Visualizing Graph Algorithms User Guide

Welcome:

I would first like to thank you for taking an interest in my project, and I hope you find it helpful in your journey of learning graphing algorithms. **DISCLAIMER:** When you are designing your graph, the screen does flicker or look glitchy. This is because it is redrawing everything to the screen after any change. (I apologize, though if I had more time, I would have a more efficient and less flickering method.)

Note: For some of you, a lot of the below information may not be as helpful as it would be for others. However, I do ask everyone to read the section: **Get Up To Date Program** found on page 12. If you don't read this section, you may run into an issue with the file. Hope you enjoy my program.

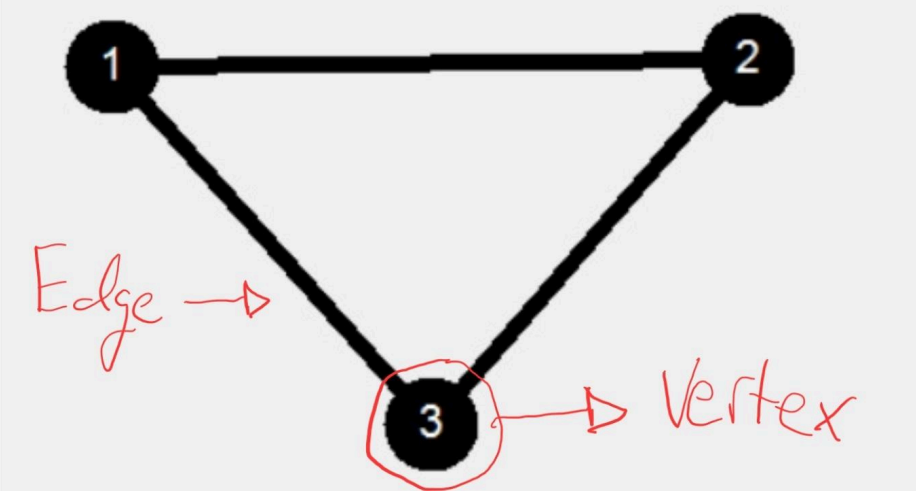
What You Need:

- Computer or Laptop
- Visual Studio 2022 *(older versions may work, but unsure)*
- Updated project zip file (instructions below).
- A basic understanding of what a graph is

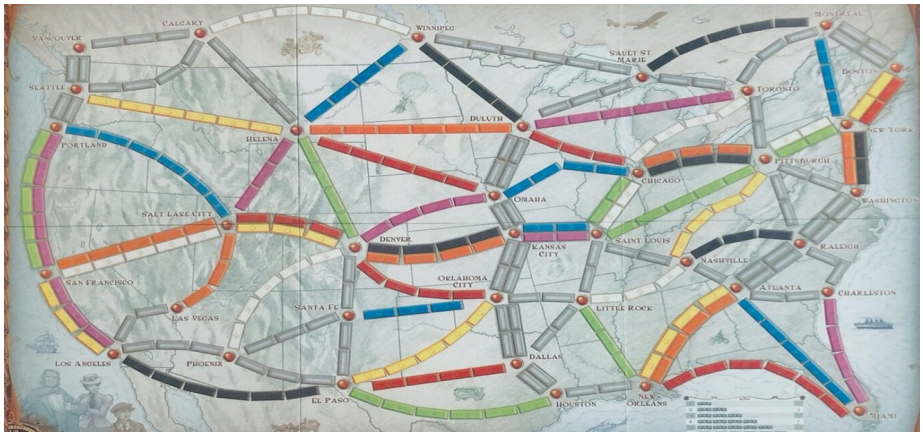
What Is a Graph in Computer Science?

- This section will aim to teach you about what a graph is in computer science. If you already have an understanding of what a graph is, you may skip this section, though I highly recommend reading as it will help you later when using the tools.

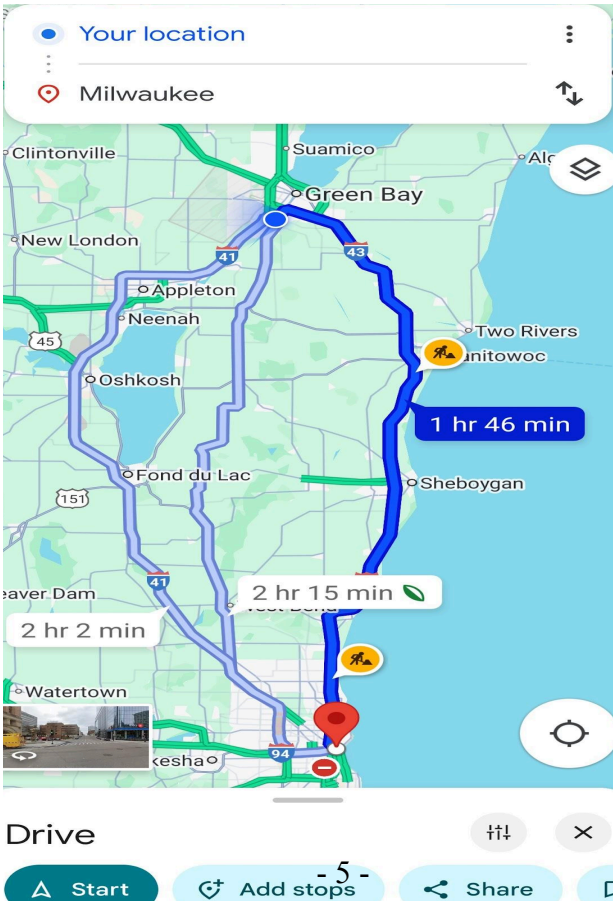
- In computer science, we use a graph to display modal networks like maps, relationships, data flow etc.
    - For example, in our world today, cities are connected by roads. The cities are points,s or what we call (nodes or vertices), and the roads are connections between two vertices or nodes, creating an (edges).
- To further elaborate, in computer science, a graph is made up of the following **(Note:** Below this will be an image of what a vertex and edge look like for my project):
    - Nodes (also called vertices): These are the items or locations (like cities, people, or stations), and are typically represented as circles on the graphs.
    - Edges: These show how nodes/vertices are connected (like roads, friendships, or train routes in the modern world).



- Another real-world example would be the board game Ticket to Ride:
    - The cities on the map are vertices/nodes.
    - The train routes connecting them are edges.
    - So if you're traveling from Seattle to New York, you're moving across the board, or in this case, a graph.
- Meaning, there are multiple paths to go, however, an algorithm like Dijkstra's can use it to find the shortest path from Seattle to New York. Below is an example of the Ticket to Ride board (Days of Wonder, Inc):



- Another example of a graph is Google Maps (and similar map apps). Destinations provide you with a from and to location mark. Google Maps gives you multiple paths to get there, some of which are the shortest in time, some of which are the shortest in miles, and some of which are just alternating paths.
- Each step within that route, i.e., each road you turn onto, can be viewed as a step within an algorithm to find the best path.
- Depending on what algorithm you select will determine what the optimal path to take is. The image below shows 3 routes from point A, which is Saint Norbert College (my current location), and point B, which is Milwaukee. Each path is very different from the others. Now you could have many points, such as Saint Norbert College in Milwaukee, then to Janesville, then to Madison, and so on.

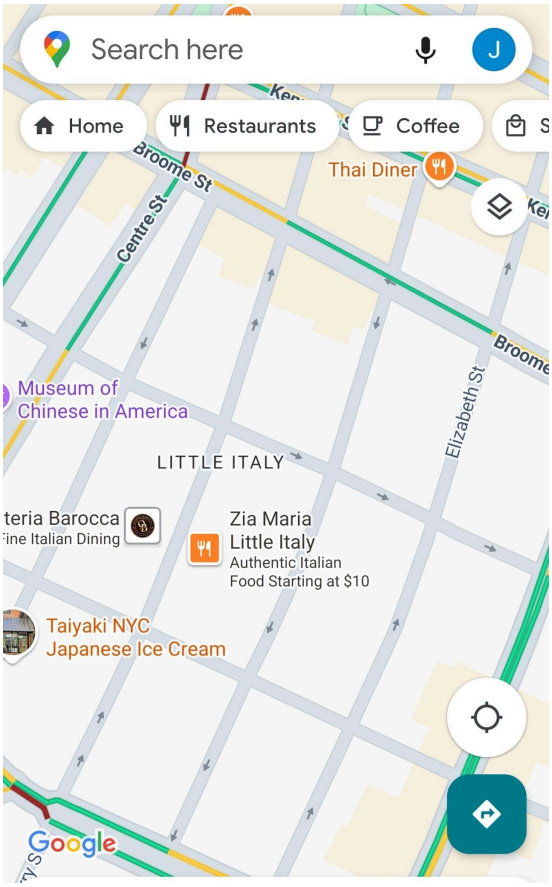


Types of Graphs:

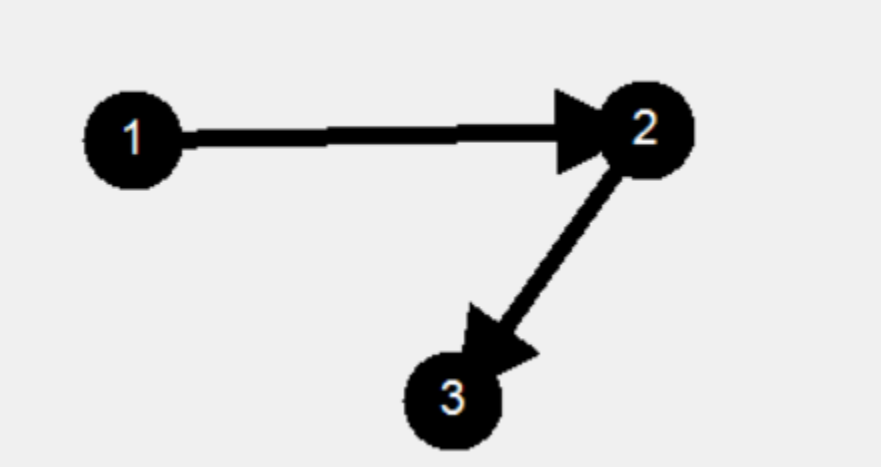
- There are different types of graphs that I will explain and talk about. There are:
  - Directed
  - Undirected
  - Weighted
  - Unweighted

Directed Graphs:

- This type of graph depicts a specific direction in which you can take from one vertex or point to another along the path or edge. Think of this like a road. In the United States, every car drives on the right side of the road. You aren't allowed to drive on the left side of the road, seeing it is highly illegal and dangerous.
- This also helps with the flow of traffic, as if you have cars going in 2 different directions on the same side of the road, it will be hectic, chaotic, and very slow.
- By funneling it into one direction, it allows you to control the flow of traffic on a specific road. In computer science, we have directed edges or paths, which help with data flow and ensure information is only passed in one direction.
- A perfect real-world example of this is a one-directional road, as depicted in the Google Maps photo below and the picture of a one-way sign.

