Data Structure Assignment3

Hotel Finder

In this project, I implemented a hotel finder which allows users to find, delete, sort hotels and find all hotels in a city. The data structure behind the implementation is HashMap.

Basically I created two HashMap classes, one for finding, deleting, and sorting functions and another one for AllinCity (find all hotels in a city) function.

For HashNode class, each object has a key and a value. The key is a string while the value is an array of strings storing information related to the string. The class also has some accessing function such as getKey() and getValue().

In the first HashMap class, every hotel is an object of hashnode which has a key and a value. I used the combination of hotel name and city name separated by a comma as the key, and the rest information are put in a storing array as the value of the hashnode. Also, I have an array of size greater than 1.3 times the input that is used to store the pointer to each hashnode of hotel. For its functionalities, it first has a function called hashcode which maps an input string to an almost unique integer. I ues this hashcode to determine the associated index. Then this class contains functions including add(), find(), findindex(), remove() and sort(). Adding function takes in a string and an array as ingredients for an object of hashnode. It will use the index determined by hashcode to get the position in the large node array where the hashnode should go. If the indicated position is occupied already, do linear probing to get a free slot. Finding function takes in a string and output matched result of information of a hotel as an array. It will also use the hashcode and the index to find potential, possible position. If not found, continuing linearly checking the next position until found or iterated the whole array. Findindex basically does the same, but instead of returning the array of information, it returns the exact index of matched result for later use. Remove function has the same finding process as the finding function, but instead of returning the result, it deletes such found node by placing a placeholder available. The sorting function simply uses a vector to extract all hotels and put their information in a whole long string. Then it uses built-in algorithm to sort the whole vector alphabetical-increasingly and outputs the sorted vector for dumping. The class also has some trivial function like getSize() to return the number of nodes inputed.

In the second Hashmap, its functions include add(), remove(), listall().However, there are some major differences. Firstly, the whole container becomes an array storing vectors containing all hotels in a city as hashnodes. Namely, the vectors act as buckets storing hotels located in the same city. Secondly, the key of each node becomes just the city name instead of combination of hotel and city because this is used to realize AllinCity function. Thirdly, the adding function is different. Instead of linear probing the whole array like HashMap1, when inputting a node with key (city name), we use hashcode to get the index in the array and then we access the vector at the index and

push back the node. In this way, since the key is the city name, we ensure that all hotels located in the same city go into the same vector. For listall() function, it takes in a string of the city name and outputs the vector containing all hotels in such city. Then it will just use the printing function to display information of the hotels. The class also has some trivial function like getSize() to return the number of nodes inputted and cleaning function as well.

In the main structure, the program first reads in a file containing all hotels. It will first go through every line to count the total number of inputs and then get the prime number greater than 1.3 times that number in case to avoid collisions in the hash map as much as possible. Then I create two hash maps of type 1 and 2 with size of that prime number. Then read in the file again ad insert nodes to both of the maps at the same time. Notice the keys and values for the nodes to be inserted in different maps are different.

Then the program allows following functions: <find add delete dump allinCity quit> Entering “find” leads to find function and user will input hotel name and city name as combination key to perform find functionality in hashmap1. If found, it will print the results accordingly along with the comparison and execution time. Otherwise, it informs the user no results matched. Entering “add” lead to insert function. User will input all information about a hotel and then I will create two hash nodes accordingly to be inserted into two maps. Entering “delete” leads to delete function. User will input hotel name and city name as key. Then the program will go look for this key in hash map 1 and delete it in both maps using delete function accordingly. If not found in hash map 1, it also doesn’t exist in hash map2, so inform user nothing found. Entering “dump” followed by a file name leads to the sorting and dumping function in hash map2, which will extract every hotel along with its information as a long string and sort these strings in a vector by the built-in algorithm sorting. Then by the input file name from the user, the program will dump the sorted results to the file line by line. Entering “allinCity” followed by a string of a city name leads to AllinCity function. It will take in a city name as find key and go to the index given by the hashcode. Then it will access the vector at the index and print all hotels information in the vector. If the position is empty or the input city name and the one at the index don’t match (hash codes coincide somehow), then it means nothing found in such city/position. Entering “quit” simply leads to the exit of the program and entering any other things is illegal and the program will ask for input again.

Regarding the efficiency, since the underlying data structure is hash maps, they help us increase the efficiency a lot. Specifically, finding, adding, deleting, and allinCity only take constant time O(1) with the use of hash code and given that the nodes are distributed well by the design of hash code. For dumping function, the worst-case time complexity is O(nlogn) because sorting algorithm in STL takes O(nlogn) time and extracting all hotels and dumping the information onto a file take O(n) time so overall the function takes O(nlogn) time. Therefore, the program is very effective in

general.