Project Readme Template

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: Twin Track Trio</td></tr>
<tr><td>2</td><td>Team members names and netids
Ethan Sotka
esotka
Freeman Nkouka
jnkouka
Marco Tchernychev
mtcherny</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects: Binpacking (Brute Force, Back Tracking, Best Case)</td></tr>
<tr><td>4</td><td>Overall success of the project: Complete, all working</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: 12 hours</td></tr>
<tr><td>6</td><td>Link to github repository:https://github.com/Freemankn/Project1-TOC.git</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>src/helper/bin_packing_helper.py</td><td>Runs functions in bin_packing.py while having supporting functions to read in and output data, all under the BinPackingAbstractClass.</td></tr>
<tr><td>src/bin_packing_Twin_Track_Trio.py</td><td>Contains functions to find working knapsack problem solutions for various algorithms (Brute Force, Back Tracking, Best Case).</td></tr>
<tr><td>src/plots_Twin_Track_Trio.py</td><td>Plots the data from the output files on their own graphs, putting the number of coins on the</td></tr>
</table>
</td></tr>
</table>

	x-axis and time on the y-axis. Successful attempts are green, while unsuccessful ones are red.
main.py	Runs the entrypoint file
src/entrypoint.py	Finds the chosen project of the group and runs it
Test Files	
input/check_Twin_Track_Trio.txt	Contains test cases for various cases of knapsack problems
Output Files	
results/output_best_case_check_Twin_Track_Trio.csv	Data of results (successes and failures) for the best case function
results/output_brute_force_check_Twin_Track_Trio.csv	Data of results (successes and failures) for the brute force function
results/output_btracking_check_Twin_Track_Trio.csv	Data of results (successes and failures) for the backtracking function
Plots (as needed)	
src/plots_binpacking_best_case_check_Twin_Track_Trio.png	Plots the data from the best-case algorithm results. The number of coins is on the x-axis, while time is on the y-axis, creating an exponential relationship. Successful attempts are in green, while unsuccessful ones are red.
src/plots_binpacking_brute_force_check_Twin_Track_Trio.png	Plots the data from the brute force algorithm results. The number of coins is on the x-axis, while time is on the y-axis, creating an exponential relationship. Successful attempts are in green, while unsuccessful ones are red.
src/plots_binpacking_btracking_check_Twin_Track_Trio.png	Plots the data from the backtracking algorithm results. The number of coins is on the x-axis, while time is on the y-axis, creating an exponential relationship. Successful attempts are in green, while unsuccessful ones are red.

8	Programming languages used, and associated libraries: Python

9	Key data structures (for each sub-project): Brute Force: Lists Back Tracking: List/queue Best Case: Lists
10	General operation of code (for each subproject) Brute Force: The program first generates a subset of all possible combinations. A for loop then goes through each subset and checks if the sum is equal to the target bin. If so, that subset would be appended to a results list. However, if there were no combinations whose sum was the target, [[0]] would be returned to signify that there was no solution. Back Tracking: There are “global” variables in the top function to keep track of current coins and working coins. Then uses recursion, having base cases of failure and success. In recursive calls of the function, cases of using the first coin available and skipping the coin get called. Best Case: I made bestCases list whose first element is the closest distance to the target, and whose second element is a list of all of the subsets that give a sum which is the first element away from the target. I go through all of the subsets and update the bestCases list. So if I find a subset who is the first element away from the target, I’ll add it to the second element of the bestCases list. If I find a subset whose sum is less than the first element away from the target, then I’ll update the first element and also start a new list as the second element. I then return the second element of the bestCases list which is all of the subsets with the best solution.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. We generated many random test cases to get an even spread of possibilities. We made sure that these randomly generated test cases included successes and failures as well as edge cases to see how our code would react to these varying scenarios.
12	How you managed the code development: We handled our code development by creating separate branches in GitHub for each team member. This allowed everyone to implement and test their own changes independently. After completing our updates, we committed and pushed the work to our individual branches. Then, one team member was responsible for merging the finished changes from those branches into the main branch.
13	Detailed discussion of results: We saw similar results in the program timing throughout the various algorithms. As the number of coins in the knapsack grows, the timing for the function grows exponentially. Additionally, throughout the algorithms, we see unsuccessful knapsacks of the same

	length being quicker than successful ones. This may be due to some additional computing required to process successful cases.
14	How the team was organized: Our group met weekly to plan out the work and assigned each subproject to a member of the project. We then worked together to create the code to plot the data we gathered from each function. Whenever we ran into challenges or needed clarification, we reached out to the professor or TA for support.
15	What you might do differently if you did the project again Look at good GitHub workflow practices for pushing code in a team-based setting. There were some mishaps with understanding when we could push and when we could pull which made our work go unsmoothly. We also could have communicated better regarding how our code works and is meant to be run, since we were usually delving into broken code on an ad hoc basis – waiting to solve problems after they arose. Aside from that, the team worked very well.
16	Any additional material: