Plot and Navigate a Virtual Maze_pdf

December 14, 2016

1 Machine Learning Capstone Project

Machine Learning Nanodegree (Udacity) Project submission by Joaquin Bejar Garcia (joaquin.bejar@gmail.com).

1.1 Definition

1.1.1 Project Overview

Understanding mazes has been contemplated for quite a long time and years in arithmetic, which implies there are an excessive number of calculations to name here. The least difficult maze comprehending calculation is the irregular calculation, which just goes ahead in the maze until it goes to a wall. It then haphazardly picks right or left and afterward goes ahead once more. With this calculation, the mouse will dependably discover the goal (if there is no time restrict), however it will definately utilize an exceptionally wasteful course to arrive. Obviously, there will be an ideal way and less ideal ways and in addition productive and wasteful methods for finding those ways. The virtual robot's score for a given maze is a figuring based upon what number of steps it utilizes first to investigate and afterward race through the maze.

The parameters and setting of this project are propelled by the Micromouse robot rivalry [1] and is, in actuality, a virtual adaptation of the Micromouse issue. The robot mouse crosses the maze twice. The principal traversal offers the chance to investigate and delineate maze while the second run requires the robot mouse endeavor to achieve the focal point of the maze as fast as it can utilizing the information procured while investigating the maze.

The basic role of this project is to program a virtual robot to investigate a basic maze, discover a course from a side of that maze to its middle, and cross the way through and through. The objective of this project is to characterize and actualize a technique to reliably find an ideal way through a progression of test mazes that exist inside a virtual world roused by the Micromouse issue.

1.1.2 Problem Statement

Every maze utilized as a part of the project takes after a strict particular. The maze is a completely encased square with an even numbered measurement along every side. The test mazes one, two and three are measured 12x12, 14x14, and 16x16 units separately. At the focal point of every maze is a 2x2 zone encased by seven walls and one passageway. This zone is the objective of the maze, and the robot must enter this space to finish a fruitful run. The robot will dependably begin in the base left corner of the maze where the cell will dependably have three walls with a solitary opening at the top.

Each 1x1 cell inside the maze occupiable by the robot can have one of 16 conceivable shapes characterized by the nearness or nonappearance of a wall on every side. There is a seventeenth conceivable shape where every one of the four walls are available, yet this cell would not be occupiable by the robot and does not show up in any of the test mazes. It is significant that, as said over, the begin cell is dependably a similar shape and edge cells and also the middle objective cells have less than 16 potential shapes given the limitations that the maze is completely encased and the objective territory has a solitary passageway.

On account of the virtual environment, a content record characterizes every maze. The esteem on the primary line of the record expresses the measurement of the maze took after by a progression of lines of comma isolated qualities. Because of cluster ordering, the primary line is the left half of the maze and the main esteem in the principal line speaks to the base left corner. Every esteem depicts a cell inside the maze. These qualities are four-piece whole numbers from 1 to 15 (0 speaks to the distant, completely encased space). Every piece in the four-piece number speaks to a wall or opening, 0 or 1 separately, in favor of the cell. The bit grouping begins with the 1s at the highest point of the cell with each extra piece (2s, 4s, and 8s) characterizing the edge clockwise around the cell. For instance, the computation of the esteem for the beginning cell, that has a wall on each side with the exception of the top, is as per the following:

```
1*1 + 0*2 + 0*4 + 0*8 = 1
```

The robot is thought to be in the focal point of the cell it involves and can just face one of the four cardinal headings. The robot is equipped for taking flawless sensor readings and making impeccable developments. This implies the robot is a totally deterministic operator, yet its sensors can just distinguish the separation to the following wall in each of the three bearings, forward, left, and right. The robot is just equipped for advancing and in reverse yet is fit for climbing to three spaces in either course. Toward the start of every time-step, the robot gets its sensor readings in view of the bearing it is confronting after the last revolution. It can then kept its present bearing or turn either left or appropriate by 90 degrees before advancing or in reverse. On the off chance that the robot hits a wall before finishing its development, the robot will remain where it is confronting the wall that hindered its way. The demonstration of moving finishes the time-step permitting the robot to get its new sensor readings beginning the following sense-move circle.

All the more particularly, the robot's `next_move` work gets the sensor readings as a rundown of 3 numbers speaking to the separations to one side, forward, and right nearest walls in a specific order. On the off chance that the wall is an edge of the right now involved space, the separation is 0. The `next_move` work should then return two qualities speaking to the robot's turn and development in a specific order. The pivot esteem can be one of - 90, 0, or 90 speaking to counterclockwise, no turn or, clockwise revolution individually. The development esteem must be a number between negative three and three including those qualities. A negative number is a regressive development, and positive is forward.

In its first keep running of the maze, the robot may move unreservedly inside the maze to investigate and outline. In the event that the robot closes the investigation run, it can return "Reset" for both its pivot and development values. This activity resets the robot's position to the base left corner or the maze and starts the second run. The second run closes when the robot achieves the objective at the focal point of the maze.

The robot should investigate the maze adequately and keep up a guide of the investigated cells to take care of the issue laid out above. While investigating, the robot should find no less than one way to the objective and visit it before endeavoring the second run. On the off chance that the robot investigates the entire maze, then it ought to have the capacity to decide the ideal way to the objective and utilize that way in the second run. The perfect robot procedure for this issue effectively and reliably finds the ideal way to the objective while minimizing the means required

to investigate the maze. The Algorithms and Techniques segment of this report incorporates a dialog of a few approaches to take care of these route and mapping issues.

An answer for this issue will require, in any event, the accompanying center procedures:

1. Take the sensor readings, join this data with the robot's area and heading and overhaul the robot's information of the maze.
2. With the redesigned information of the maze, figure out where the robot ought to go to take in more about the maze.
3. Move to the wanted area and overhaul the robot's heading and area.
4. Keep investigating the maze until no less than one way to the objective is found, yet ideally locate the ideal way to the objective.
5. Begin the following keep running of the maze.
6. Take after the picked way from the begin to the objective.

This is a gross rearrangements of the rationale required to take care of this issue however fills in as a beginning stage. Later areas of this report will expound on this procedure investigate every part of the usage in much more prominent detail.

1.1.3 Metrics

For each of the test mazes, the virtual robot travels through the maze twice. In the main run, the virtual robot can move unreservedly through the maze trying to investigate and outline. It is allowed to keep investigating the maze even subsequent to entering the objective zone. Once the virtual robot has found the objective, it might end the investigation keep running whenever. For the second run, the virtual robot is relied upon to cross the maze and achieve the objective as fast as possible. The score granted to the robot for every maze is the aggregate of the two after terms:

- The quantity of steps taken while investigating the maze amid the main run partitioned by 30.
- The quantity of steps taken to achieve the objective amid the second keep running of the maze.

A virtual robot with a littler score performs superior to one with a bigger score.

Every keep running of the maze is restricted to 1000 stages.

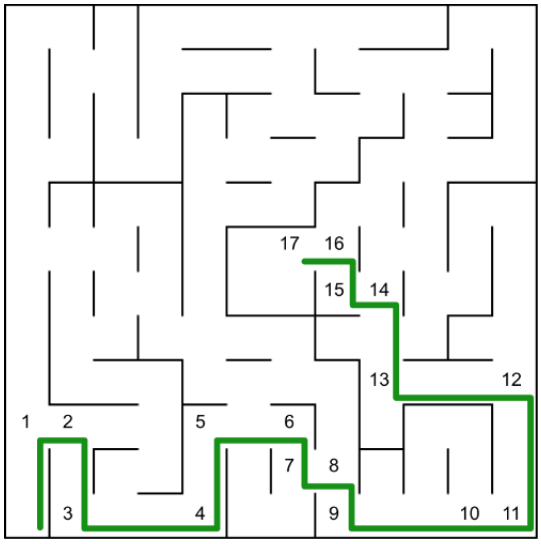
Note that this scoring metric punishes both robots that neglect to locate the ideal way to the objective and additionally those that investigate the maze in a wasteful way. All things considered, a robot that breaking points investigation and neglects to locate the ideal way to the objective is probably going to, yet may not really, perform more terrible than a robot that sets aside the opportunity to investigate the maze adequately and finds the ideal way to the objective.

