COT 5405

Programming Assignment 3

Submitted by

Rajib Dey

Chapter 4, Problem #6

The problem

- Your friend is working as a camp counselor, and he is in charge of organizing activities for a set of junior-high-school-age campers. One of his plans is the following mini-triathalon exercise: each contestant must swim 20 laps of a pool, then bike 10 miles, then run 3 miles. The plan is to send the contestants out in a staggered fashion, via the following rule: the contestants must use the pool one at a time. In other words, first one contestant swims the 20 laps, gets out, and starts biking. As soon as this first person is out of the pool, a second contestant begins swimming the 20 laps; as soon as he or she is out and starts biking, a third contestant begins swimming . . . and soon.) Each contestant has a projected swimming time (the expected time it will take him or her to complete the 20 laps), a projected biking time (the expected time it will take him or her to complete the 10 miles of bicycling), and a projected running time (the time it will take him or her to complete the 3 miles of running). Your friend wants to decide on a schedule for the triathalon: an order in which to sequence the starts of the contestants. Let’s say that the completion time of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathalon, assuming they each spend exactly their projected swimming, biking, and running times on the three parts. (Again, note that participants can bike and run simultaneously, but at most one person can be in the pool at any time.) What’s the best order for sending people out, if one wants the whole competition to be over as early as possible? More precisely, give an efficient algorithm that produces a schedule whose completion time is as small as possible

Test Cases with Brute-Force correct answer

```
Random Generated Set ======

 set([(37, 69, 35), (17, 16, 8), (98, 89, 17)])

Combination number 1 =

((37, 69, 35), (17, 16, 8), (98, 89, 17))

Time needed for this combination is ===  258


Combination number 2 =

((37, 69, 35), (98, 89, 17), (17, 16, 8))

Time needed for this combination is ===  241


Combination number 3 =

((17, 16, 8), (37, 69, 35), (98, 89, 17))

Time needed for this combination is ===  258


Combination number 4 =

((17, 16, 8), (98, 89, 17), (37, 69, 35))

Time needed for this combination is ===  256


Combination number 5 =

((98, 89, 17), (37, 69, 35), (17, 16, 8))

Time needed for this combination is ===  239


Combination number 6 =

((98, 89, 17), (17, 16, 8), (37, 69, 35))

Time needed for this combination is ===  256


List of all the time values for this set===

[258, 241, 258, 256, 239, 256]

Best Time for this set--according
to BruteForce Method====================  239
```

As the lowest time among all of the time was chosen by the Brute-Force Method, Correctness of the Test case is Validated

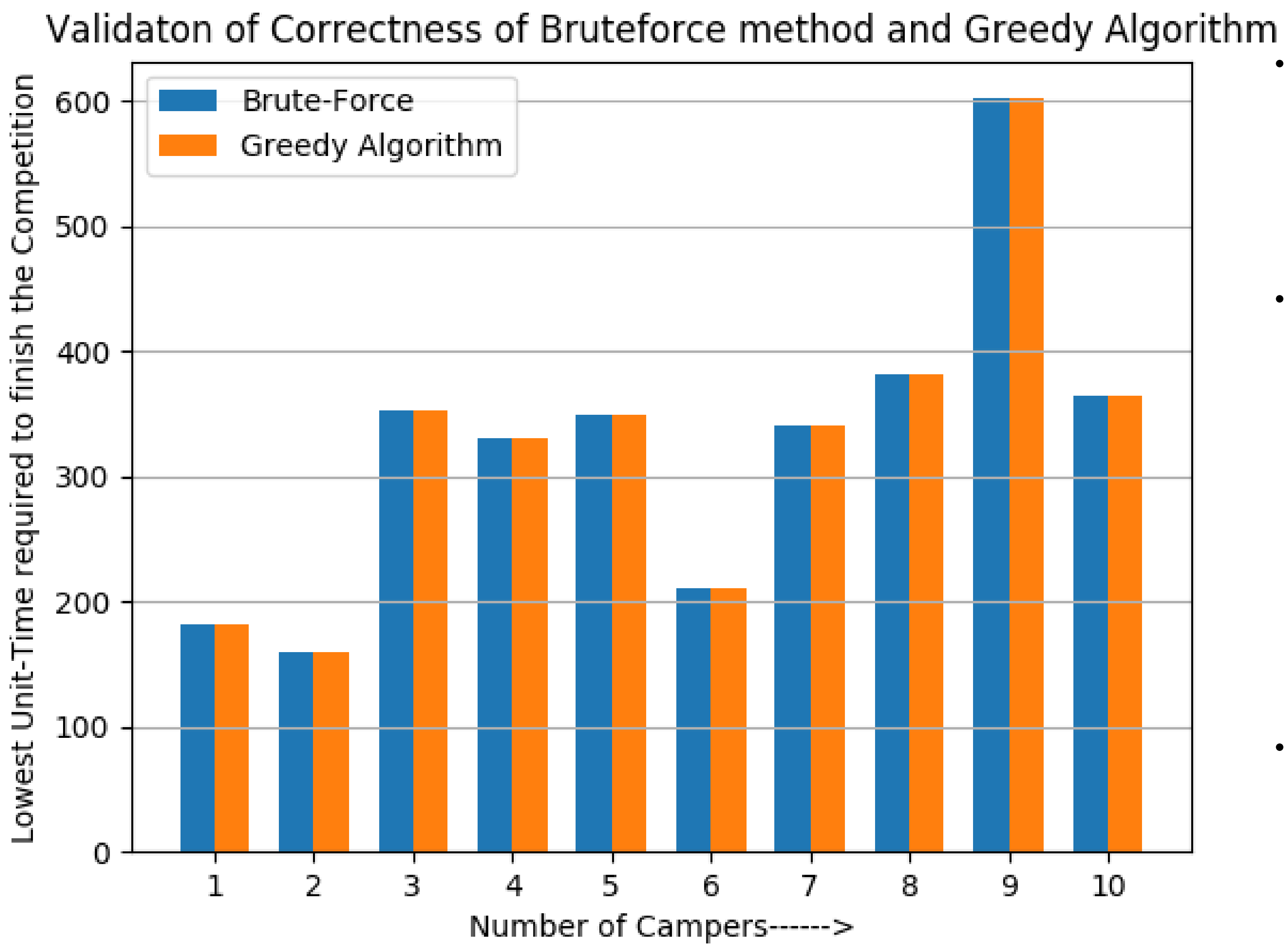
```
According to Bruteforce Method, The best combination is

((98, 89, 17), (37, 69, 35), (17, 16, 8))
```

