Jordan Edginton

CS-415 SP ‘21

Project 4 report: task 3

**Task #1**

I had a feeling that the memory function version would require less operations than the traditional method since we don’t need to calculate the entire table. What surprises me though, is that 07 has less operations for the traditional version, albeit it’s only ~2% difference. Perhaps I had an extra counter where I shouldn’t have put it. The only other reason I can think of is that the memory function had to fill out most cells on top of doing the recursive calls and operations.

Overall, the memory function is superior to the traditional in most cases.

**Task #2**

I was surprised how much shorter this was compared to the dynamic programing functions, even if they didn’t give the correct output (still was close though each time).

The difference in algorithms is… interesting. The heap is better than the regular greedy until 07 and 08 (I separated them since their numbers are too great and you can’t see the smaller numbers on the graph). I had to go and double check my outputs to make sure. My guess is that the cost to reheapify (I don’t know what word to use) increases with more items in the input and will eventually overtake the nlogn sorting algorithm used for the regular greedy (mine was merge sort). It’s the difference of doing an O(nlogn) algorithm once vs an O(n) multiple times.

Overall, for smaller inputs the heap is more efficient, but for bigger inputs the greedy algorithm with the nlogn sorting is more efficient.

**Values for the graphs**

Task 1a:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 63 | 3657 | 327 | 2677 | 818 | 1896 | 2739 | 24040 | 320209062 |

Task 1b:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 82 | 2500 | 267 | 1528 | 771 | 1525 | 1681 | 24643 | 187631722 |

Task 2a:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 48 | 158 | 66 | 82 | 101 | 119 | 102 | 276 | 491 |

Task 2b:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 45 | 152 | 60 | 62 | 74 | 84 | 92 | 405 | 990 |