1.2 Analysis

1.2.1 Data Exploration

We will investigate Test Maze 1 trying to comprehend the structure of the maze all the more totally. The following is the maze demonstrating the ideal way to the objective. It finds a way to take after this way.

There are a couple of basic components display in this maze that are significant. The crisscross walls that meet the two right corners of the objective zone imply that all ways to the objective



Test Maze 1 with a green line demonstrating the ideal way from the begin to the objective and the arrangement of projects along that path.

should first go to one of the correct corners of the maze before achieving the objective. Given that the upper right corner of the maze is the uttermost region from the begin position, it is not astonishing the ideal way goes around the base right corner. It is likewise fascinating to note that there are not very many level walls in the left third of the maze permitting the robot to investigate that part of the maze more effectively than it would something else. On the off chance that more corners were available, they would keep the robot from seeing down long 'halls'. This shows exactly the amount of the maze the Robot can investigate with only a couple steps. The main sensor perusing alone will give enough data to demonstrate that each and every edge inside the furthest left cells of the maze are openings.

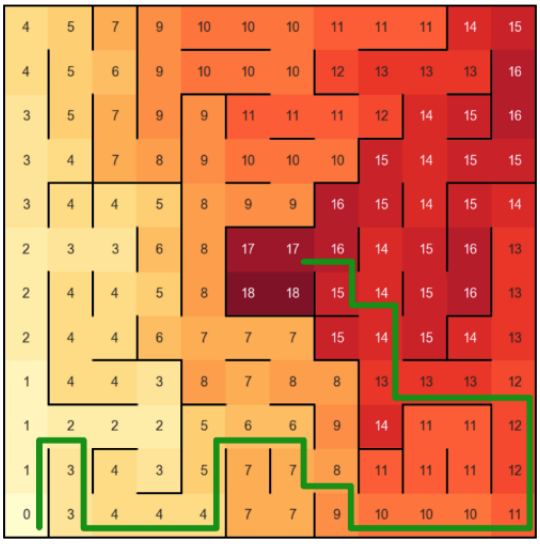
1.2.2 Exploratory Visualization

The structure of Test Maze 1 examined in the past segment is much more clear when laid over a heatmap that portrays the way cost to achieve every cell. These qualities are the base number of steps required for the robot to achieve every cell through any way.

The heatmap makes it clear how the robot needs to visit one of the right-hand corners before navigating the furthest right triangular wedge of the maze that prompts to the objective. The heatmap likewise demonstrates what a small number of steps are required for the robot to navigate the left third of the maze. Indeed, even without the green line demonstrating the ideal way to the objective, it would be obvious that the robot's best system is to go to the base right corner as opposed to the upper right corner. All things considered, this is only a representation indicating how the ideal way thinks about to the options. As a general rule, the robot will compute the ideal way and utilize a productive procedure to investigate the maze exploiting any long halls inside it.

1.2.3 Algorithms and Techniques

The demonstration of first investigating the maze and afterward finding the ideal way are both pursuit issues. There are a couple of calculations that can be utilized to take care of hunt issues. These incorporate Dijkstra's calculation [2], A* [3], and in addition a large group of diagram [4]



Test Maze 1 with a heatmap demonstrating the way cost for every cell alongside the ideal way from the begin to the goal.

and tree [5] traversal calculations. The accompanying will clarify for which, assuming either, seek issue the algorithm(s) would be appropriate.

- **Dijkstra's algorithm** is utilized to locate the most limited way between hubs in a diagram [2]. The calculation takes a chart and after that iteratively projects through every conceivable way through diagram putting away the littlest way cost for every hub. The calculation chooses the hub with the littlest way cost and chooses the majority of the neighbors for that hub. In the event that the way cost through the present hub to achieve a given neighbor is not exactly the present way taken a toll for that neighbor, then the way cost is upgraded for the neighbor with the lower esteem. This procedure is rehashed until the speediest way to the objective hub is found. The ideal way can then be developed by strolling through the diagram and taking after the hubs with the most reduced way costs.Once the robot completely investigates the maze and there is a chart that portrays every fork, then Dijkstra's calculation could be utilized to locate the ideal way through the diagram.

- **A*** is a calculation utilized for pathfinding and diagram traversal [3]. At the point when given a beginning position and target position, A* will dynamically seek through every single conceivable way of length n expanding n through the inquiry space and around any discovered deterrents until the objective is found. When the objective is found, that way of length n must be the briefest way to the objective. There are less gullible variations of A* that will likewise ascertain the immediate course to the objective until a deterrent is found and at exactly that point innocently scan for a way around the impediment by assessing every conceivable way around the hindrance. Once the impediment is bypassed, the calculation keeps on taking an immediate way to the objective either until it is found or until the following obstruction is found. Like Dijkstra's calculation, A* could be utilized to locate the ideal way through the maze once it has been investigated, yet could likewise be utilized to help the robot investigate the maze notwithstanding when the robot's information of the maze is fragmented. The robot could, given whatever its learning of the maze, endeavor to move toward the focal point of the maze. As walls hindering the robot's way are found and the

robot's information of the maze expands, the calculation can be utilized to evade the obstructions reliably. This system will discover a way to the objective, however may not locate the ideal way. Once the robot achieves the objective, A* could then be utilized to achieve the unexplored zones inside the maze. At face esteem, this sounds culminate, however may not be extremely effective as the robot would likely go through as of now investigated regions bringing about a scoring punishment.

- **Graph traversal algorithms**. There are two sorts of diagram traversal calculations, profundity first pursuit and broadness first hunt [3]. As the names recommend, profundity first and expansiveness first vary by they way they select the following hub to extend as a part of the inquiry procedure. Profundity first hunt will extend a solitary way until it arrives at an end and afterward begin with the following neighboring hub. Broadness first pursuit will extend every neighboring hub dynamically working through the diagram until the objective is found. The two systems can either be a decent or poor decision depending what data is thought about the diagram and the relative areas of the begin and target hubs. On the off chance that the diagram is not exceptionally very much associated and there is an approach to channel the spokes of the chart driving from the begin hub as to pick a spoke with the most astounding probability of including the objective, then profundity first pursuit can demonstrate to discover the objective more proficiently than profundity first hunt. Assuming, be that as it may, the objective hub could be anyplace in the chart yet is thought to be nearer to the begin hub than the normal profundity of the majority of the ways driving from the begin hub, then broadness first hunt is probably going to be more productive. Both groups of calculations could be utilized to both, however autonomously, investigate the maze and additionally locate the ideal way once the investigation is finished. Both methodologies would finish these undertakings, however they would not get a similar score. The expansiveness first inquiry, which is an exceptional instance of Dijkstra's calculation, requires that the robot ceaselessly return to investigated parts of the maze to get to the distinctive parts of the unexplored limit. This procedure would bring about a huge investigation scoring punishment.

- **Tree traversal algorithms** are a subset of diagram traversal calculations. T2hese traversal calculations visit every leaf hub precisely once [4]. A chart can be a tree, however a tree is not really a diagram. A tree is characterized as having precisely one root hub and every branch from the root prompts to another hub. Each hub in the tree, aside from the root hub, must have precisely one parent hub. Each hub in the tree can have at least zero kids with the exception of the root that must have no less than one kid. Expecting a tree information structure is a reasonable representation of a maze, then the tree traversal calculations would work in a fundamentally the same as route to the profundity first and expansiveness first inquiry calculations talked about as of now. They do contrast in that trees don't have any circles to battle with and the fractal structure of trees make them exceptionally reasonable for recursive calculations. This permits tree traversal calculations to be exceptionally effective. Be that as it may, the strict meaning of the tree information structure implies it would just be an appropriate representation of a maze if all ways from the begin position prompt to at least one deadlocks with the exception of the one way which would prompt to the objective. The objective must be come to by means of that solitary way. This maze structure would not be reasonable for the substantial larger part of mazes including every one of the three test mazes that each incorporate no less than one circle. Therefore, utilizing tree traversal calculations would not present a change over chart traversal calculations.

