



AMRITA

VISHWA VIDYAPEETHAM

21AIE203 Introduction to Biological Systems

TERM PROJECT (END SEMESTER)

BATCH A GROUP_4

TEAM MEMBERS	
ABINAYA.N	[CB.EN.U4AIE21001]
DILIP.P	[CB.EN.U4AIE21009]
M.C.S. SANJANA	[CB.EN.U4AIE21029]
SIDDHARTH.D	[CB.EN.U4AIE21064]

TABLE OF CONTENTS

Trie Data Structure

Trie Data Structure Representation

Trie data structure implementation in Bioinformatics

Restriction Enzymes

ECORI Restriction Enzyme In A DNA Sequence

Properties of the Trie for a set of the DNA sequence

Speciality of Trie data structure

Prefix Search

DNA substring searching using Trie

Obtaining the sequence

The Restriction enzymes we implemented

Outputs Obtained for HMGB2

Outputs Obtained for Human DNA

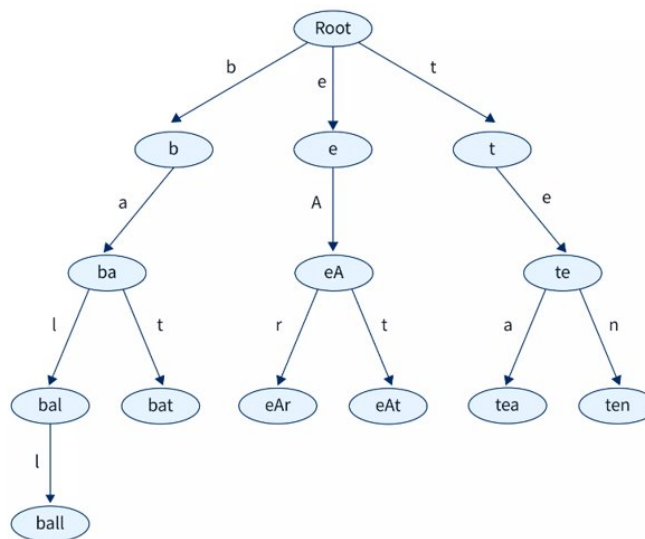
Conclusion

DNA Substring search with Trie

Trie Data Structure

- A Trie is a tree data structure that stores a dynamic set or associative array where the keys are usually strings.
- Unlike a binary search tree, no node in the tree stores the key associated with that node; instead, its position in the tree defines the key with which it is associated.
- All the descendants of a node have a common prefix of the string associated with that node, and the root is associated with the empty string.
- Values are not necessarily associated with every node.
- Rather, values tend only to be associated with leaves, and with some inner nodes that correspond to keys of interest.

