REPORT

The main purpose of this report is to explain methods, techniques, solutions and memory efficiency of the code that is written for Data Structure Lab final project. The project is consist of 4 parts which are searching, sorting, hashing and linked list part. In this short report, of course it is not possible to talk about all line of the project , but I will give you information about the most significant parts.

Sorting: The algorithm which I have preferred to sort 10.000 passwords is Quick Sort because before using quick sort , I tried to use Bubble sort algorithm but the time complexity of sorting was too much. After trying bubble sort, I have tried to use Heap sort, but I faced memory errors because of overflow. So , I implemented Quick Sort algorithm to the project and it sorts array of passwords correctly and quickly. For both ascending and descending sorting, I used only one function which takes an additional variable. If the variable equals to 1, sorting algorithm will sort ascending, if equals to 0, sorting will be in descending order. Although it sorts given array quickly, there was still a problem. The problem was that if you tried to sort an array which is already sorted in opposite order, time complexity will be worst case (n^2). To solve this, I read file again between ascending and descending sorting.

Searching: After solving sorting problem, I used binary search algorithm because binary search algorithm needs a sorted array. By using quick sort and binary search algorithm, the time complexity of the program was very small. Also , I added a condition which controls that if array is sorted or not. If array is sorted already, we don't need to sort it again. By adding this condition, if you want to search more than 1 password, the program will sort the array only once.

Hashing: In hashing part, I created a hash function, inserting function and searching function. The purpose of using hash function is getting a value which is created by using every character of string. Searching and inserting functions uses this hash function and the same probing method. In this project , I used quadratic probing method to reduce number of collision. Also, I selected a big hash table and size of this hash table was an prime number. So, the number collisions was very small.

Linked list: In linked list part, I preferred to use doubly linked list and used a function that insert at the end of the linked list. Also, I used a function which measures strength of given password. So that , Linked list struct includes 4 types (password, strength , address of previous node, address of next node). After insertion , I created sorting function which sorts all nodes by strength for doubly linked list.

Memory efficiency :

Since I used quick sort, memory usage of sorting is very small. If I used merge sort, it will be too much. But in hashing part, I created a hash with a huge table size (5077) to reduce collision. So that memory usage is too much here. As a result of this, memory usage of this program is normal.