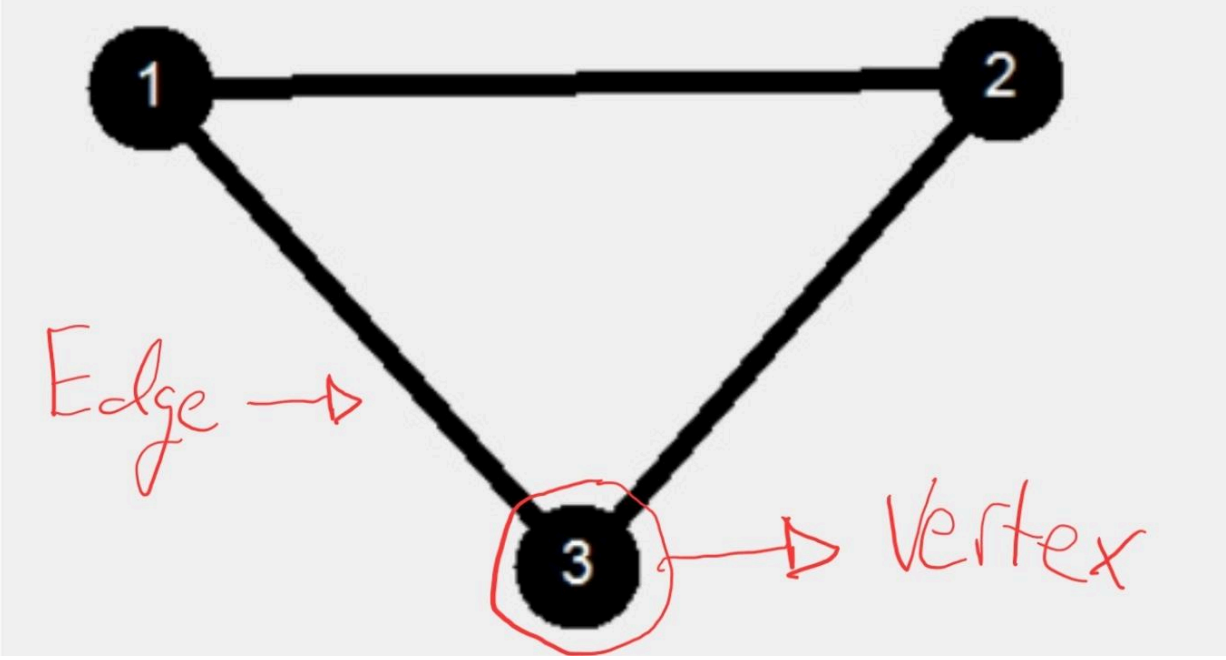


- Below is an example of what it would look like for my project. Vertex 1 has a path to vertex 2, and vertex 2 has a path to vertex 3, but vertex 2 does not have a path to vertex 1.



Undirected Graphs

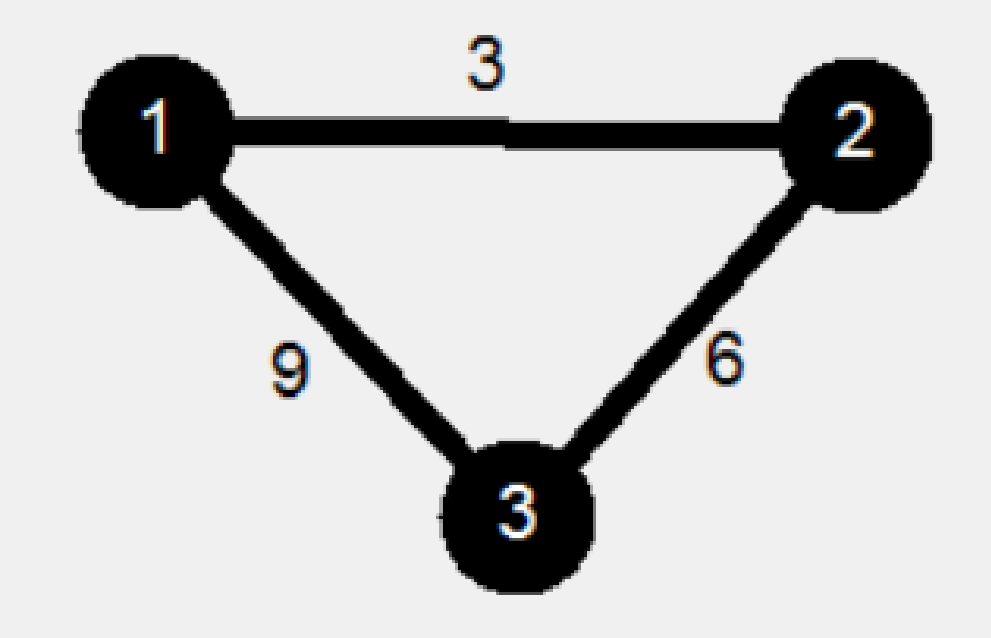
- No set direction. You can go back and forth on a connection.
- Think of a two-way street or a friendship between two people.
- For example the below image I used earlier, vertex 1 is connected to vertex 2, vertex 1 is connected to vertex 3, and vertex 2 is also connected to vertex 1 and is connected to vertex 3. Vertex 3 is also connected to vertices 1 and 3 too.



Weighted Graphs

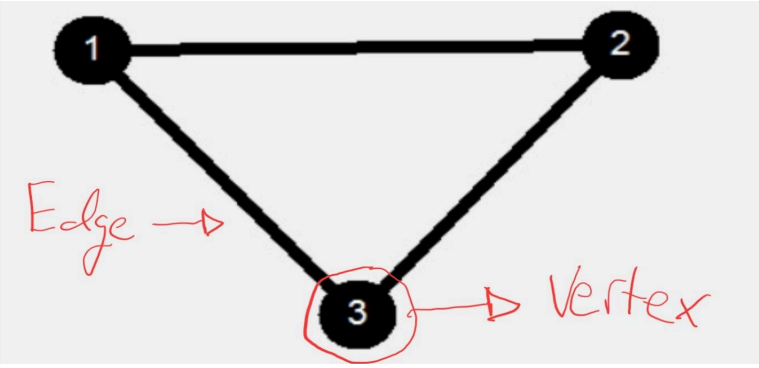
- What a weighted graph means is that each edge has its own weight, which is the cost associated with taking that path.
- This weight can represent distance, time, cost, or any other metric.

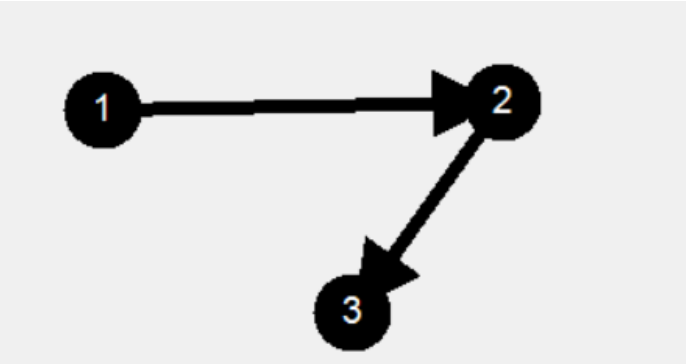
- For example, using the same Google Maps picture from above, there are 3 listed paths to get from Saint Norbert College to Milwaukee. You could categorize the weights as 106 minutes (1 hour and 46 minutes path), 135 minutes (2 hours and 15 minutes path), and 122 minutes or (2 hours and 2 minutes path). 104 minutes is the shortest path time-wise.
- The above Google Map picture is an example of a weighted graph (due to having a cost of time or miles i.e., distance needed to travel), and a weighted graph example from my program is shown below.



<u>Unweighted Graphs</u>

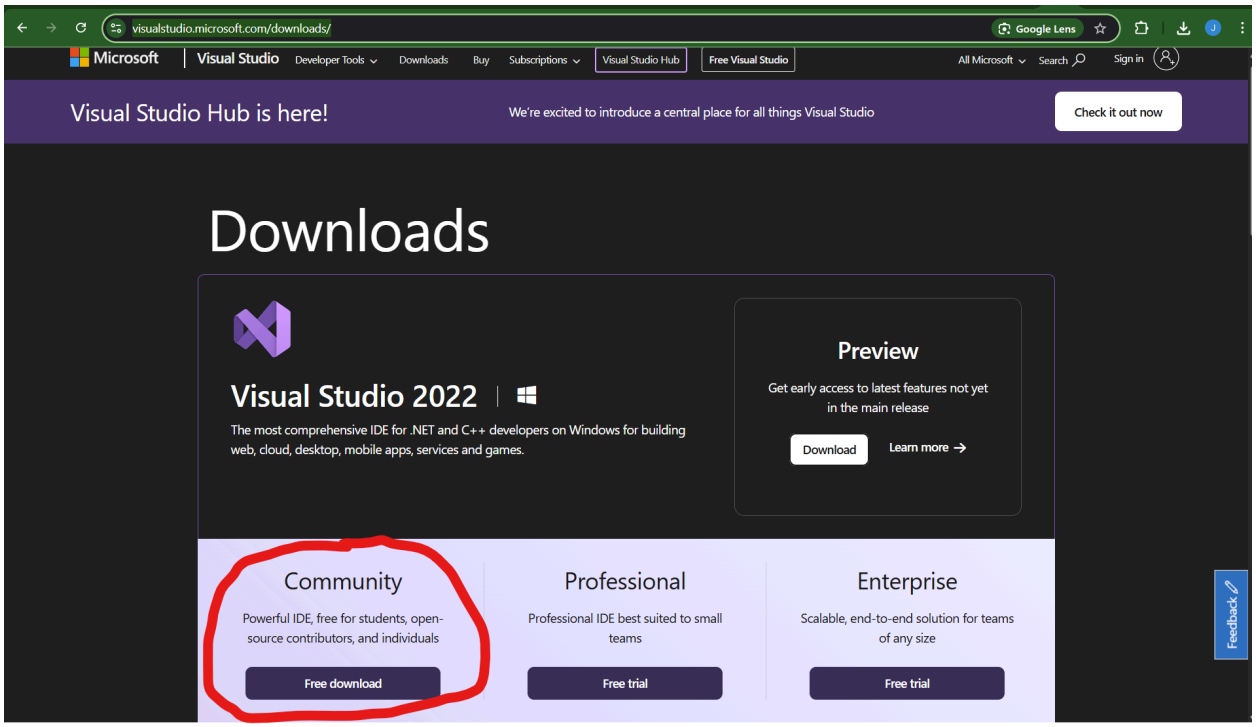
- All edges are treated equally, meaning there is no edge that has a special cost or weight associated with it.
- You can only count the number of steps between nodes, or basically use it when you only care about the shortest number of connections, not distance or cost.
- Below are two examples of an unweighted graph, one is directed and the other is undirected. **Note:** Weighted graphs can also be directed and undirected.





