Suffix Tree project 3 Report

Matthew Pham & Trenton Bergdahl
March 26, 2025

1. System Configuration
Matthew's System:
CPU: AMD Ryzen 7 5700X3D @ 4.50GHz (8 cores, 16 threads)
RAM: 48GB DDR4
Cache: L1 512KB, L2 4MB, L3 96MB

Trenton's System:
CPU: Intel Core i9-11900H @ 4.90GHz (8 cores, 16 threads).
RAM: 32GB DDR4.
Cache: L1 640KB, L2 10MB, L3 24MB.

2: matrix

A) Similarity Matrix

	1	2	3	4	5	6	7	8	9	10
1	29893	34867	30447	30166	32924	23	23	23	104	104
2	34867	29876	4623	8896	34873	23	23	23	104	104
3	30452	4620	29854	30458	30458	23	23	23	104	104
4	30167	8896	30453	29882	23769	23	23	23	104	104
5	32924	34873	30453	23769	29903	23	23	23	104	104
6	23	23	23	23	23	30055	26952	31868	20	20
7	23	23	23	23	23	26939	30123	26746	20	20
8	23	23	23	23	23	31848	26739	30123	20	20
9	104	104	104	104	104	20	20	20	29644	35190
10	104	104	104	104	104	20	20	20	35172	29727

B) LCS Length Measures Matrix

	1	2	3	4	5	6	7	8	9	10
1	29893	11082	7961	13980	19064	23	23	23	104	104
2	11082	29876	4620	8896	11082	23	23	23	104	104
3	7961	4620	29854	7961	7961	23	23	23	104	104
4	13980	8896	7961	29882	23769	23	23	23	104	104
5	19064	11082	7961	23769	29903	23	23	23	104	104
6	23	23	23	23	23	30055	2890	3182	20	20
7	23	23	23	23	23	2890	30123	3094	20	20
8	23	23	23	23	23	3182	3094	30123	20	20
9	104	104	104	104	104	20	20	20	29644	7878
10	104	104	104	104	104	20	20	20	7878	29727

C)

Looking at the similarity matrix, we can see how closely related some of the strains are based on their similarity scores. Strains that have longer common substrings tend to have higher

scores, which tells us they're likely more similar or related. One clear example is strains 1 through 5 they consistently show high similarity with each other, and you can really see this grouping stand out in the matrix.

As for performance, computing the longest common substring first definitely helped cut down on computation time. It gave us a solid starting point for alignment and made the rest of the process more efficient. Interestingly, we noticed that when it took longer to build the suffix tree, the alignment itself usually ran faster. So there seems to be a bit of a tradeoff between those two steps.

3: performance

Performance Summary:

```
==========================================
Pair    Tree Time       Align Time      Total Time
------------------------------------------
0v0     7.9523          0.0000          7.9523
0v1     3.1613          54.9446         58.1059
0v2     4.7261          81.5367         86.2628
0v3     11.2717         44.2545         55.5262
0v4     7.6019          26.1025         33.7044
0v5     4.1487          106.5084        110.6571
0v6     5.2329          109.7586        114.9915
0v7     4.6064          112.8049        117.4113
0v8     3.3394          85.7880         89.1274
0v9     3.7886          104.3682        108.1568
1v0     8.8106          107.3311        116.1417
1v1     18.4172         0.0000          18.4172
1v2     9.8129          57.5634         67.3762
1v3     9.5718          21.9594         31.5312
1v4     10.5809         82.5521         93.1330
1v5     5.0140          101.3527        106.3667
1v6     5.5375          101.4962        107.0338
1v7     6.0880          93.5699         99.6579
1v8     5.1019          125.2671        130.3690
1v9     5.5594          133.6511        139.2105
2v0     6.7223          69.7315         76.4537
2v1     5.5006          38.8276         44.3282
2v2     9.9545          0.0000          9.9545
2v3     4.2229          51.1338         55.3568
2v4     4.5648          51.2399         55.8048
2v5     3.2614          55.8298         59.0912
2v6     3.4530          56.1215         59.5745
```

