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"

1	Team Name: Rademacher
2	Team members names and netids: Grace Rademacher grrademac
3	Overall project attempted, with sub-projects: Bin Packing Problems #1 (Coin Problem)
4	Overall success of the project: Pretty successful
5	Approximately total time (in hours) to complete: ~5 hours
6	Link to github repository: https://github.com/grademac/Theory_Project1
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). Add more rows as necessary.

File/folder Name	File Contents and Use
Code Files	
coinstester_Rademacher.py	dumbSat equivalent for the bin packing problem in which it solves for the combination of coins for a given amount. It reads in a file of targets and coin amounts and outputs in both terminal and an output file if it's satisfiable and the amounts for each coin. It also outputs the execution time. The solver skips a case if it takes longer than 5 minutes to solve.
Test Files	
knapsackTestCaseGen_Rademacher.py	Creates a file of randomly generated targets and coin values. The user can adjust how many types of coin values are generated and how many cases in total are generated. These values are outputted into a file.
Test_Rademacher (folder)	Test files generated from knapsackTestCaseGen.py for each of the number of coins cases
Output Files	

Timeresults_Rademacher (folder)	Time results from each of the number of coins cases (2-5 representing how many types of coins were available) – used to create graphs
SATresults_Rademacher (folder)	Satisfactory results from each of the number of coins cases (2-5 representing how many types of coins were available) – used to create graphs
results_Rademacher (folder)	Output results from each of the number of coins cases (2-5 representing how many types of coins were available)
Plots (as needed)	
Project 1 Graphs_Rademacher	Shows number of types of coins vs microseconds, number of types of coins vs log microseconds, and the number of types of coins vs the average number of microseconds

8	Programming languages used, and associated libraries: python, import time, import random, import argparse, import datetime, import textwrap
9	Key data structures (for each sub-project): list (of tuples)
10	General operation of code (for each subproject): similar to dumbSat – reads a file of targets and coin values and brute force solves for a combination of coins that equals the target value. Then based on if there is a possible solution, outputs the success and the number of coins of each type that is used to achieve the desired target value.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code: through knapsackTestCaseGen_Rademacher.py I was able to generate a bunch of test cases that had varying levels of size and number of types of coins used. Since I was getting both outputs of unsatisfiable and satisfiable, I was able to look at the unsatisfiable numbers and could tell it was correct.
12	How you managed the code development: Since this was a version of dumbSat for a different problem, I used dumbSat as a road map for creating this code. I knew I was going to have to read in numbers, preferably from a file, so I started there and used a similar method for brute forcing the combinations.
13	Detailed discussion of results: As expected, when only using 1 or 2 types of coins, the code ran pretty smoothly. Once I ran it with 3, the code slowed down exponentially, and 4 types of coins was even slower. This matches with our knowledge that 2Sat problems are much more manageable than 3Sat and greater problems.
14	How team was organized: Since it was just me, the team wasn't organized in any particular way.
15	What you might do differently if you did the project again: If I did this project again, I would change how I tested my data as it was tedious to run the code each time I wanted

	to test a different amount of types of coins
16	Any additional material: