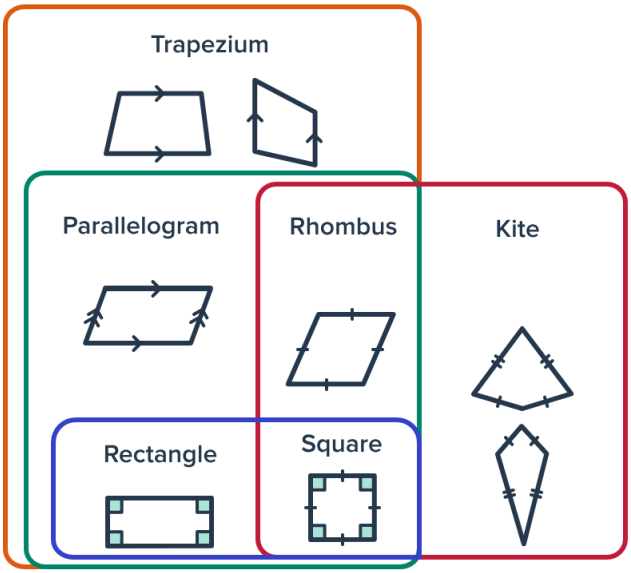
Year 9 CAT Investigation – Mathematical Modeling and Quadrilaterals

INTRODUCTION

In a not-too-distant future, (friendly) bipedal robots will be part of daily life. As they navigate a human-centered environment, they will need to detect affordances such as walls, windows, doors, tables, fridges, dishwashers and other objects with quadrilateral faces. One such robot will likely 'see' quadrilaterals at an angle so that a square tabletop might look like a rhombus, and a rectangular TV might look like a trapezium or a parallelogram. This is due to how squares and rectangles are projected onto retinas and image sensors. Quadrilateral detection is therefore an important area of computer vision research in industry and academia (online-search "Real-time quadrilateral object corner detection algorithm based on deep learning" if interested).

For this investigation, we will imagine that a robot is surveying a visual scene (you've asked it to load the dishwasher) and it is able to detect edges and corners using its neural networks. These corner coordinates will be superimposed on a Cartesian plane and an algorithm will be required to identify the quadrilateral faces that are detected.

You will be given four sets of Cartesian coordinates that represent an abstract quadrilateral. These can be joined together to form some of the *convex* polygons shown below.



As a human with keen perceptual faculties, you can 'just see' which quadrilateral it is. The robot, however, will need your help. You are required to design and implement algorithms that can identify the quadrilateral specifically. For example, given coordinates `(2,2), (-1,-1), (-1,2), (2,-1)`, your algorithms should identify this as a square (at the lowest level of abstraction) even though it's also a trapezium, parallelogram and a kite. Your algorithms should display the reasoning process they followed as they investigated the relevant properties of quadrilateral sides, interior angles and diagonals.

As you go through the design process, make sure to document your progress in presentation slides. The designs in your slides from now on should be in plain English using brief dot-points (with indentation to indicate blocks of instructions) or you can use diagrams/pseudocode/flowcharts if you prefer).

Please note that it is important that you understand all formulae you use in your investigation. It is not recommended that you use vector methods such as the scalar product (to be covered next year) unless you derive them in full in your investigation slides. The same applies to techniques that make use of the tangent addition formula (to be covered in year 11). Essentially, you need to demonstrate a deep understanding of the algorithms and formulae that you use as you design your algorithms.

You will likely need to make assumptions as you engage in design work. Make sure to document them in your slides (*these slides must be saved in the same folder that contains your code*).

Below are properties of quadrilaterals to get you started. As you conduct additional research, if you choose to, you can add to the tables but do not delete any rows.