2v7	3.7087	56.2744	59.9831
2v8	4.1545	67.9601	72.1147
2v9	4.2468	67.5884	71.8352
3v0	6.1424	28.2673	34.4096
3v1	4.3961	12.3128	16.7089
3v2	4.3701	51.4735	55.8435
3v3	10.1779	0.0000	10.1779
3v4	7.8373	1.0570	8.8943
3v5	3.4137	56.1663	59.5801
3v6	3.4032	56.2715	59.6747
3v7	3.4038	56.3593	59.7631
3v8	3.3682	68.5069	71.8751
3v9	3.5002	68.0289	71.5291
4v0	6.5455	20.8830	27.4285
4v1	5.3886	55.9241	61.3128
4v2	5.0381	52.6375	57.6756
4v3	8.1176	1.0964	9.2141
4v4	10.3827	0.0000	10.3827
4v5	3.5975	56.9697	60.5671
4v6	3.6125	56.9101	60.5226
4v7	3.6921	57.1252	60.8173
4v8	3.6642	69.2563	72.9205
4v9	3.5777	69.0594	72.6370
5v0	3.7274	58.5803	62.3077
5v1	3.8008	58.5206	62.3214
5v2	4.0854	58.5292	62.6146
5v3	4.0546	58.0951	62.1498
5v4	4.0996	58.0912	62.1908
5v5	10.8283	0.0000	10.8284
5v6	4.3814	62.4454	66.8268
5v7	4.5816	86.8876	91.4693
5v8	4.2082	72.2806	76.4888
5v9	4.4027	73.1197	77.5223
6v0	4.3342	58.3443	62.6785
6v1	4.3074	58.6147	62.9221
6v2	4.3205	58.6021	62.9226
6v3	4.4226	58.7388	63.1613
6v4	4.4196	59.0439	63.4635
6v5	4.7962	64.0757	68.8719
6v6	11.4719	0.0000	11.4719
6v7	5.1331	60.3626	65.4957
6v8	4.7978	72.6845	77.4823
6v9	4.9340	73.7839	78.7179
7v0	4.9948	59.7239	64.7187

7v1	4.9825	60.1255	65.1080
7v2	5.1099	60.1172	65.2271
7v3	5.0028	60.0340	65.0368
7v4	4.9974	59.5422	64.5397
7v5	5.6516	90.5907	96.2422
7v6	5.5319	60.9913	66.5232
7v7	12.1295	0.0000	12.1295
7v8	5.4975	73.0076	78.5051
7v9	5.6313	74.0933	79.7246
8v0	5.5471	70.8785	76.4256
8v1	5.5198	71.3244	76.8442
8v2	5.6849	71.8164	77.5013
8v3	5.7788	72.6249	78.4037
8v4	5.8846	72.4839	78.3684
8v5	5.8950	74.7166	80.6116
8v6	5.8712	74.5967	80.4680
8v7	5.8769	74.7192	80.5961
8v8	12.3880	0.0000	12.3880
8v9	6.9494	80.9237	87.8731
9v0	5.7116	70.3837	76.0952
9v1	5.2708	70.7030	75.9738
9v2	5.3979	71.6579	77.0558
9v3	5.7313	72.0020	77.7333
9v4	5.9764	72.0738	78.0502
9v5	5.8360	73.9870	79.8230
9v6	5.9325	74.5460	80.4785
9v7	6.1091	74.7043	80.8134
9v8	7.0081	81.3208	88.3290
9v9	12.6979	0.0000	12.6979
TOTAL	585.4	5978.31	6563.75
AVG	5.8545	59.7831	65.6376

==========

4.

Notes: problem 2C

Our data shows the relationship between the length of common substring and it similarity score.

Yes we do see logical grouping visible through the two matrices, an example could be strings 1-5. They all have very high similarity scores and it showed up visually in the matrix screenshot above.

Based on observation it seems like they have an inverse relationship. Where building trees takes longer, alignment time generally is faster.