**The Chinese postman problem**

**Background to project:**

Gilly has a paper round around her local area. Currently she just goes from street to street and often crosses over herself and it takes her a long time. Sometimes the streets she delivers to change. Because of the number of streets and combinations she is unsure of the optimal route and would like me to create a computer program that will detect the ultimate route for her.

To do this I will use the Chinese postman problem.

The Chinese postman problem is the following:

* A postman must deliver post to every street on his route
* He must go down every street at least once
* He must start and finish at the same place

The objective is to make sure Gilly does delivers all her papers to all houses while taking the shortest path to save the most time.

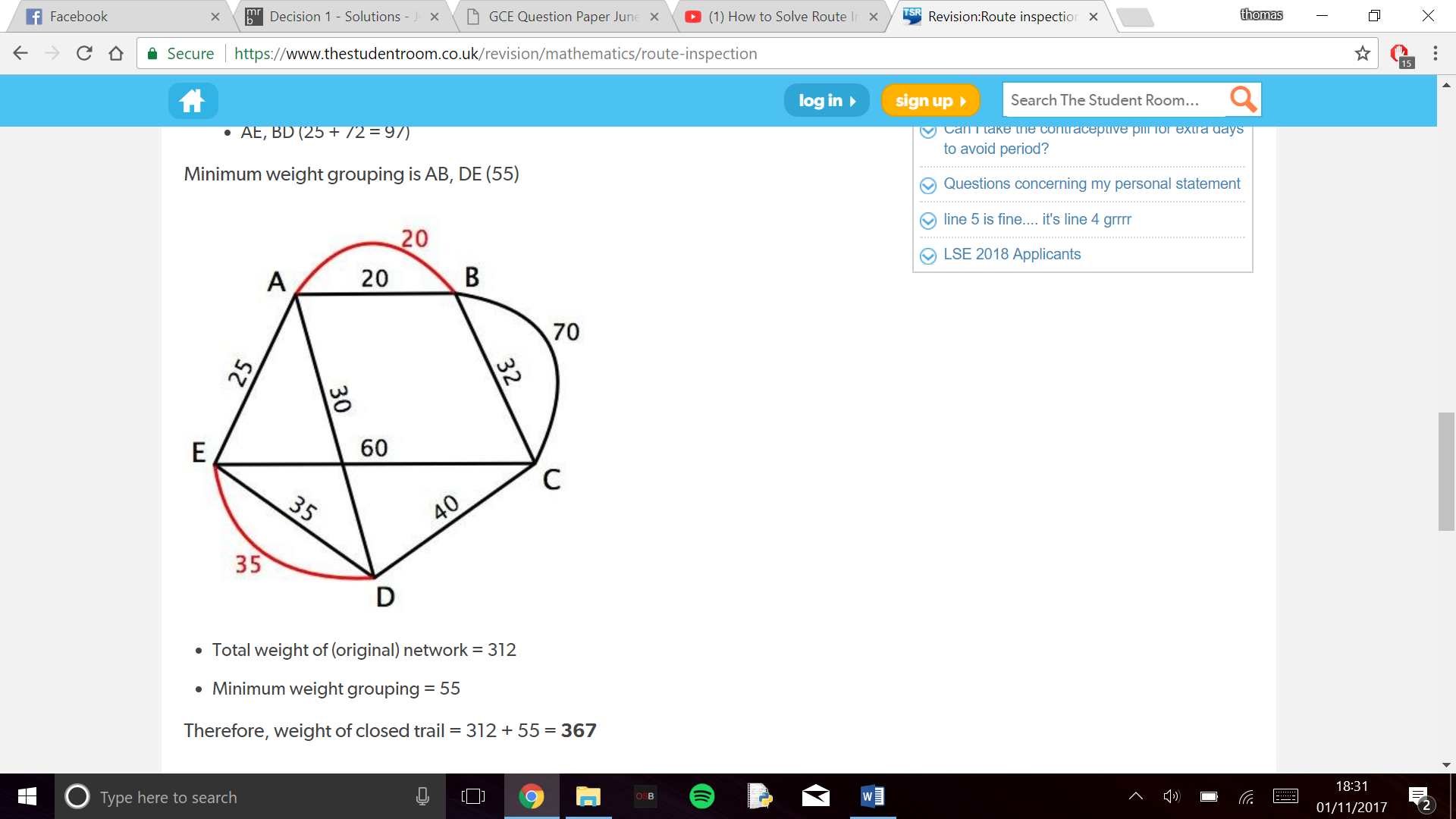
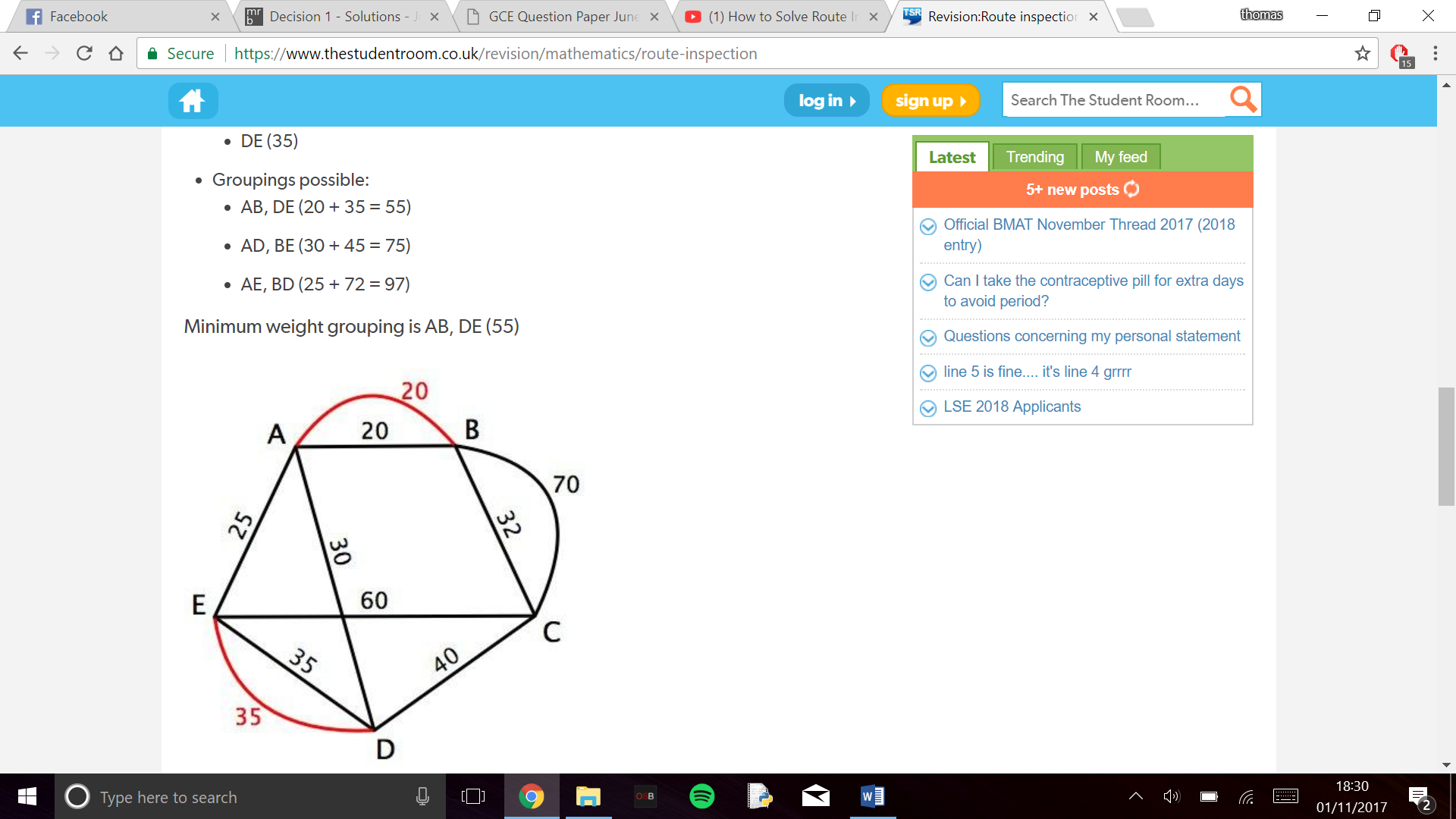
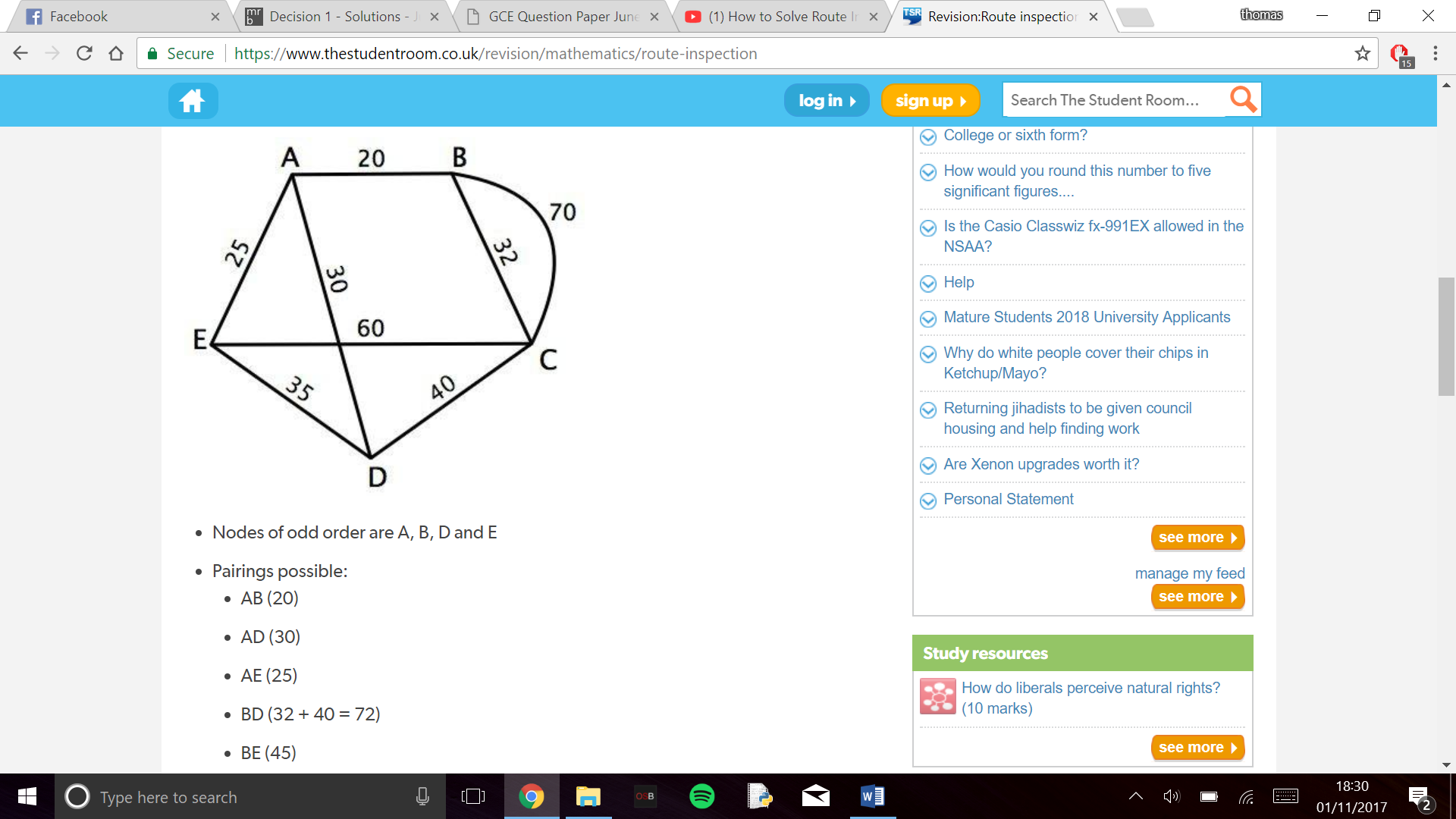
**My solution:**

