Project 1 Readme Team ekoran

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: ekoran</td></tr>
<tr><td>2</td><td>Team members names and netids
Ethan Koran
NetID: ekoran</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects:
Project: Hamiltonian Path Problem
Sub-Project: DumbSAT Solver</td></tr>
<tr><td>4</td><td>Overall success of the project: My project was successful</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: ~6 hours</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/ethankoran/TOC_project01_ekoran</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>hamiltonian_path_ekoran.py</td><td>DumbSAT solver code</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>data_ekoran.csv</td><td>CSV file with 40 graph test cases</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td>output_ekoran.csv

stdout_ekoran.txt</td><td>CSV output file used to plot results in Excel

stdout from code showing the results of each test case</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th colspan="2">Plots (as needed)</th></tr><tr><td>plot_ekoran.jpeg</td><td>Screen shot of plot made in Excel</td></tr></table></td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries: I used Python to write the dumbSAT solver script, utilizing the itertools and time libraries</td></tr>
<tr><td>9</td><td>Key data structures (for each sub-project):
- List storing the vertices of each graph
- List of tuples storing the edges of each graph, where each tuple has a starting and ending vertex</td></tr>
<tr><td>10</td><td>General operation of code (for each subproject)
My code starts by interacting with two files: it opens input_ekoran.csv to read in test cases and creates output_ekoran.csv to write results to. Then, it reads in/processes each graph test case in the new_graph() function, which returns the list of vertices and list of edges from the input file (note: if the graph is undirected, you must append an edge going both ways onto the list). Then, find_hamiltonian() is called on each test case. In this function, first we compute the start time of the computation using the time library. Then, using the itertools library and a for loop, I iterate through every possible permutation of the order of vertices of the graph until I find a hamiltonian path. For each permutation, cycle through the vertices and make sure that there exists a path between each consecutive vertex. If a path does not exist, skip to the next permutation. Once we get to the last two vertices and confirm there is an edge between them, we can compute the elapsed time and return True. Otherwise, the function we go through all iterations without finding a hamiltonian path and return the elapsed time with False. Back in main, we write the result of each test case to our output file and to stdout.</td></tr>
<tr><td>11</td><td>What test cases you used/added, why you used them, what did they tell you about the correctness of your code.
I used the 40 Hamiltonian graph test cases provided by a fellow classmate, which contained graphs of between 1 and 14 vertices. The result of whether the graph contains a Hamiltonian path is already known, so I was able to compare the result of my dumbSAT solver to the expected result to verify that my code functioned correctly.</td></tr>
<tr><td>12</td><td>How you managed the code development
First, I brainstormed the steps required to create my dumbSAT solver. To do this, I utilized resources such as a white board to visualize my strategy. I broke the process into three major steps:
1. Writing and debugging code for taking in test data input
2. Writing debugging code for determining if a hamiltonian path exists (dumbSAT)
3. Writing and debugging code for writing results to a new output file
Breaking up my development process like this helped keep it more manageable as I worked</td></tr>
<tr><td>13</td><td>Detailed discussion of results:
I was successfully able to implement a dumbSAT solver in python. To analyze my results, I used my output file to create a plot of the results. Through this, I found an</td></tr>
</table>

	exponential relationship between the number of vertices of each graph vs the time it took to compute the hamiltonian path. This makes sense, because my code uses brute force to go through every permutation of vertex combinations to determine if a hamiltonian path exists. Thus, if no hamiltonian path exists in a graph, the code can go through anywhere between 1 and upwards of 87 billion permutations depending on how many vertices are in the graph, (for this data, there were graphs that had between 1 and 14 vertices). Therefore, naturally the time it takes to do these computations will grow exponentially, since there are n! permutations for n vertices.
14	**How team was organized**: I worked on this project alone, so I did each aspect of the project, starting with brainstorming, then the actual code, and finally the
15	**What you might do differently if you did the project again**: If I were to do this project again, I would try to implement another script that generated even more test cases for my code. Although, I still achieved great results with the test cases that were provided to me.
16	**Any additional material**: N/A