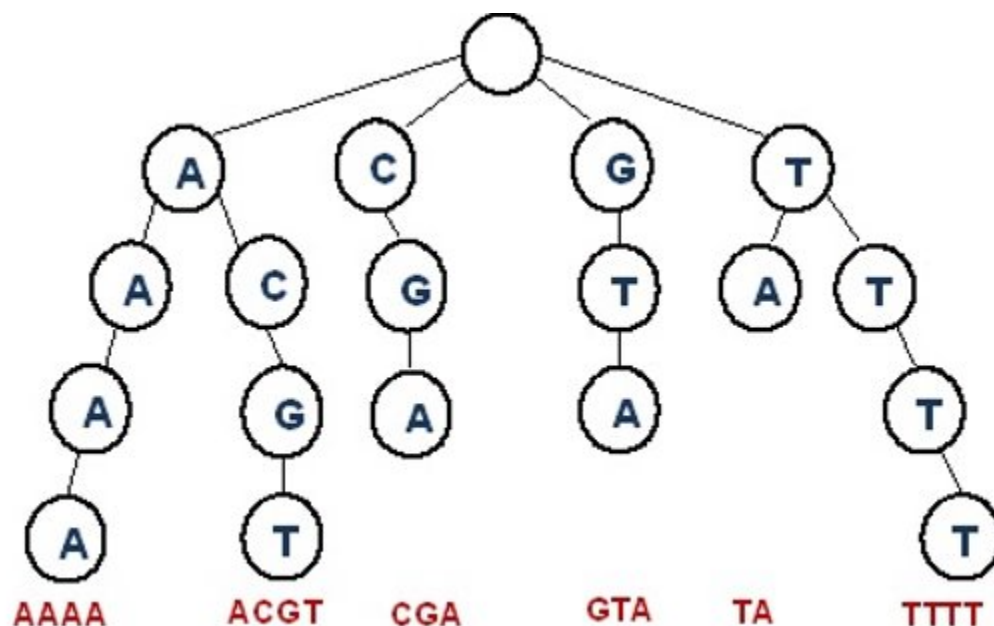
Trie Data Structure Representation



Trie data structure implementation in Bioinformatics

- A Trie, also known as a prefix tree, is a tree-like data structure that is commonly used in bioinformatics for tasks such as storing and searching large sets of DNA or protein sequences.
- In a Trie, each node represents a single character in a sequence, and edges between nodes represent the relationships between characters in the sequence.
- The Trie allows for efficient search and retrieval of sequences based on their prefixes, which is useful in applications such as identifying potential protein-coding regions in a genome.
- Trie data structure is also used to detect the repeat sequence in the genome and in the de-novo assembly of genome sequencing.

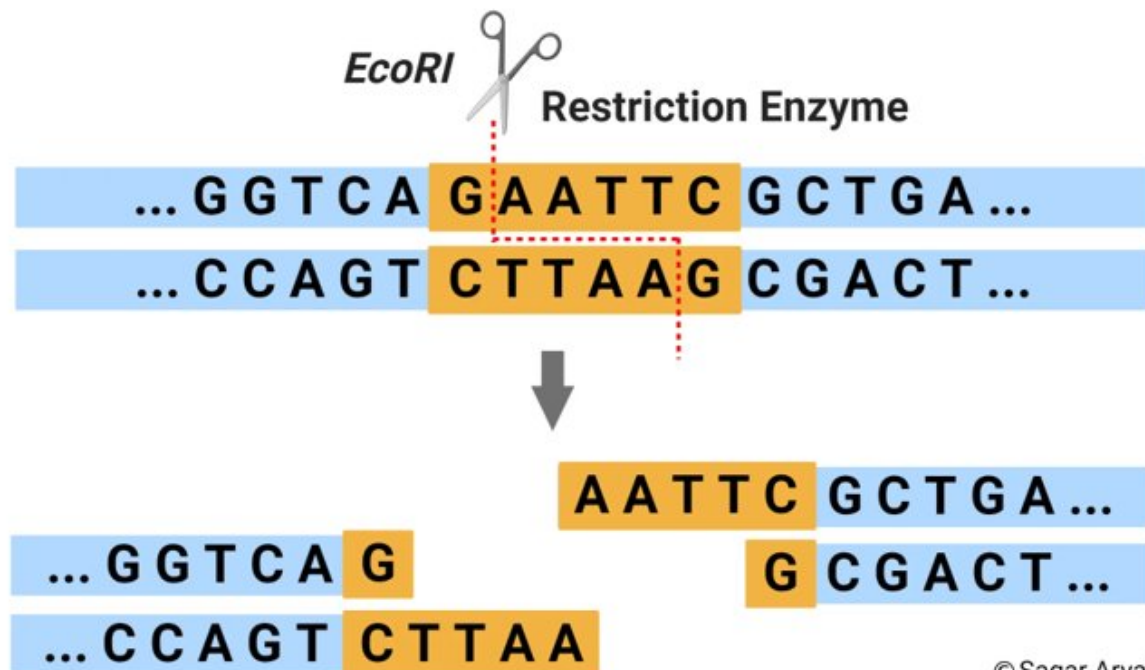
Model using the sequences



Restriction Enzymes

- Restriction enzymes are enzymes that cut DNA at specific locations, called restriction sites.
- The restriction enzymes protect the live bacteria from bacteriophages. They recognize and cleave at the restriction sites of the bacteriophage and destroy its DNA.
- Restriction enzymes make a double-stranded break at a specific location within that sequence.
- This creates a restriction fragment that can be separated from the rest of the DNA.
- There are many different types of restrictive enzymes, each recognizing and cutting a specific sequence of nucleotides in the DNA. Some of the most used restrictive enzymes in molecular biology are EcoRI, BamHI, and HindIII.