The general algorithm to find the shortest path is as follows:

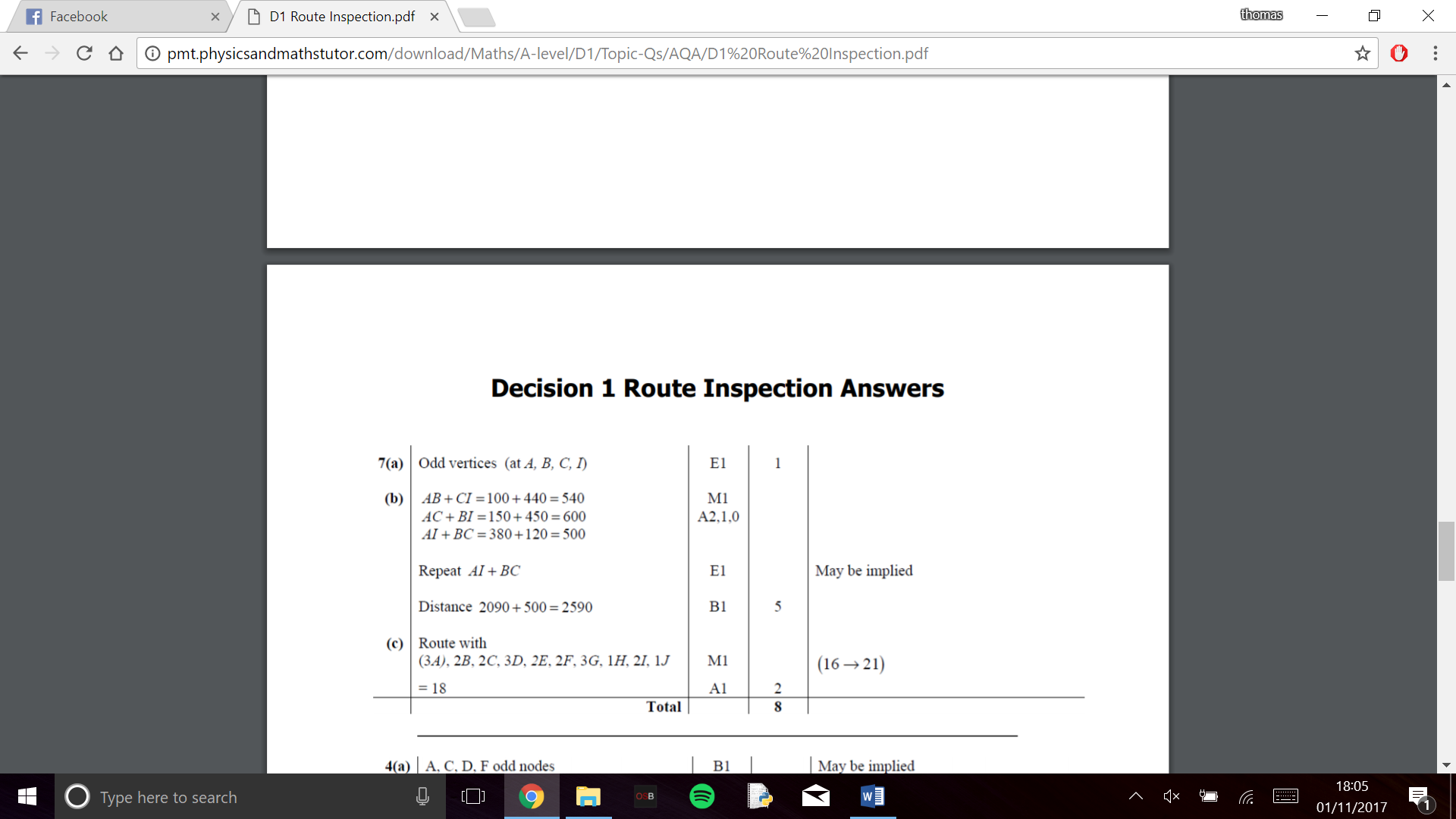
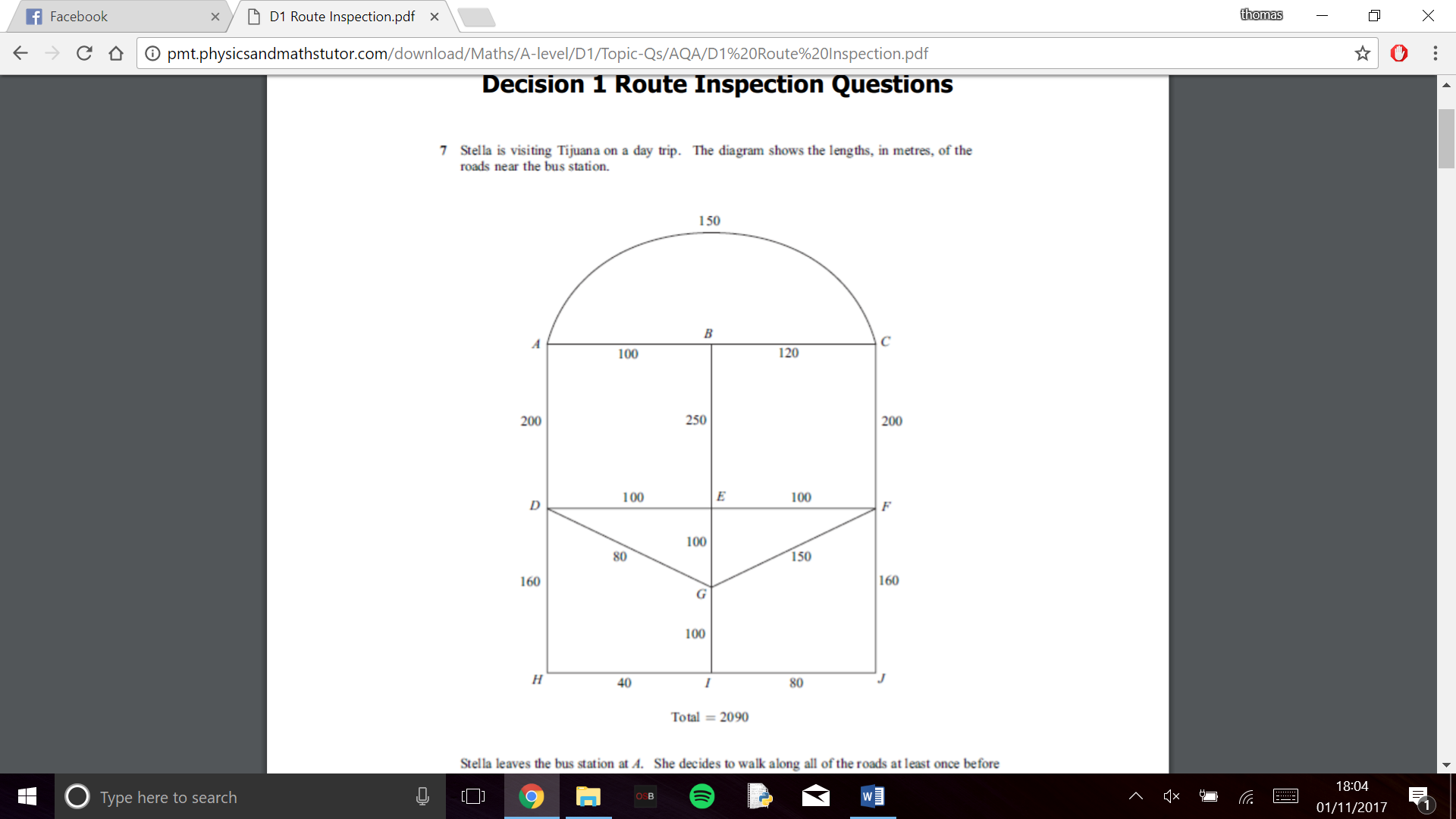
* Identify any vertices with odd valency
* Consider all possible complete pairings of these vertices
* Select the complete pairing that has the least sum
* Add a repeat for the arcs indicated by this pairing to the network