Side length and angle properties	Trapezium	Kite	Parallelogram	Rectangle	Rhombus	Square
One pair of parallel sides	Yes		Yes	Yes	Yes	Yes
Two pairs of parallel sides			Yes	Yes	Yes	Yes
All sides equal					Yes	Yes
All angles 90 degrees				Yes		Yes
Two pairs of adjacent equal sides		Yes			Yes	Yes

Properties of diagonals	Trapezium*	Kite	Parallelogram	Rectangle	Rhombus	Square
Diagonals are perpendicular		Yes			Yes	Yes
One diagonal bisects the other		Yes	Yes	Yes	Yes	Yes
Both diagonals bisect each other			Yes	Yes	Yes	Yes
One diagonal bisects the angles it passes through		Yes			Yes	Yes
Both diagonals bisect the angles they pass through					Yes	Yes
Diagonals are equal in length				Yes		Yes

* you will need to decide what to do about the trapezium (document this in your slides)

You might think that it is repetitive to use two sets of properties to identify quadrilaterals. Keep in mind that redundancy is a core design feature of machines that operate in the real world in order to have fault tolerance. Usually, the redundant subsystems would be implemented in physically separate circuits as backups of each other.

Represent each quadrilateral in terms of its properties (5 marks)

Once you have the four sets of coordinates, you will use them to ascertain whether each of the properties in the tables above is present. For example, if the only three properties present are; diagonals being perpendicular, one diagonal bisecting another and one diagonal bisecting the angles it passes through, then your algorithm needs to classify this as a kite. You therefore need to somehow represent each quadrilateral in terms of the properties that identify it.

Constructing a flexible representation is an important aspect of Mathematical Modeling. That way, if a new quadrilateral (such as an isosceles trapezium) is added to the list, your design should not need much modification to accommodate it. You should therefore try to think of a more elegant way to identify the shape than using lots of conditional (if) statements. Detail your design decisions in your slides so that you can explain them to your classmates during presentation time. You might find it helpful to come back and think about this section once you have read the rest of these instructions.

Also in your slides, research and explain the concept of abstraction in the context of classification of quadrilaterals.

PART 1: VERTEX COORDINATES ENTERED IN ORDER

For this part of the investigation, the entered coordinates must be interpreted as being the vertices of a quadrilateral, entered in *anticlockwise* order. Thus the vertices entered as `(2,2), (-1,2), (-1,-1)` and `(2,-1)` consecutively could be labelled as A, B, C and D respectively.

This simple mapping allows us to identify the sides (edges) as AB, BC, CD and DA and interior angles (corners) as ABC, BCD, CDA and DAB. We can therefore design algorithms to identify parallel and perpendicular sides, ascertain side lengths and recognize adjacent sides. The mapping also allows us to identify the diagonals as AC and BD and thus inquire if they bisect each other or if they are perpendicular. Finally, we can find out if the diagonals bisect the angles they pass through.

1.1 Input and validation (5 marks)

We will continue to use `(2,2), (-1,-1), (-1,2)` and `(2,-1)` as an example. After asking for vertex coordinates, your algorithm should validate the input. Only comma-separated ordered pairs should be accepted. If an invalid input is detected, your design should respond appropriately. Make sure you do basic checks that ensure that the points can form a quadrilateral *in principle* (repeated points should be rejected, for example). Detail your input validation checks (and any assumptions) in your slides. An example interaction is shown below.

```
Enter comma-separated coordinates in anticlockwise order:
Vertex A: 2,2
Vertex B: -1,y
Error: Vertex B: -1,-1
Vertex C: -1,2
Vertex D: 2,-1
```

After four coordinates are entered, you should store them in a format that makes it easy for you to pass them on to the modules that check for various quadrilateral properties as described below. As an optional feature without any marks allocated, you can use a framework like Pygame and let the user click on a cartesian plane to input the points.

1.2 Algorithm components (20 marks)

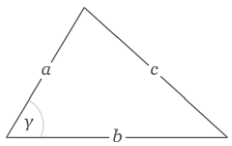
In your presentation slides, detail your design for algorithms that determine the presence of properties listed below based on the four coordinates entered:

- One pair of parallel sides
- Two pairs of parallel sides
- All sides equal
- All angles 90 degrees
- Two pairs of adjacent equal sides
- Perpendicularity of diagonals
- One diagonal bisecting another
- Both diagonals bisecting each other
- One diagonal bisecting the angles it passes through
- Both diagonals bisecting the angles they pass through
- Diagonals being equal in length

The above properties are mandatory, and you can add more if you choose to. If you do, document them in your slides. As you design your algorithms, remember that you must demonstrate your knowledge of year 9 and year 10 mathematical techniques therefore choose your approach carefully in order to maximize marks earned. Simply importing and using functions such as `math.dist()` that do all of your work for you will not meet this requirement.

