INPUT

To randomly generate input for our test cases we wrote a program (testdatagenerator_jim.py) which generated these results for us. It generates 15 total test cases, 5 small, 5 medium, and 5 large cases. We determined the appropriate parameters for min and max values of coin denominations, as well as amount of each coin, and target coin, for each test size. The output of this program generates an array randomly shuffled with all the coins in a jar. The lines of output are in the following format *[target, [coin1, coin2, coin3, … , coinN]].* Below is the output generated by the program which we will use as input to test our algorithms:

Output After Running testdatagenerator_jim.py
Small Cases: [10, [50, 5, 24, 5, 24, 24, 2]] [19, [1, 16, 39, 16, 16, 7, 7, 7, 16, 7, 16, 1, 7]] [6, [6, 40, 6, 40, 6, 40, 40, 2, 6, 18, 18, 40, 2]] [40, [27, 2, 15, 15, 15, 8, 27, 15]] [20, [16, 16, 16, 6, 1, 6, 16, 1, 1, 1]] Medium Cases: [300, [10, 29, 2, 12, 10, 10, 2, 12, 29, 12, 2, 2, 29, 12, 2, 2, 10, 10, 12, 29, 2, 10, 10, 29, 10, 12, 29]] [402, [3, 9, 9, 3, 24, 3, 35, 24, 9, 3, 9, 3, 9, 24, 3, 35, 35, 24, 3, 3, 35, 35, 24, 9, 24, 35]] [495, [49, 11, 9, 49, 1, 11, 49, 9, 9, 11, 1, 11, 1, 1, 11, 1, 49, 11, 11, 9, 49, 1, 49, 9, 49, 11, 9]] [364, [21, 6, 21, 31, 3, 31, 3, 31, 3, 31, 6, 3, 21, 3, 3, 31, 21, 6, 31, 6, 3, 31, 6, 6, 31, 21, 6, 21, 6, 3, 21]] [227, [47, 3, 9, 3, 47, 13, 3, 9, 9, 13, 3, 9, 47, 13, 3, 13, 47, 9, 47, 9, 13, 13, 13, 9, 9, 3, 47, 47, 13, 3]] Large Cases: [812, [30, 16, 16, 3, 30, 3, 6, 6, 16, 30, 16, 3, 3, 3, 6, 16, 3, 16, 30, 16, 3, 6, 6, 3, 3, 6, 30, 16, 3, 3, 6, 6, 30, 6, 3, 16, 6, 30, 3, 6, 6, 6, 30, 30, 16, 3, 6, 6, 30, 16, 3, 6, 6, 30, 30]] [435, [2, 2, 11, 2, 45, 9, 9, 9, 45, 11, 45, 2, 45, 45, 2, 11, 45, 9, 2, 2, 45, 45, 11, 11, 45, 45, 9, 11, 9, 45, 45, 11, 11, 2, 9, 9, 11, 9, 2, 45, 45, 2, 9, 11, 45, 2, 9, 2, 2, 2, 2, 2, 45, 11, 9, 9, 45, 9, 2, 2, 9, 2]] [809, [4, 4, 33, 4, 4, 3, 3, 4, 24, 3, 4, 24, 4, 33, 33, 4, 24, 3, 3, 4, 24, 4, 33, 24, 33, 24, 24, 4, 4, 4, 4, 33, 4, 24, 3, 24, 24, 24, 24, 33, 33, 3, 4, 24, 3, 33, 3, 33, 3, 33, 24, 4, 4]] [626, [8, 8, 37, 8, 8, 37, 22, 8, 22, 8, 8, 8, 2, 2, 37, 8, 37, 22, 22, 37, 8, 22, 37, 22, 37, 8, 2, 2, 37, 22, 22, 22, 8, 8, 8, 22, 37, 22, 8, 8, 2, 22, 22, 2, 8, 22, 2, 2, 8, 8, 2, 22, 37, 2, 37, 22, 8, 37, 37]] [218, [1, 19, 1, 48, 48, 48, 48, 19, 10, 10, 48, 19, 19, 19, 48, 48, 10, 10, 19, 10, 48, 19, 19, 1, 48, 10, 10, 10, 19, 48, 48, 48, 48, 1, 10, 19, 10, 10, 48, 10, 1, 48, 19, 10, 10, 10, 48, 19, 10, 10, 19, 1, 19, 19, 19, 19, 1, 1, 48, 1, 19, 48, 48, 1, 1, 19]]