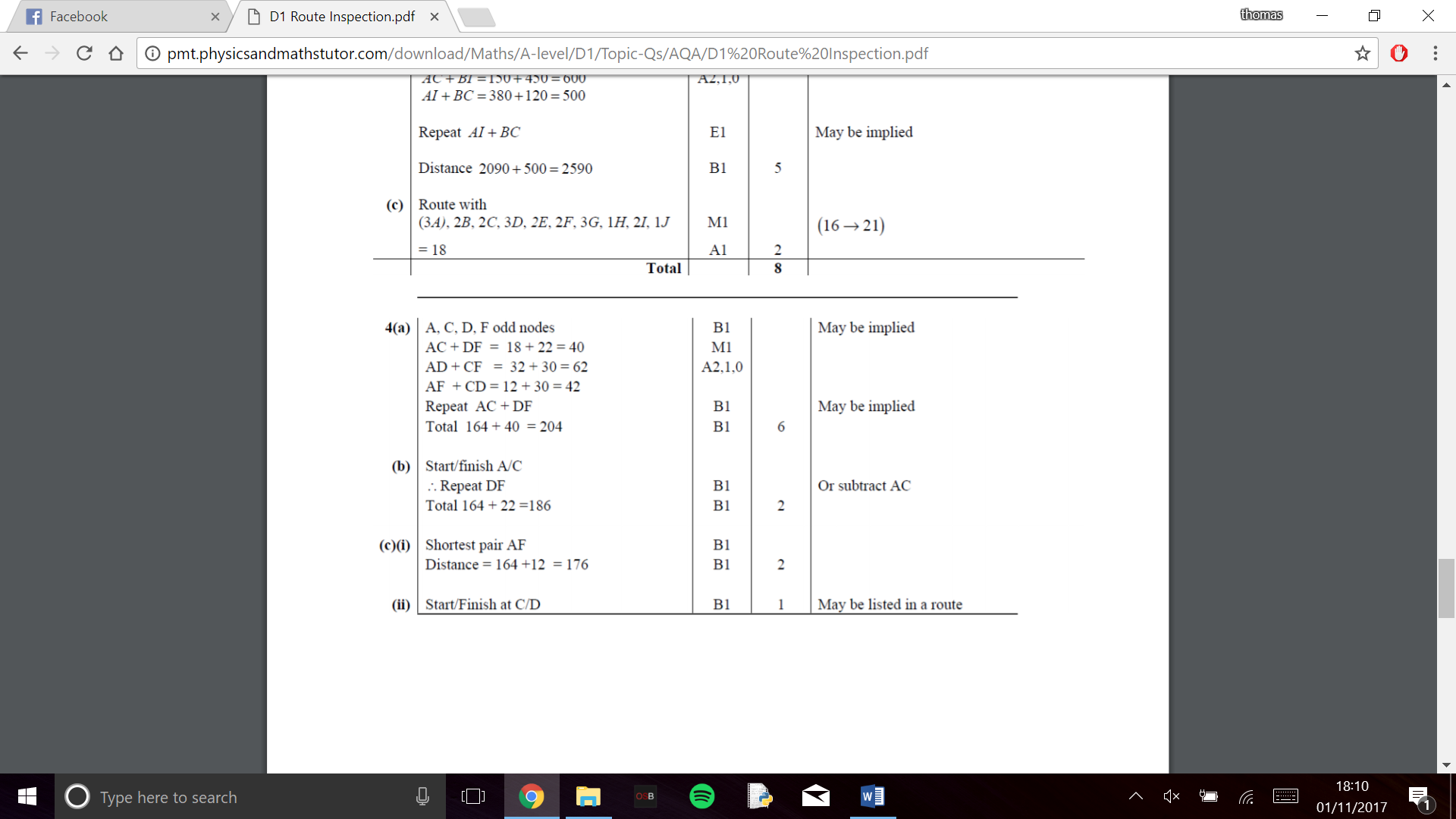
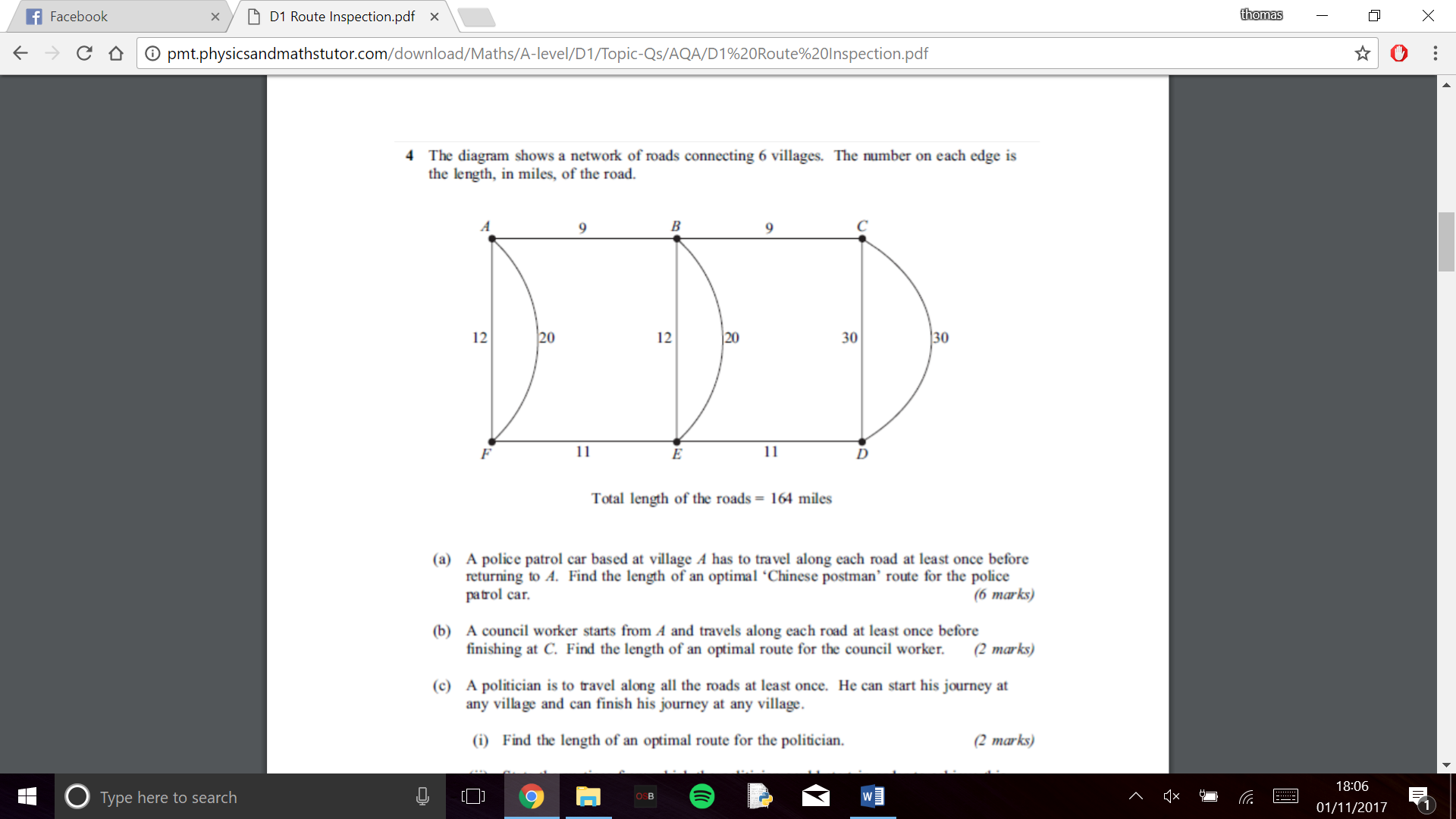
The algorithm works as follows:

Worked example:



Done examples:



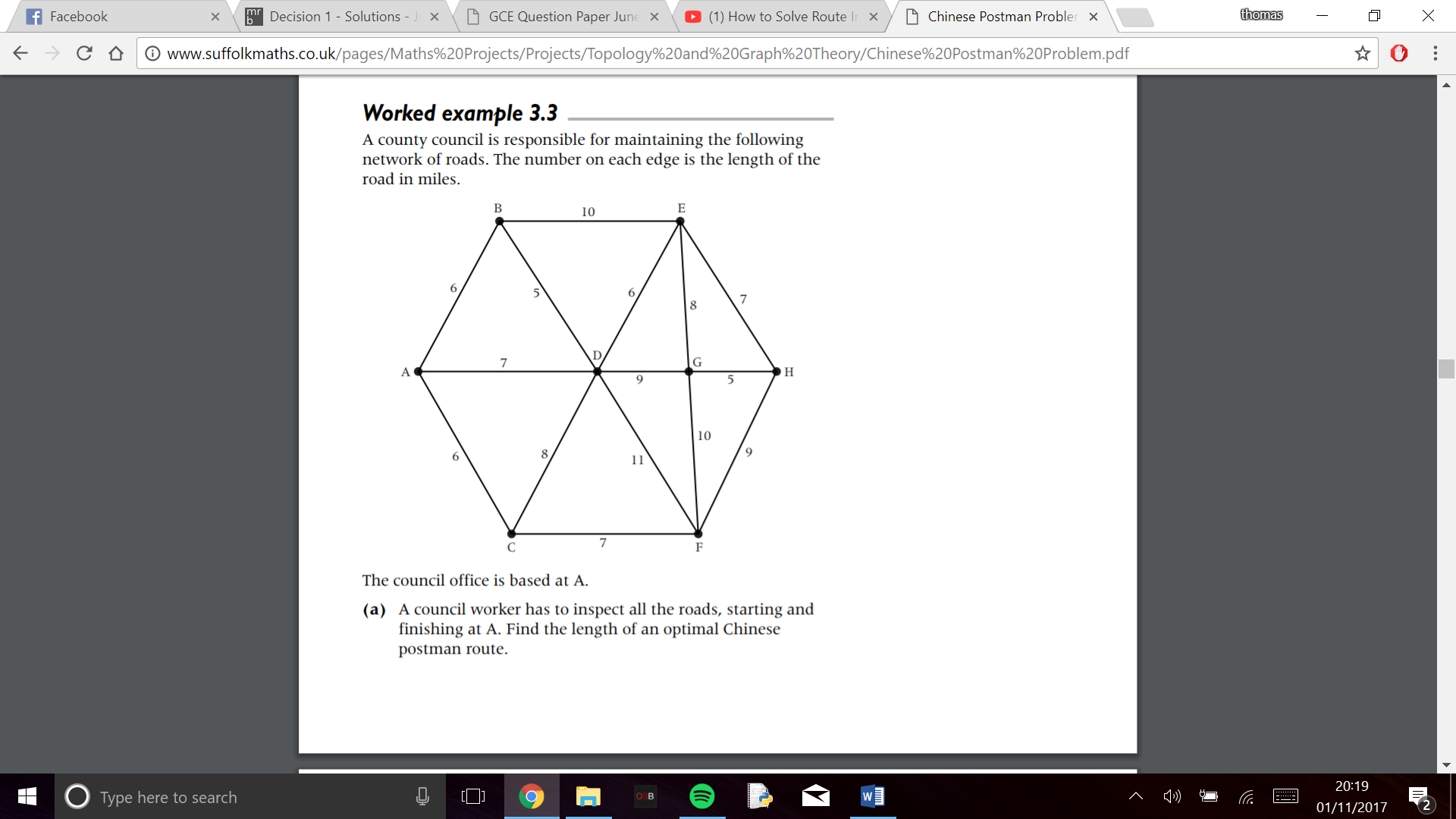
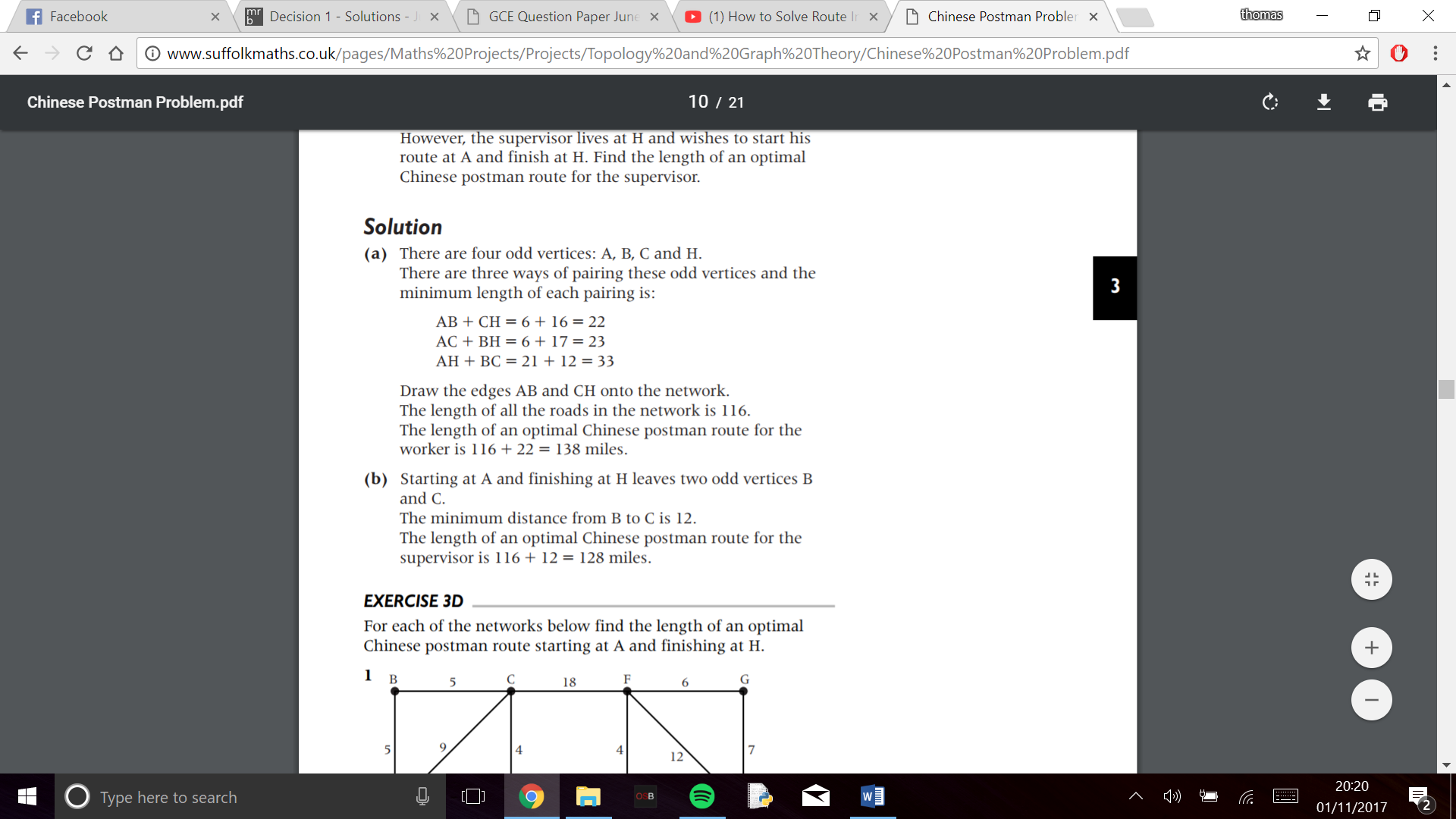


Due to the enormus amount of combos that can vary for different reasons there is no one algorithm to calculate. However, you can use three different rules (for different circumstances) to solve this problem.

If your start and finish points happen to both be odd verticies you can do the following:

From your odd calculations find the one where the start and stop are paired. You no longer have to add them as they do not need to be repeated. The shortest distance is now the sum of the rest of the repeated routes.

