**Walaa Jamous**

INM702 Programming and Mathematics for Artificial Intelligence (PRD1 A 2020/21)

Task 1

Task 1

1. The algorithm find the shortest bath by the following technique:

A path is created in the start square where it starts to seed to all the adjucent sequare and those paths are saved in a list this list is referred to as Seeding path list “*seedingpath*”.

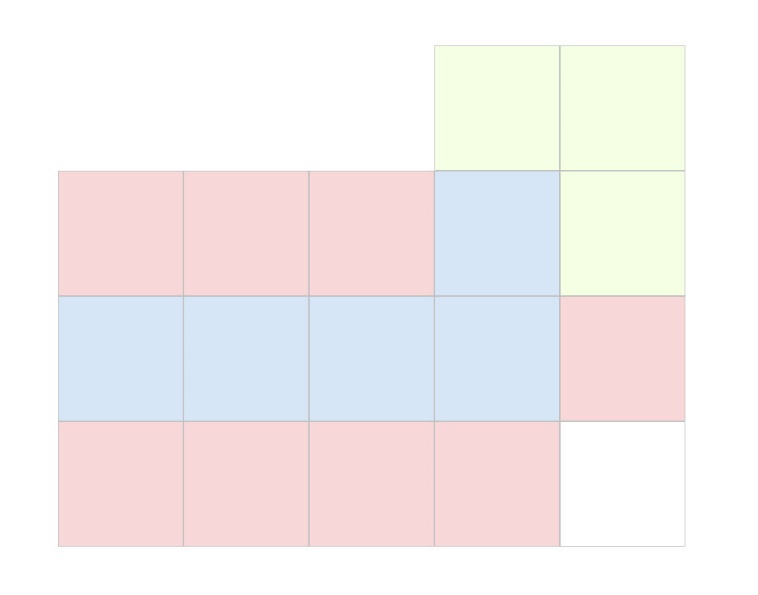
Note : the terms” path is seeding” will be used to refer to a status of finding the adjucent square and seeded paths will be created if the square was not visited before

Each path gets a timing value where it is saved in variable inside the path object

When the main method is called the method start seeding by entering through a loop where it starts “ticking” each tick takes down one from the timing value in the *seedingpath* those paths that reach the value zero are allowed to move on and create their own seeding path and then added to the seedingpath list. It is also crucial to know that once a path reaches a new square, it saves the time estimated to reach this square in a parallel matrix with same size to the original grid.

In case one of the path ends up in closed loop (which is all considered prior to any path seeding) or reached a square that has been visited the path terminate and stored in a different list called terminated path. Also if the path has no other way then an already visited cell by any path it terminates.

A path is not allowed to seed to pass through a cell twice neither it is allowed to pass to any adjucent cell to the cells it passes through it. This is to avoid a path from wasting resources on a path that it could have gone to in a shorter route the illustration below domenstrate this point



If we assume that the path moves along the blue square, then it is not allowed to go to any adjacent location as this would waste of time

After all. I will give a general idea in how this works: A path that has large time is halted while the other who has less time cost moves faster therefore we can ensure that the shortest path to reach the end square is the first path to reach it without the need of checking if there is another shorter path. Thus makes this algorithm fast and always finds the best solution the disadvantage is that the method requires way more resources when the grid size grows as many paths will be seeding.

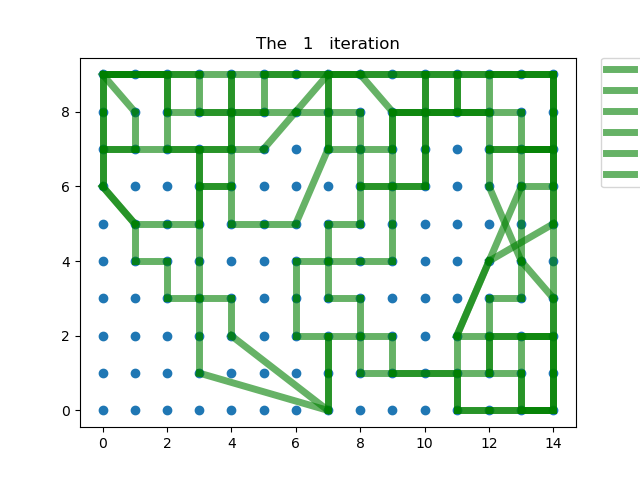
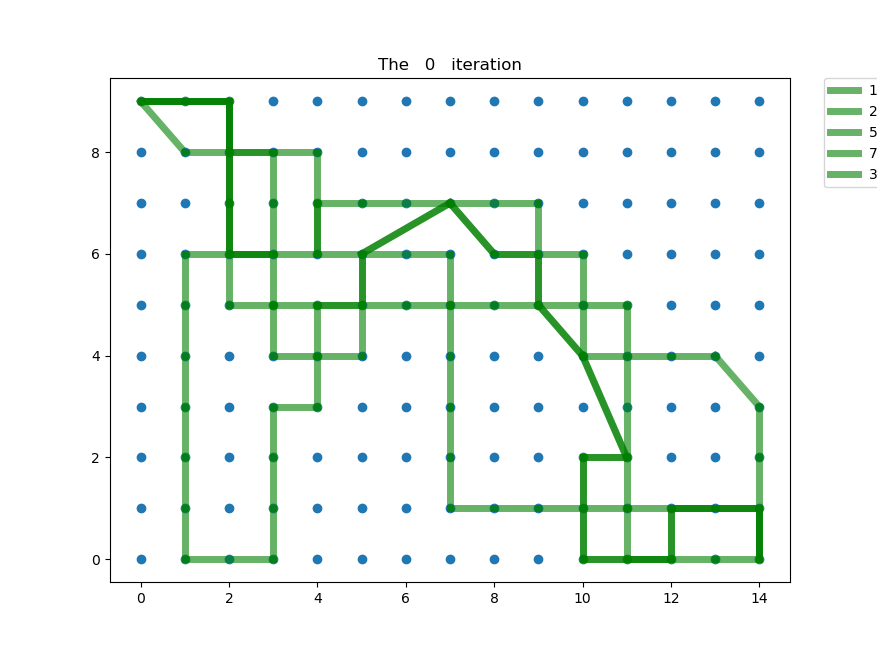
1. Dijkstra algorithm.

