Name: Jon Abrahamson
ID: 107084898

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 1 **Spring 2019, CU-Boulder**

Your colleagues from the biology department have reached out to you for help. They are trying to complete an analysis for an important study, but the program they have written is taking too long to complete. Here is their email:

Dear Computer Scientist I know,

We are studying a specific set of genes that we think play some role in chemoresistance. To test how these genes respond to chemo, we tested the activity of all genes (i.e., the expression) before and after treatment in both normal healthy skin tissue and the tumor. We now need to measure how these genes respond to treatment by comparing their expression before and after being exposed to the chemo. Our data includes

- A list of genes we are studying
- A list of all genes
- Gene expression before treatment in normal skin tissue (e.g., TP53 5.1, BCRA1 3.1,).
- Gene expression before treatment in the tumor (e.g., TP53 6.2, BCRA1 4.1,).
- Gene expression after treatment in normal skin tissue (e.g., TP53 10.5, BCRA1 2.1,).
- Gene expression after treatment in the tumor (e.g., TP53 20.5, BCRA1 1.1,).

For each gene on our list, we want to find the difference in expression from before and after treatment, then take the mean of differences for the tumor and for the normal skin tissues. To compute these values, we take one gene from our list and scan the four expression files until we find the matching gene name. We then use the corresponding four expression values to compute the two differential values. We repeat this for all of the genes we care about, then find the mean of the normal and tumor values. While this seems to work, we would like to test many different gene sets, but our current program is taking too long. Please help.

Thanks in advance,

Your colleague in the bio department

Name: Jon Abrahamson
ID: 107084898

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 1 **Spring 2019, CU-Boulder**

Help your colleagues understand why their method is taking so long, suggest to them a different algorithm and explain why they can expect this new method to run faster.

While the current algorithm may work, it is very time ineffective. The algorithm is required to search through all four expression files to find matches, one gene at a time In other words, for every gene, there are four separate loops that must be ran in order to find a match in their respective files. This causes lots of wasted time looking through files that are presumably unsorted. Finally, another issue with this algorithm is that we are doing one gene at a time. We are not simultaneously running multiple trials and so we must wait to run the the same gene before chemo and when we have found the matching expressions, then run it again after chemo finding the matching expressions. From there the difference in expressions can be found due to the chemo treatment. This is tedious because each single gene must run through four loops twice, so eight total loops until you can determine the effect of chemotherapy on the genes.
Alternately we should approach the problem first in a way to obtain the most parallel algorithm we can, meaning we can run more than one gene at the same time and simultaneously look for matching gene expressions in the four files. Assuming that the processes are mutually independent and that you are repeating the same process with all genes, we should be able to drastically parallelize the algorithm speeding up the run time. Next, we can sort the four expression files in ascending order. If the files are sorted in ascending order we can run a binary search algorithm that instead of running through each individual data point one by one checking for a match like in the original algorithm, cut the data set in half until a match is found. The worst case scenario for a binary search is taking lg(n) time to sort through the entire file and not finding the matching expression, with n being the size of the data set. Alternatively, in the previous algorithm, a sequential search was probably being used that has a worst case scenario of (n) time to sort through the entire expression file and not find a match. Looking at the rate of growth of these two equation, it is clear that as n, the data set size, continues to increase, lg(n) results in a far less amount of time than (n), the original algorithm's search structure. So with large data sets it is obvious that applying a binary search algorithm to search for matches between the gene and the expression files is the most efficient. This is also applied to all four of the expression files so with more genes mean far more expression files to run through meaning more and more times the search algorithm needs to be used to look for matches. In the end this original algorithm can be made far more efficient by sorted the files, applying a binary search algorithm to search for matches between the genes and the expression files, and finally by making a more parallel algorithm, meaning making it so more loops/trials can be ran at the same time.