The following example shows if the start and finish are at A and H respecivly

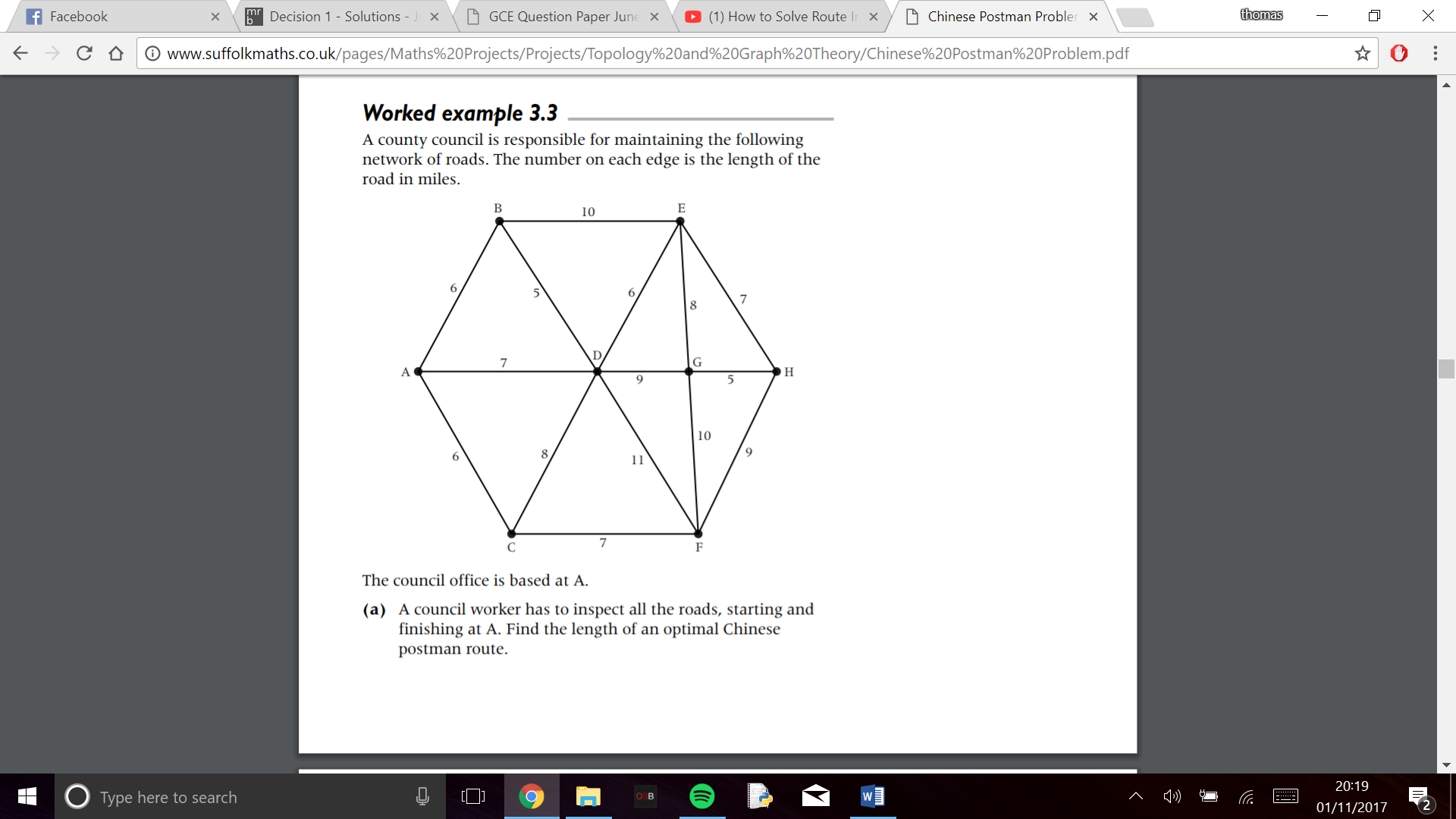
If the start or stop (but not both) are on a odd verticie you can do the following:

You must use a new algorithm called dijkstras which goes as follows:

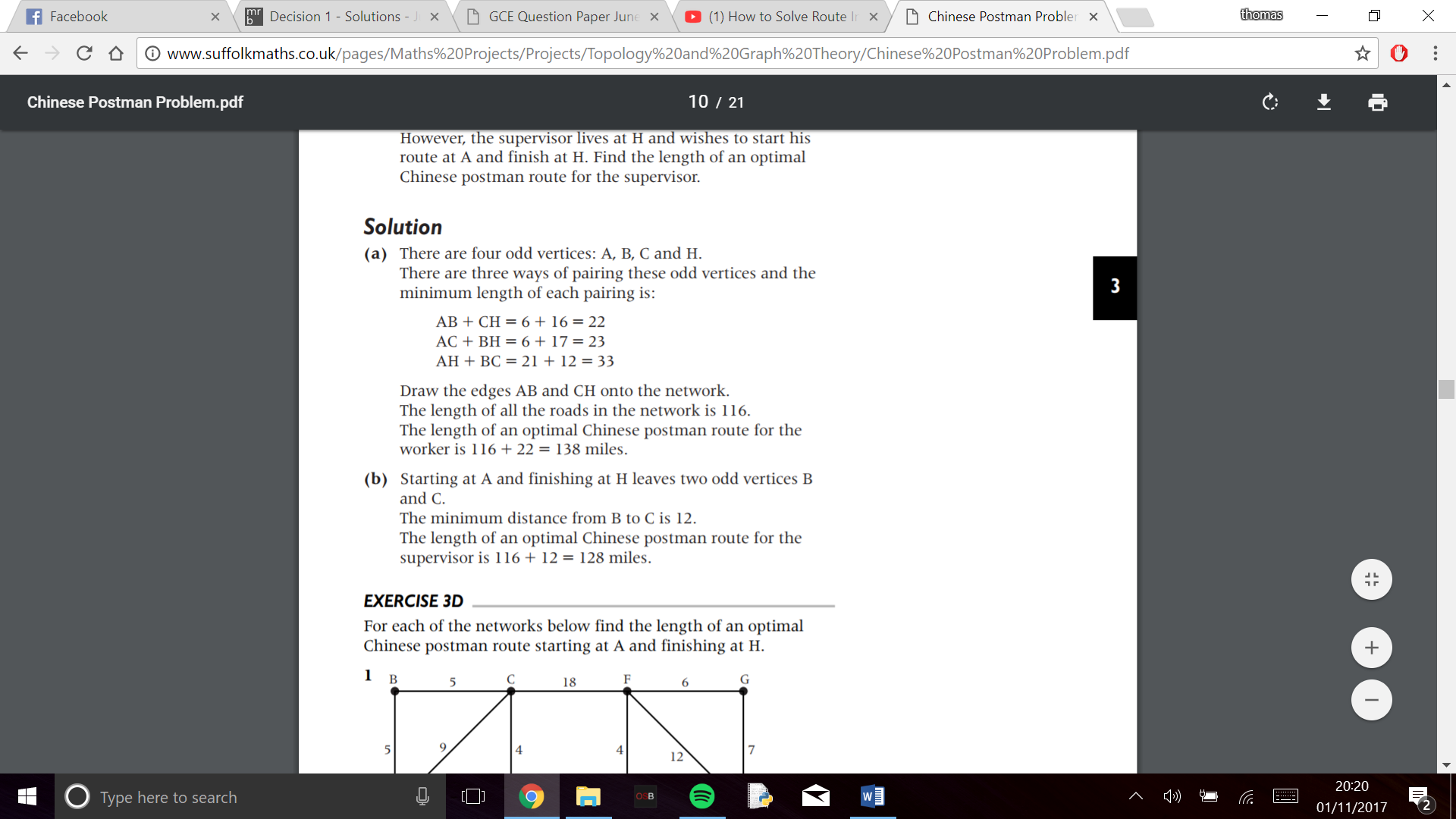
* Use start vertex (S) and final vertex (0)
* Record a working variable at every vertex, Y that is directly connected to the vertex, X, that has just recived its final label.
  + Working value of Y = final value at X + weight of acr XY
  + If there is already a working value at Y, it is only replaced if the new value is smaller
  + Once a vertex has a final lable it is not revisited and its working values are no longer considered
* Look at the working values at all vertices without final lables. Select the smallest working value. This now becomes the final label at the vertex. (If two verticies have the same smallest working value either may be given as its final lable first)
* Repeat steps 2 and 3 until the destination vertex, T, recives its final label
* To find the shortest path, trace back from T to S. given that B already lies on the route, include arc AB whenever final label of B – final lable of A = weight of arc AB

