Suffix Tree Construction 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. Construction Performance

Suffix Tree Construction Performance Report

File Name	Length (bp)	Time (ms)	Nodes
chr12.fas	1078175	459914	1777696
Covid_Australia.fasta	29893	420	48985
Covid_Brazil.fasta	29876	414	48948
Covid_India.fasta	29854	420	48915
Covid_USA-CA4.fasta	29882	412	48958
Covid_Wuhan.fasta	29903	408	49003
Human-BRCA2-cds.fasta	11382	105	18683
Opsin1_colorblindness_gene.fasta	7145	57	11855
s1.fas	6	0	11
s2.fas	11	0	19
Slyco.fas	155461	15415	254435

3. Justification

The performance results match what we expected: construction time for the suffix tree grows as the input string size increases, which makes sense given that McCreight's algorithm has a linear time complexity (O(n)). As the input data gets larger, like with the chr12 and Slyco genomic sequences, the number of suffixes, nodes, and suffix links grows, causing the processing time and memory usage to go up as well. This is especially noticeable when comparing smaller datasets to larger ones. So, the increase in runtime and memory usage is completely reasonable, reflecting how the algorithm performs when dealing with large, complex genomic data.

4. Implementation Space Constant

Each byte of the input is copied. Integers are stored in nodes, a start and end index into the string, as well as the suffix index. 1 Parent pointer and 5 child pointers. All are 8 bytes. So in addition to the copy made of the input, there are (Node Count) * 64 bytes. Assuming we make an average of 1 node per three input bytes, we use (Input Size / 3) * 64 bytes (roughly).

5. BWT Index

Are in the files