Homework #5

Name: Purnabhishek Sripathi
Email: ps747@nau.edu
User id: 6274051

Problem #1 (of 1): Prefix Trie

Create a class called ***Prefix_Trie.*** The purpose of the class will be to contain a dataset of genomic sequences (queries) and all of the functions needed to operate on this set. Use the **prefix trie** data structure to store the genomic fragments of a given size. Here you will be performing fuzzy matching, tolerating up to 1 mismatch.

At a minimum, the class must contain(1**5pts**):

A default constructor

ondemand.hpc.nau.edu/pun/sys/dashboard/files/edit/fs/home/ps747/hw5/headers.h

Save /home/ps747/hw5/headers.h Key Bindings Default Font Size 12px

```
// Default constructor
PrefixTrie() : root(nullptr), wordLength(36) {}
```

At least one custom constructor to build a trie from a set of queries (of size n)

ondemand.hpc.nau.edu/pun/sys/dashboard/files/edit/fs/home/ps747/hw5/headers.h

Save /home/ps747/hw5/headers.h Key Bindings Default Font Size 12px

```
PrefixTrie() : root(nullptr), wordLength(36) {}

// Custom constructor to build a trie from a set of queries (of size n)
PrefixTrie(const char **queries, int n) : wordLength(36) {
    root = new TrieNode();
    for (int i = 0; i < n; ++i) {
        insert(queries[i]);
    }
}
```

A function to traverse (*search*) the trie using a genome of size G. Note that you can assume that G >> n. You will need to implement a *fuzzy search tolerating up to 1 mismatch* (substitutions only). Hint: use a stack to keep track of branches in the tree that need to be explored.

ondemand.hpc.nau.edu/pun/sys/dashboard/files/edit/fs/home/ps747/hw5/headers.h

Save /home/ps747/hw5/headers.h Key Bindings Default Font Size 12px Mode

```
//A function to traverse (search) the trie using a genome of size G.
bool fuzzySearchHelper(TrieNode *root, char *ch, int index, bool flag){
    if(index==36)//condition to check index equals to 32
    return true; // returns true
    if(root==nullptr) // condition to check root is equals to null or not
    return false; // returns false

    int i=getIndex(ch[index]); // getting the index
    if(root->children[i]!=nullptr){ //condition to check root children equlas to null or not
        return fuzzySearchHelper(root->children[i],ch,index+1,false); // returns the output from fuzzyhelper method
    }
    else //if not it executes this else block
    {
        if(flag) //consition to check the flag true or not
        return false;//returns false
        int j=0;//initialize the variable j=0
        while(j<4){//loop to iterate until j becomes leass than 4
            if(fuzzySearchHelper(root->children[i],ch,index+1,true))
            return true;//return true value
            ++j;//increment the looping variable
        }
    }
    return false;//at the end return the false nothing validates
}
```

A destructor

ondemand.hpc.nau.edu/pun/sys/dashboard/files/edit/fs/home/ps747/hw5/headers.h

Save /home/ps747/hw5/headers.h Key Bindings Default Font Size 12px

```
    long long int countTrieNodes(){
        return countTriesNodesHelper(root,0,wordLength);
    }


};

// Destructor
~PrefixTrie() {
    deleteTrie(root);
}
```

Part A. (20pts) Basic prefix trie

For each of the 36-mer datasets, what are the sizes of the trie (# of nodes)? Explain the pattern that you observed.

ondemand.hpc.nau.edu/pun/sys/dashboard/files/fs//home/ps747/op_A

```
Data read successfully: 3057195757
Question A started
For 5000:
total number of nodes in prefixTrie: 144747
total hits: 4801
For 50000:
total number of nodes in prefixTrie: 913284
total hits: 31629
For 100000:
total number of nodes in prefixTrie: 1233950
total hits: 43292
For 1000000:
total number of nodes in prefixTrie: 1395469
total hits: 49963
```

Building a prefix trie from a section of the human genome and producing random 36-mers for various dataset sizes demonstrates a rise in trie size as dataset size increases, according to the data presented.

The pattern is consistent with expectations: because there are more unique substrings in larger datasets, there are greater tries. Determining the experiment's meaning, however, is difficult without background information on its goal and an evaluation of the relevance of the findings. For a thorough evaluation, more research on trie efficiency and matches discovered in genomic data with up to one mismatch is required.

Iterate through all possible 36-mers in the segment, using each to search/traverse the prefix trie with up to 1 mismatch. How many of your 36-mers had a match? Does it make sense? Explain why.

ondemand.hpc.nau.edu/pun/sys/dashboard/files/fs//home/ps747/op_A

```
Data read successfully: 3057195757
Question A started
For 5000:
total number of nodes in prefixTrie: 144747
total hits: 4801
For 50000:
total number of nodes in prefixTrie: 913284
total hits: 31629
For 100000:
total number of nodes in prefixTrie: 1233950
total hits: 43292
For 1000000:
total number of nodes in prefixTrie: 1395469
total hits: 49963
```

A popular method for approximative string matching in genomic data processing involves iterating through all possible 36-mers in the segment and searching/traversing the prefix trie with up to 1 mismatch.