After gathering our randomly generated input, we parsed it and put it into txt files which can be read appropriately by our algorithms: “*target \n coin1, coin2, coin3, … , coinN”.* Repository naming conventions are *ex. data_small1_jim.txt.* Below are what the txt files look like:

Input Files used for Independent	
Input Text File	Content
small1.txt	10 50, 5, 24, 5, 24, 24, 2
small2.txt	19 1, 16, 39, 16, 16, 7, 7, 7, 16, 7, 16, 1, 7
small3.txt	6 6, 40, 6, 40, 6, 40, 40, 2, 6, 18, 18, 40, 2
small4.txt	40 27, 2, 15, 15, 15, 8, 27, 15
small5.txt	20 16, 16, 16, 6, 1, 6, 16, 1, 1, 1
medium1.txt	300 10, 29, 2, 12, 10, 10, 2, 12, 29, 12, 2, 2, 29, 12, 2, 2, 10, 10, 12, 29, 2, 10, 10, 29, 10, 12, 29
medium2.txt	402 3, 9, 9, 3, 24, 3, 35, 24, 9, 3, 9, 3, 9, 24, 3, 35, 35, 24, 3, 3, 35, 35, 24, 9, 24, 35
medium3.txt	495 49, 11, 9, 49, 1, 11, 49, 9, 9, 11, 1, 11, 1, 1, 11, 1, 49, 11, 11, 9, 49, 1, 49, 9, 49, 11, 9
medium4.txt	364 21, 6, 21, 31, 3, 31, 3, 31, 3, 31, 6, 3, 21, 3, 3, 31, 21, 6, 31, 6, 3, 31, 6, 6, 31, 21, 6, 21, 6, 3, 21
medium5.txt	227 47, 3, 9, 3, 47, 13, 3, 9, 9, 13, 3, 9, 47, 13, 3, 13, 47, 9, 47, 9, 13, 13, 13, 9, 9, 3, 47, 47, 13, 3
large1.txt	812 30, 16, 16, 3, 30, 3, 6, 6, 16, 30, 16, 3, 3, 3, 6, 16, 3, 16, 30, 16, 3, 6, 6, 3, 3, 6, 30, 16, 3, 3, 6, 6, 30, 6, 3, 16, 6, 30, 3, 6, 6, 6, 30, 30, 16, 3, 6, 6, 30, 16, 3, 6, 6, 30, 30
large2.txt	435 2, 2, 11, 2, 45, 9, 9, 9, 45, 11, 45, 2, 45, 45, 2, 11, 45, 9, 2, 2, 45, 45, 11, 11, 45, 45, 9, 11, 9, 45, 45, 11, 11, 2, 9, 9, 11, 9, 2, 45, 45, 2, 9, 11, 45, 2, 9, 2, 2, 2, 2, 2, 45, 11, 9, 9, 45, 9, 2, 2, 9, 2
large3.txt	809