Note that you can use the formula below to calculate the angle between two sides (the angle gamma must be included by sides a and b and must be opposite side c). It may come handy at some point (the derivation of this formula using right-angle triangle trigonometry is in the year 9 Connect Library, CAT folder).

$$\gamma = \cos^{-1}\left(\frac{a^2+b^2-c^2}{2ab}\right)$$



1.3 Modular algorithm design (15 marks)

You may have noticed that some of these checks are repetitive, meaning that you will need to think about a modular architecture where some core functions are designed first and then reused. For example, since you repeatedly need to work out lengths of lines and diagonals, you should only have one algorithm to calculate and return the length given two sets of coordinates. Document your design in your slides, emphasizing its modular construction.

As your algorithm checks through the properties of quadrilaterals, it should display those that are present as shown in the sample output below. You should then make use of the quadrilateral representation designed in the introduction to finally decide on the *specific* quadrilateral that embodies all of the properties.

```
Side and angle properties for (2,2), (-1,2), (-1,-1) and (2,-1):
One pair of parallel sides
Two pairs of parallel sides
All sides equal
All angles 90 degrees
CONCLUSION: The quadrilateral is a SQUARE

Diagonal properties for (2,2), (-1,2), (-1,-1) and (2,-1)::
One diagonal bisects the other
Both diagonals bisect each other
Diagonals are perpendicular
One diagonal bisects the angles it passes through
Both diagonals bisects the angles they pass through
Diagonals are equal in length
CONCLUSION: The quadrilateral is a SQUARE
```

Remember, your algorithm needs to be able to cope with the quadrilateral in any orientation. If all of your checks are inconclusive and the quadrilateral cannot be identified, your algorithm should report this.

1.4 Optional: Give details for each property (5 marks)

You can give more details in your output as suggested below. You may choose to format the output in a different way as long as the information presented is correct.

```
Side and angle properties (2,2), (-1,2), (-1,-1) and (2,-1):
One pair of parallel sides:          BA, CD
Two pairs of parallel sides:         Also BC, AD
All sides equal:                     AB, BC, CD, DA = 3.0 units
All angles 90 degrees:               DAB, ABC, BCD, CDA
CONCLUSION: The quadrilateral is a SQUARE

Diagonal properties (2,2), (-1,2), (-1,-1) and (2,-1):
One diagonal bisects the other:      AC bisects BD
Both diagonals bisect each other:    Also BD bisects AC
Diagonals are perpendicular:         AC, BD
One diagonal bisects the angles it passes through:        AC bisects BAD, BCD (2 x 45.0 deg)
Both diagonals bisects the angles they pass through:      Also BD bisects ABC, ADC (2 x 45.0 deg)
Diagonals are equal in length:                            AC, BD = 4.2 units
CONCLUSION: The quadrilateral is a SQUARE
```

Using colour to enhance presentation is highly recommended.

PART 2: VERTEX COORDINATES ENTERED IN RANDOM ORDER

This part of the investigation is intended to make your algorithm more flexible. By the time you are done, you will be able to keep your existing design as is, and simply pre-process the coordinates to place them in the required order.

You can imagine that the camera module of our bipedal robot that identifies edges and corners simply stores the coordinates as they are computed (due to a quirk of its neural networks). It then passes them on to a perceptual module that reconstructs the visual scene as it makes sense of the available information.

