Luke Erlewein

Dr. Denton Bobeldyk

CIS 263

7/19/2025

Assignment 7

Program Output

```
PS D:\Git Repos\CIS 263> python -u "d:\Git Repos\CIS 263\Homework\Week 4\Assignmentseven.py"
Number of activities in list: 25
Activity time range between 0 and 100

Brute Force: 8 activities selected in 47.95784 seconds
Greedy: 8 activities selected in 0.0000000 seconds

Number of items in array: 10000

Brute Force: max sum = 7194, time = 5.50209 seconds
Divide & Conquer: max sum = 7194, time = 0.03756 seconds
PS D:\Git Repos\CIS 263>
```

Figure 1: Output of the program after running. With the same N value for greedy and brute force.

Greedy Analysis

After running both the brute force method and the greedy method of the activity selection problem, it can be determined that the Greedy method is a much faster method. As can be seen in Figure 1 where the greedy method is virtually instantaneous. As there are 25 activities in the list, the brute force method takes much longer to run through as it need to run through, I believe if I am correct 2^{25} Or 33,554,432 possible scenarios. It checks all possible routes that it could take before choosing the best one. At the same time, the greedy method chooses the number of activities based on predetermined scenarios, so it only checks each of the activities once before determining the route it wants to take. To show that the greedy algorithm is much faster and more efficient. I changed the amount of the activities for the greedy method to 2,500. In Figure 2,

you can see that the amount of time it took to analyze a list of 2,500 activities was much faster than the time it took for the brute force method to analyze a list of only 25 activities.

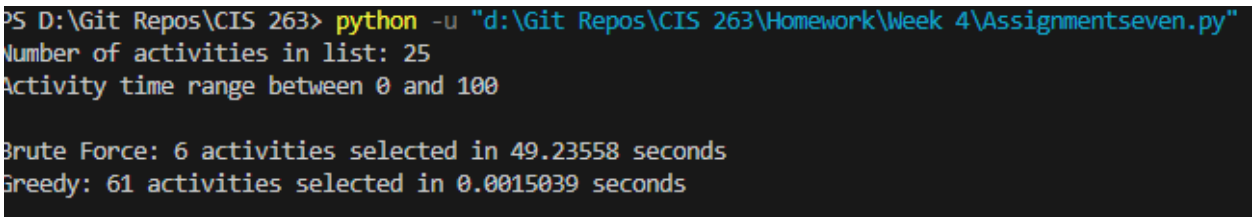

Figure 2: 2,500 activity list for comparison.

Divide and Conquer Analysis

For the divide and conquer method. It indeed was faster than the brute force method. It did not optimize as much as the greedy method, but that is because it is looking at a different kind of problem. While both methods yielded the same correct answer, comparing the completion times reveals that the divide and conquer method is much faster. This is because the brute force method does more comparisons; it will run through every single possible combination. The divide and conquer method takes a different approach by halving the number of comparisons it is doing every time it recurses through the program. This recursive halving results in a complexity that is more efficient than the brute force method. As can be seen in Figure 1, the divide and conquer method is 146.5 times faster than the brute force method.