Data Structures Project 1

Nurgazy Seidaliev
Long Nguyen
Lab section: Wednesday 1:15pm - 3:15pm

February 2019

1. Project:

- The project file contains main.cpp, myfunctions.cpp, myfunctions.h, Makefile, and the Dictionaries.
- The main function we implement in the whole project is the Binary Search since the dictionaries are already sorted. This will help optimize the time complexity from O(n) to O(logn) by reducing the number of comparisons needed to find a word.

2. Algorithm:

- There are 3 key functionalities, which we divide into 3 types:
  - Type 1: search for a full word.
  - Type 2: search for and list the words that match the prefix (*).
  - Type 3: search for and match the words that match the prefix and postfix (?).
- For our functions binSearch and prefixBinSearch, they take 4 arguments: vector<string>data, string word, int n, int &cnt.
  - vector<string> data: contains the words in the dictionary and are represented as a sorted vector.
  - word: the word that the user inputs.
  - n: is the size of the vector.
  - cnt: is the number of comparisons that are carried out to find the matching word.
- For our function FirstIndex which is executed for type 2 and 3 after we found the prefix position, we have 2 additional arguments: int &pos, int len.

  - pos: the position of the returned prefix.
  - len: the size of the input.
- Next, we implement the Binary Search in our project.
  - If (right < left), there is no match. We output “word not found”.
  - Find the middle element (mid), and compare input with mid.
  - If word == mid:
    - If type 1, we output “Word found” and number of comparisons.
    - If type 2 and 3, we run while loops (FirstIndex) to find the range that matches the prefix.
      - If type 2, from the range we found, we compare to the limit of maximum outputs that were given and print the number of comparisons. The counting for number of comparisons continue even when we have reached the first element that matches in the binary search.
      - If type 3, from the range we received by FirstIndex, we go through it to find if there is any words that match the postfix. We do linear search in this step.
  - Else:
    - If word < data[mid], we continue the BinSearch for the range (left, mid-1)
    - If word > data[mid], we continue the BinSearch for the range (mid+1, right)