It makes sense since there is a chance that genetic sequences contain faults or variants, and the chance of identifying meaningful matches grows with each mismatch.

The trie's ability to capture changes in genomic sequences and its potential use in tasks like variant calling and sequence alignment can be inferred from the number of 36-mers that matched.

Part B (20pts) Impact of error rate on trie structure:

For each of the 36-mer datasets, what are the sizes of the trie (# of nodes)? Explain differences (if any) between the trie sizes in part A and part B.

ondemand.hpc.nau.edu/pun/sys/dashboard/files/fs//home/ps747/op_B

```
Data read successfully: 3057195757
Question B started
For 5000:
total number of nodes in prefixTrie: 143969
total hits: 4527
For 50000:
total number of nodes in prefixTrie: 925465
total hits: 30677
For 100000:
total number of nodes in prefixTrie: 1295583
total hits: 42571
For 1000000:
total number of nodes in prefixTrie: 2280576
total hits: 49963
```

Part A shows that the total hits increase from 4801 to 49963, and the trie sizes range from 144747 to 1395469 nodes.

In Part B, total hits range from 4527 to 49963, and trie sizes range from 143969 to 2280576 nodes.

In general, Part B consistently exhibits greater trie sizes than Part A for all datasets, suggesting possible variations in the types of data used or the techniques used to design trie architecture.

Iterate through all possible 36-mers in the segment, using each to search/traverse the prefix trie with up to 1 mismatch. How many of your 36-mers had a match? Does it make sense? Explain why.

ondemand.hpc.nau.edu/pun/sys/dashboard/files/fs//home/ps747/op_B

```
Data read successfully: 3057195757
Question B started
For 5000:
total number of nodes in prefixTrie: 143969
total hits: 4527
For 50000:
total number of nodes in prefixTrie: 925465
total hits: 30677
For 100000:
total number of nodes in prefixTrie: 1295583
total hits: 42571
For 1000000:
total number of nodes in prefixTrie: 2280576
total hits: 49963
```

With more options for matching sequences in larger datasets, it is natural that the number of matches rises as dataset size increases.

Furthermore, in genetic data, where faults and variances are widespread, allowing only one mismatch boosts the probability of identifying matches.

The outcomes demonstrate how well the trie-based method captures sequence similarities with up to one mismatch in a variety of sample sizes.

Part C (20pts) Full prefix trie experience:

How long did it take you to find all 32-mers of 100K, 1M, and 100M character segments within the prefix trie? Estimate how long it would take to search the entire human genome.

ondemand.hpc.nau.edu/pun/sys/dashboard/files/fs//home/ps747/op_C

```
Data read successfully: 3057195757
Question C started

fragments read successfully

For100000:
total trie nodes:1562545999
total hits: 7011
time taken: 0.122104

For1000000:
total trie nodes:1562545999
total hits: 72621
time taken: 1.18527

For100000000:
total trie nodes:1562545999
total hits: 7387072
time taken: 118.266
```

Time taken for 100k: 0.122104 seconds.

Time taken for 1M: 1.18527 seconds.

Time taken for 100M: 118.266 seconds.

Estimated time for entire genome = (Total fragments in the genome/Fragments tested) ×Time taken for 100,000,000 fragments

Estimated time for entire genome= (3057195757 / 100000000) ×118.266

Estimated time for entire genome = 30.57195757×118.266

Estimated time for the entire genome = 3613.303 seconds.

Therefore, to search the full human genome, it would take around 1 hour and 0.22 minutes, or about 1 hour and 13 minutes.

How many 'hits' did you find for the 100K, 1M, and 100M segments? Estimate how many you would find in the full genome.

ondemand.hpc.nau.edu/pun/sys/dashboard/files/fs//home/ps747/op_C

```
Data read successfully: 3057195757
Question C started

fragments read successfully

For100000:
total trie nodes:1562545999
total hits: 7011
time taken: 0.122104

For1000000:
total trie nodes:1562545999
total hits: 72621
time taken: 1.18527

For100000000:
total trie nodes:1562545999
total hits: 7387072
time taken: 118.266
```

- **Estimated hints for entire genome =** hpf * Total number of fragments in entire human genome.
- **For the 100,000 fragments dataset: Estimated hits =** 0.07011 * 3057195757 = 214474 hits.
- **For the 1,000,000 fragments dataset: Estimated hits =** 0.072621 * 3057195757 ≈ 222037 hits.
- **For the 100,000,000 fragments dataset: Estimated hits =** 0.07387072 * 3057195757 = 22556071 hits.

Therefore, the estimated number of hits for the entire human genome would be around 214474 to 22556071 hits, depending on the assumed hits per fragment ratio.