As we have seen, the lion's share of these calculations would be reasonable to locate the ideal

way once the investigation is finished, however are less appropriate for the real investigation of the maze. The other perception is that these calculations, as would be normal, are innocent to the particular subtle elements of this maze issue. For instance, the robot's sensors give data about more than one space. As we saw in the Data Exploration segment of this project, the main sensor perusing for Test Maze 1 gives the robot data around 12 cells inside the maze. This data permits the robot to investigate the maze more productively than the guileless variant of the above calculations would permit. It would be more productive to tailor one of the pursuit calculations to utilize this extra data.

In actuality, every sensor perusing can fall the information of various cells all the while. This learning may permit the robot to know precisely the state of an unvisited cell or in any event constrain the quantity of conceivable shapes the cell may take. This information in blend with extra sensor readings may then permit the robot to decide with all assurance, as the sensor readings are flawless, the state of the unvisited cell. The ramifications of this is the robot could know the state of the entire maze without expecting to visit the greater part of the cells. The investigation procedure could be further advanced by verifying that the ideal way has been found even with fragmented learning of the maze. In the event that the obscure components of the maze couldn't bring about a more ideal course, then the current ideal way is the way that robot ought to take. Now, the robot would augment its score by ending the investigation run instantly and continue to the second keep running of the maze.

The perfect technique is to program the robot to monitor the what is thought about the maze, the guide, and also monitor conceivable shapes for every space given what has been learnt about the maze in this way. A* pursuit can then be utilized to investigate the maze guiding the robot to the zones the robot knows minimum about. This ought to expand information of the maze while exploiting the way that the robot can find out about the unexplored ranges without going to them. It ought to likewise minimize the quantity of steps required to learn enough about the maze to locate the ideal course. Dijkstra's calculation can then be utilized to locate the ideal way through a diagram representation of the maze given what is known.

1.2.4 Benchmark

The benchmark score for a given maze is the whole of a sensible way to the objective and one thirtieth of the sensible number of steps expected to investigate the maze given its size. The trap is to characterize what is 'sensible'. There is no motivation behind why the ideal way through the maze can not be found the length of the virtual robot finishes its investigation of the maze. Whatever other separation for the last way bit of the benchmark would be altogether subjective so the ideal way will be utilized for the benchmark. For Test Maze 1, examined prior, the ideal way is 17 stages.

The investigation bit of the benchmark is much harder to characterize. This uncertainty recommends that two benchmark scores would be fitting. An Upper Benchmark score will characterize a limit that any robot utilizing even a guileless investigation method ought to beat while a Lower Benchmark will characterize a much harder to accomplish edge that ought to require a robot to utilize a substantially more advanced and custom-made investigation rationale. Another approach to look at the two scores is that the Upper Benchmark is the biggest worthy score a working robot needs to beat while the Lower Benchmark is the score that a robot ought to attempt and beat to demonstrate the nature of the usage.

To visit each cell of a maze, it would be outlandish for a robot to cross the whole maze and visit each cell without halting at a cell more than once. Guileless investigation rationale could be actualized so as to train the robot to explore the maze and stop at each unvisited cell. When the

sum total of what cells have been gone by, then the robot will have a total guide of the maze and will have the capacity to locate the ideal way to the objective. This proposes the Upper Benchmark needs to take into account more investigation projects than there are cells in the maze, however even guileless investigation rationale ought to permit the robot to cross the whole maze without utilizing the same number of projects as double the quantity of cells and in the event that it does, then that is an indication that something isn't right with the investigation rationale. Therefore, the investigation segment of the Upper Benchmark score will be the measurement of the maze squared duplicated by two and isolated by thirty.

As talked about above, it would require sensibly advanced investigation rationale to locate the ideal way without going by even the same number of spaces as there are in the maze. The investigation bit of the Lower Benchmark score will be the measurement of the maze squared duplicated separated by thirty or, as it were, a large portion of the investigation part of the Upper Benchmark.

The race part of the two benchmark scores will be the same and match the quantity of steps required to take after the ideal way to the objective.

The aggregate benchmark scores for Test Maze 1 are as per the following:

```
Upper Benchmark: 17 + (12 x 12 x 2) / 30 = 17 + 9.6 = 26.6

Lower Benchmark: 17 + (12 x 12 ) / 30    = 17 + 4.8 = 21.8
```

1.3 Methodology

1.3.1 Data Preprocessing

The way of this project leaves a nonappearance of information preprocessing. The virtual robot sensor particular and plan of the virtual environment have been given as a part of the project definition.

1.3.2 Implementation

Subsequent to concentrate the issue for this project and thoroughly considering which calculations and systems could be utilized to take care of various parts of the issue, it turned out to be obvious that the bigger maze mapping and route issue ought to be separated into the accompanying sub-issues:

- Decide how to store and overhaul the robot's information of the maze.
- Determine how far the robot ought to move and in which course to expand learning of the maze while investigating yet minimize the quantity of steps taken amid investigation.
- Find the objective area and ensure the Robot has gone by it (this is a necessity to permit the robot to end the investigation run early).
- Once a way to the objective has been found, figure out whether it is the ideal way.
- Once the ideal way has been discovered, spare it, end the investigation run, and execute the second (race) run.

The way toward taking care of these sub-issues and executing the code that permits the robot to comprehend the maze and effectively entire both runs brought about a further in a roundabout way related sub-issue.

- Determine how to rapidly pinpoint and investigate sensible mistakes in the usage.

Decide how to store and update the robot's knowledge of the maze. It was felt that the robot's knowledge of the maze should be stored in two two-dimensional lists. One would store knowledge about the edges bordering each cell of the maze, the Wall Map. This would allow the robot to know where the walls and openings exist within the maze and which edges are of unknown state (wall or opening). The robot's prior knowledge of how the mazes are constructed allowed some of this information to be pre-populated upon initialisation. It could know that all exterior edges are walls and the four edges interior to the goal area must be openings.

The second two-dimensional list would maintain knowledge of the possible shapes each cell could have, the Uncertainty Map. As discussed in the Problem Statement of this report, a given interior cell could have one of 15 different shapes given the permutations for walls and openings with the one exception being four walls. As a result of the prior knowledge of the maze walls, some of the cell shape possibilities could be narrowed down upon initialisation. Corner cells could only have two possible shapes, edge cells could have 7, and goal cells could have 3 possibilities. When the robot is certain of the shape of a cell, the uncertainty for that cell decreases to 1.

At the beginning of each time-step, when the robot receives a new set of sensor readings, the data structure storing information about the walls would be updated with the knowledge of the walls and openings on the left, front, and right sides of the robot given its current location and heading. This updated Wall Map could then be used to update the Uncertainty Map. While the Robot explores new areas of the maze, the uncertainty surrounding the robot will decrease.

The data structure storing the cell shape possibilities is very straightforward. As every maze is square, the outer list, as well as each sublist, has the same length as the dimension of the maze. The data structure that stores knowledge about the cell edges is a bit more complicated. To conform to the array indexing used elsewhere in this project (ie. the first index of the first sublist is the bottom left corner of the maze), the edge information would be stored from left to right and from bottom to top. The first sublist represents the left edge of exterior vertical walls and the next sublist represents the horizontal walls separating the leftmost column of cells within the maze. This pattern then repeats with each sublist alternating between vertical and horizontal edges. What complicates this data structure is that there is one extra horizontal wall for every vertical wall and the total number of sublists is twice the dimension of the maze plus one. This results in atypical and somewhat confusing array indexing, but it does result in a single data structure that stores information about every edge in a maze without repeating edges (as would be the case if all four edges were stored for every cell even though two neighbouring cells share an edge).

