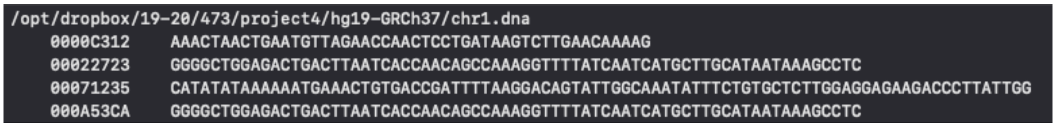
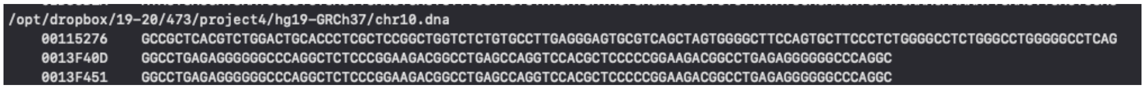
Juliana McCausland
Ling 473
Project 4

Results:



```
/opt/dropbox/19-20/473/project4/hg19-GRCh37/chr1.dna
    0000C312     AAACTAACTGAATGTTAGAACCAACTCCTGATAAGTCTTGAACAAAAG
    00022723     GGGGCTGGAGACTGACTTAATCACCAACAGCCAAAGGTTTTATCAATCATGCTTGCATAATAAAGCCTC
    00071235     CATATATAAAAAATGAAACTGTGACCGATTTTAAGGACAGTATTGGCAAATATTTCTGTGCTCTTGGAGGAGAAGACCCTTATTGG
    000A53CA     GGGGCTGGAGACTGACTTAATCACCAACAGCCAAAGGTTTTATCAATCATGCTTGCATAATAAAGCCTC
```

…



```
/opt/dropbox/19-20/473/project4/hg19-GRCh37/chr10.dna
    00115276    GCCGCTCACGTCTGGACTGCACCCTCGCTCCGGCTGGTCTCTGTGCCTTGAGGGAGTGCGTCAGCTAGTGGGGCTTCCAGTGCTTCCCTCTGGGGCCTCTGGGCCTGGGGGCCTCAG
    0013F40D    GGCCTGAGAGGGGGGCCCAGGCTCTCCCGGAAGACGGCCTGAGCCAGGTCCACGCTCCCCCGGAAGACGGCCTGAGAGGGGGGCCCAGGC
    0013F451    GGCCTGAGAGGGGGGCCCAGGCTCTCCCGGAAGACGGCCTGAGCCAGGTCCACGCTCCCCCGGAAGACGGCCTGAGAGGGGGGCCCAGGC
```

…
etc.

Description:

To execute the program: $ ./wrap.sh

This was the most challenging project thus far. The code runs as described in the project spec; the target sequences are loaded into a trie before the program filters through each file to search for matching sequences from the trie. I also managed to get the extra credit in there, which was less difficult than I expected.

I encountered many problems while completing this project. The primary obstacle was optimization. I originally looked into using defaultdict() for optimization purposes, but I could not get it to work. In an attempt to ensure a fast-ish runtime, I tried to avoid unnecessary variables, and I wanted to use dictionaries, and avoid any procedures that may consume significant time. I also read files as bytes using ‘rb’. Despite my efforts, my original code took about 12 hours to run. I managed to cut that down to 6 hours through more careful editing of my code, but that was still too long.

I also tried my hand at parallel processing. After days of failed attempts, I started to see some progress. I eventually managed to manipulate my condor.cmd file in such a way that it simultaneously submits 24 jobs (one for each file). I also included a wrap.sh file, which invokes condor_submit, performs a condor_wait action, and then concatenates all output files to a final output.out file. With the parallel processing, the program takes about 45 minutes to run.

I hit a roadblock while concatenating the output files, which I later realized was due to the condor_wait command. When I invoke condor_submit using $ ./wrap.sh, the terminal waits for all the jobs to complete before allowing me to enter any commands into the terminal. I wanted to obsessively refresh condor_q, so I would close my terminal, reopen it, and check condor_q, along with my output files. In doing this, I was causing a disturbance to the wrap.sh file, ultimately terminating the concatenation process. Walking away from the computer once I submit the jobs to condor seemed to resolve this issue.

The program runs by the $ ./wrap.sh command. I didn’t realize that this may be the wrong way to run the program (according to the project description) until the very last minute, so that is something I would have liked to fix.

My issues with this project mostly stemmed from a lack of understanding in the areas of optimization, condor, and shell scripts. I learned a lot in the past couple of days alone, and I am grateful to have been forced to face this steep learning curve. I expect it will come in handy in the future!