Project 1 Readme Team Farmer

Version 1 9/11/24

A single copy of this template should be filled out and submitted with each project submission, regardless of the number of students on the team. It should have the name readme_"teamname" Also change the title of this template to "Project x Readme Team xxx"

<table>
<tr><td>1</td><td>Team Name: Farmer</td></tr>
<tr><td>2</td><td>Team members names and netids

Connor
cmacdon4</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects:

Hamiltonian Path Problem: DumbSAT Solver</td></tr>
<tr><td>4</td><td>Overall success of the project:

perfect!</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete:

4 hours coding and 3 hours on documents</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/cmacdon4/Theory_Project_01</td></tr>
<tr><td>7</td><td>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). Add more rows as necessary.
<table>
<tr><td>File/folder Name</td><td>File Contents and Use</td></tr>
<tr><td colspan="2">Code Files</td></tr>
<tr><td>hpath_farmer.py</td><td>Hamiltonian Path checker

./hpath_farmer.py $FILE</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>data_unit_test_farmer.txt</td><td>40 undirected and directed graphs various</td></tr>
</table>
</td></tr>
</table>

		numbers of vertices Use for hpath_farmer.py
	Output Files	
	output_data_farmer.csv	Output from hpath_farmer.py
	output_terminal_farmer.png	Screenshot of terminal output
	output_excel_farmer.xlsx	Excel sheet of csv data in easy to read format
	Plots (as needed)	
	plot_farmer.png	screenshot of plot (number vertices vs time)
8	Programming languages used, and associated libraries: Python; OS, sys, time, itertools	
9	Key data structures (for each sub-project): I implemented a class called Hamiltonian that had the following attributes in Figure 1.	

```
class hamiltonian:
    def __init__(self, hpath, case_no):
        self.is_hpath = hpath
        self.ncase = case_no

        self.verts = []
        self.edges = []

        self.nedges = 0
        self.nverts = 0

        self.undirected = False
        self.result = False
        self.time = 0
```

	Figure 1 – Attributes of the Hamiltonian Class The boolean is_hpath describes whether the given graph is a Hamiltonian path, which is found from the data file; ncase is the graph number (ranging from 1-40); verts is a list of the vertices derived from the data file; edges is a list of tuples containing 2 vertices that are connected; nedges and nverts are the number of vertices and the number of edges given in the data file; undirected in a boolean describing whether the graph was directed or undirected which is given in the data file; the boolean result is determined by the is_hpath function which algorithmically determines whether the graph has a Hamiltonian path (discussed later); and the time was the total time to determine whether the graph was a Hamiltonian Path. While reading the data file, a Hamiltonian object was created for each graph and appended to a list.
10	General operation of code (for each subproject) The code reads in a number of graphs from a data file. Then it checks if the graph has a Hamiltonian path. Finally, it writes to stdout and a CSV file for the results to be processed for correctness and speed. A much more detailed description as well as its project management are given in 12.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. I obtained my data file from the course website, containing 40 different graphs. It also provided both undirected and directed graphs which added a layer of complexity to the code. This file was a great resource as it provided both edge cases–when there was only 1 vertex–and complex Hamiltonian paths, some having as many as 13 vertices. The file also provided whether the graph had a true Hamiltonian path, making it easy to check if my checker algorithm had worked.
12	How you managed the code development First, I analyzed the requirements for the project. To get this project off the ground, I would need a number of graphs that my code would have to read in. Luckily, there was a source on the canvas page that would allow me to read into 40 different graphs. Then I actually began implementing the reading in. This required me to

	brainstorm how to organize the data in a way that could be easily shared across functions. As mentioned above, I settled on a list of objects of the Hamiltonian class. After making that list, I read in each line from the input file and made an object for each graph. I tested this code first with one graph at a time using my check_test_farmer.txt file and eventually moved up to the data_unit_test_farmer.txt as I ironed out the bugs. The next part of the project required me to check if there was a Hamiltonian path in each of the graphs. To do this, I made the is_hpath function that uses all possible permutations of the given vertices. To do this I imported the permutation function from itertools. This method worked because if you can generate all possible combinations of paths, then the Hamiltonian path must exist in it. For each permutation, the code checked to see if an edge between perm[n] and perm[n + 1] existed. If not, the loop broke out moving on to the next permutation. The permutation is classified as successful if vertex n equals nverts - 2. This means the code has checked if the edge between the second to last and the last vertex exists in the graph. While inefficient, this eventually produced the right results. To test this function, I stored my function's result and compared it to the given result that data_unit_test_farmer.txt provides. This was the gist of the project.
13	Detailed discussion of results: To analyze the results of the program, I wrote the case number, the number of vertices, the time, and a boolean value representing whether there was a Hamiltonian path in a CSV file. I then opened this CSV file in Excel and plotted the points against time and the number of vertices. In the graph, the data fit an exponential trendline pretty well. Because I used permutations to check if the given graph has a Hamiltonian path, the time complexity of the program is likely P(n!), which is incredibly slow. This makes sense for a Dumb Solver.
14	How team was organized I worked alone
15	What you might do differently if you did the project again I might reduce the memory overhead by producing results and time for each graph one at

	a time instead of storing them in a list.
16	Any additional material: No