CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean
Webpage: `http://www.cs.clemson.edu/~bcdean/`
Handout 8: Lab #6

Fall 2020
MWF 9:05-9:55
Flour 132

1 Dividing and Conquering an Epic Midterm Exam

Prof. Dean has been pondering new fun ways to potentially design the midterm exam for CpSc 2120. He is particularly proud of his latest idea: a non-stop week-long tag-team take-home exam! The exam would be released on Monday morning (right at midnight), and due the next Monday (also right at midnight).

The N students in the class would work on the exam as one large team, but in individual shifts. The first student would tackle the exam for a certain amount of time, then pass the results so far to the second student, and so on. The students would divide up the entire week into N blocks of time, each assigned to a different student, where the student working in one block of time would pass results to the student working in the next block of time. Blocks of time cannot overlap, as only one student is allowed to be working at any given time.

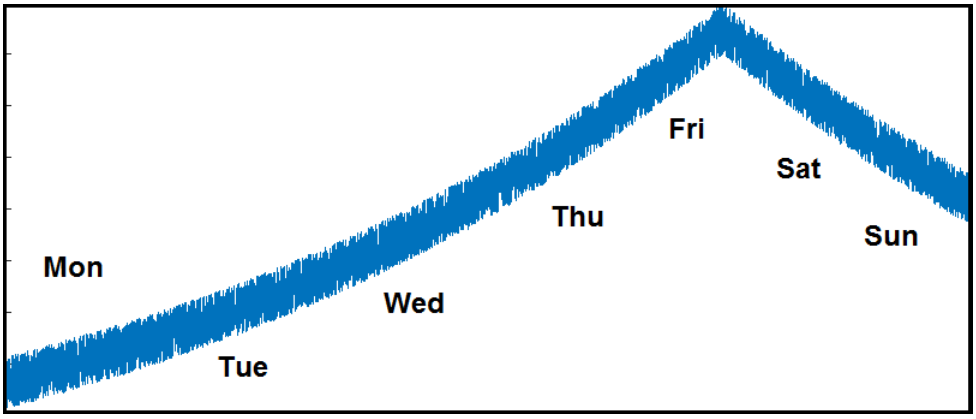

Figure 1: Michael's productivity rate during every individual minute over the course of the week.

Depending on the student's schedule, a student can be more productive at some points in time during the week than at other points in time. For example, Figure 1 shows Michael's rate of productivity over the course of the week, which, despite some random fluctuation, slowly increases from Monday through Friday. However, on Friday afternoon, sad that CpSc 2120 lectures for the week are done, his rate of productivity diminishes through the rest of the weekend. It would therefore make sense to assign Michael a block of time during which his productivity rate is high.

2 Input and Output

The productivity rate of every student over an entire week is given in the file:

`/group/course/cpsc212/f20/lab06/productivity.txt`

Each of the N lines of this file corresponds to a single student, giving the name of the student followed by $7 \times 24 \times 60$ doubles. The ith double in this list gives the student's rate of productivity from minute $i-1$ to minute i during the week (a student's productivity rate is constant during these minute-long intervals).

Your goal is to assign each student to a contiguous block of time during the week, so that:

- No two blocks overlap, and
- The total productivity of each student during their block is at least $1/N$ of their total productivity during the entire week, so each student is performing "better than average" during their time block.

3 Algorithm Design

This problem can be solved with an elegant application of the "divide and conquer" principle. In fact, it uses *three* distinct "layers" of divide and conquer[1]: one that breaks up the time range of the week by successive halving, one that performs selection to find a median point suitable for division in the outer algorithm, and finally a binary search. The expected running time is $O(N \log N)$ in terms of its dependence on N (the running time also has some dependence on the complexity of the productivity time series for each student, but we'll focus here on how it scales with N). For simplicity, we'll assume N is a power of 2 in this lab ($N = 64$ for the input file provided[2]). This isn't a fundamental assumption at all — the algorithm can be easily adapted to work for arbitrary values of N, but the details just get a tiny bit messier.

Full details of the overall algorithm will be discussed at the beginning of lab. For a bird's-eye view of the process, it starts with a problem involving N students and a time range T, which it divides into two recursively-solved subproblems: an "early" subproblem involving $N/2$ students and the beginning part of T, and a "late" subproblem involving the remaining $N/2$ students and the later part of T. As a base case, when we reach a subproblem with a single student and a short time range T, we assign T as the block of time for that student.

3.1 Subtask 1: Binary Search to Compute Halfway Cutoffs

To divide the N students into the "early" group of size $N/2$ and the "late" group of size $N/2$, we first compute for each student their *halfway* cutoff point — that value of time t such that exactly half their productivity within T occurs by time t. Your first task is to fill in the function `get_halfway_cutoff` that accomplishes this via binary search on t. Conveniently, you are provided with a function `get_productivity` that computes the total productivity of a student over any range

[1] A divide and conquer "Turducken"?

[2] There are 68 students currently in the class, but four of them are named Jacob, so the most convenient way to reduce the input file to 64 students was to pass it through a de-Jacob-fication filter.

of time. Using this, you can easily tell if a prospective value for t is too high or too low, enabling effective binary search.

3.2 Subtask 2: Selection

After computing the halfway cutoff points for all the N students in our current problem, we want to find their median[3]. For this, you should implement the randomized quickselect algorithm as described in class. Empty functions for quickselect and its partitioning subroutine are provided for you to fill in. The starter code also provides a function that helps in testing the output of quickselect.

4 Grading

For this lab, you will receive 8 points for correctness and 2 points for having well-organized, readable code. Zero points will be awarded for code that does not compile, so make sure your code compiles on the lab machines before submitting!

Final submissions are due by 11:59pm on the evening of Monday, October 12. No late submissions will be accepted.

[3]Note that N is even, so there are technically two medians, at ranks $N/2-1$ and $N/2$. Let's find the first of these — i.e., if our elements have ranks $0 \ldots 7$, we want to find the element of rank $N/2-1=3$.