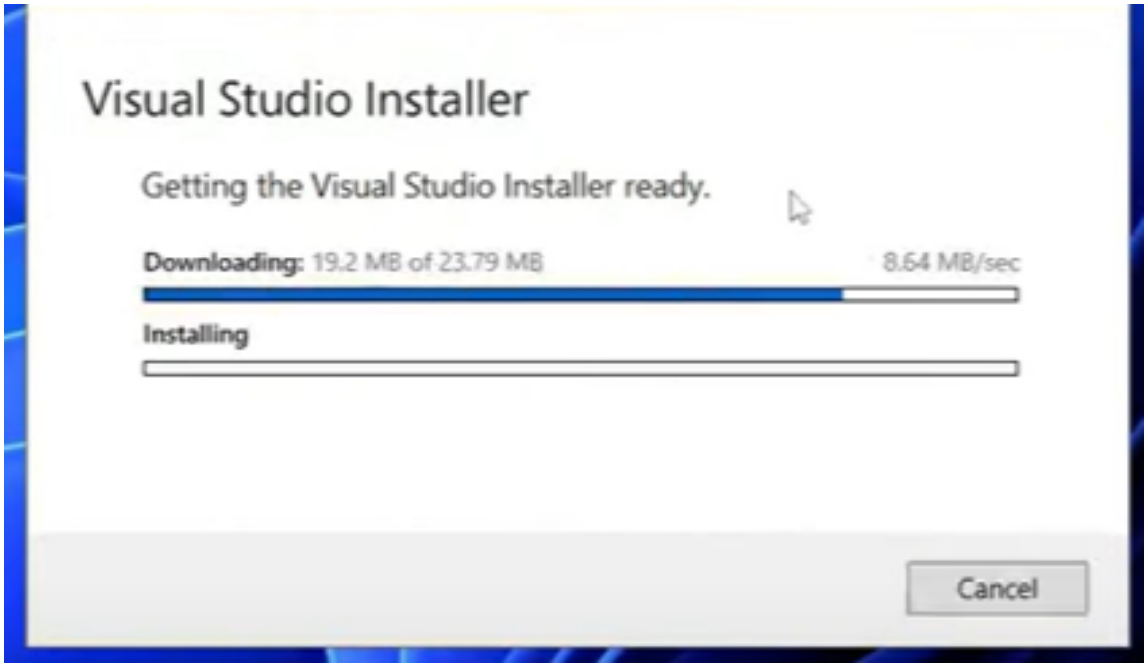
If You Don't Have Visual Studio:

- To reiterate, you will need a computer with Visual Studio installed. (I used version 2022, you may be able to use 2019 or 2017, it may prompt you to update the file, which it will handle)
- To download Visual Studio 2022, you can search for Visual Studio 2022 download, and it should be the first link. Or use this URL: https://visualstudio.microsoft.com/downloads/
- Next, you want to download the Community Free Version shown below:

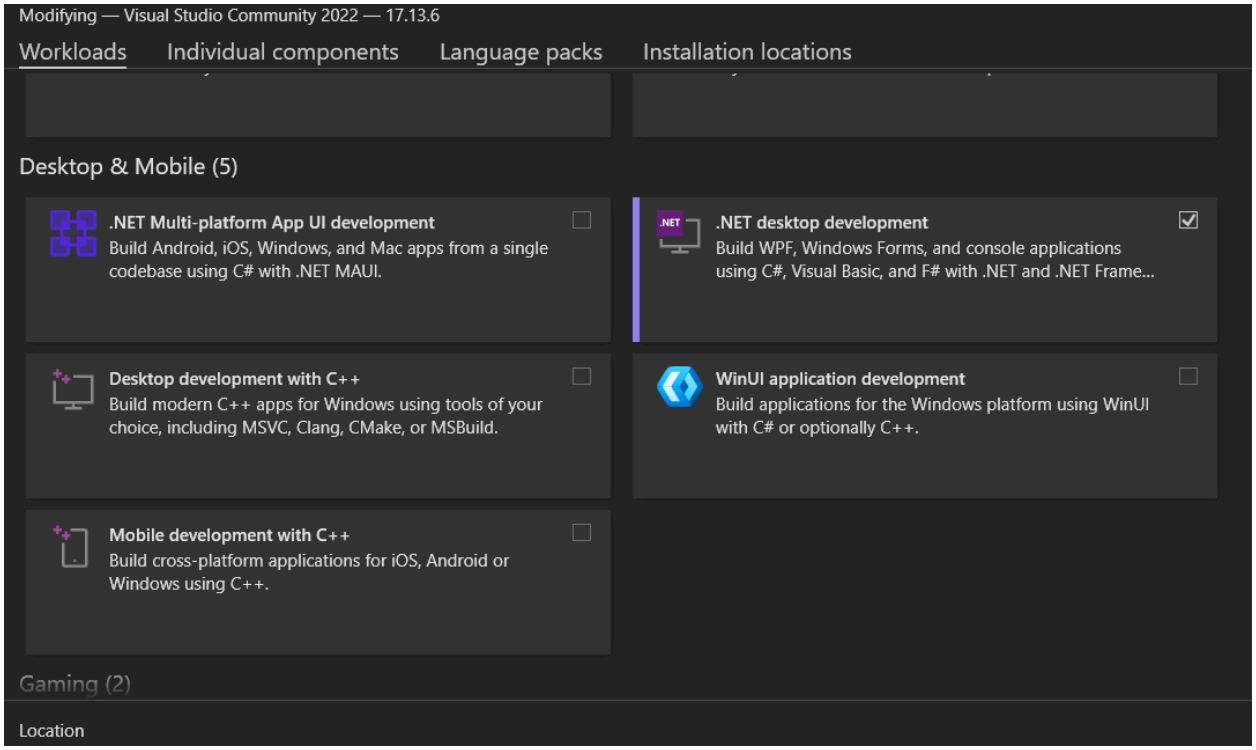


- You can download it anywhere on your computer; it doesn't really matter.
- Once downloaded, you need to open the file. You may be prompted with, Would you like Visual Studios to make changes, Click allow.
    - Clicking "Allow" is safe, and this doesn't give Visual Studio control of your system. It will simply allow Visual Studio to use the resources it needs to execute algorithms and animations.
    - Without this permission, the application won't be able to run.
- Next, we need to follow the prompts as needed to set up. The first is shown below, and you want to click **Continue**.
    - Please note this could take several minutes, maybe even longer depending on network speed to download.

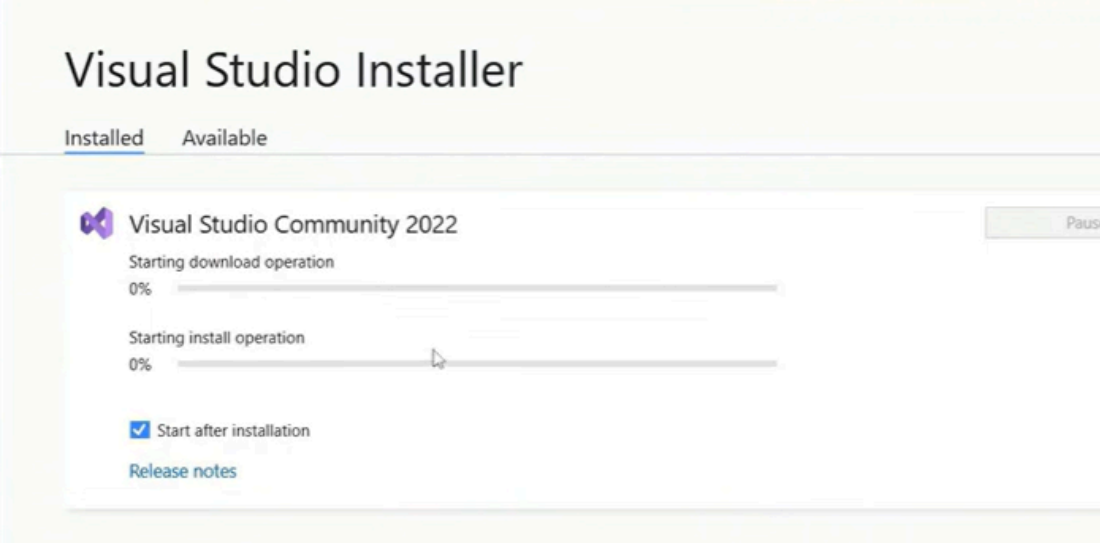
- After that, you should see the screen below:



- The screen below will then show up. You need to make sure you select **.NET desktop development** as selected and shown below.



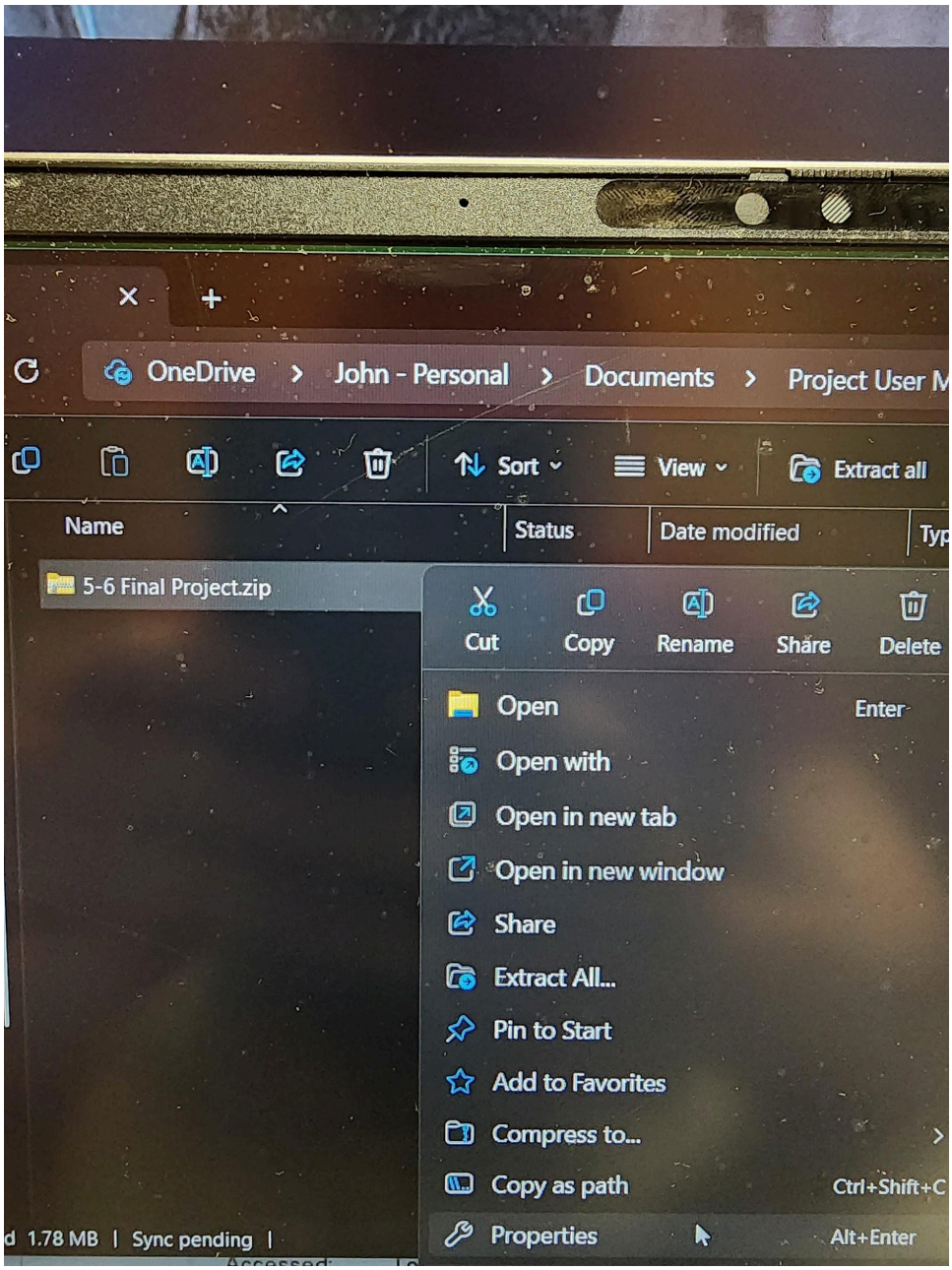
- Once you do that, you want to click install in the bottom right corner. This should then bring you to the screen below:



