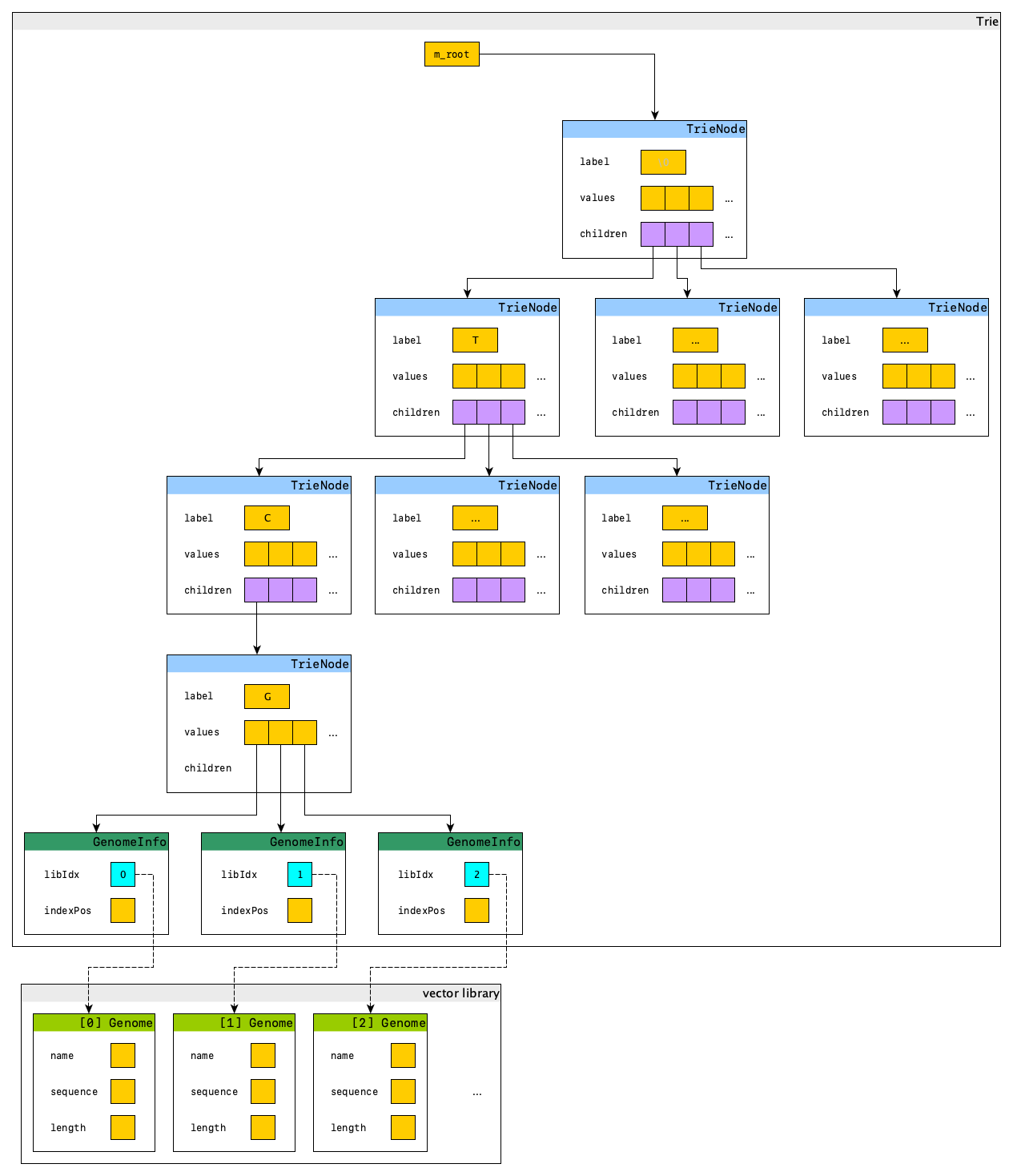
**Project 4 Report**

**Part 1. Illustration**



The classes I implemented are bug-free, according to all the tests I have performed.

Each of the class methods satisfies the big-O requirements.

**Part 2. Pseudocode**

vector<V> Trie<V>::find(**const** string &key, **bool** exactMatchOnly) **const**

|  |
| --- |
| if exact match only  ask the root node to do an exact-match find with the given key  else  ask the root node to do a SNiP find with the given key |

vector<V> Trie<V>::Node::findExact(**const** string &key) **const**

|  |
| --- |
| if the given key is an empty string  found. return the values hold within this node  denote the first character of the key string as **key label**  try to find the child with the key label above  if there is no such child  not found. return an empty vector  else  ask the found child node to do a exact find with the rest of the given key  return the found result |

vector<V> Trie<V>::Node::findSNiP(**const** string &key, **const** **bool** &first) **const**

|  |
| --- |
| if the given key is an empty string  found. return the values hold within this node  denote the first character of the key string as **key label**  create a **new result vector** to hold the results  for each child of this node  if the child has the **key label**  ask the child to find do a SNiP find with the rest of given key  append the found result to the result vector  else if this is not a first level trie node  ask the child to find do a exact find with the rest of given key  append the found result to the result vector  return the result vector |

**bool** GenomeMatcherImpl::findGenomesWithThisDNA(

**const** string &fragment, **int** minimumLength, **bool** exactMatchOnly,

vector<DNAMatch> &matches) **const**

|  |
| --- |
| if **fragmentLength** is smaller than **minimumLength**  no need to search; there will be no match. return false  if **minimumLength** is smaller than **minimumSearchLength**  "hey you can't do that" return false  clear the **matches** vector  use the first **minimumSearchLength** characters as the key to search the trie with  the matching option and call the returned genomes **candidates**  for each **candiate**  extract the **minimumLength** characters starting from the key matching position  if possible; otherwise, extract whatever is left  if the extracted string is not long enough  pass this **candidate** (continue)  check how long a prefix of the candidate's can match that of the fragment's  if the matching prefix is longer than **minimumLength**, we have a match  construct a match record  check if there is already a match result from this candidate genome  if there is not  add the match record  else if there is already a matching record  if the existing match result is shorter  replace the earlier result with this longer result  flatten the records into a vector of matches and return them |