	4, 4, 33, 4, 4, 3, 3, 4, 24, 3, 4, 24, 4, 33, 33, 4, 24, 3, 3, 4, 24, 4, 33, 24, 33, 24, 24, 4, 4, 4, 4, 33, 4, 24, 3, 24, 24, 24, 24, 33, 33, 3, 4, 24, 3, 33, 3, 33, 3, 33, 24, 4, 4
large4.txt	626 8, 8, 37, 8, 8, 37, 22, 8, 22, 8, 8, 8, 2, 2, 37, 8, 37, 22, 22, 37, 8, 22, 37, 22, 37, 8, 2, 2, 37, 22, 22, 22, 8, 8, 8, 22, 37, 22, 8, 8, 2, 22, 22, 2, 8, 22, 2, 2, 8, 8, 2, 22, 37, 2, 37, 22, 8, 37, 37
large5.txt	218 1, 19, 1, 48, 48, 48, 48, 19, 10, 10, 48, 19, 19, 19, 48, 48, 10, 10, 19, 10, 48, 19, 19, 1, 48, 10, 10, 10, 19, 48, 48, 48, 48, 1, 10, 19, 10, 10, 48, 10, 1, 48, 19, 10, 10, 10, 48, 19, 10, 10, 19, 1, 19, 19, 19, 19, 1, 1, 48, 1, 19, 48, 48, 1, 1, 19

OUTPUT

We ran the test files and calculated the execution time for each algorithm, by running *time ./algorithm.py < input.txt* in the terminal and recorded the output below. If the time exceeded 20 minutes, we stopped the program. The user time is the most relevant to us in order to evaluate the efficiency and speed of our algorithms.

Input, Output, and Execution Time for Brute Force Algorithm time ./bruteforce_jim.py < input.txt			
Size of Test	Input File	Output of Program	Ex. Time (s)
small	small1.txt	Target: 10 [5, 5]	real 0m0.123s user 0m0.033s sys 0m0.009s
	small2.txt	Target: 19 No possible combination.	real 0m0.050s user 0m0.035s sys 0m0.009s
	small3.txt	Target: 6 [6]	real 0m0.051s user 0m0.043s sys 0m0.004s

	small4.txt	Target: 40 [2, 15, 8, 15]	real 0m0.102s user 0m0.034s sys 0m0.008s
	small5.txt	Target: 20 [1, 16, 1, 1, 1]	real 0m0.046s user 0m0.031s sys 0m0.011s
medium	medium1.txt	Target: 300 [29, 12, 12, 29, 12, 29, 12, 2, 10, 10, 12, 29, 2, 10, 10, 29, 10, 12, 29]	real 2m1.933s user 2m1.589s sys 0m0.012s
	medium2.txt	Target: 402 [9, 24, 35, 24, 9, 9, 9, 24, 35, 35, 24, 3, 35, 35, 24, 9, 24, 35]	real 1m3.143s user 1m1.543s sys 0m0.014s
	medium3.txt	Target: 495 No possible combination.	real 1m55.552s user 1m55.128s sys 0m0.054s
	medium4.txt	**program stopped: exceeded 20 minutes**	X
	medium5.txt	Target: 227 [47, 47, 13, 13, 47, 47, 13]	real 14m37.424s user 14m34.526s sys 0m0.012s
large	large1.txt	**program stopped: exceeded 20 minutes**	X
	large2.txt	**program stopped: exceeded 20 minutes**	X
	large3.txt	**program stopped: exceeded 20 minutes**	X

	large4.txt	**program stopped: exceeded 20 minutes**	X
	large5.txt	**program stopped: exceeded 20 minutes**	X

Input, Output, and Execution Time for Pruning Algorithm time ./pruning_jim.py < input.txt			
Size of Test	Input File	Output of Program	Ex. Time (s)
small	small1.txt	Target: 10 [5, 5]	real 0m0.045s user 0m0.032s sys 0m0.010s
	small2.txt	Target: 19 No possible combination.	real 0m0.051s user 0m0.039s sys 0m0.006s
	small3.txt	Target: 6 [6]	real 0m0.049s user 0m0.031s sys 0m0.009s
	small4.txt	Target: 40 [2, 15, 8, 15]	real 0m0.047s user 0m0.034s sys 0m0.009s
	small5.txt	Target: 20 [1, 16, 1, 1, 1]	real 0m0.095s user 0m0.031s sys 0m0.011s
medium	medium1.txt	Target: 300 [29, 12, 12, 29, 12, 29, 12, 2, 10, 10, 12, 29, 2, 10, 10, 29, 10, 12, 29]	real 2m37.180s user 2m36.625s sys 0m0.021s