We will then have to use this algorithm to track the distance from the start or finish (which ever is not on an odd vertice) to each odd vertice that connects to the (start or finish).

We will use the example from before and say that the new start is E and the finish is A:



We know that the 3 possible cominations are:



So we must use dijkstras from E to B,C and H

Following the algorithm we find the follwing

EB = 10

EC = 14

EH = 7

If we eliminate the A to corrisponding vertice from each combination and add the value from dijkstras we get the following values

AB = 10(EB) + 16(CH) = 26

AC = 14(EC) + 17(BH) = 31

AH = 7(EH) + 12(BC) = 19

So we find that the minumu is 19 + 116 = 135

If neither the start or stop points are an odd verticie you can do the following:

We must use dijkstras for both the start and finish and examine each combo to find the shortest distance like before

**Hardware investigation:**

The computer that runs the computer must have a decent amount of ram due to dijkstras algorithm being recursive and the sheer amount of data being stored the computer may crash with too manny streets inputed.

The rest of the spec dosent matter though if a drag and drop funtion is added to points on the map a touch screen may speed up the process of producing a route.

**Software investigation:**

I will be using python for the following reasons:

* Use of 2D and 3D arrays to map routes
* Ability to store routes and maps into documents
* Can use pygame to create GUI
* Can have an interactive and editable GUI
* Easy to use and already understood language

**My solution:**

The objective of my code would be as follows:

* Allow people to plan a route they are going
* Map the route for them
* Calculate the shortest distance for them and show possible routes they may take to achive it saving them time
* I will use python to calculate and store verticies and paths inputed by the program
* I will use pygame to create an interface to map the paths created by the user

**Prospective users:**

Along with gilly people who have to deliver panflets to ceartin areas such as other people with paper rounds will use this to save them time and be more efficent with their delivering time. Instead of having to go down routes 3, 4 or 5 times to finish their trip they can more efficently calculate their route. They also don’t have to sit down and go over the thousands of combinations as the computer can calculate it instantly.

**Gillys Objectives:**

* Gilly has identified that sometimes she must go down different routes in different weeks so she would like preset routes she can save and add to if new streets are added or taken away
* Gilly has identified that often she goes to a friends house or straight to school after her paper round and so would like the start and finish points to be different
* Gilly finds it hard to follow text on a screen and would like a visual representation if possible to make it easier for her to understand and use.

**Personal Objectives:**

Must:

* Allow the user to input any number of position with any amount of connections to other positions
* Calculate the shortest path starting and finishing at the same point
* Display the user a potential path that will be shortest for them at the same start and finish point
* To be able to save shortest paths found in a file

Should:

* Allow the user to input different start and finish points
* Display and calculate a shortest route between a different start and stop point for the user
* Be able to save routes in a database to be later editied
* Have a GUI interface where it shows the points and verticies that the user has inputed

Could:

* Have a GUI interface where useres can drag and drop points and create distanced connections to other points for easier use
* Allow useres to input “no go roads” so there may be a connection between paths but the user only wants to go a certin distance and then turn around

**Transcript:**

Tom(T) Gilly(G)

T: Please describe the situation and what you would like the program to do.

G: I do a paper round twice a week my route often changes but is always in the same area. I often spend a long time going all the way back over myself for one house I would like the program to calculate the best route for me to save time.

T: Is there anything you would like the program to do?

G: I would like to be able to come back to what I was doing and pick up where I left of as sometimes my route only changes by one or two roads. Sometimes I finish at a friends house or school so I would like to be able to factor that into the decision of the route.

T: Do you have any personal needs of the code?

G: I find text quite hard to read and follow so if their was a drag and drop or some sort of visual aid that would help me an awful lot

T: Is there any special cases the code may need to take into effect?

G: The only one I can think of is sometimes a road may be closed due to festivals or road workings and sometimes I only need to go halfway up a road but it leads onto other roads I need to go on so it may be quicker to turn round or may be quicker to move onto the road.