Compressed Direct-Address Table - practical index for genomic sequences

Katarzyna Jabłonowska

Michał Sabaciński
Norbert Dojer

Segovia
10 October 2019

The problem

ATTATATAGTTGTAA

✓GTTGT
✓TAA
×*CTA*

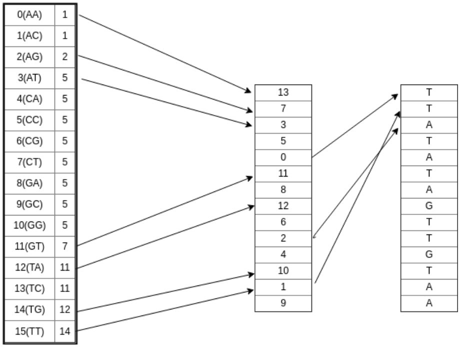
Our goal is to find an algorithm that:

- scans the whole text only once
- is flexible
- is fast and memory efficient

The basic idea

Two steps:

1. Building an index
2. Searching for patterns



Building an index

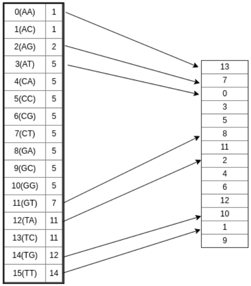
Step 1

0(AA)	1
1(AC)	0
2(AG)	1
3(AT)	3
4(CA)	0
5(CC)	0
6(CG)	0
7(CT)	0
8(GA)	0
9(GC)	0
10(GG)	0
11(GT)	2
12(TA)	4
13(TC)	0
14(TG)	1
15(TT)	2

Step 2

0(AA)	1
1(AC)	1
2(AG)	2
3(AT)	5
4(CA)	5
5(CC)	5
6(CG)	5
7(CT)	5
8(GA)	5
9(GC)	5
10(GG)	5
11(GT)	7
12(TA)	11
13(TC)	11
14(TG)	12
15(TT)	14

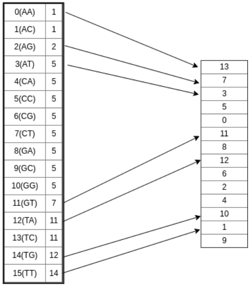
Step 3



0(AA)	1
1(AC)	1
2(AG)	2
3(AT)	5
4(CA)	5
5(CC)	5
6(CG)	5
7(CT)	5
8(GA)	5
9(GC)	5
10(GG)	5
11(GT)	7
12(TA)	11
13(TC)	11
14(TG)	12
15(TT)	14

13
7
0
3
5
8
11
2
4
6
12
10
1
9

Step 4

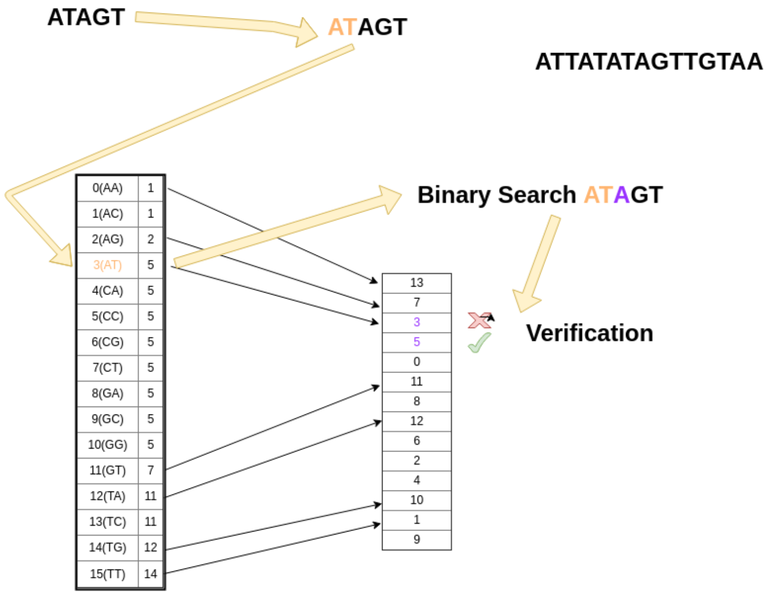


0(AA)	1
1(AC)	1
2(AG)	2
3(AT)	5
4(CA)	5
5(CC)	5
6(CG)	5
7(CT)	5
8(GA)	5
9(GC)	5
10(GG)	5
11(GT)	7
12(TA)	11
13(TC)	11
14(TG)	12
15(TT)	14

13
7
3
5
0
11
8
12
6
2
4
10
1
9

Prefix length = 2
Substring length = 3

Searching for patterns



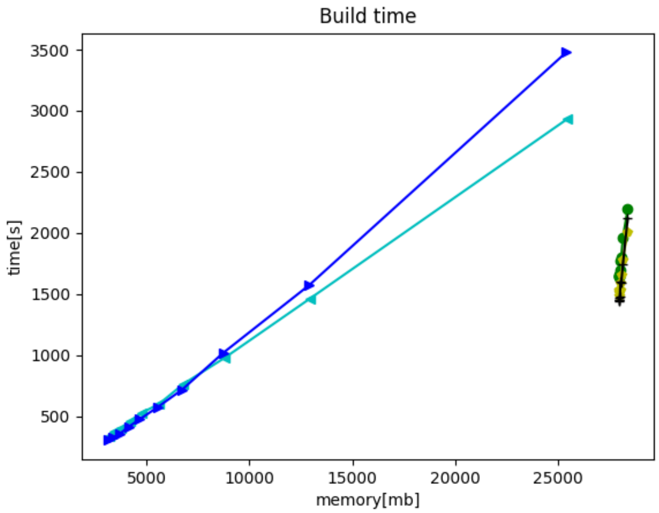
Compression

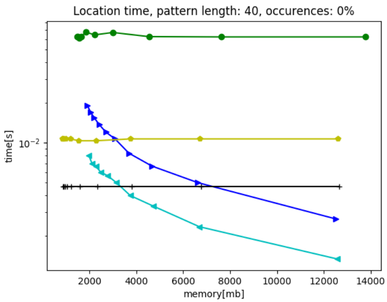
1. 'GTCCAT'
2. 'GTCCAT'
3. 'GTCCAT*', $* \in \Sigma$
4. '*GTCCAT', $* \in \Sigma$
5. 'GTCCAT**', $* \in \Sigma$

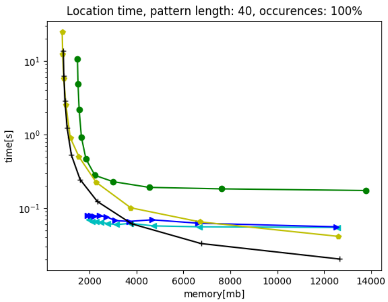
Experiments

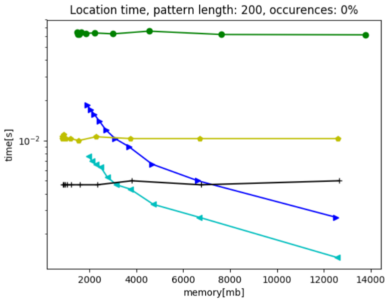
- human genome
- C++
- prefixLength = 10 and 12
- patterns of length 40 or 200. All or none of them appearing in the text

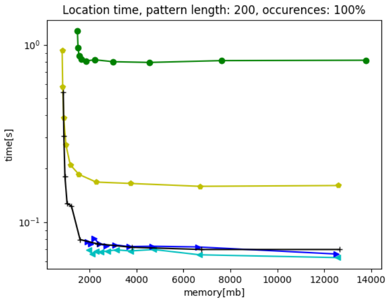
Results


Build time
time[s]
memory[mb]
3500
3000
2500
2000
1500
1000
500
5000
10000
15000
20000
25000


Location time, pattern length: 40, occurences: 0%
time[s]
10^{-2}
memory[mb]
2000
4000
6000
8000
10000
12000
14000


Location time, pattern length: 40, occurences: 100%
time[s]
10^{1}
10^{0}
10^{-1}
memory[mb]
2000
4000
6000
8000
10000
12000
14000


Location time, pattern length: 200, occurences: 0%
time[s]
10^{-2}
memory[mb]
2000
4000
6000
8000
10000
12000
14000


Location time, pattern length: 200, occurences: 100%
time[s]
10^{0}
10^{-1}
memory[mb]
2000
4000
6000
8000
10000
12000
14000

Thank you