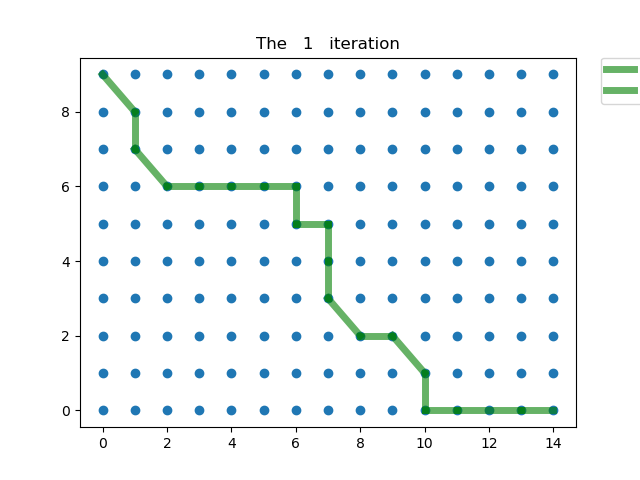
This algorithm find the best path by using three parallel list in which the list record the path and the order of each node the algorithm is set to find always the optimal path but yet would consumes more resources then the ant colony and also the to an extent to the algorithm that I have designed as this algorithm might go to any direction and that is where A+ algorithm set to fix

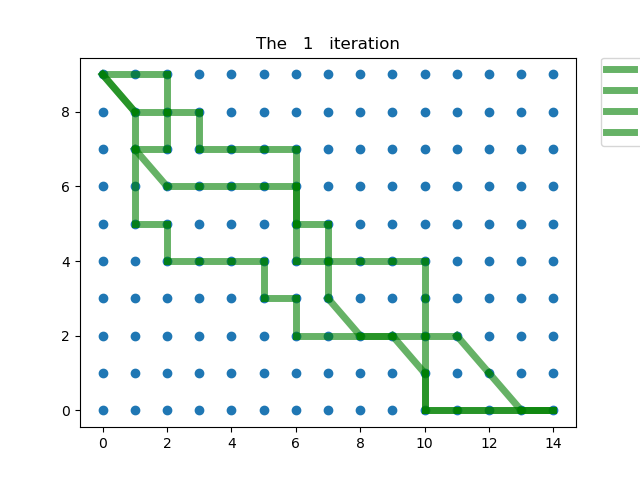
1. Ant Colony algorithm:

By far the ant colony optimization is the least rescourse consuming method between the three above though it is worth to know the for each grid size the same parameter will guarantee a solution as the ant might lose their path and never reach the end point. There is some sort of intuition on how many ants must be in a colony in grid size or the effect of pharmon on the probability and lastly the evaborating. But yet a changing the parameter might result on finding faster solution.

Note: in my implementation I tend to kill those ant who lose their way in a big grid or I added a condtion that does not the ant goes in a endless or extremely long path and I named the feature the ant life span



At first ant tend to follow random paths After few iteration longer path fades as evaboration become obvious



At after many iteration few paths are above is the best path which ants has lifts found

* The diagnle line is to indicate that the cell has zero value and that is why ant cross it as it cost nothing to cross.
* As a summary it is clear that the ant colony has advantage over both algorithm once the size of the grid grows exponentially and that sets apart. Where both other finds the optimal solution they lack the efficiency once the grid size grows. And with time it manages to find an almost optimal path as the evaporation and works in favor of the probability of choosing the shorter path