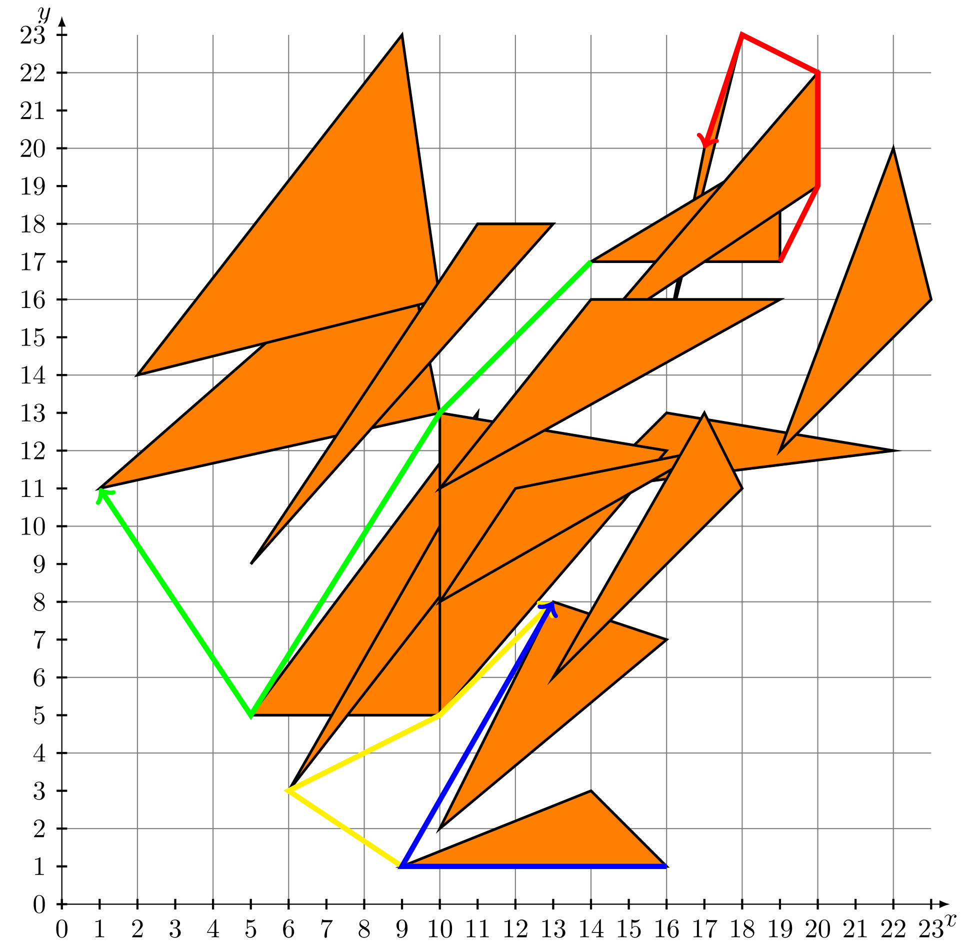
Triangular Navigation Puzzle

Consider an integer grid where points have coordinates (x, y) where $0 \leq x < 24$ and $0 \leq y < 24$, which is populated by 16 triangles as illustrated:



Now consider the problem of travelling from coordinate $(14, 17)$ to coordinate $(1, 11)$, subject to the constraints that the journey is made up of one or more straight-line segments. Moreover, none of the line segments are permitted to overlap with the interior of any of the triangles. The interior of a triangle are those points contained with the triangle which are not on its perimeter. Thus it is permissable for a line segment to pass along the edge of a triangle, or through a vertex pf the triangle, since both are contained within its perimeter.

The three green line segments illustrate one solution to the problem of navigating from $(14, 17)$ to $(1, 11)$. This solution can be described by the following sequence of vertices: $(14, 17)(10, 13)(5, 5)(1, 11)$ The two blue line segments illustrate a solution for navigating from $(16, 1)$ to $(13, 8)$. This solution is $(16, 1)(9, 1)(13, 8)$. Another solution is $(16, 1)(9, 1)(6, 3)(10, 5)(13, 8)$ which is illustrated in yellow. The four red line segments depict a solution for travelling from $(19, 17)$ to $(17, 20)$.

