Analysis of Algorithms Homework 3 Report

Süleyman Can MUTLU
150130020

01.12.2017

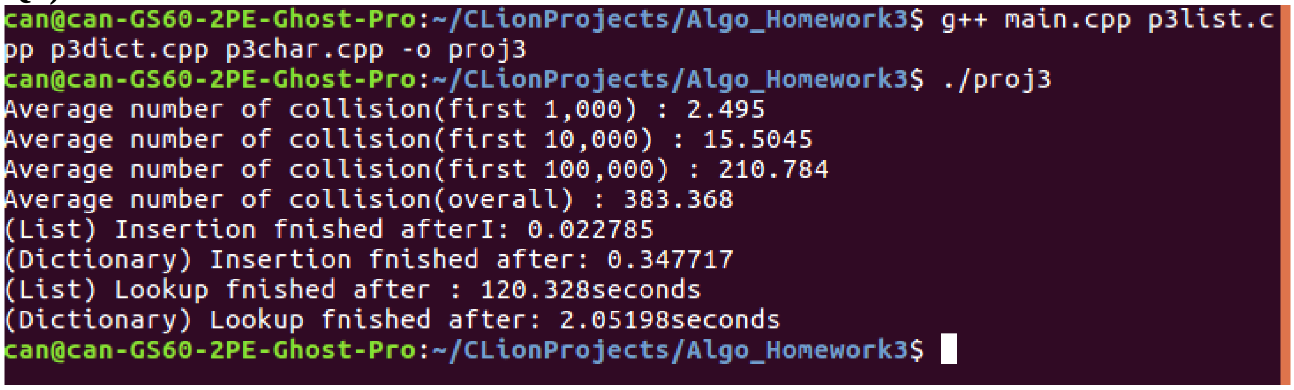
Q1)

Runtime (seconds)	**Block Insertion**	**Block Lookup**
Dictionary	0.347717	2.05198
List	0.022785	120.328

Q2)

The program runs the same process to both structures and there is noticeable difference between implementation and lookup procedures. Firstly, the read text file is inserted into list structure. List implementation is faster than dictionary implementation, because the list has a simple adding function which adds the items to vector ordinary. However the dictionary implementation function has complex algorithms such as hashing and probing, also hashing cause some collisions and probing function executes until no collision occur. Therefore, insertion of dictionary needs a bit more time but when we look to lookup procedure, we can see huge difference. Lookup function executed by dictionary after 2.05 seconds, but list executed after 2 minutes. This variables show us how dictionary can easily find using hash key. In addition, our situation is average-case, because the book characters given the random order via input file.

Q3)



```
can@can-GS60-2PE-Ghost-Pro:~/CLionProjects/Algo_Homework3$ g++ main.cpp p3list.c
pp p3dict.cpp p3char.cpp -o proj3
can@can-GS60-2PE-Ghost-Pro:~/CLionProjects/Algo_Homework3$ ./proj3
Average number of collision(first 1,000) : 2.495
Average number of collision(first 10,000) : 15.5045
Average number of collision(first 100,000) : 210.784
Average number of collision(overall) : 383.368
(List) Insertion fnished afterI: 0.022785
(Dictionary) Insertion fnished after: 0.347717
(List) Lookup fnished after : 120.328seconds
(Dictionary) Lookup fnished after: 2.05198seconds
can@can-GS60-2PE-Ghost-Pro:~/CLionProjects/Algo_Homework3$ 
```

While number of items increasing, hashing a number and finding a available place getting harder. First 1000 items can find a index using hash key easily, but after 100000 items, it can be hard to find a index and number of probing executes increases exponentially. The hash table keeps the book characters and take index which means hashed unique keys. Elements of input.txt files are read and assign as a bookCharacter object. Then, it create a unique key. The key is passed to hashing function and a hash key created. After that, the dictionary class which has a has table, insert objects to the table if the index is empty. If not, the hash key is passed to probing function until the key index is empty. While the number of element is closing to the size, number of executes of probing (collision) also increased.

Q4)

```
    lookup(unsigned long key) {
    unsigned long keyHash = hashing(key);
    unsigned long index = keyHash;
    int x = 1;
    while (hashTable[index]->createUniqKey() != key){
        index = probing(keyHash,x);
        x++;
    }
    return hashTable[index]->getValue();
}
```

Lookup function of dictionary has a while loop which so similar to insertion function. The worst case of lookup procedure is the being collision number maximum. For example, for 10 items, it can be 0+1+2+3+4+5+6+7+8+9 collisions. It means n(n-1) collisions can be occur and while loop of function executetes n(n-1) times. Therefore we can say the O(n^2) is the worst case complexity of lookup function.