFS

- A* and DFS :We will compare these algorithms at the level of the medium layout

The astar algorithm has a lower score number than the DFS score number
The astar algorithm has a lower nodes number than the DFS nodes number

- A* and BFS :

astar has a lower number of score and nodes than the BFS algorithm.

c. Justification of the differences :

A* allows optimality and completeness, two valuable property of search algorithms but bfs and dfs explaned the possible node in graph