Directions and hints

You will receive an automatic e-mail which contains a pdf document, that is individual to you, which includes a table like that given below to the left:

triangle	vertex$_1$	vertex$_2$	vertex$_3$
0	(9, 1)	(16, 1)	(14, 3)
1	(16, 15)	(18, 23)	(17, 20)
2	(14, 17)	(19, 17)	(19, 20)
3	(5, 5)	(10, 5)	(11, 13)
4	(10, 2)	(16, 7)	(13, 8)
5	(1, 11)	(10, 13)	(9, 18)
6	(2, 14)	(10, 16)	(9, 23)
7	(14, 11)	(22, 12)	(16, 13)
8	(6, 3)	(13, 12)	(10, 10)
9	(14, 15)	(20, 22)	(20, 19)
10	(10, 5)	(16, 12)	(10, 13)
11	(10, 8)	(17, 12)	(12, 11)
12	(19, 12)	(22, 20)	(23, 16)
13	(10, 11)	(14, 16)	(19, 16)
14	(13, 6)	(17, 13)	(18, 11)
15	(5, 9)	(13, 18)	(11, 18)

problem	start	finish
0	(2, 14)	(17, 20)
1	(10, 5)	(14, 3)
2	(19, 16)	(18, 23)
3	(19, 17)	(17, 20)
4	(17, 13)	(20, 19)
5	(1, 11)	(14, 17)
6	(13, 8)	(14, 3)
7	(16, 1)	(13, 8)
8	(5, 9)	(18, 23)
9	(16, 13)	(20, 22)
10	(14, 17)	(1, 11)
11	(17, 20)	(5, 5)

This table will give the vertices of your 16 triangles. The document will also contain 12 navigation problems, as in the table given above to the right. Your mission, should you decide to accept it[1], is to generate solutions for each of the 12 problems. Each solution should be written to a separate output file and placed in the submission directory detailed below. For example, for problem 7, you will need to generate `7.txt` which contains either:

```
(16, 1) (9, 1) (13, 8)
```

or

```
(16, 1) (9, 1) (6, 3) (10, 5) (13, 8)
```

or indeed any other sequence of line segments which connects coordinate $(16, 1)$ to $(13, 8)$.

Note that a single space separates each coordinate and the coordinates are themselves formatted with a single space following the comma. Since part of the assessment will be automatically marked, it is vital that you conform to this format otherwise 0 marks will be awarded automatically. To de-risk formatting, you are advised to check each of your solutions using `FormatChecker.java` which is supplied on the course webpage. You should check your solutions before and after copying them to the submission directory. You are also advised to check your solutions using the diagram provided in the pdf document.

You are free to make use of `Vertex.java` which is available on the webpage. `Vertex` provides the function `linesIntersect(Vertex u1, Vertex u2, Vertex v1, Vertex v2)` for checking whether the line between two vertices `u1` and `u2` intersects with the line between `v1` and `v2`. The class also provides the function `vertexIntersect(Vertex u, Vertex v1, Vertex v2)` to check whether `u` occurs on the line between `v1` and `v2`. Finally, `vertexInterior(Vertex u, Vertex v1, Vertex v2, Vertex v3)` which determines if `u` is an interior point of the triangle defined by the three vertices `v1`, `v2` and `v3`. The class `Vertex` is specifically designed to support this assessment; you are advised to read and carefully consider the comments given in this source code.

[1]This is not meant to imply that the assignment is impossible, or that I will disavow any knowledge of it.

Mark scheme

Twelve marks will be awarded for a correction solution to each problem and a further twelve marks will be awarded for optimal solutions, that is, solutions that minimise the number of intermediate vertices. In addition to placing the 12 solutions in the submission directory, you will also need to submit the source code used to generate your solutions; without providing the code itself no marks be will be awarded for the solutions themselves.

This code must be written in Java and should be clearly (but not overly) documented. All source files need to be marked with your login and surname. The code should include a `main()` method so that it can be executed without using BlueJ. Also, for testing, your code should only use language features supported in Java 1.8. The source code should be submitted a one or more .java files (not a .jar file, nor a .zip file, nor one or more .class files). Code that is clear and succinct will attract the highest marks. Particular attention should be paid to your `next_configs` method.

Deadline

You need to place your solutions in: `/proj/co528c/triangular/`xyz on raptor where xyz is your own personal login. Permissions have been set so that only xyz can access files in the directory xyz. The deadline for submission is *23.55* on the Wednesday of week 18 (February 21st). Section 2.2.1.1 of Annex 9 of the Credit Framework states that, “Academic staff may not accept coursework submitted after the applicable deadline except in concessionary circumstances”.

A Frequently Asked Questions document on Plagiarism and Collaboration is available at: `www.cs.kent.ac.uk/teaching/student/assessment/plagiarism.local`. The work you submit must be your own, except where its original author is clearly referenced. Checks will be run on submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism. When you use other peoples’ material, you must clearly indicate the source of the material.