Note: As printout of the test cases of n=1 to n=10 is a massive text file (around 888MB), a printout of test cases of n=1 to n=7 is supplied along with the source code. Look for the filename TestCase-BruteForce-Correct Answer.txt . Here only one set with 3 campers is shown as an example.

Pseudocode Of the Greedy Algorithm

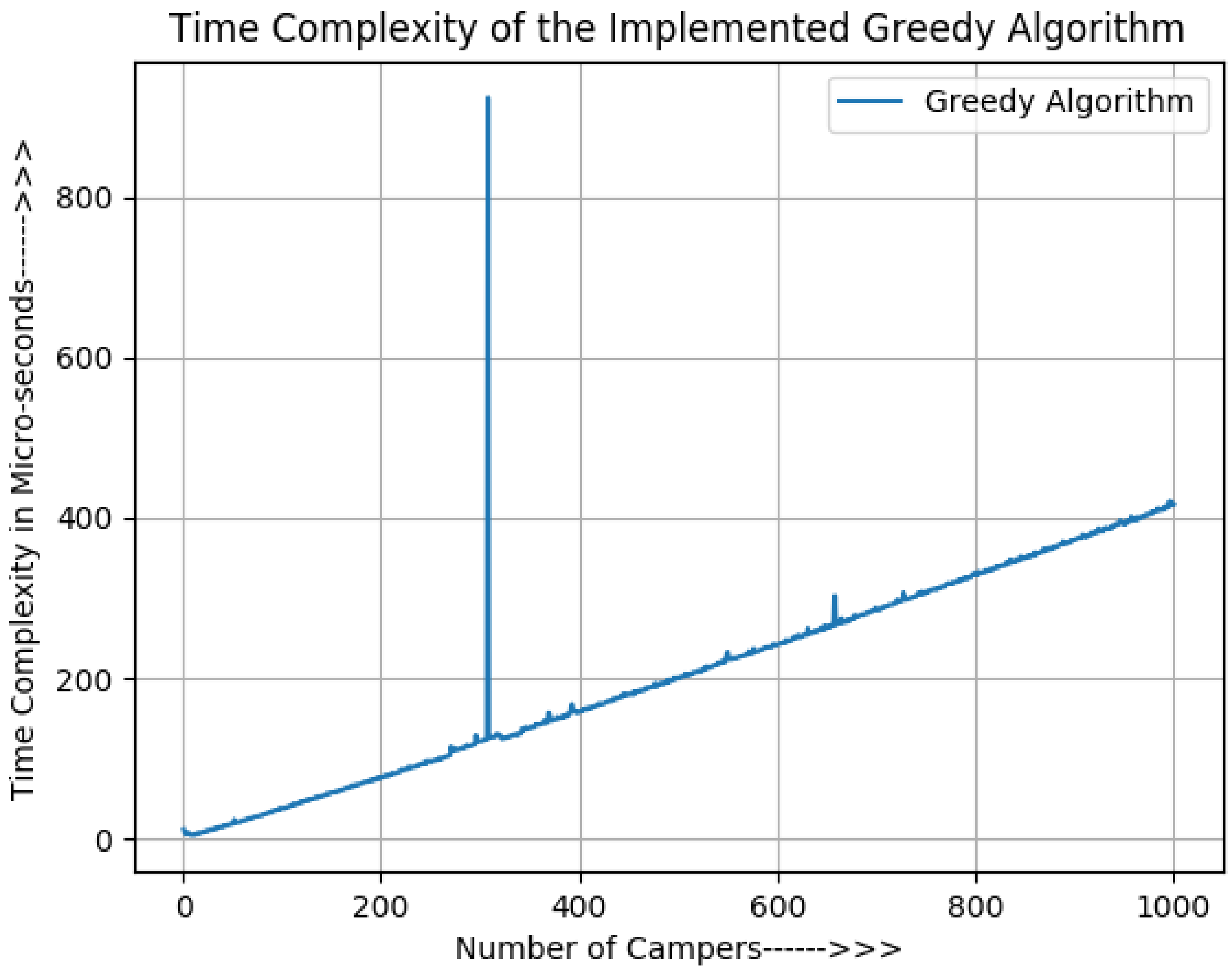
- If each of the campers have a swimming, biking and running times of si, bi and ri.
- Then according to the Greedy algorithm designed, we will sort the campers in decreasing order of [bi+ri] and while sending them we will follow this order.
- According to few tests that I did, it seems that this algorithm will always give us the lowest completion time for the whole competition.
- Even when there are multiple orders exists which can give us the lowest completion time(we can see this using Brute-Force method), this algorithm will never give us a wrong order. It will choose one of the correct order.
- Time complexity of this algorithm is O(n log n) . As we are just sorting the campers in decreasing order of [bi+ri]. Which would be visible when you look at the image (Time Complexity of Greedy Algorithm.png) produced by the Program.
- **Example:** Let, there are 3 campers with a set of individual times of (8,4,9),(3,5,7),(5,8,9). Where first, second and the third number represent their individual swimming, biking and running times respectively. In our algorithm we will add the last 2 numbers from each of these sets and sort them in decreasing order.
  This order should give us the lowest completion time for the competition.
  Here, that order is (5,8,9),(8,4,9),(3,5,7). As, 8+9=17,4+9=13 and 5+7=12. Which will give us the lowest completion time of 28. So, According to the Greedy Algorithm designed, we will have to send the 3rd camper first then the 1st camper and then the 2nd camper should go into the pool.

Validation of Correctness of the Greedy Algorithm compared to Brute-Force for Test Cases