ECORI RESTRICTION ENZYME IN A DNA SEQUENCE



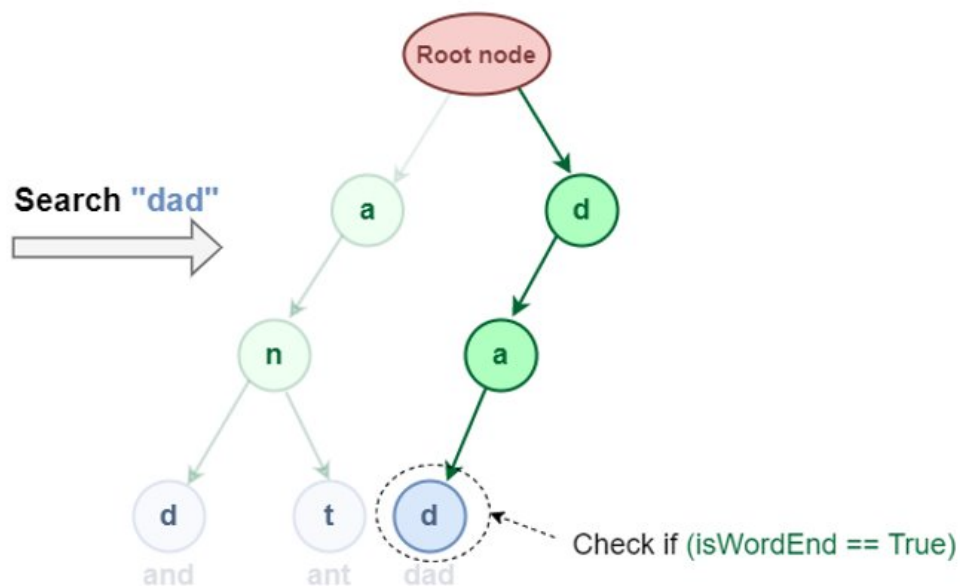
Properties of the Trie for a set of the DNA sequence

- The root node of the Trie always represents the null node.
- Each child of nodes is sorted alphabetically.
- Each node can have a maximum of 4 children (ATCG)
- Each node (except the root) can store one letter of the alphabet.

Speciality of Trie data structure

- Trie is an efficient information retrieval data structure.
- Using Trie, search complexities can be brought to optimal limit (key length).
- If the Trie is balanced, then the time taken to search is the fastest compared to all other substring searching methods.
- Search time linearly increases with the length of substring.

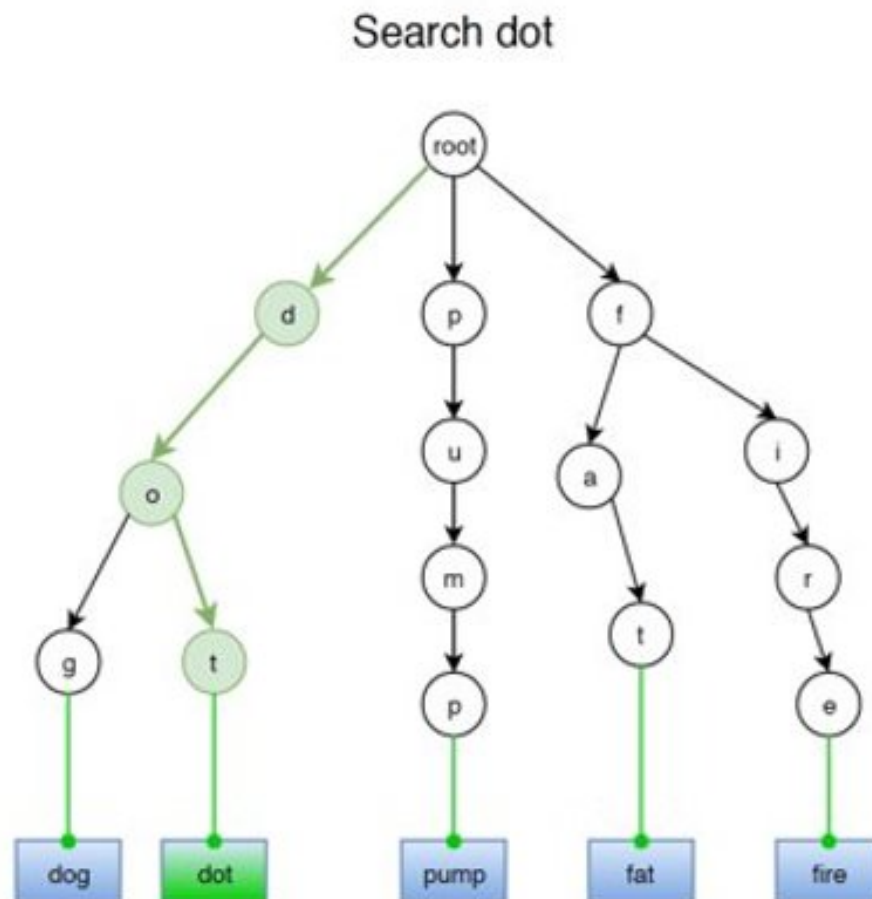
SEARCHING IN TRIE



PREFIX SEARCH

Due to the way how Trie stores data, its known to perform good to search anything related to prefixes, like finding most common prefixes, autocomplete with prefixes and more.

