Saad Teeti

New York University in Abu Dhabi

In this project I have developed a program that takes an input of file csv. The input file includes a list of hotel Name, city Name, stars, price, country Name, and address. After I learned how to implement hash tables. I realized how my code would be like. In order to make this code I created multiple classes such as Hotel, Hash Nodes, and the HashMap class. Hotel class, which included hotel Name, city Name, stars, price, country Name, and address. Hash Node Class which included the key and the value, and HashMap class which included the different methods required for the assignment like insert, remove, search, all in city and dump. This report includes the decision that I used in order to make my program in terms of efficiency. It also the flow of the program and some challenges that faced though out the process.

Decision in terms of efficiency:

Efficiency:

I decided to use only separate chaining instead of both probing and separate chaining. I created two hash tables, the key for first one includes the hotel name and the city name, and the second hash tables, my key was only the city name. For both Hash tables my value was the same, which was an object of type hotel that included hotel Name, city Name, stars, price, country Name, and address. The reason for that is that the code would be clearer, and in terms of efficiency, if I decided to do either both chaining and probing or only probing, my code, would have included the methods for chaining in both cases, therefore I thought it would be better to just use chaining. My program would still as efficient if I only implemented chaining.

The flow of the Program:

After we created the Class HashMap, I used different methods in order to implement the hash table. The first method that I tend to use was the insert method. I had two type of insert methods, the first one was for the first HashMap, and the second one was for the second was for the second HashMap. For both the insert methods would be O(1) which is in average complexity and O(n) in worst case complexity. For the first insert, we had a condition that we cant repeat the same key, so for each key we had one value. For the second insert method we were able to repeat the key which was the city name, then we go through the list and compare if the key matches the value, then we had to check that if the value is already in the has table to not update it. For remove it was also the same as the insert function, we check if the key matches the value, and we erase.

The search function used the hotel and city name. it compared if the key for the hotel and the city name matched the key for the value, and if it did, then it would display the value. The function gets through all the city name and the hotel name key, and goes through all the nodes of the buckets, to find the matched value. The same implementation was used for all in city function where the function gets through city name key, and goes through all the nodes of the buckets, to find the matched value. For both the search methods and all in city method, O(1) would be the average complexity and O(n) would be the worst case complexity. Whenever I am deleting or inserting into a hash table, I need both function of insert and both functions of delete, to be called, in order to make sure that when I display all in city, I would have either a value deleted or inserted depending on the user preference.

I also had an is Prime function that I would use to find the size of the map. The is prime function check takes the number and find the module for it from 2 until the square root of the number, if at any point the module would be 0 then the functions return false in which the

number is not prime, if it doesn’t return 0 then its prime. This process is little bit slower if the input value is bigger.