Graph Theory:

A branch of computer science and mathematics. It revolves around networks of points connected with lines.

Raspberry Pi Pico:

A Micro controller with 26 GPIO pins (General Purpose Input Output pins). It has a 133 MHz processor with 264 KB SRAM (Static Random-Access Memory) and 2 MB of flash memory.

Problem: To explore and solve a maze in the shortest path.

Solution: Graph Theory and Maze Solving Algorithms.

Criteria:

1. The robot must guide itself from start to end.
2. Once the robot has started, no human interaction must be made.
3. The algorithm must be efficient enough to run on a microcontroller.
4. Self calibration is required to make accurate movements.

Applications of maze solving algorithms:

- Manufacturing/Warehousing - An example of this are the Amazon robots, they calculate the shortest path from locations to keep everything efficient.
- Home Automation - Robot vacuum cleaners and home monitoring navigate themselves through the house.
- Route Optimization - Firefighters and paramedics use this to find the best and shortest route to the destination so they can get to the victim as fast as possible.
- Rescue Missions - *"Mobile robots have been employed widely in the search and rescue operations, such as searching for victims in dangerous areas which are harmful for humans."*[1]

Definitions

Stack (data type):

Like a stack of pancakes, First in First out. Values are added and removed from the same side.

Queue:

Like a line of people added to one side and removed from the other. A dequeue is a double ended queue, you can add and remove from either side.

Node:

Refers to a point or vertex.

Graph:

A system of connected nodes. There are commonly 2 ways to store graphs: Adjacency Matrix and Adjacency List. I did not use either in my project, instead I used a grid layout to store the maze using a python list/array.

EX. let 0 represent walls and 1 represent open space:

```
Map = [[0,0,0,0,0],
       [0,1,1,1,0],
       [0,1,0,0,0],
       [0,1,1,1,0],
       [0,0,0,0,0]]
```

Linked List:

An item stores “data” and “next item’s index number”. It can be implemented using classes or lists/arrays.

Iterative Function:

Repeated code with a loop (For or While loop).

Recursive Function:

A function that calls itself again and again to get the repeated code.

Pinout

Pin	**What it does**
GND	Provides ground (0 Volts) for circuits
Power	Provides power for the Pico or for circuits
UART, SPI, I2C	Used for communication with other devices
GPIO, PIO, PWM	GPIO(General Purpose Input Output) allows you to take input and give output to and from electronic circuits. PWM (Pulse Width Modulation), in short, allows you to control the amount of average voltage you give using pulses. PIO (Programmable Input/Output) allows you to manually communicate with other devices
ADC	Analog-to-Digital Converter is used to measure an analog signal such as temperature, and convert it to a digital signal the processor can read
System Control	Used to control the processor (reset it, disable it, etc)

Breadth First Search

The Breadth First Search Algorithm explores a graph in a layered fashion. It starts at a node, explores all its neighbors, then explores the neighbors after that, and so on. It uses a queue to keep track of where it is in the process. It is not typically seen in a recursive form. You can find the shortest path using this algorithm by keeping track of the path as you move along. If you reach the target node, loop back through the path until you hit start and then reverse the order of the path.

Depth First Search

The Depth First Search Algorithm explores the graph by focusing on searching depth at first. You keep visited nodes until you hit a dead end, then you backtrack to the most recently visited node that is still being visited. Typically you would see this algorithm in a recursive form, however, you can change it to a derivative form by using a stack.

Collecting Maze Data
(My Algorithm)

I custom made this algorithm to collect maze data as the maze is being explored. You can then use this data to find the shortest path between any two nodes. It starts with a point surrounded by a wall. If the next point is outside this boundary, then add a wall in that direction and mark that point as open. This might present a problem when trying to go up or left. All the points must be shifted down/right in order not to distort the data.

Robot Configuration
(My Algorithm)

As I was testing my prototypes, I noticed that the robot moves differently on different surfaces and with different voltage levels (as the battery loses capacity, voltage is reduced). To fix this, I made a program that calculates turns and movements. The robot would move forward until a wall is hit and based on the distance specified, I can calculate the time to move 1 mm. With this value I can move and turn the robot consistently the same amount, no matter what surface condition or battery voltage are.

I combined Breadth First Search and Depth First Search to create my own algorithm. My algorithm can efficiently explore a looped maze and find the shortest path.

Steps Followed to Create Computer Program and Build Robot Car

Learning Experience from Using Various Hardware

Step 1 - Research

- Explored Graph Theory
- Explored Micro controllers

Step 2 - Creating the computer simulation

- Pygame (visual extension of Python)
- Breadth first search
- Depth first search

Step 3 - Adding physical elements to the program

- Collecting maze function
- Direction tracking

Step 4 - Finding the right hardware and understanding it

- Worked on the first dimension (moving forward then backward)
- Detecting a wall using IR sensors

(my original idea was to find a remote control car and hack into it. Realization, toy cars these days go way too fast. I tried to slow down the motors using PWM but it reduced the torque. I ended up buying separate motors; and designing and building car body from scratch)

Step 5 - Creating first car model

- Stepper motors (realization: they were way too slow)
- Prototype failed

Step 6 - Creating second car model

- Used geared down DC motors
- Wall avoidance was not reliable

Step 7 - Fine Tuning hardware

- Reduced size of the model car
- Found perfect place for the sensors

Step 8 - Optimizing software to be used on microcontroller

- Removed all recursive function and replaced with iterative functions
- Removed Numpy module since it does not run on Raspberry Pi Pico
- Removed all bugs in the program

Prototype 1:

- Motors were too fast and not consistent
- Car was too big
- Turning was hard to perfect
- Static friction was too high

Prototype 2:

- Motors did not hold enough torque (needed to be geared down)
- Turning was still hard to perfect
- The size was perfect but the surface area to put my electronics was not large enough

Prototype 3:

- The stepper motors held lots of torque
- The turning was incredibly accurate
- They were too slow
- To get both motors moving at max speed, I needed to use another thread, the other thread can be useful for later

Prototype 4 (Final Design):

- Speed and torque were just right
- Turning was accurate
- Configuration program was working well

Prototype 4 is when I came to the realization that I do not need an electronic wall avoidance system (IR Sensors). It made the execution simpler.

From this entire experience I can conclude that an engineering project is not about the end result, it is about the journey to reach the end result. In an engineering project there are hills and valleys, mistakes and eureka moments. I started this project knowing nothing, I ended up acquiring so much knowledge about this field. I learned about computer algorithms, I even created my own algorithm. I learned how to program a microcontroller, how to put circuits together and work with hardware. I got introduced to Graph Theory and different ways of programming. I enriched my skills of Engineering.

Although this particular model can't be used in the real world, the concepts behind it have huge potential to do great things. What would normally take hours for a human to do, my collected and modified algorithms can do in a few seconds. When people eventually set camp on Mars, they would need help from robots to explore the possibly dangerous environment. This could be in the form of drones mapping the area, or rovers in hard to reach areas.

My next steps include:

- Debugging my computer program so the robot car can find the shortest path in a physical maze.
- Experiment with the 3rd dimension of maze solving
- Consider going back to the automatic wall avoidance system (it was working but was not 100% reliable).

Next steps for robotics equipped with Graph Theory Algorithms are infinite. They might include:

- Equipping a drone with Graph Theory and AI so it can map its surroundings and find the best path in an emergency situation.
- Self driving vehicles shoveling snow off the roadways.
- Robots sorting and placing books.
- The list goes on and on.

[1] Alamri, Shatha & Alshehri, Shuruq & Alshehri, Wejdan & Alamri, Hadeel & Alaklabi, Ahad & Alhmiedat, Tareq. (2021). Autonomous Maze Solving Robotics: Algorithms and Systems. International Journal of Mechanical Engineering and Robotics Research. 10. 668. 10.18178/ijmerr.10.12.668-675.

[2] Zhentao-Lin & JinbaoPeng & DenzelChen (2022, December 29). Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi_Pico [Github - PDF]. Retrieved from https://github.com/Freenove/Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi_Pico

[3] WilliamFiset. (2018, April 1). Depth First Search Algorithm | Graph Theory [Video]. Youtube. https://youtu.be/7fujbpJ0LB4

[4] WilliamFiset. (2018, April 2).Breadth First Search Algorithm | Shortest Path | Graph Theory [Video]. Youtube. https://youtu.be/oDqjPvD54Ss

[5] WilliamFiset. (2018, April 20). Breadth First Search grid shortest path | Graph Theory [Video]. Youtube. https://youtu.be/KiCBXu4P-2Y

[6] Learn DS & Algorithms [Course]. (n.d.). Retrieved from https://www.programiz.com/dsa

[7] Anonymous. (2017). *PROJECT LAB INSTRUCTION MANUAL*. Elenco Electronics.

Thank you Google Search for providing so much knowledge at my fingertips.

Thank you to my Computer Science teacher for mentioning the Depth First Search Algorithm. I searched up how it works, and the ball started rolling from that point on.

Last but not least, thank you to my father for helping with the woodwork required for building the robot car.