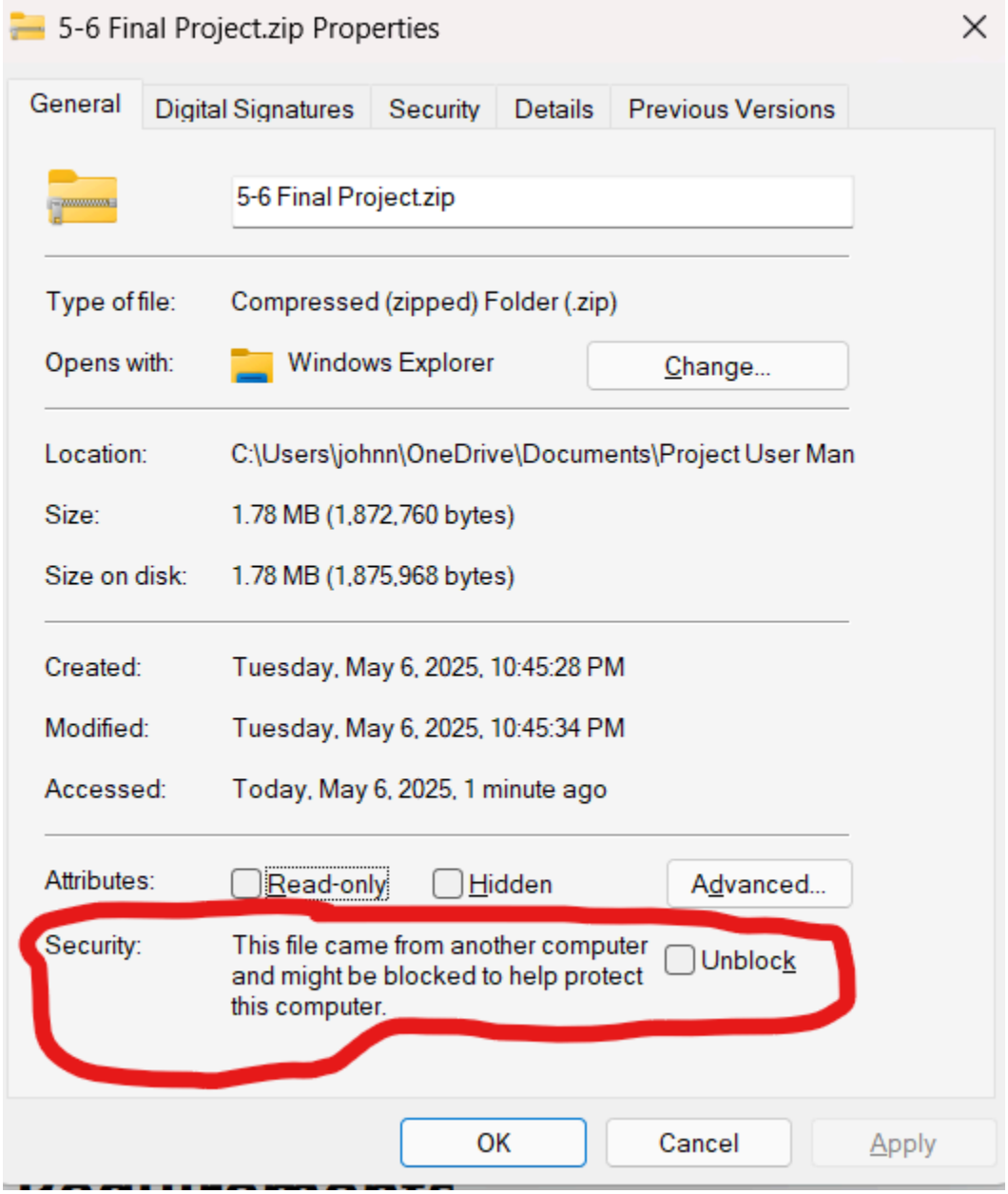
- Once you are done installing, you should have the following screen. **NOTE: Nothing more needs to be done after this step in regards to downloading Visual Studio; the rest is optional for now.** From here, you can choose to launch, log in, and create an account, or skip. You can also choose what theme you'd like, such as light or dark mode (Note you may need to do this later on as well, but I am not 100% sure).

Get Up To Date Program File:

- Now you will need the most recent version of my project, which can be downloaded from my website under the project section, and my last blog post as well. Here is the URL: https://compsci04.snc.edu/cs460/2025/johnolson/website/project.html
- Then you want to click the most recent uploaded file (which will be shown) and click the download button. You may put this file anywhere. **NOTE: DO NOT Extract the file yet, just download and follow further steps.**
- Once downloaded, you will need to **right-click** on the file, navigate to **properties** as shown below, or click on the file and hit **ALT+Enter**:



- Then you will be brought to a similar screen shown below. If you look down at the bottom under securities (shown in the red circle), if you have a similar message as I do below, you're going to want to **<u>select Unblock</u>**, then **Ok or Apply.**

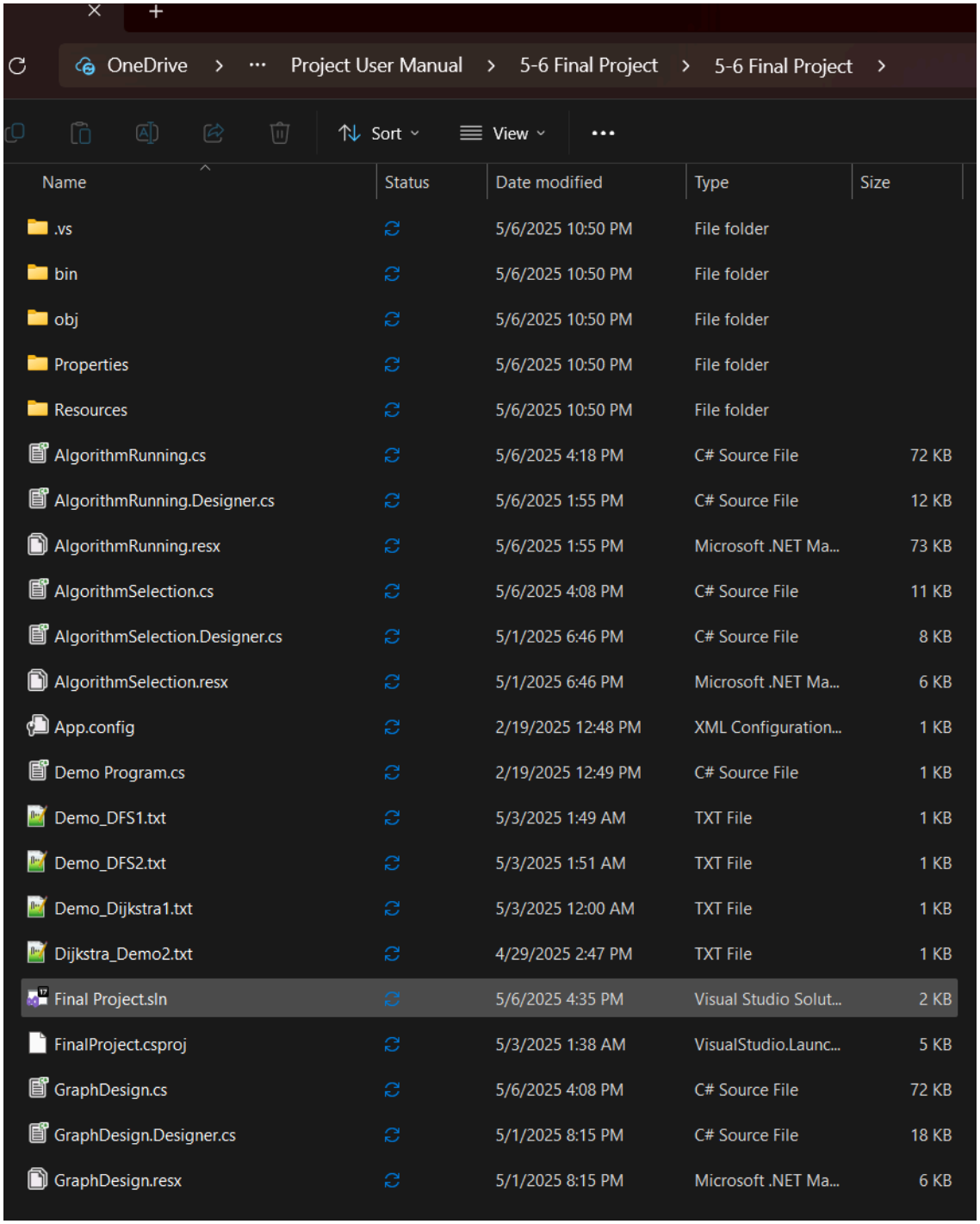


- Now you can unzip or extract the file. If you click on the file, it should say in the bar to extract all, or when you right-clicked on it earlier, it was right under the share.
- Now you are ready for the next step

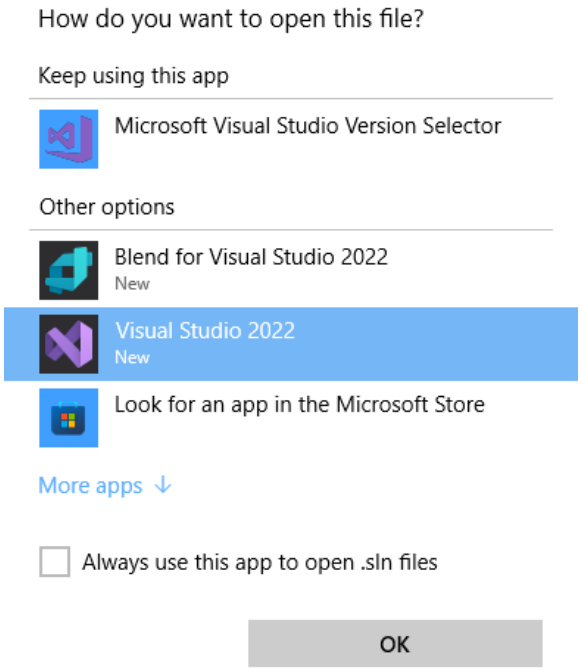
<u>Running / Launching a program</u>

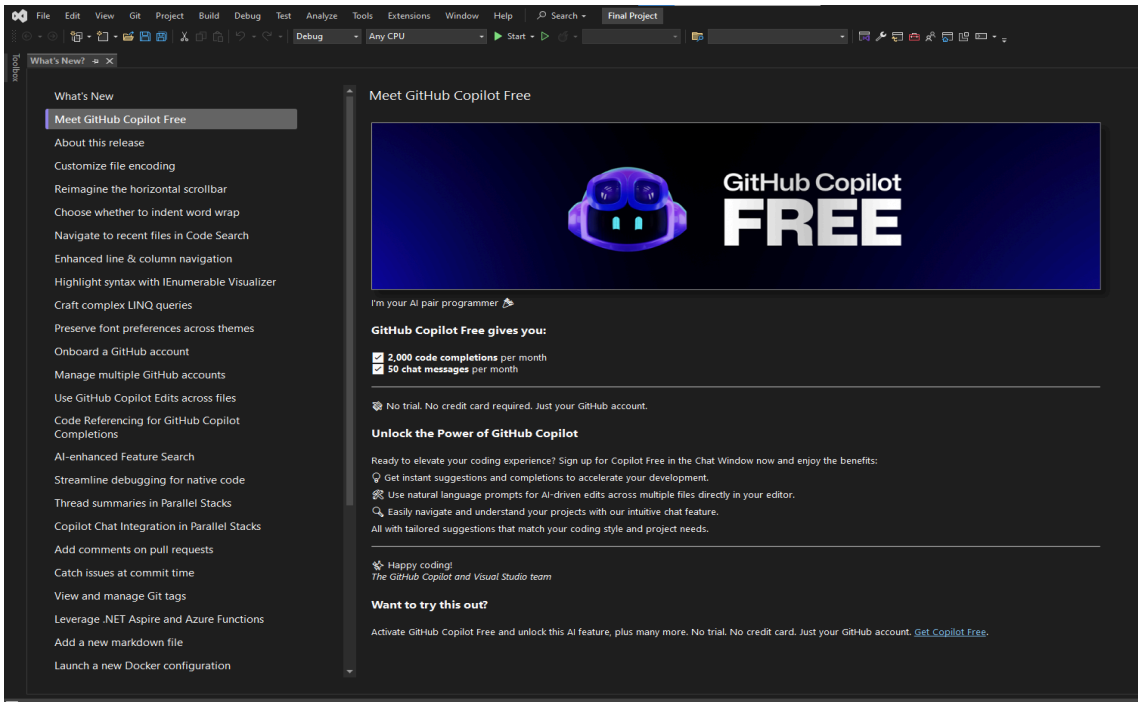
- **DISCLAIMER:** The design, graph form, flickers, or may look glitchy. This is because it is redrawing everything to the screen after any change. (I apologize, though if I had more time, I would have a more efficient and less flickering method.)

- Now you want to navigate to where you saved and extracted your file and open it. You should see a screen similar to what is below:

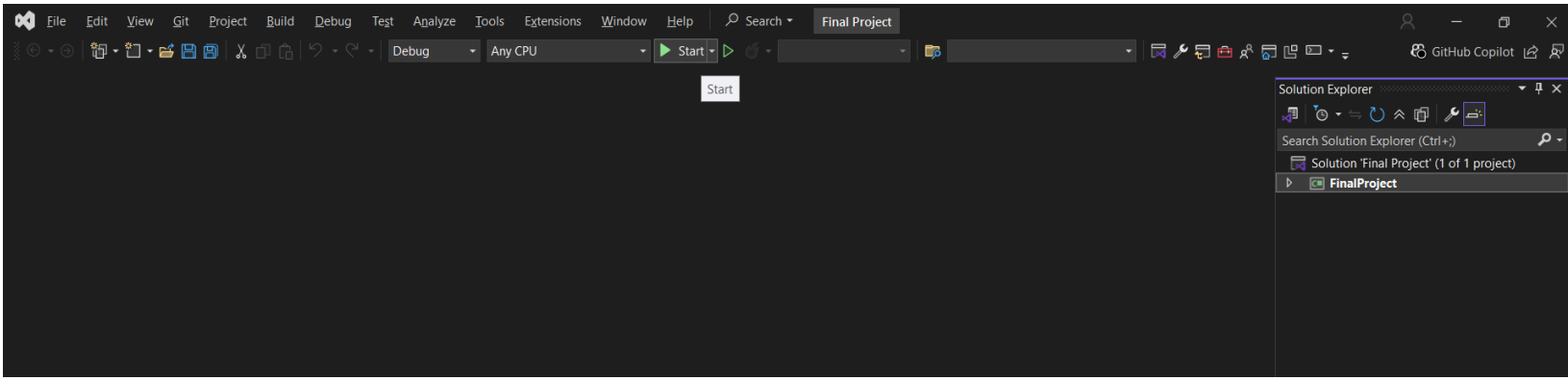


- Now you need to open up the Final Project.sln (.sin may not display; it will be Microsoft Visual Studio Solution (under Type)), which is highlighted on the above screen.
- Below screen may appear. Please select Visual Studio 2022

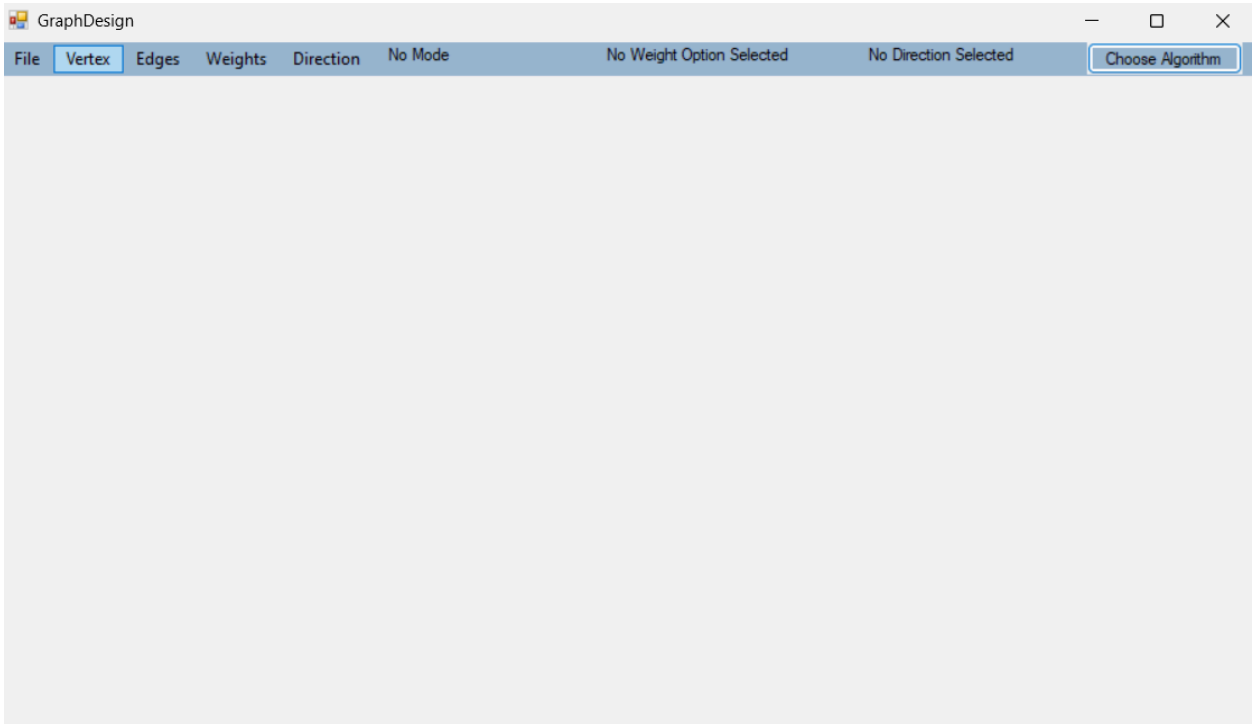




- You should come to a similar screen shown either above or below, and you want to click the start button located in the top middle of the screen. (Shown highlighted below)

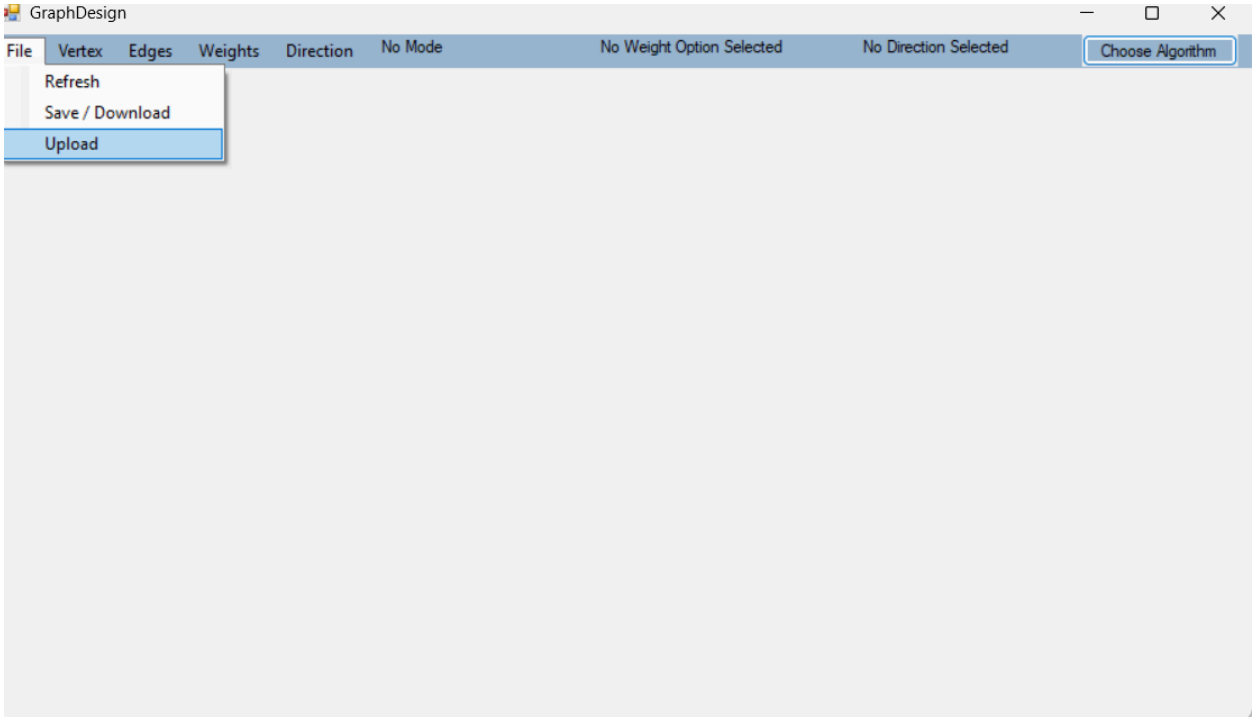


- Now it's time to have some fun, if you are met with the below screen, and now you can design your own graph. (More instructions to detail how this process works).

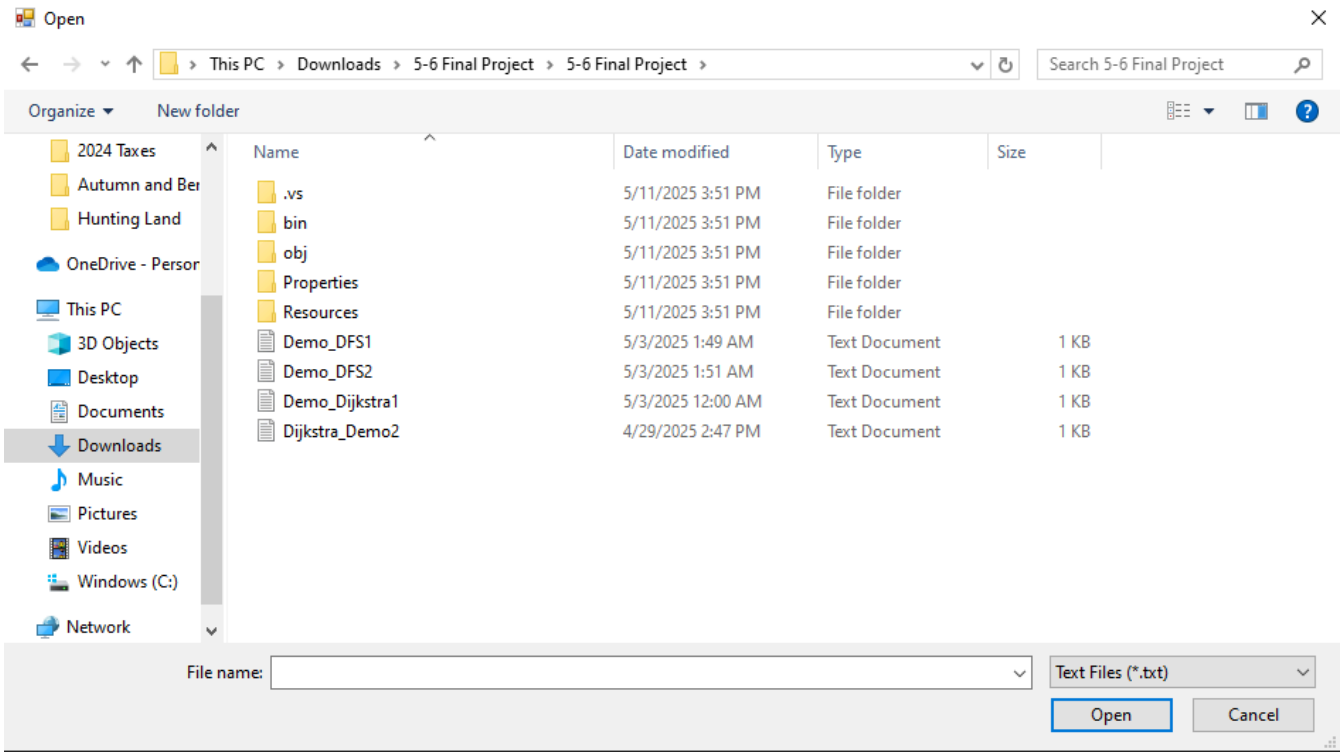


Loading Graphs:

- I have four graphs that I have saved in the folder. If you want to open any of them up to get a feel for what graphs look like, feel free to do so. How that works is, you want to click on the file button, and it will bring a drop-down menu as shown below.



