Project TeamworkTemplate

Version 1 9/11/24

A ***separate copy*** of this template should be filled out and submitted by each student, regardless of the number of students on the team. Also change the title of this template to "Project x Teamwork <Czaplewski> - <jczaplew>"

1	Team Name: Czaplewski
2	Individual name: Jason Czaplewski
3	Individual netid: jczaplew
4	Other team members names and netids: I worked alone
5	Link to github repository: [11jac11/HamiltonianPath_Czaplewski: In this repository is the contents of Jason Czaplewski's Project 1 Hamiltonian Path (github.com)]
6	Overall project attempted, with sub-projects: Hamiltonian Path and Cycle and Traveling Salesman Problems
7	List of included files (if you have many files of a certain type, such as test files of different sizes, list just the folder): (Add more rows as necessary)

File/folder Name	File Contents and Use
Code Files	
HamiltonianCode_Czaplewski.py	In this file is the entirety of the code that runs the Hamiltonian path. It contains all the libraries, main, functions, as well as the output code for the text and graph. It is used to run the code and is where all of the work was actually done to create the project
Test Files	
TestFile_Czaplewski	This is a pdf file of what I used to test my code. I did not use csv files but instead randomly generated the graphs within the code. Essentially what is in here is snippets of codes that are used to check the variety of metrics within the main code. I did not use any outside tests since it all had to be done internally to make sure that it was testing what was randomly created. Since no csv files were used I

	thought that this would suffice as to how to show that I checked my code internally every time that it runs since it creates the graphs on its own.
Output Files	
TextOutput_Czaplewski	This is a pdf file that shows the output text files that you receive when you run the code. These are different every time so I simply picked a random execution and inserted it into the document. It tells you the size of the vertex and the average run time for that size. It also gives you one of the possibly many Hamiltonian cycles found when it ran that execution.
Plots (as needed)	
GraphOutput_Czaplewski	This is a pdf showing the graph you are given when you run the code. SI]ince the graphs are randomized each time I picked an execution and put it in this file. It puts a green dot where the Execution time for each point is at each of the vertices (4-10). It then draws a blue line through the vertices showing the average execution time for the runs and you can see the exponential growth as the vertex number gets higher.

8	Individual Student time (in hours) to complete: ~5 hours
9	Your specific activities and responsibilities: I completed the whole thing
10	What was personally learned (topic, programming, algorithms): From this code I learned about the brute-force approach to solving the Hamiltonian cycle problem. The algorithm checks all permutations of vertices in a graph to identify if any form a valid Hamiltonian cycle. This exposed me to the concept of computational complexity and the performance challenges that arise with these types of problems as graph sizes increase. In most of my previous work I have not had to worry about execution time so tracking that and also making a visual to see its exponential growth highlights its importance especially when writing larger code. Also by testing and timing the execution for different graph sizes I gained insight into the inefficiency of these search methods and the importance of choosing the right algorithm when dealing with such problems.
11	How the team was organized, and what might be improved: I worked alone so I just worked on my own schedule. Not much could be improved since I was able to get any questions I had about the project answered. Overall it was a success and GitHub allowed me to organize all of my files easily to submit.
12	Any additional material: Issues: When running the code I kept encountering zero execution time for many of the random iterations of the trials even at high vertices. I presumed that these were the non hamiltonian paths being graphed however when I tried to go back into the code and remove them it would mess up the randomization.

	Therefore, on the graph you see the two different y intercepts. The one on the left is the execution time for each point and the one on the right is the average execution time. I broke these up so that you can visually see the linear growth because without that the slope did not increase in an expected manner.