- To do a prefix search, we begin at the root node and proceed along the route of child nodes that correspond to the prefix's characters.
- The last node in the path, if a matching one exists, is the end node of the prefix. By moving through the end node's child nodes, the Trie can be searched for all words with the prefix.



DNA substring searching using Trie

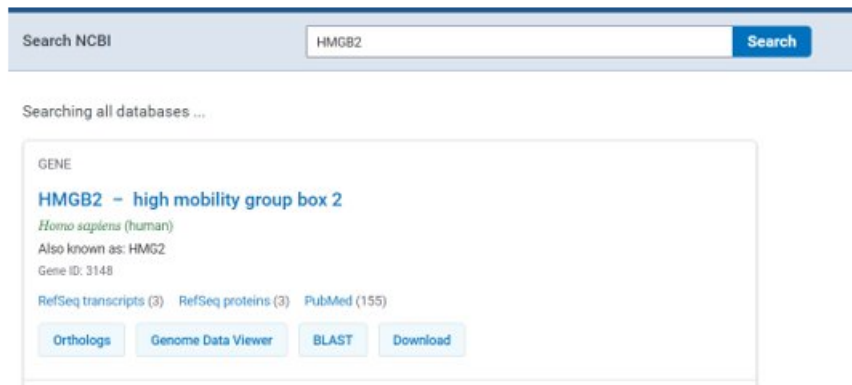
- A Trie data structure can be used to efficiently search for DNA substrings by representing each DNA base (A, C, G, T) as a node in the trie.
- Each path from the root of the trie to a leaf node represents a sequence of DNA bases.
- To search for a specific DNA substring, we can start at the root of the trie and traverse the tree, following the path of nodes corresponding to the bases in the substring.
- If the path leads to a leaf node, then the substring is present in the set of DNA sequences represented by the trie.
- If the path leads to an internal node, it means the substring is not present or is a prefix of some other sequence present in the trie.

Obtaining the sequence

In this we took a Nucleotide sequence of the HMGB2 (a differentially expressed protein).

STEP 1 :

- We give the input of which protein or nucleotide sequence we need in the NCBI National Center for Biotechnology Information.



The screenshot shows the NCBI search interface. At the top, there is a search bar with the text "Search NCBI" on the left and "HMGB2" in the center. To the right of the search bar is a blue button labeled "Search". Below the search bar, it says "Searching all databases ...". The search results are displayed in a box with the heading "GENE". The first result is "HMGB2 - high mobility group box 2", followed by "Homo sapiens (human)", "Also known as: HMGB2", and "Gene ID: 3148". Below this information, there are links for "RefSeq transcripts (3)", "RefSeq proteins (3)", and "PubMed (155)". At the bottom of the results box, there are four buttons: "Orthologs", "Genome Data Viewer", "BLAST", and "Download".

STEP 2 :

- From that NCBI we take the FASTA SEQUENCE of the respective nucleotide or protein and produce that FASTA in the blast.

FASTA ▾

Homo sapiens chromosome 4, GRCh38.p14 Primary Assembly

NCBI Reference Sequence: NC_000004.12

[GenBank](#) [Graphics](#)

```
>NC_000004.12:c173334358-173331376 Homo sapiens chromosome 4, GRCh38.p14 Primary Assembly
AAACCAAGTTTACGCGGAGCCCCGTGAGGGAAGCGTCTCCGTTGGGTCCGGCCGCTCTGCGGGACTCTGA
GGAAAAGCTCGCACCAAGGCAAGAATACCTCCAATACCTCGGGTTCGTGGACCTGCCCTTCCCCATTCC
CTCAGAGCTCTACTCGGTCTCGGCGCAGTGGCTCTCGGGGTCTGACCCGGCGAGCGGCATTGGGGTG
GGGCCGGCGAGGGCTGGGTCTGTGGAGGGCCGGCGGGCAGTCGGAGGAGGCGGAACTGCCCTGACCGG
GCCCGGTTCTGGGAGTTTCAATGTCGGTCACGAGGTGCTTAGCGGTTAAGTGAGGCTCGCGACGTGGGA
GCTGGGGGACATTTCTGTACGGAAGGCGTTTTGAGGGCGCTTGCGGAGTCACCGACTCGGTGAAGGGC
GGGCGGCGGCGTCTCGGGCCCCGGGAGCCACTTGCTGGGCTGAGCCGCGGCCGCTCGGGCCGGGAGGAGG
AGGAGGAGGAGGGTGTGCGCCGGGGCGCGGGCGGGGGCGCGGGCGGCCGCCCTTGGGAGGACGCCCTCGG
GAGGAGGGGGCGCGGGCAGGGCGGGAGCGGATTGGGCGGGAAGCGGAGCCCCGCCAGCGCCGCCCTG
GCAGCTGCGGGCTCCGCGCCGACCTTCGGCTTCCCTCTCCCCCTCGGGCCCGTCAGGTGGACGCGGA
TCTGTCAACATGGGTAAGGAGAGCCCCAACAGCCGCGGGGCAAAATGTCCTCGTACGCCCTTCTCTGTC
```

STEP 3:

- The obtained FASTA we will put in the respective protein or nucleotide blast and get the final DNA sequence we required.
- A FASTA sequence is a simple text-based format used to store biological sequences, such as DNA or protein sequences.
- Example DNA Sequence ATG CAG CTA GCTAGC
- It is widely used in bioinformatics and computational biology to store and share genetic data.

STEP 4

- The final obtained DNA Sequence we will place in the place of the string in our code.
- The application here we are doing using the Trie is finding the substring of the restriction enzymes. We give the respective string of that enzyme to the input and searching for that substring of that restriction enzyme sequence.

- Here we took the sequence of the HMGB2 and Human DNA and found the restriction enzymes mentioned below.

The Restriction enzymes we implemented are:

StuI

- AGG/CCT

ClaI BspDI

- AT/CGAT

EcoRI EcoRI-HF®

- G/AATTC

MnII

- CCTC

OUTPUTS OBTAINED FOR HMGB2

```
PS C:\Users\Siddharth\VScode\pythonprojects\project1> cd "c:\Users\Siddharth\VScode\pythonprojects\project1\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
CCTC found at:
48 86 149 213 276 334 485 662 699 811 1058 1068 1100 1108 1514 1612 1624 1633 1708 2190 3899 4049 4055 4081 4087 4193 4211 4217

AGG found at:
16 19 40 70 115 120 124 127 139 239 292 315 327 375 406 421 448 468 472 478 537 642 681 762 776 822 938 944 987 1030 1046 1180 1196 1215 1218 1274 1294 1330 1335 1345 1376
1445 1448 1451 1454 1457 1460 1507 1521 1524 1538 1646 1676 1816 1821 1825 1830 1843 1852 1869 1872 1880 1884 1891 1953 1967 2009 2057 2116 2169 2198 2209 2221 2241 2480
2495 2534 2540 2611 2642 2813 2819 2822 2851 2981 3077 3101 3117 3125 3136 3144 3176 3179 3182 3185 3188 3209 3212 3218 3282 3406 3526 3561 3903 4011 4155 4175 4240 4321

CGAT found at:
1960 4030 4070

GAATTC found at:
0 1785 3306 4335
```

OUTPUTS OBTAINED FOR HUMAN DNA

```
CCTC found at:
97 107 139 147 554 653 665 674 749 1231 2937

AGG found at:
26 69 85 219 235 254 257 313 333 369 374 384 415 484 487 490 493 496 499 547 561 564 578 687 717 857 862 866 871 884 893 910 913 921 925 932 994 1008 1050 1098 1157 1210 1
239 1250 1262 1282 1521 1536 1575 1581 1652 1683 1854 1860 1863 1892 2022 2118 2142 2158 2166 2177 2185 2217 2220 2223 2226 2229 2250 2253 2259 2323 2447 2567 2602 2941

CGAT found at:
1001

GAATTC found at:
826 2347
```

Conclusion

In conclusion, we have successfully implemented DNA substring searching using the TRIE data structure. Our implementation effectively allows us to find restriction enzymes on HMGB2 and human DNA. Our results demonstrate the efficacy of the TRIE data structure in solving the problem of DNA substring searching. This project has added to the existing knowledge base on efficient algorithms for DNA sequence analysis and has potential applications in the field of molecular biology and biotechnology.