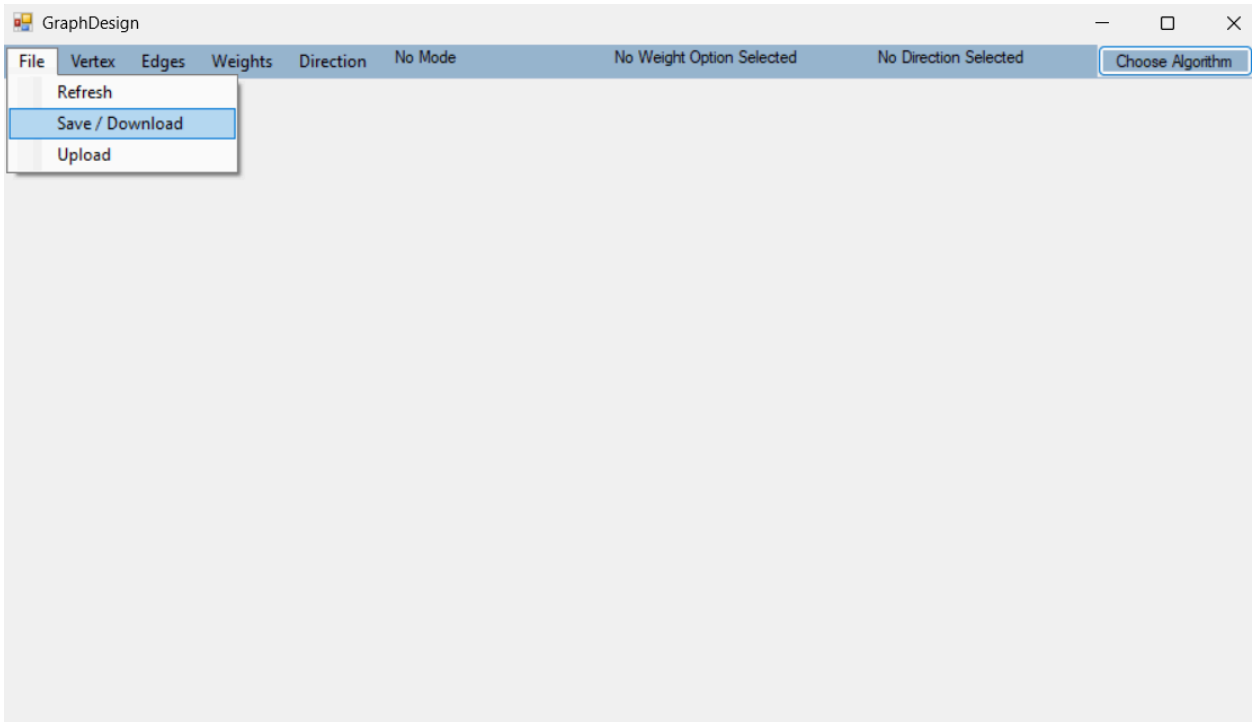
- Then you want to click on Upload, which is highlighted blue in the above picture. This will pull up a file explorer page as shown below. You want to navigate to where you saved the file, and there should be 4 .txt (Text Document) files that show up. Choose one of your choices. If you'd like to look at them all, you can repeat this process and choose a different graph to load.



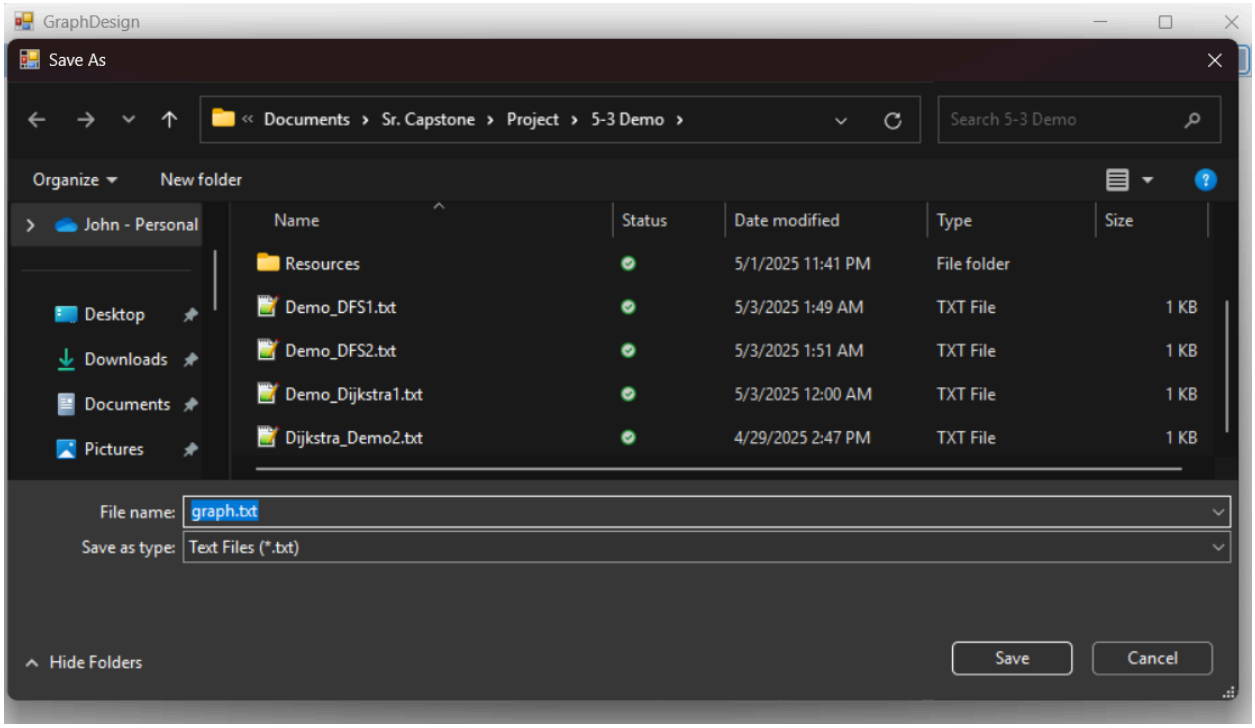
- Later on, I will discuss how you can save your own graph. But when you have your own graph saved, you can follow the same steps I have listed above to upload your own graph.

Save Graph:

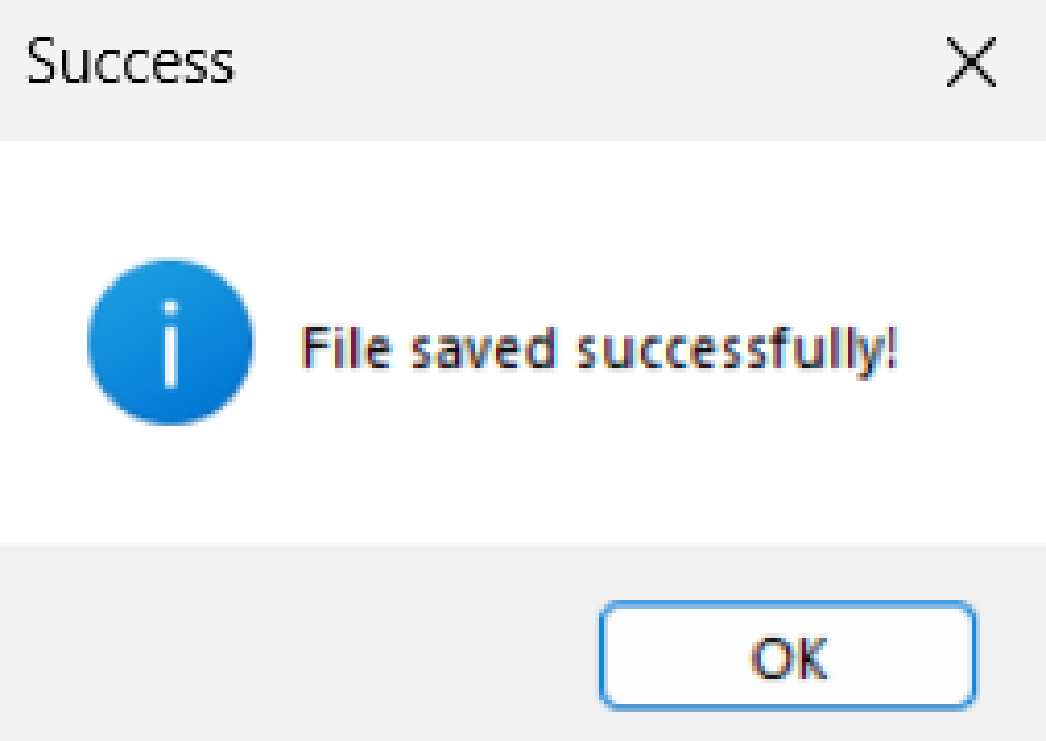
- I will discuss how to create a graph in the next portion, but once you have a graph you like, and you want to save it, you always have it. Then you want to click the file button. You will see a drop-down menu, where you can select/click Save/Download as shown highlighted in the picture below.



- Once you click Save/Download, the program will pop up a file explorer. It will suggest the default name of graph.txt as shown below. **Note:** This file type can't be changed from a text file. You can save it to any location on your computer, along with naming it whatever you want. File must stay as a text file.



- Once you have chosen a specific location in which you'd like to download your graph, and a name, you want to click save in the bottom right corner of the above picture. If your file has successfully been saved, you will get the following prompt.

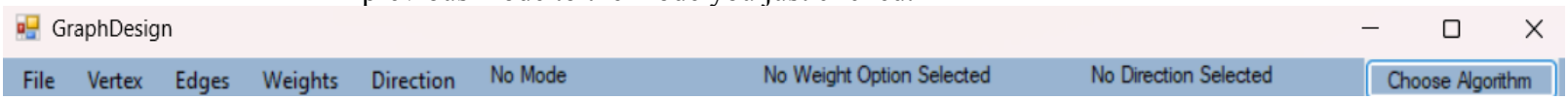


Graph Creation:

- Now I will discuss how to create your own personal graph. You will need at least one edge and one vertex in order to move on to choosing an algorithm.

Buttons / Features:

- Below, I will discuss all of the buttons/features that you can find when you start. **Note:** The first line of text to the right of the direction will tell you what mode you are in. Meaning if you are adding or deleting anything, this will be very helpful for when you are designing your graph. Also, any time you click on a new button, it will change the previous mode to the mode you just clicked.



- **File Menu Button:**
  - Refresh Button:
    - This button will refresh your screen. Essentially, if you want to start all over, you can click File, Refresh to do so. **Note:** You may see a prompt or alert pop up when you click it. If you do, that is letting you know your graph is unsaved and there were changes made. If you click **YES**, then it will refresh the screen. **No/Closing** the window will result in nothing happening and returning to your previous graph.

- Save/Download Button:
  - This button will allow you to save your graph, as discussed above in the Saving Graph section.
- Upload Button:
  - This button will allow you to upload a graph, as discussed above in the Loading Graph section.

- **Vertex Menu Button:**
  - Add / Draw Button:
    - Purpose: Adds a new vertex to the screen.
    - How to Use: Click this button, then click anywhere on the screen to place a vertex.
  - Delete / Erase Button:
    - Purpose: This will remove the existing vertex and renumber the rest.
    - How to Use: Click this button, then click on any vertex you want to delete. **Note:** Any edges connected to it will also be removed.
  - Drag Button:
    - Purpose: Moves a vertex to a new location.
    - How to Use: Click this button, then click and hold a vertex to drag it across the canvas.
    - **Note:** Again, this will look very "glitchy" as the screen flickers. This again is to redrawing everything, and you aren't doing anything wrong, this is just how the program works.

********* **NOTE:** Direction and Weight must be selected before drawing edges *********
********* Also, once you choose weighted/unweighted or directed/undirected, you can't go back. So make sure you

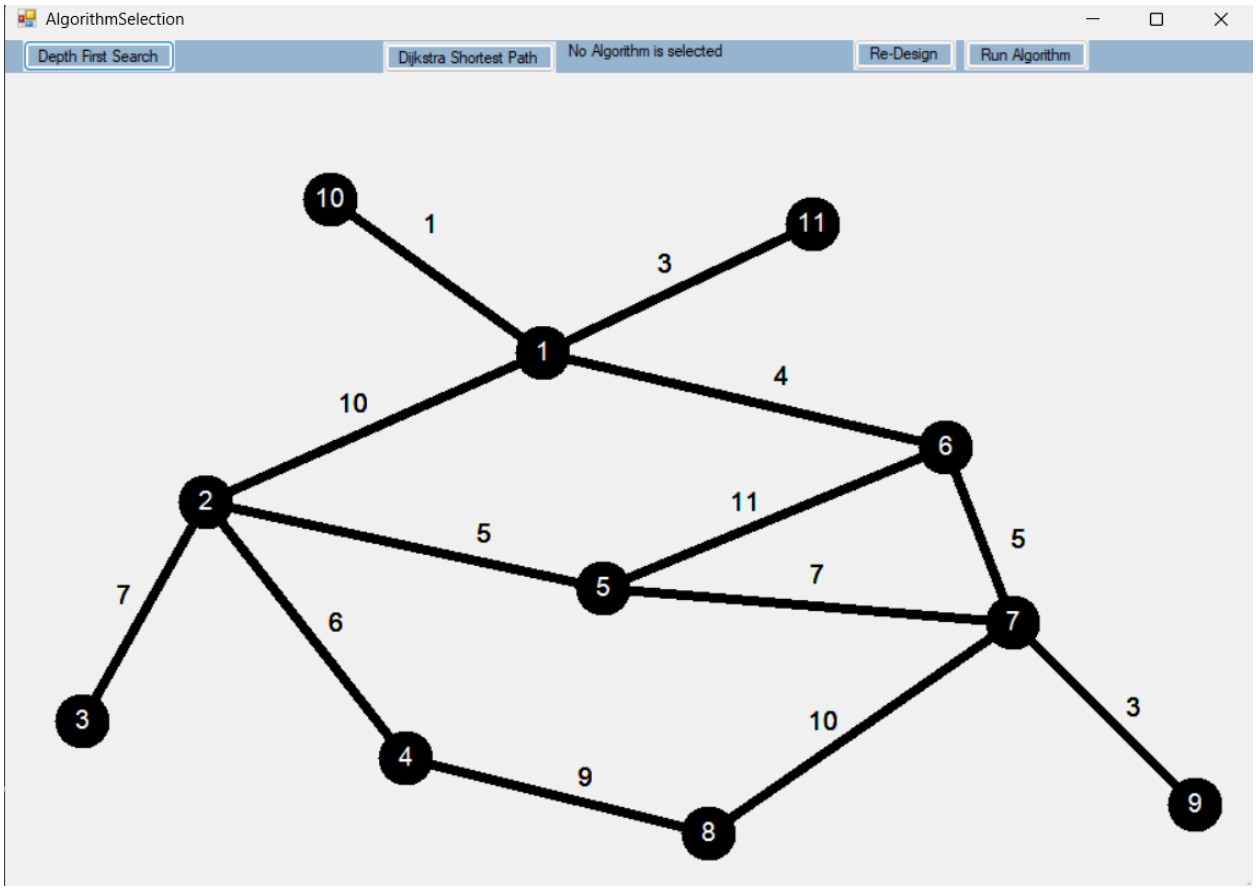
- **Direction Menu Button:**
  - Undirected Button:
    - Purpose: This will set the edge creation mode to have no direction.
    - How to Use: Click this before drawing edges to ensure they appear without arrows.
  - Directed Button:
    - Purpose: This will set the edge creation mode to have a direction, i.e, draws one-way edges with arrows.
    - How to Use: Click this before drawing edges to indicate direction from one vertex to another.
- **Weights Menu Button:**
  - Unweighted Button:
    - Purpose: Sets edges to display no weights.
    - How to Use: Click this to ensure edges will be non-weighted when created.

        - Note: This is an edge mode. So it makes it when you create/add an edge, it will make it unweighted.
    - Weighted Button:
        - Purpose: Sets edges to display weights.
        - How to Use: Click this to ensure edges will be weighted when created.
        - Note: This is an edge mode. So it makes it when you create/add an edge, it will make it weighted.
    - Dragging Weight Button:
        - Purpose: This will move the weight label's position on the screen.
        - How to Use: Click this, then you want to nagovate to the weighted label (or number on the screen) you'd like to reposition to make it easier to see.
        - Now you must click and hold over that label/number, and drag it to your new desired location on the screen. (I recommend keeping it close to the edge it is associated with too.
        - **Note:** Again, this will look very "glitchy" as the screen flickers. This again is to redrawing everything, and you aren't doing anything wrong; this is just how the program works.
    - Changing Weight Button:
        - Purpose: This will modify the numerical value of a weight.
        - How to Use: Click this, then click on the weight label/number to edit its value. You will be prompted with the below

- **Edges Menu Button:**
    - Add / Draw Button:
        - Purpose: Connects two vertices with an edge.
        - How to Use: Click this button, then click on the first vertex. It will be highlighted in red. Then you want to click on a second vertex. An edge will then be created between them.
        - **Note:** For directed edges, the first vertex clicked is the "from" vertex, and the second one clicked is the "to" vertex. I.e., the second vertex will have the arrow pointing to it.
    - Delete / Erase Button:
        - Purpose: This will remove an edge between two vertices.
        - How to Use: Click this button, then it works the same as adding an edge. You click on one vertex, then the second, and it will delete the edge between the two vertices.
        - **Note:** If you are trying to delete a directed edge, you **MUST** follow the arrow. The vertex that an arrow points to must be the second vertex clicked. Otherwise, it will recognise no edge and won't delete it.

Graph Design Finished:

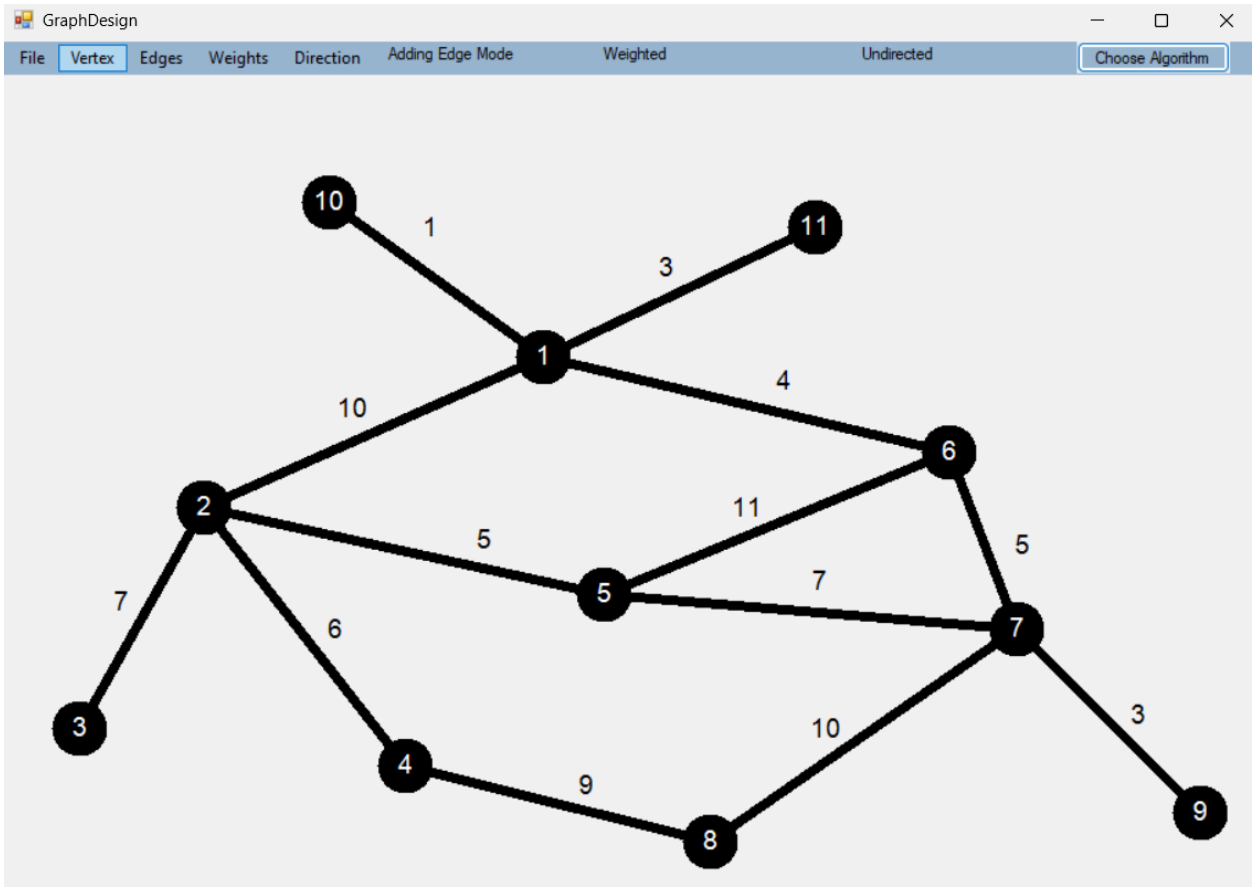
- Once you've finished creating and customizing your graph and you have added all the necessary vertices, edges, weights, and directions, you're ready to apply an algorithm to it.
- Please click on the "Choose Algorithm" button, located on the right side of the menu.
- Once you have clicked on the button, you will be presented with the window below. You will see your graph and a new menu bar, which I will discuss.



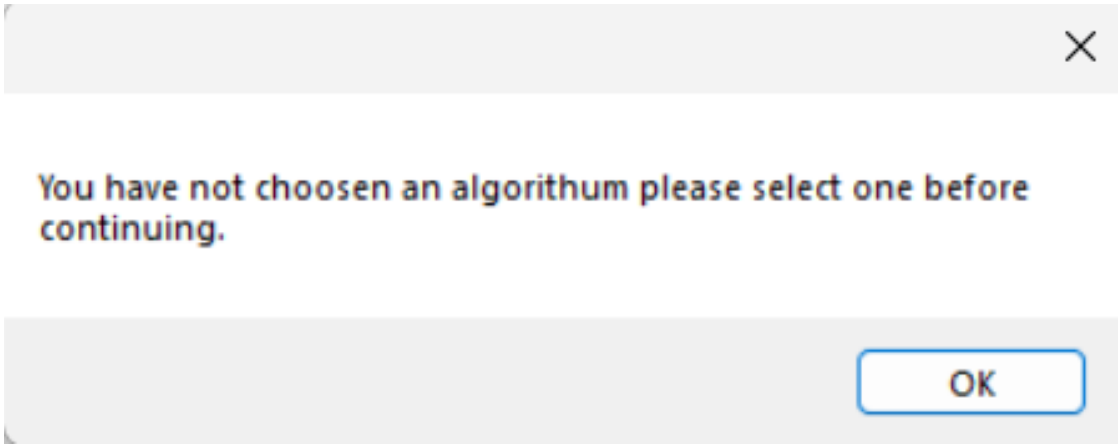
********** This is an example picture of my graph. Your graph (you created) will be displayed in its place. This is to show the option bar on top to run your algorithm.**********

Algorithm Selection:

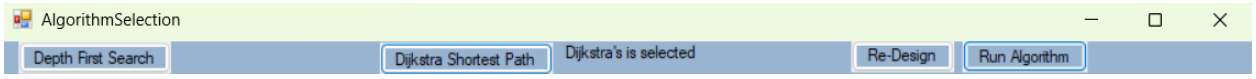
- From here, there are a few options. If you messed up or realized your graph isn't finished, and you would like to go back and edit it, please select the "Re-Design" button, and you will be brought back to the page below. (i.e., the previous page).



- You must select an algorithm before continuing to run it. You may have noticed that it says "No Algorithm is selected." If you try clicking "Run Algorithm" (this is where you run the algorithm you choose), it will present you with the following pop-up message.



- There are currently only two algorithms implemented, so please select one of them. Dijkstra's is a shortest path algorithm, and it will show you the shortest path to get through your graph, from the starting vertex.
- Whereas Depth First Search or DFS explores your graph by going as deep as possible along one path before backtracking. It's great for understanding a traversal order and works with both weighted and unweighted graphs, though it ignores weights.
- Once you have selected your desired algorithm, you can now click on "Run Algorithm" in the top right corner of your screen. For example, below I selected Dijkstra's algorithm, and the text changes accordingly, now I can select Run Algorithm.



Depth First Search Algorithm:

- Below, I will discuss statuses and what the animation colors mean
- Black Vertex/Edge: Has not yet been reached by the algorithm
- Fuchsia (Pink)
    - Vertex: Is an unvisited vertex that is being considered.
    - Edge: An edge currently under consideration.
- Green Edge: This is the edge that is chosen for this traversal. Once this occurs, the vertex is marked as visited.
- Orange Vertex: A vertex that has been marked visited. It becomes visited if we choose the Fuchsia path, and it turns green.
- Blue Edge (Back Edge):
    - This occurs when the edge leads to a vertex that has already been visited or is done.
    - This indicates we've already been there, and that it is ignored.
- Green Vertex: A vertex is considered done once all its adjacent vertices have been explored, and thus turns green, signifying completed.

Dijkstra's Shortest Path Algorithm:

- There are many parts to this algorithm. One is the visualization or animation of colors, then there's a table, and a feedback section. I will explain what each part does.

Dijkstra's Shortest Path Animation Guide:

- Green Edge: The edge chosen as part of the current shortest path.
- Orange Vertex: A visited vertex—its shortest distance is finalized.
- Fuchsia (Pink):
  - The vertex or edge is currently being considered.
  - Represents potential paths under evaluation.
- Black: Vertex or edge that has not been reached yet.
- Blue Edge: An edge that was considered but not chosen as part of the shortest path.

Understanding Dijkstra's Table:

- The table updates in real-time to match each animation step.
- Each row shows:
  - The current guaranteed shortest path, denoted on the leftmost portion of the graph.
  - It shows the vertex, its parent, and the total cost associated with reaching that vertex.
  - It then also checks other paths and considers them, with Possible Parent and Possible Cost.
  - What that means is that the vertex has not been chosen yet, but the current known shortest for that vertex is through the Possible Parent, with a possible cost.

Feedback of Dijkstra's Table:

- Why a vertex or edge was chosen.
- Why an edge / path wasn't chosen.
  Which paths are being updated.
- How the shortest path is built over time.

Running Algorithm:

- Now it is time to run your algorithm. **Note:** Depending on your selected algorithm, the screen visuals may be different. However, do not make the screen smaller as this could cause issues.
- The menu bar will remain the same, however, so let me discuss what each option does. You can see a picture of it below:

Choose Algorithm Button:

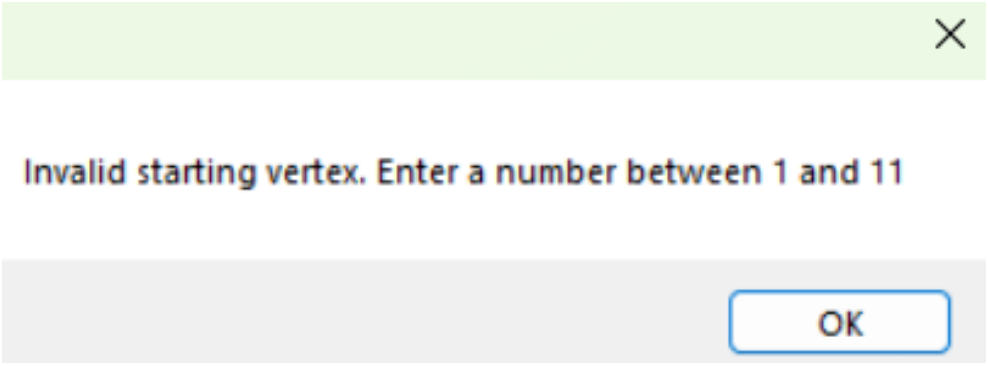
- This button is for if you would like to go back and select a different algorithm to run, or if you accidentally selected the wrong algorithm. Note: If you would like to redesign your graph, you would first have to click Choose Algorithm, and then it will redirect you to the Algorithm Selection window. Then you would need to click the Re-Deisgn button discussed above.

Starting Node:

- This is a text box. You type the number of the vertex/node in which you would like to start the designed algorithm.

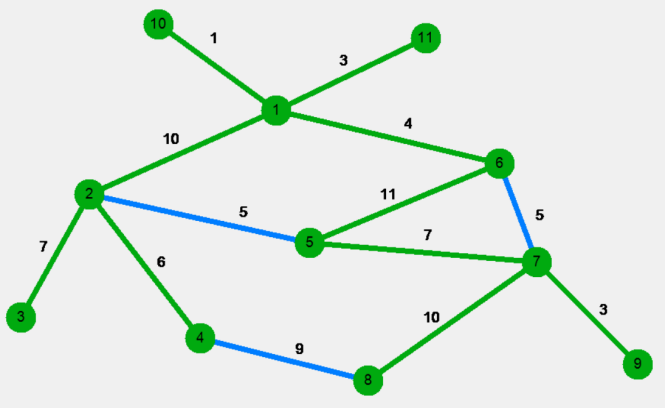
Start Button:

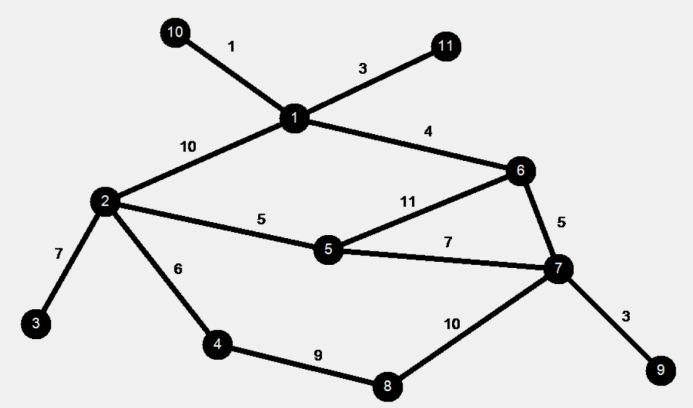
- Once you have a valid starting node, you may click the starting button. This will initialize the start of the algorithm and run off the speed bar. Note: Read through what everything else does before clicking this button.
- **Also to Note:** If you select a vertex out of range, it will let you know, and you will have to select another vertex. See the pop-up below (note this is for my example graph so you may have more or less than 11 vertices:



Refresh:

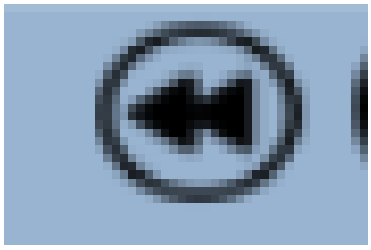
- This button is used when the algorithm is finished running. If you want to go back to seeing your default graph, you can. **Note:** If you attempt to click this button while the algorithm is running, you will be met with an error box letting you know you need to wait till after the algorithm finishes.
- Check out the before (There will be colors not just black) and after (all black) example:





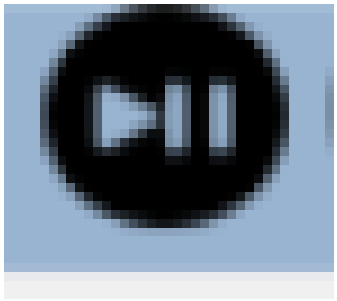
Rewind Button:

- This button isn't implemented, and therefore, if you click it, it will not do anything. This is what the button looks like:



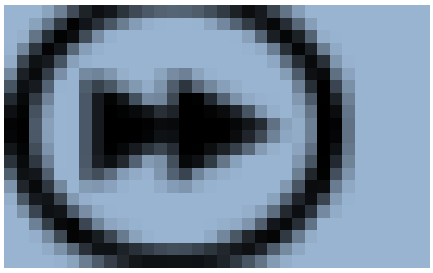
Play / Pause Button:

- This button will pause the algorithm animation display or play it, depending on whether or not it was previously paused. For example, upon starting the algorithm, the first time you click the button, it will pause. The second time it will play. Below is what the button looks like:



Fast Forward Button:

- This button only works if the animations are paused. This will allow you to step through the algorithm one step at a time. **Please Note:** This button may not work on the first click, so wait a second, if no visual happens, click it again. For Dijkstra's, there will be feedback notes that are printed. Below is what the button looks like:



Speed Slide Bar:

- This controls how fast you see the animations. The far left is the least amount of time it delays. To the right is the maximum amount of time it is delayed. 0 seconds is all the way to the left, and 10 seconds is all the way to the right.
- Every dash is 1 second, and it is defaulted to a 5-second delay.
- **Note:** You may change this while the algorithm is running, however, it won't update the delay time until after the animation is finished, then the next one will be the delay you just set.