***N.B.**: Even though one of these two-dimensional lists could have been stored as a NumPy matrix, the other could not as the sublists do not have consistent length. As a result, it was decided not to use Numpy for either two-dimensional list for the sake of consistency.*

Determine how far the robot should move and in which direction to maximise knowledge of the maze while exploring yet minimise the number of steps taken during exploration. Since the robot can monitor the edges inside the maze, through the Wall Map, and the quantity of conceivable shapes for every cell, by means of the Uncertainty Map, the time has come to choose how to proficiently investigate the maze. The best approach is decide the cell with the best level of vulnerability inside the maze and explore the robot toward that cell. On the off chance that the Uncertainty Map has more than one pinnacle, the robot will endeavor to achieve the pinnacle nearest to it. To discover the zone of most noteworthy vulnerability, it seemed well and good proportional the qualities to misrepresent territories of most prominent instability. This felt vital as the qualities inside the guide would, generally, have a similar esteem particularly toward the start of the investigation run. To separate between these qualities, a scaling component was contrived and after

that connected to separate between the cells with most noteworthy vulnerability. The technique for scaling in the underlying usage was to duplicate the phones of most prominent vulnerability by the range between that phone and the nearest neighboring cell with a littler instability esteem. A gullible approach was disentangled the usage.

For every cell with most extreme vulnerability, take a gander at the instability esteem for the eight cells encompassing it. In the event that those phones likewise have the most extreme instability esteem, add one to the span and take a gander at the eight cells that characterize the corners and center of the edges of the ring around the first cell being referred to. Once a phone was found with a vulnerability esteem not exactly the phone being referred to, duplicate the first cell's instability by the sweep of the ring where the more certain phone was found. This new esteem was then put away in an alternate two-dimensional rundown to maintain a strategic distance from a multiplicative impact. This approach is a long way from flawless however felt like a sensible first way to deal with finding the regions of most prominent vulnerability inside the maze.

This, in any case, was just the initial segment of the arrangement. The robot still should have been ready to discover a way from its present area to the area of most noteworthy instability. Eventually, it was concluded that Dijkstra's calculation would be less demanding to execute and was utilized to discover ways through the maze. To utilize Dijkstra's calculation, the Wall Map of the maze would should be changed over into an undirected diagram. The diagram could then be bolstered through Dijkstra's calculation. The way costs created by Dijkstra's calculation would then be utilized to locate the speediest way from the present cell to the objective cell. This was accomplished by working in reverse from the objective to the present cell taking after the inclination of littlest way costs.

There were a couple of intricacies while executing Dijkstra's calculation. Unmistakably the ideal ways were not being found as the way expenses were not right for each cell. At initially, it was thought this was because of the very associated nature of the diagram and that a given cell could be gotten to by means of countless. It was later found that there were a couple of blunders with the execution. An underlying fix for the issue was to rehash the calculation various circumstances until the way costs did not change anymore and had merged upon the right values. This clearly was not the right arrangement but rather functioned admirably enough to move onto different issues with the code. At last, it was found that sorting the unvisited hubs list by way cost was returning wrong values when the way cost was set to the underlying estimation of `float("inf")`. As opposed to utilizing this expression for endlessness as the underlying way cost, the quantity of cells in the maze was utilized. Given that no way could be longer than the aggregate number of cells, the code was left coherently predictable. This little change settled the examination while sorting hubs by way cost and at last settled the first issue with the calculation. The code then precisely and reliably mapped the way costs and gave back the ideal way through the maze.

The way was developed in a manner that the robot could move encourage in a solitary time-project the length of it didn't go through a cell with an instability esteem more prominent than 1. This guarantees the Robot would logically and precisely investigate the maze without skipping indeterminate cells.

For every time-project in the investigation run, this procedure would be rehashed and the robot would move the quantity of cells and in the course as characterized by the initial phase in the way created by Dijkstra's calculation.

Find the objective area and ensure the robot has gone to it. Knowing the state of the objective zone, a 2x2 square of cells encompassed by seven walls and a solitary opening, the robot could

find how to get to the objective of the maze by either discovering every one of the seven walls or the single opening. Upon either happening, the vulnerability of every one of the four objective cells would fall to an estimation of 1 regardless of the possibility that each of the 8 edges had not yet been detected by the robot.

Once the passageway to the objective is found by the robot, its objective area inside the maze would turn into the objective itself. This guarantees the robot has gone by the objective before endeavoring to end the investigation run. Once the objective has been gone by, the robot could keep investigating as it had some time recently.

Once a way to the objective has been found, figure out whether it is the ideal way. Figuring out if a discovered way was the ideal way demonstrated less demanding than anticipated. The calculations used to explore to the territories of most prominent vulnerability inside the maze could be re-utilized. The procedure would basically require finding the best way through the maze from the begin to the objective once utilizing a variant of the Wall Map expecting every single obscure edge are walls (keeping the way going through unexplored territories inside the maze) and again utilizing the Wall Map accepting every single obscure edge are openings. This second way would locate the quickest conceivable way to the objective accepting the sum total of what walls had been found. On the off chance that the two ways made by this procedure are of a similar length, then the ideal way has been found.

Once the ideal way has been discovered, spare it, end the investigation run, and execute the second (race) run. Once the ideal way has been found, the robot just stores it alongside the developments and turns required to take after the way. For whatever length of time that the objective area has been gone by, the robot could then total the investigation keep running by returning "Reset" as both the development and turn values. The demonstration of playing out the second race keep running of the maze is as basic as monitoring the time-step and taking after the development and pivot guidelines for that time-project to take after the ideal way.

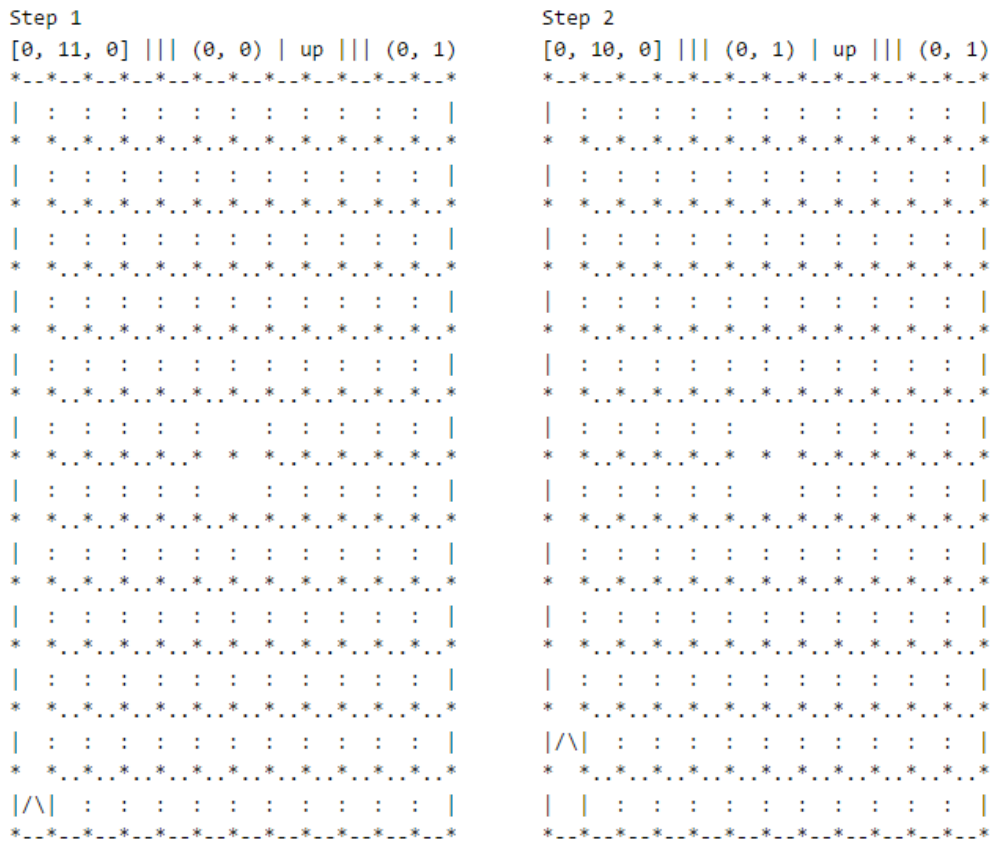
Determine how to quickly pinpoint and debug logical errors in the software implementation. The somewhat entangled ordering of the Wall Map made investigating by means of customary means (utilizing PUDB and reviewing variable qualities) extremely tedious and not exceptionally beneficial. To help this procedure, it felt like the best approach was to compose a technique that printed an Ascii attracting of the maze to the terminal while the robot investigated it. This was later enhanced by including the area and heading of the robot to the drawing. Indeed, even later, the capacity was corrected to show the way costs for every cell inside the drawing. This had an incredible effect when endeavoring to troubleshoot the issues with the usage of Dijkstra's calculation talked about beforehand. Having this data printed to the terminal alongside sensor readings, area, heading, development, and turn values implied distinguishing issues in the robot's rationale and when the conduct strayed from the desire turned out to be quick and simple.

As a representation, these are the initial six terminal yields for the robot while investigating Test Maze 1 alongside the last guide with way costs.

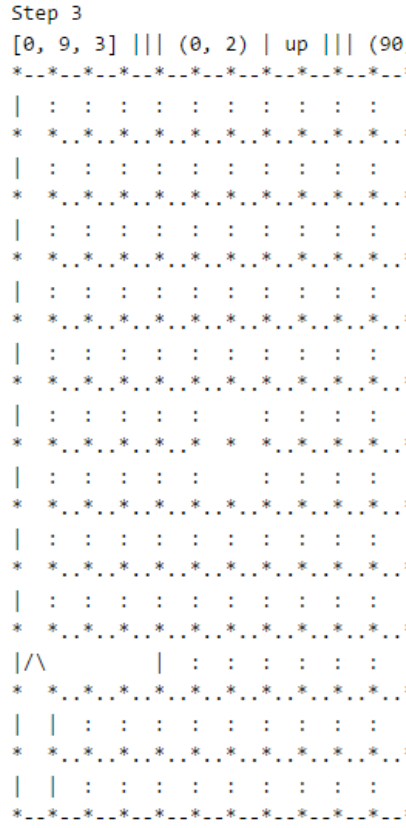
```
The information above each drawing of the maze:
Sensor Readings [left, forward, right] ||| Location (x,y) |
Heading ||| Instructions for the Next Step (rotation, movement)

Knowledge of each edge is represented by lines for walls,
```

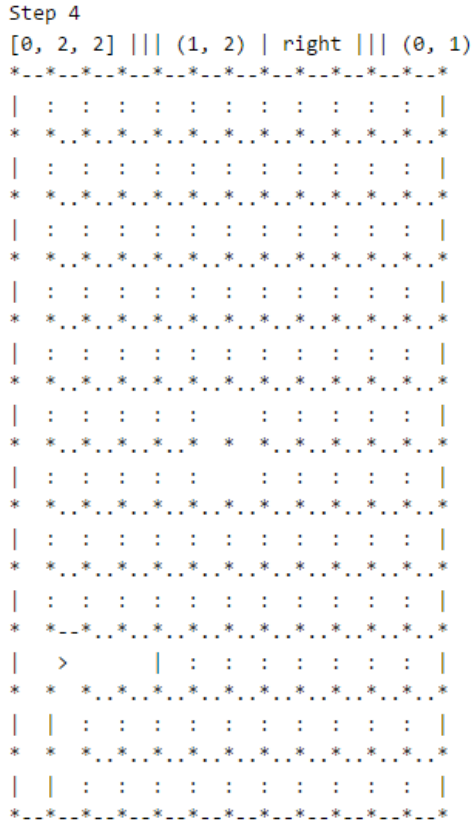
spaces for openings, and dots for unknown states.



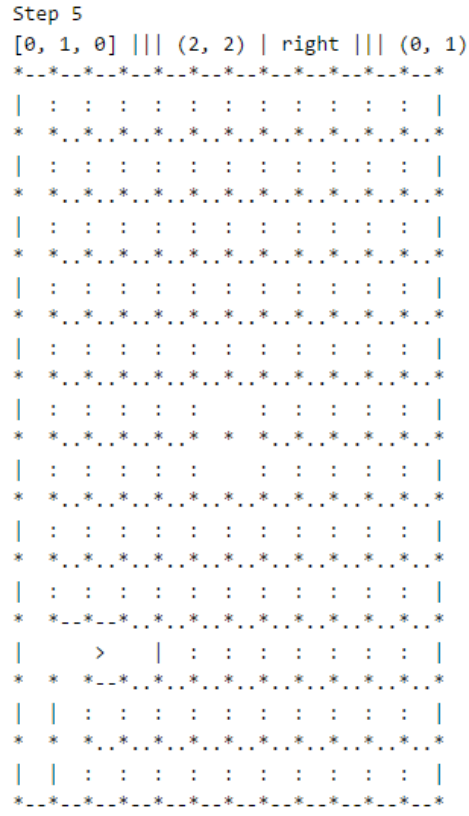
Step 3



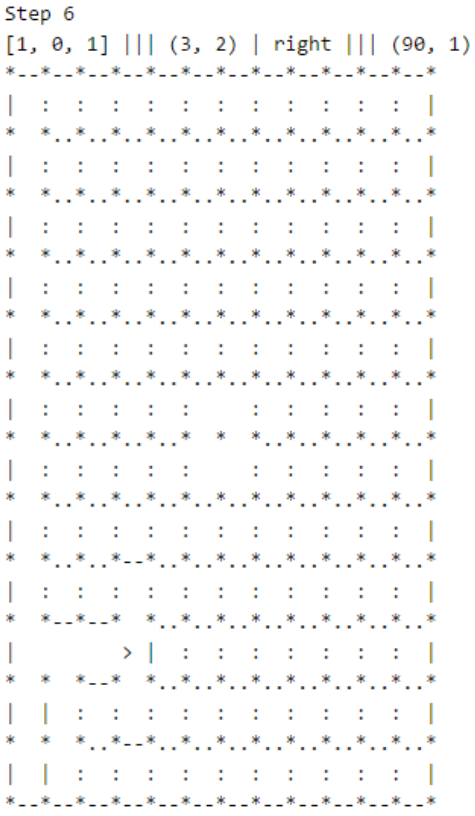
Step 4



Step 5

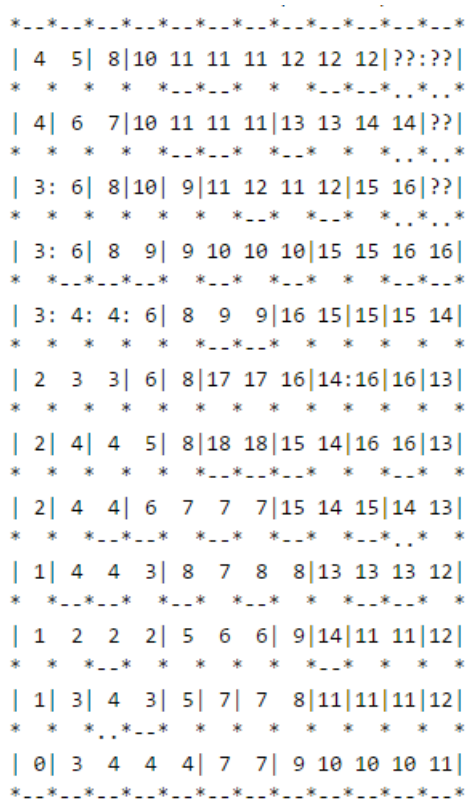


Step 6



Goal Location is: (6, 6)

```
Path costs for each explored space within the maze:
```




```
Where the path cost can not be calculated due
to the cell not having been explored
sufficiently, the value is '??'.
```

Plainly the robot's learning of the maze increments with each extra stride and sensor perusing. It is likewise significant that the way costs ascertained by the robot, for the completely investigated regions of the maze, coordinate precisely the heatmap delineation indicated before in this report.

