Megan Chu
A12814536
Assignment 3
5/14/18

Q1.

Answers:

a) E(number L-mer matches) = m*(0.25^L)*n, the 0.25^L term represents the probability of L nucleotides matching. The m term (technically m-L+1) represents the number of positions these L nucleotides can be in the query. The n term (technically n-m+1) represents the number of positions the query can be in the random database.

b) If m is very large, the expected number of L-mer matches goes up since there are more possible L-mers in the query that can be compared with the database. As m decreases towards L, the expected number of L-mer matches decreases since there are less possible L-mers in the query that can be compared with the database. For example, if m = 10 and L = 2 then our expected number of L-mer matches, E1, is 10*(0.25^2)*n. If m decreases towards L, and supposing m = 3 and L = 2, then E2 = 3*(0.25^2)*n. Clearly E1 > E2 since 10 > 3. Thus for large m, E is high and as m decreases towards L, E decreases. One can think of this like a lottery, where m represents how many lottery tickets you have. The more lottery tickets (greater m) the more likely you are to win the lottery (more L-mer matches), and the less lottery tickets (m close to L) the less likely you are to win the lottery (less L-mer matches).

c) E(number L-mer matches for one query) = r*(0.25^L)*n, so the E(number L-mer matches for all queries) = $\sum_{m/r}$r*(.25^L)*n, simplifying the summation over all m/r queries gives us (m/r)*r*(.25^L)*n = m*(.25^L)*n. This is the exact same equation as when L-mer matches spanned multiple query strings. Thus we conclude that the expected number of L-mer matches does not change for matches spanning multiple query strings and matches not spanning multiple query strings.

d) Pr[C] = Pr[G]=0.4 and Pr[A] = Pr[T]=0.1
5-mer matches for query string AATAAGCCGC, with m = 10
E(5-mer matches) = P(match each 5-mer in the query)*n = [(0.1^5) + (0.1^4)(0.4) + (0.1^3)(0.4^2) + (0.1^2)(0.4^3) + (0.1^1)(0.4^4) + (0.4^5)]*n
= .01365*n is the expected number of 5-mer matches for the given query string.

Q2.
Note a code for this seed and extend strategy is provided in the Q2algorithm.py file, although its was not tested since there was no given queries and database to test it on.

Description of Algorithm used:

```
for each query of length L
     take the first w nucleotides as the key
     iterate through i positions the database
          if database at position i matches key
               compute global alignment b/w query and L nucleotides
               at position i in database (with m = 1, s = d = 0)
               if score >= .85*L (we have more than 85% matches)
                    add L nucleotides at position I in database as a
                    homolog for this query
```

Method to compute speed-up:
Local alignment runtime: O(nm) steps
From algorithm above we calculate runtime of seed-and-extend search, we assume keyword searching does not consume any time and that alignment computation is the time consuming step.
thus for seed-and-extend,
E(#hits with length w) = m*(.25^w)*n
Number computations = [m*(.25^w)*n]L*L for global alignment of 2 strings length L
thus speed-up = nm:nm(.25^w)LL => 1:(.25^w)LL ~ $[(.25^w)LL]^{-1}$