- As we can see, for test cases when the number of campers is 1 to 10, the Brute-Force method and the Greedy Algorithm implemented always produces the same result. So, The Correctness of the Algorithm is validated for the test cases.
- A copy of the printout for this test is provided with the source code. This will show you that for each test cases what is the lowest unit time calculated by both methods and these time values are stored in separate lists called *LowestTimeGreedy* and *LowestTimeBruteForce* . You will see the list growing with the growing numbers of campers. Please look for the file Validation-Correctness-Test-Cases.txt
- Every time the code is run a new version of test cases will be generated and this image will be generated comparing the lowest time for Brute-Force and Greedy Algorithm.

Validation of performance for the Greedy Algorithm



- Time taken to calculate the lowest time to finish the competition for the Greedy Algorithm is plotted against the number of campers. Here, the number of campers is 1 to 1000.
- Time complexity of this algorithm is O(n log n). Which is visible from the graph shown.
- A few aberrations exists in the graph, but it follows the (n log n) curve right after that. Even with the aberrations, the calculations happen in Micro-seconds. Where for the Brute-Force method, calculating n=1000 might take days. As 1000! Is a number that has 2610 digits. Which means the Brute-Force method would have to go through that many combinations.
- A copy of this graph is also provided along with the source code. Please look for the file Time Complexity of Greedy Algorithm.png
- Every time the code is run a new version of random cases for n=1 to n=1000 will be generated and this image will be generated which plots the time taken for the greedy algorithm to calculate the lowest possible completion time for each set vs the Number of campers for that set.