1.3.3 Refinement

The way toward refining the robot usage brought about three essential attempts. The code was refactored so that as opposed to keeping the greater part of the rationale in the robot class according to the underlying execution, the greater part of the rationale was moved to pilot and mapper classes both to clean up the code and to help seperate and typify the intelligent responsibility for between the classes. This helps both the association and the intelligibility of the robot, pilot, and mapper code.

Past the upgrades and refinements as of now examine in the Implementation segment above, there were two key extra refinements.

The first was to change the scaling of instability qualities when the guide endeavors to locate the best target cell to explore to while investigating the maze. Different scaling procedures were investigated including taking the entirety of instabilities and adding or increasing them to crest vulnerability values, yet this at last had little effect to the robot's execution. The underlying

thought of finding the area of most prominent vulnerability at last left the robot investigating in an extremely wasteful way. The robot was found to investigate one a player in the maze for a brief span, then move over the maze, investigate there, and after that arrival to the main region of vulnerability. Endless supply of how the calculation functioned, this conduct was not that shocking. On the off chance that the maze had at least two ranges of instability, the robot would investigate one until it was a tad bit more sure than another region and soon thereafter it would move to the next region. When it was somewhat more sure than the primary region, it would then rehash this procedure until it had adequately investigated both. This left the robot investigating wastefully as it was making many strides moving between the ranges of instability as opposed to investigating one preceding moving onto the following. The system that demonstrated to bring about the best score was to leave the vulnerability values unscaled. This permits the robot to dynamically search out every phone of most prominent vulnerability without making a trip too far to discover them. The robot still ricochets around the maze to some degree near the end of the investigation run when there are far less cells with an instability esteem more prominent than one. There is, point of fact, space for extra change, yet this little change had the best effect on enhancing the robot's score.

The second change was fundamentally executed to take care of an infrequent issue with the robot's conduct however wound up enhancing the effectiveness of the robot's investigation of the maze also. Now and again, the robot was found to stall out between two cells skipping forward and backward between them inconclusively. This was plainly on the grounds that the ideal way the robot's objective cell when at cell A was to move to cell B, however when at cell B, the robot's ideal way to the objective implied moving back to cell A. This was effectively settled by refactoring the investigation rationale. Instead of just utilizing the initial step as a part of the way, the robot would utilize the initial two stages in the way produced by Dijkstra's calculation unless one of the accompanying special cases happened:

- The created way just had a solitary stride.
- The sensor readings subsequent to venturing out the way uncovered walls that would hinder the robot when making the second stride.

In the wake of making the second stride in the way or experiencing both of the above special cases, the investigation procedure just returned to the first rationale searching out the best way to the nearest cell of most noteworthy instability.

Testing the robot against a substantial number of haphazardly produced mazes (talked about in more prominent detail in the following segment), represented that, in specific situations, the robot could in any case stall out circling between more than two hubs. This implied a more strong arrangement was required to battle these circles. Extra rationale was added to scan for rehashed designs in the most as of late went to hubs. For every circle length somewhere around 2 and 6, the pilot figures out if the robot has navigated an arrangement of hubs, remembered its means back to the start of the grouping and afterward rehashed the main arrangement of hubs. At the point when this happens, the guide guarantees that robot makes an extra stride in the way to its objective trying to break the circle. A most extreme circle length of 6 was chosen by experimentation to minimize the shot of the robot stalling out without bringing about a huge effect on execution. The execution was a worry as this procedure is performed for each progression taken by the robot amid the investigation run. This extra rationale resolved the limitless circle issue for many arbitrarily produced mazes, however not every one of them. A more nitty gritty examination of this issue is incorporated into the Free-Form Visualization segment of this report.

The progressive change in performance is displayed in the table below. The changes follow the order in which they were made to the code.

Implementation	Test Maze 1		Test Maze 2		Test Maze 3	
	Score	% change	Score	% change	Score	% change
First working version	26.700		31.400		34.567	
Take second step	23.633	11.49%	34.033	-8.39%	34.467	0.29%
Use optimal path	21.233	10.16%	29.433	13.52%	33.267	3.48%
Simplify uncertainties	20.667	9.73%	28.567	2.94%	31.767	4.51%
Break recursive loops	20.667	0	28.567	0	31.767	0
Total Improvement	**6.033**	**22.60%**	**2.833**	**9.02%**	**2.8**	**8.10%**

These refinements along with fixing the implementation of Dijkstra's algorithm resulted in an average improvement of 13% for the scores achieved on the three test mazes provided for this project with the greatest improvement coming from the performance against Test Maze 1.

It helps to know the Upper and Lower Benchmark scores for each maze to put the improvements into context and better understand how important these improvements have been.

	Robot's First Score	Upper Benchmark	Lower Benchmark	Robot's Final Score
Test Maze 1	26.700	**26.600**	**21.800**	20.667
Test Maze 2	31.400	**35.067**	**28.533**	28.567
Test Maze 3	34.567	**42.067**	**33.533**	31.767

The first working version of the robot did not even pass the Upper Benchmark for Test Maze 1 though it did for the other two mazes. It was not until the improvements were complete that the robot was able to beat the Lower Benchmark score for Mazes 1 and 3 and come very close to that threshold for Test Maze 2.

1.4 Results

1.4.1 Model Evaluation and Validation

To sufficiently assess the execution of the robot route and mapping rationale, it felt as if the three test mazes would not be sufficient to affirm how well the robot would perform when gone up against with a more prominent assortment of maze shapes and sizes. Rather, it was chosen that a bigger specimen of arbitrarily created mazes would be required. They were made utilizing an altered rendition of the Recursive Division maze era calculation [6]. This procedure is very easy to execute and guarantees that all cells are open and, by definition, guarantees that each maze is resolvable. Generally, the maze era prepare takes a clear maze beginning with the underlying conditions required to comply with this present project's issue articulation and afterward continuously substitutes between haphazardly chose areas of flat and vertical edges. The essential Recursive Division calculation takes extends of walls from edge to edge between zones where walls/openings have as of now been characterized, yet relying upon the irregular choice, this technique can bring about mazes that have long passageways with couple of openings at last restricting the assortment in mazes created. Rather, the edge determination criteria confine every choice to an extend

of just four edges. From these four edges, a solitary opening and three walls are characterized. This demonstrated to bring about a much more prominent assortment of maze shapes than the fundamental Recursive Division calculation. Unavoidably, there will be similitudes in the maze structure given the technique for era, yet this felt to be a sensible trade off.

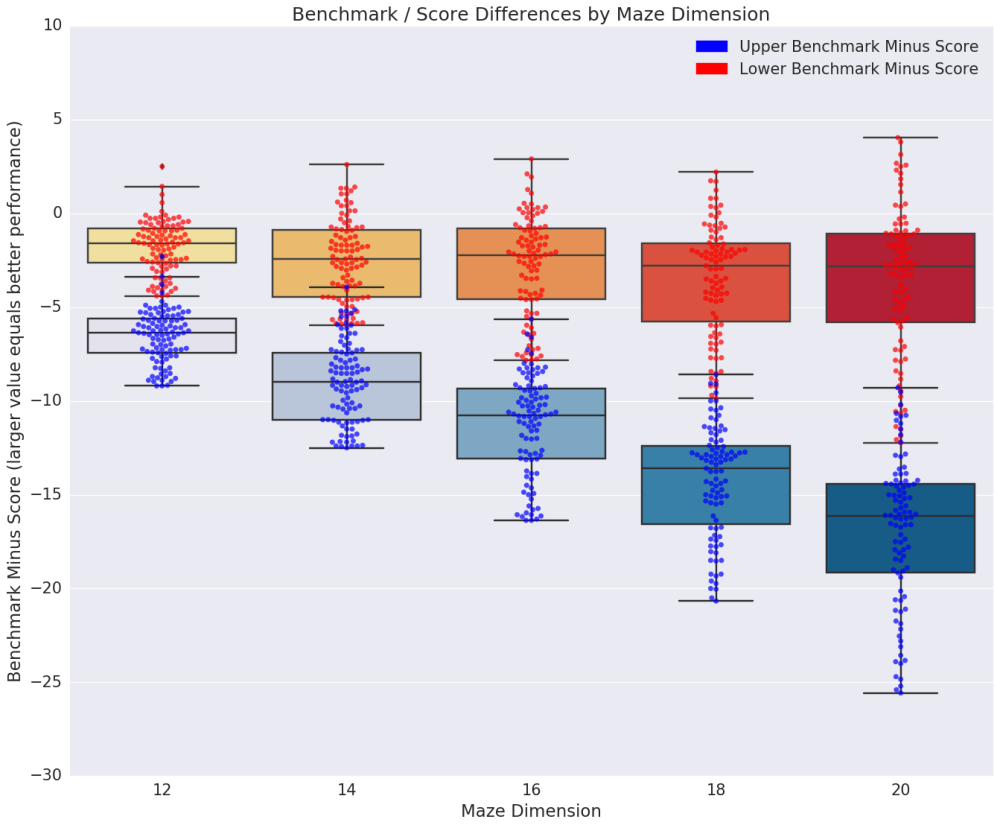
The maze generator was then set to produce 100 mazes for each even measurement from 12 to 20 making an aggregate of 500 test mazes.