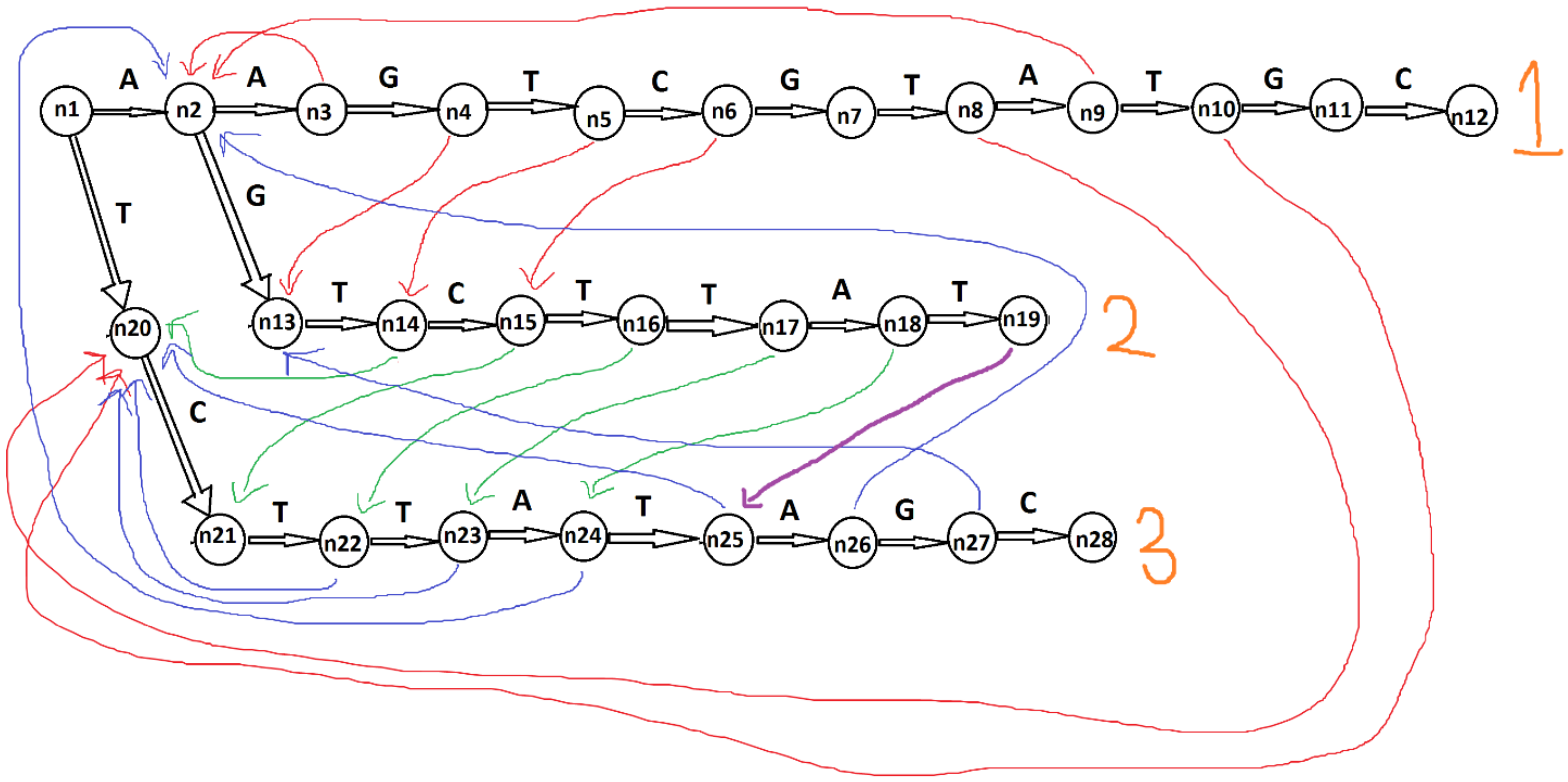
Method to compute sensitivity:
Sensitivity = Pr(any true homolog of any query is recovered)
Local alignment sensitivity = 1.0
Seed-and-extend sensitivity = ?
Assume that mismatches are randomly distributed and we are looking for matches with >= 85% sequence identity, then what is the probability that there is no string of length w where the query and the database match exactly? This is ~ (1-(.85)^w)^(L-w). Then what is the probability of an exact match of a string of length w between the query and the database? This is 1-(1-(.85^w))^(L-w), which is the seed-and-extend sensitivity.

Table with speed-up and sensitivity values (computed by Q2.py code):
Assume n = m = 10^7, L = 100.

w	Speed-up = 1:(.25^w)LL	Sensitivity = 1-(1-.85^w)^(L-w)
5	1:9.766 ~ .1024	1 <- rounds to 1
11	1:.00238 ~ 419	.9999999165
15	1:9.313E-6 ~ 107374	.9995776317
20	1:9.095E-9 ~ 109951163	.957678239
25	1:8.882E-12 ~ 1.126E11	.7277557777
30	1:8.674E-15 ~ 1.153E14	.4150340252
35	1:8.470E-18 ~ 1.181E17	.1978430327
40	1:8.272E-21 ~ 1.209E20	.0862569282

Q3.

Note black arrows are transition links and colorful arrows are failure links, n1 is the root node. The purple arrow from n19 to n25 is used if you want to match more than one word (since if you have reached n19 you clearly have a match for word #2 and can stop if you are only looking for one match). Also note that the nodes are arbitrarily numbered for clarity (since we don't know what order they are traversed in unless given an actual string to find matches).



Q4.
Note, I will define a “hit” as matching a string to any one of the words (1, 2, or 3) in the trie above.
E-value

```
      = E(# hits to trie)
      = E(# hits to 1) + E(# hits to 2) + E(# hits to 3)
      = (.25^11)*(n−11+1) + (.25^8)*(n-8+1) + (.25^9)*(n-9+1)
      = (.25^11 + .25^8 + .25^9)*n by eliminating constants
      = (1.931190491E-5)*n
```

p-value

```
      = 1 – P(no hits)
      = 1 – P(no hits to 1 and no hits to 2 and no hits to 3)
      = 1 – [((1-.25^11)^n)*((1-.25^8)^n*((1-.25^9)^n))
      = 1 – ((1-.25^11)*(1-.25^8)*(1-.25^9))^n
      = 1 - .9999806882^n
```

At what size of the database are matches no longer significant? You must explain your reasoning in a concise fashion.
Matches are no longer significant when the p-value >= 0.05 (0.05 is the accepted cutoff for significance in the scientific community). The greater the p-value, the more likely a hit is due to random chance (since probability for at least one hit is high), the smaller the p-value, the less likely a hit is due to random chance (since probability for at least one hit is low). Using the p-value equation from above, we calculate the n at which the p-value = 0.05 = 1 - .9999806882^n

```
      ⇨ .95 = .9999806882^n
      ⇨ log(.95) = log(.9999806882^n)
      ⇨ .95 = nlog(.9999806882)
      ⇨ .95/log(.9999806882) = n
      ⇨ n = 2656.034088
```

Thus, when the size of the database reaches about 2656 bp, the matches are no longer significant.