2.1 Design an algorithm to sort the coordinates appropriately (20 marks)

To simulate the above, you should allow the user to enter coordinates as before but in any order they please (if using Pygame, click in random order, neither clockwise nor anticlockwise). As pointed out earlier, you, a carbon-based lifeform endowed with human cultural learning, can simply plot these coordinates on a Cartesian plane and 'just see' the quadrilateral. Since the algorithm designed in the pervious part does not have this ability, you will need to think of a way to order the coordinates appropriately so that your existing algorithm can make use of them.

As before, most of the marks will be allocated to your design of this algorithm so make sure to include a detailed explanation of the mathematical principles involved in your slides, with diagrams and examples of how it works. You should be able to complete this section using the mathematical knowledge you have learned so far this year and keep in mind you will need to explain this to a random group of your classmates. Part of what you are being assessed on is communication skills – you need to explain your algorithm in a way that your classmates can follow and rate you on clarity.

DOCUMENT YOUR CODING PROCESS AND PROGRESS BY MAKING FREQUENT AND DETAILED COMMITS (30 marks)

Once you complete the design process, you are required to implement it in code using the Python programming language. You are expected to use code comments and functions extensively and there will be marks allocated for these. Consult the year 9 connect notice posted on 6 February 2023 for a refresher on functions (more in in Grok Academy's "Introduction to Programming 2" intermediate course, module 7).

Do NOT use Grok to write code for this investigation. You must use VS Code or PyCharm and you need to have installed Git so that you can document your coding process with frequent commits ('diary entries'). Writing a separate diary of your progress elsewhere other than by 'making commits' as instructed will not be awarded any marks. If any of these instructions are unclear, make sure to contact Mr. Kigodi during term 3 weeks 9 and 10.

Before you start coding, create a new folder called `Y9 CAT Inv2 Name Surname` (with your name and surname), open it in VS Code or PyCharm and initialize a new repository (follow the instructions posted on the year 9 connect on 23 August 2023). Create a `main.py` file to hold your code and *save your design slides into the same folder that contains main.py*. You should then immediately make your first commit with a diary entry saying you have just created a new project with your design slides and a blank file to hold your code.

As you code, debug and test what you have written, and once you have made a notable addition that works (such as a welcome message or a new function), commit it with a detailed description of what you've done. Continue adding code as per your design, and each time you write enough lines to make a notable change, make another commit with a detailed description. If you make a mistake or change your mind about some of your code or design, make the necessary changes and commit again with a description of what you've done. You will not be penalised for this – just make sure to adjust or refine and commit your design slides to reflect any new ideas you come up with as you code (this is normal). The code and the design in your slides must match.

The code at each commit checkpoint needs to be able to run and you should be able to describe the features that the code implements up to that point. Remember that frequent and detailed commits are required to earn the marks in this section (Git records the date and time of each commit). Inserting tens or hundreds of lines of code per commit will not meet the requirement. Essentially, your commit history should make it very clear that you have been gradually working on your investigation and that you understand intimately the code that you are committing. Remember that you will need to show your commit history when presenting to a random group of classmates.

SUBMIT YOUR WORK (COMMIT AND UPLOAD BY THE DUE DATE)

The due date for the slides and the code produced for this investigation is Friday 20 October 2023 (end of week 2 in term 4). Make sure your presentation slides are in the same folder that contains your code and make the final commit by the due date. The code and slides in this final commit are what you will present to your classmates and the timestamp will be used to confirm that the submission was made on time.

In addition to making the final commit, you will also upload the code and slides to an online submission form that will be made available the week the investigation is due. The slides can be uploaded as is, but due to restrictions on the online form, the code (together with libraries you wrote, if any) must be pasted into a special word document first. Full submission instructions will be issued in week 2 of next term.

During week 3, you will do a 12-minute presentation of your computational and algorithmic thinking processes to a random group of peers with teacher supervision. You will use VS Code or PyCharm to present your code and simulations. You will also need to showcase the slides in the final commit documenting the process you went through and the designs you generated. Email isaac.kigodi@education.wa.edu.au or come to the Maths office if you need assistance and remember to `'import this'` to get a refresher of the Zen of Python.

You can research and use the language and concepts in the Computational Thinking Practices chart below as you prepare your presentation.