The robot effectively finished everything except 17 of these mazes. While endeavoring the 17 anomalies, the robot entered a recursive circle it couldn't get away. The 12x12 case from these 17 mazes will be examined in more detail in the Free-Form Visualization segment of this project.

The robot performed so well when contrasted with the maze benchmark delineated in the Benchmark segment of this report, a stricter benchmark was characterized to better break down the outcomes.

The race benchmark is the same for both the Upper (less strict) Benchmark and Lower (more strict) Benchmark, yet the investigation scores contrast by a solitary variable. The Upper Benchmark incorporates an explor1tion score that depends on the robot halting in every cell twice while the Lower Benchmark permits the robot to, all things considered, just stop at every cell once.

Figure 3 demonstrates the distinction between the robot's score and each of the two benchmarks for each effectively finished maze for each of the five maze measurements. This distinction is characterized in a manner that a bigger positive esteem demonstrates that the robot beat the Benchmark score by that sum. This implies a bigger positive number is superior to anything a littler number. Negative numbers show that the robot did not beat the benchmark.



The robot's scores compared to the benchmark scores across all 483 completed randomly generated mazes.

It is clear that for the very large majority of mazes, the robot beat both of the Upper and Lower Benchmark scores. As the maze dimension increases, the robot outperformed the benchmarks by a larger amount.

1.4.2 Justification

As displayed previously, the Benchmark scores for the original 3 test mazes are as follows along with the maze dimensions, optimal path length, and the robot's final scores:

	Dimension	Optimal Path	Upper Benchmark	Lower Benchmark	Robot's Score
Test Maze 1	12	17	26.6	21.8	20.667
Test Maze 2	14	22	35.067	28.533	28.567
Test Maze 3	16	25	42.067	33.533	31.767

The robot's scores for these three mazes are 20.667, 28.567, and 31.767 individually. This implies the robot figured out how to locate the ideal way in just 110 stages for the primary maze (34 stages less than the quantity of cells), 197 stages for the second maze (1 stage more than the quantity of cells), and 203 stages for the third maze (53 stages less than the quantity of cells). The robot outflanked the Lower Benchmark for two out of three mazes and missed this benchmark by just a solitary stride for the second maze recommends that the arrangement executed for this project sufficiently meets the execution necessities.

A nearer examination of the robot's execution against the 483 finished haphazardly created mazes demonstrates the accompanying:

- The robot outflanked the Lower Benchmark for 420 of the 483 mazes (86.96%).
- all things considered, the robot beat the Lower Benchmark by 2.77.
- all things considered, the robot beat the Upper Benchmark by 11.53.

Given these outcomes, it feels as if this execution of the robot effectively explains the maze mapping and route issue as characterized for this project.

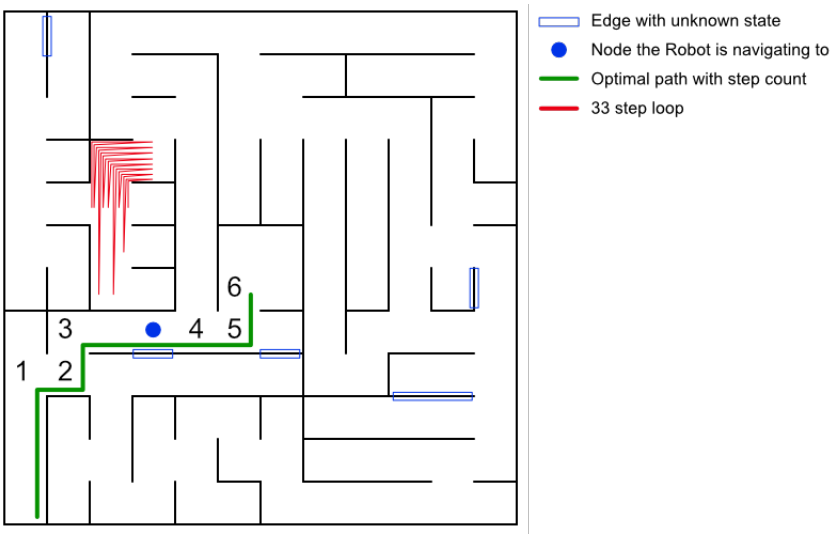
1.5 Conclusion

1.5.1 Free-Form Visualization

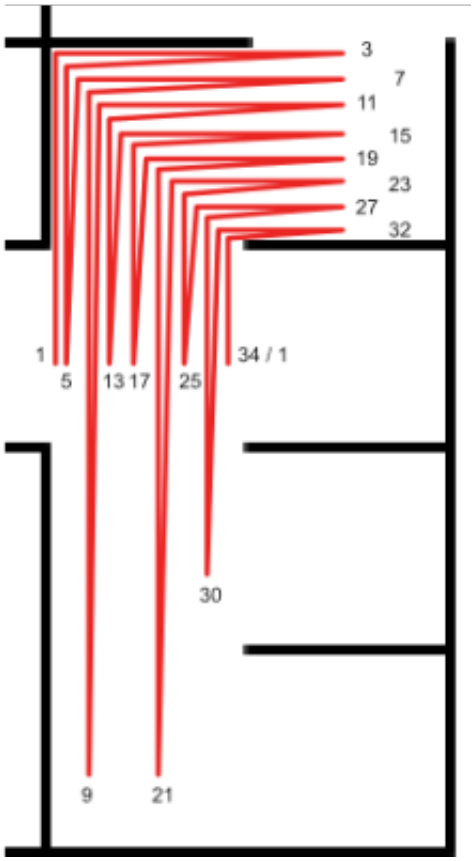
The arbitrary maze generator used to assess the power of the robot route calculations figured out how to produce a few mazes that brought on the robot to stall out. Figure 4 portrays the single 12x12 illustration where the robot was not able finish the maze.

Subsequent to investigating the maze almost to consummation in 131 stages, the robot cycles uncertainly bringing about the pursue to time out 1000 stages. This is especially surprising considering the lengths taken to enhance the route heartiness particularly to evade this conduct. At the point when the robot stalls out, it rehashes the accompanying 33 hub circle and once more.

```
(2, 7) (2, 8) (3, 8) (2, 8) (2, 7) (2, 8) (3, 8) (2, 8) (2, 5) (2, 8)
(3, 8) (2, 8) (2, 7) (2, 8) (3, 8) (2, 8) (2, 7) (2, 8) (3, 8) (2, 8)
(2, 5) (2, 8) (3, 8) (2, 8) (2, 7) (2, 8) (3, 8) (2, 8) (2, 7) (2, 6)
(2, 8) (3, 8) (2, 8)
```



The problematic 12x12 maze annotated with the robot's knowledge of the maze, the loop where robot gets stuck, and the optimal path through the maze.