Q5.
Note positions in database are numbered starting with 1 as the first nucleotide, 2 as the second nucleotide, and so on.
I used the following website for reference in setting up the trie data structure in python (https://towardsdatascience.com/implementing-a-trie-data-structure-in-python-in-less-than-100-lines-of-code-a877ea23c1a1)

Program output:
For queries.txt…
```
     Pattern 1 matches starting at position 3665
     Pattern 1 matches starting at position 494569
     Pattern 3 matches starting at position 501200
     Pattern 2 matches starting at position 501416
     Pattern 1 matches starting at position 2932571
     Pattern 1 matches starting at position 4362313
     Pattern 4 matches starting at position 4531797
     Pattern 4 matches starting at position 4801795
     Pattern 1 matches starting at position 5466370
     Pattern 1 matches starting at position 5494658
     Pattern 1 matches starting at position 6614019
     Pattern 4 matches starting at position 6795823
     Pattern 1 matches starting at position 6954236
     Pattern 4 matches starting at position 8569327
     Pattern 4 matches starting at position 8871750
     Pattern 4 matches starting at position 9025414
     Pattern 4 matches starting at position 9931926
     Pattern 4 matches starting at position 10391586
     Pattern 1 matches starting at position 10737587
     Pattern 1 matches starting at position 11917210
     Pattern 1 matches starting at position 12561690
     Pattern 1 matches starting at position 12619812
```

For queries2.txt (just the first few lines of output)…
```
     Pattern 1 matches starting at position 3665
     Pattern 395 matches starting at position 32004
     Pattern 319 matches starting at position 108895
     Pattern 7 matches starting at position 108912
     Pattern 61 matches starting at position 115086
     Pattern 356 matches starting at position 115087
     Pattern 324 matches starting at position 115107
     Pattern 545 matches starting at position 116956
     Pattern 265 matches starting at position 246834
     Pattern 537 matches starting at position 265113
     Pattern 187 matches starting at position 265127
     Pattern 27 matches starting at position 265143
     Pattern 264 matches starting at position 296328
     Pattern 434 matches starting at position 313423
     ...
```

Number of matches to each keyword:
For queries.txt, this is also included in a text file named "Q5matches1.txt":

```
     Total # Matches: 22
     Keyword    # Matches
     AATAGCTAACA     12
     ACCAAACTATAGAAT 1
     CTCTTAATATTTATGAAGAAGAACATGGT    1
     GCCTGGGTGACAGAGTGAGACCCTGTCTC    8
```

For queries2.txt, the first few number of matches for each keyword is listed below, the complete results are in "Q5matches2.txt" along with my code:

```
     Total # Matches: 464
     Keyword    # Matches
     AATAGCTAACA     12
     ACCAAACTATAGAAT 1
     CTCTTAATATTTATGAAGAAGAACATGGT    1
     GCCTGGGTGACAGAGTGAGACCCTGTCTC    8
     GATCATACCATTGTACTCTAGCCTGGGTG    1
     GCCGCCTTCACATTCTCAAAGGAACTCCTGGCCCCCAAACAGGGTCCGGG     1
     TATTCAACATTCTTAAAGAAAAGAATTTTCAACCCAGAATTTCATATCCA     13
     ...
```

E-value comparisons:
Using queries.txt, the file with 4 query strings, we can calculate the E-value using the methods from Q4

The 4 queries have length 11, 15, 29, and 29, respectively. Supposing the DNA database has equal probability of A, C, G, and T, we can calculate the E-value
     = E(number of hits)
n = 13385191
E(number of hits pattern 1) = (.25^11)*(n-11+1) = 3.19127583504
E(number of hits pattern 2) = (.25^15)*(n-15+1) = .0124659175
E(number of hits pattern 3) = (.25^29)*(n-29+1) = 4.6439113E-11
E(number of hits pattern 4) = (.25^29)*(n-29+1) = 4.6439113E-11
E(total hits) = sum of 4 values above = 3.2036935011

The number of matches is much greater than what was expected for each of patterns 1-4 in queries.txt and also for the total expected hits. Patterns 4 that achieved 8 hits had the most fold change from the expected to the observed (the observed was 10^11 times greater). One reason for this could be because we were not given a randomly generated DNA database. The database given was a real sequenced string from chromosome 12 of the human genome. This causes the database sequence to be more likely to have certain queries present as those sequences serve certain functions necessary in the human genome. The expected number of hits for such a non-random database sequence would of course be much greater than that of a random sequence.