	medium2.txt	Target: 402 [9, 24, 35, 24, 9, 9, 9, 24, 35, 35, 24, 3, 35, 35, 24, 9, 24, 35]	real 1m5.629s user 1m5.362s sys 0m0.008s
	medium3.txt	Target: 495 No possible combination.	real 2m21.696s user 2m21.206s sys 0m0.021s
	medium4.txt	**program stopped: exceeded 20 minutes**	X
	medium5.txt	Target: 227 [47, 47, 13, 13, 47, 47, 13]	real 7m12.435s user 7m11.097s sys 0m0.013s
large	large1.txt	**program stopped: exceeded 20 minutes**	X
	large2.txt	**program stopped: exceeded 20 minutes**	X
	large3.txt	**program stopped: exceeded 20 minutes**	X
	large4.txt	**program stopped: exceeded 20 minutes**	X
	large5.txt	**program stopped: exceeded 20 minutes**	X

Input, Output, and Execution Time for Dynamic Programming Algorithm time ./dynamicprogramming_jim.py < input.txt			
Size of Test	Input File	Output of Program	Ex. Time (s)
small	small1.txt	Target: 10 [5, 5]	real 0m0.046s user 0m0.031s sys 0m0.011s
	small2.txt	Target: 19 No possible combination.	real 0m0.101s user 0m0.032s sys

			0m0.010s
	small3.txt	Target: 6 [6]	real 0m0.122s user 0m0.032s sys 0m0.012s
	small4.txt	Target: 40 [2, 8, 15, 15]	real 0m0.048s user 0m0.032s sys 0m0.007s
	small5.txt	Target: 20 [1, 1, 1, 1, 16]	real 0m0.100s user 0m0.034s sys 0m0.008s
medium	medium1.txt	Target: 300 [2, 12, 12, 12, 12, 29, 29, 29, 29, 29, 12, 12, 10, 10, 10, 10, 10, 2, 29]	real 0m0.044s user 0m0.031s sys 0m0.010s
	medium2.txt	Target: 402 [3, 9, 9, 24, 24, 24, 35, 35, 35, 35, 24, 24, 24, 9, 9, 9, 35, 35]	real 0m0.113s user 0m0.037s sys 0m0.007s
	medium3.txt	Target: 495 No possible combination.	real 0m0.069s user 0m0.035s sys 0m0.009s
	medium4.txt	Target: 364 [21, 21, 21, 21, 31, 31, 31, 31, 31, 31, 21, 21, 21, 31]	real 0m0.100s user 0m0.035s sys 0m0.009s

	medium5.txt	Target: 227 [13, 13, 13, 47, 47, 47, 47]	real 0m0.043s user 0m0.031s sys 0m0.005s
large	large1.txt	Target: 812 No possible combination.	real 0m0.096s user 0m0.038s sys 0m0.007s
	large2.txt	Target: 435 [2, 2, 2, 2, 11, 11, 45, 45, 45, 45, 45, 45, 45, 45, 45]	real 0m0.046s user 0m0.034s sys 0m0.007s
	large3.txt	Target: 809 No possible combination.	real 0m0.051s user 0m0.038s sys 0m0.008s
	large4.txt	Target: 626 [2, 2, 2, 22, 22, 22, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 22, 22, 22, 22, 22]	real 0m0.104s user 0m0.032s sys 0m0.012s
	large5.txt	Target: 218 [1, 1, 1, 1, 1, 1, 1, 19, 48, 48, 48, 48]	real 0m0.045s user 0m0.034s sys 0m0.005s