Closer view of the loop within the context of the maze shown above with the step numbers from 1 through 33.

```
The robot then moves to (2, 7) to reach the beginning of the loop
causing it to repeat the path again.
```

This conduct is unmistakably the aftereffect of the mix of the investigation qualities where the pilot takes after the initial two stages in a way and the way that when searching for circles inside the went by hubs, the guide endeavors to coordinate a way three circumstances (in the first bearing, backtracking those means, and rehashing the first arrangement of steps. The circle condition for a circle of 3 hubs is met a few circumstances inside the bigger circle. The accompanying is the three hub circle rehashed three circumstances that happens toward the start of the 33 hub circle and additionally inside it.

```
(2, 7) (2, 8) (3, 8) (2, 8) (2, 7) (2, 8) (3, 8)
```

The rationale used to battle this conduct is the thing that causes the hubs (2,5) and (2, 6) to appear inside the bigger circle however less as often as possible than alternate hubs.

Is shocking that the ideal way to objective is short and straight forward. Indeed, even with the robot's fragmented learning of the maze, the way to the objective just requires six stages. This recommends the robot would have in a perfect world quit investigating much sooner than it stalled out. Be that as it may, the reason it didn't is that there is a conceivable way that would achieve the objective in one less strides if the wall between (3, 3) and (3, 4) was an opening rather than a wall.

1.5.2 Reflection

At its least complex, the issue and answer for this project can be portrayed by the accompanying procedure:

1. Get a little measure of data about the maze (sensor readings).
2. Join this data with the heading and area of the robot.
3. Overhaul the maze outline any new data construed specifically or in a roundabout way from the sensor readings.
4. Given the present learning of the maze, choose where to go to augment the securing of new data.
5. Choose how to get from the present area to the wanted area.
6. Move toward the wanted area.
7. Rehash steps 1-6 until the ideal way from the begin to objective areas has been found.
8. Begin the following keep running of the maze.
9. Take after the ideal way from the begin to the objective.

As direct as this sounds, the real execution of the robot that can reliably and effectively entire this procedure in a productive way notwithstanding when confronted with an expansive number of maze changes is impressively more troublesome. Indeed, even subsequent to emphasizing through a few systems to keep the robot from stalling out, regardless it figured out how to enter recursive circles that it couldn't get away. An altogether vigorous arrangement would likely build the multifaceted nature of the investigation rationale and might require giving up the productivity by which the robot investigates the dominant part of mazes it appearances and subsequently bringing on the robot to perform more awful on the lion's share of mazes essentially to perform better for a little number of others. Whether this would be an adequate bargain is to a great extent

scholarly and is outside of the extent of this project (given the present project parameters and scoring rules).

In spite of this trouble, the calculations used to outline explore the maze are very straight forward. It is intriguing to think about how as a little number of sensibly basic calculations, when joined soundly, can bring about very mind boggling conduct and figure out how to take care of the current issue so well. The two most confounded calculations utilized as a part of the project are the calculation that changes over the maze delineate an undirected diagram and Dijkstra's Algorithm used to locate the ideal way through that chart. Notwithstanding for these two calculations, the execution is sensibly straightforward regardless of the possibility that troubleshooting and refining them was more testing than anticipated.

The most fascinating part of the project was, before an essential execution of the robot that figured out how to finish the maze was created, to discover approaches to enhance the execution of the robot and additionally its strength. Formulating approaches to test the robot and perceive how it would adapt managing to various molded mazes additionally expanded the delight in taking care of these issues.

1.5.3 Improvement

In the event that for a minute we consider the effect of developing the issue for this project, we will see exactly the amount more troublesome taking care of maze mapping and route issues could be. On the off chance that as opposed to having an all around characterized maze where sensor readings, developments, and pivots are immaculate, walls have no thickness and the robot is dependably impeccably situated inside a cell, and rather, developments and sensor readings were boisterous then the execution utilized for this project would no longer work so well. Having a thickness of the walls and a measurement fo the robot would bring about sensor readings that would be all the more difficult to work with, yet not by much.

For instance, if the wall thickness was 0.1 (0.05 in every cell between which the wall dwells) and the robot's distance across was 0.4 situated at the focal point of the possessed cell, the robot's first sensor readings for Test Maze 1 would have been `[0.25, 11.25, 0.25]` rather than `[0, 11, 0]`. The new sensor readings could be effectively changed over to the first readings by basically taking the floor of every esteem changing over them to the simple to work with numbers.

Assuming, be that as it may, the new sensor readings, developments and revolutions were do not immaculate anymore and were, truth be told, probabilistic, the robot would need to utilize an altogether different approach for mapping and route. For this sort of robot control situation, there are a couple of calculations that could be utilized to tackle distinctive parts of the issue.

- **Kalmen Filters** takes uproarious estimations saw after some time of obscure factors and delivers gauges for those factors that are more exact than a solitary estimation. The calculation applies Bayesian surmising and builds up a joint likelihood circulation for the factors inside the setting of the time allotment amid which the perceptions are made [7]. Kalmen Filters can be utilized for robot route and localisation.

- **Particle Filters** is a hereditary sort change choice molecule calculation where an accumulation of virtual representations of the robot are initialised haphazardly [8]. With each extra perception, the virtual representations are sifted in light of the probability that the given representation precisely depicts the condition of the robot. After some time, the Particle Filters focalize on the best gauge of the condition of the robot. Like Kalmen Filters, Particle Filters can be utilized for route and localisation.

- **PID Controller** is a control circle criticism component that consistently computes a blunder term that is the contrast between the wanted esteem and a perception of that esteem. The controller then adjusts the robot's development utilizing relative, fundamental, and subsidiary terms (henceforth PID) to take into consideration the consistent merging of the robot's position and the sought position [9]. This component can be utilized to control a robot's developments when those developments are loud or postponed, (for example, guiding a ship).

- **Simultaneous Localisation and Mapping (SLAM)** is a procedure that permits a robot to at the same time develop a guide of an obscure domain while exploring it [10]. Pummel depicts a gathering of calculations and strategies that, when actualized in show, cooperate to take care of the SLAM issue. Hammer applies Bayes manage taking into consideration the successive overhauling of localisation rear ends given a guide and move capacities. Through a comparable strategy, the guide can be consecutively upgraded. Pummel has clear applications for robot mapping and route.

A less difficult approach to make this project additionally difficult without having to fundamentally adjust the project parameters is to require the robot to explore and outline maze without advising it of the measurement of the maze from the beginning. This would compel the robot to not just redesign its learning of the edges inside the maze additionally grow the guide itself as new parts of the maze are found and investigated. Much all the more difficult is permit the guide to have an odd number of edges on every agree with a 3x3 objective zone in the middle yet keeping every single other control reliable. The blend of these two changes would altogether expand the test introduced by this project.

As it was characterized, this project displayed a pleasant issue to explain without being excessively troublesome.

1.6 References

1. Wikipedia contributors, "Micromouse," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Micromouse&oldid=709118923 (accessed March 9, 2016).

2. Wikipedia contributors, "Dijkstra's algorithm," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Dijkstra%27s_algorithm&oldid=745950368 (accessed October 24, 2016).

3. Wikipedia contributors, "A* search algorithm," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=A*search*algorithm&oldid=744637356 (accessed October 16, 2016).

4. Wikipedia contributors, "Graph traversal," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Graph_traversal&oldid=703421335 (accessed February 5, 2016).

5. Wikipedia contributors, "Tree traversal," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Tree_traversal&oldid=745817776 (accessed October 23, 2016).

6. Wikipedia contributors, “Maze generation algorithm,” Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Maze_generation_algorithm&oldid=745040926 (accessed October 18, 2016).

7. Wikipedia contributors, “Kalman filter,” Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Kalman_filter&oldid=744979113 (accessed October 18, 2016).

8. Wikipedia contributors, “Particle filter,” Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Particle_filter&oldid=747290664 (accessed November 1, 2016).

9. Wikipedia contributors, “PID controller,” Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=747236397 (accessed November 1, 2016).

10. Wikipedia contributors, “Simultaneous localization and mapping,” Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Simultaneous_localization_and_mapping&oldid (accessed November 1, 2016).

In [ ]: