**Report   
Programming Assignment #5**

First Name: Pratik Patel UIN: 527004337

User Name: p.pratik99 E-mail address: [p.pratik99@tamu.edu](mailto:p.pratik99@tamu.edu)

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office.

|  |  |  |  |
| --- | --- | --- | --- |
| Type of sources | 1 | 2 | 3 |
| People |  |  |  |
| Web pages (provide URL) | cplusplus.com |  |  |
| Printed material | Textbook |  |  |
| Other Sources | Lecture Slides |  |  |

1. A description of the assignment objective, how to compile and run your programs, and an explanation of your program structure (i.e. a description of the classes you use, the relationship between the classes, and the functions or classes in addition to those in the lecture notes).
   1. Objective: The main objective of this assignment is to implement a Skiplist data structure. It also asks for implementation of various function of Skiplist class such as insert, find, print etc. It also asks for a comparison to Binary Search Trees data structure in terms of Big-Oh notation.
   2. To compile this assignment, just run the following command on any Unix based system, g++ -std=c++11 main.cpp SkipList.cpp. In order to run the program on the command line, execute the following command, ./a.out.
   3. There are main three files, SkipList.h, SKipList.cpp and main.cpp. SkipList.h contains the declaration of struct Node as well as the declaration of the class SkipList. SkipList.cpp contains definitions to the functions declared in SkipList.h. main.cpp is a driver file. It interacts between the user and SkipList class.
2. A brief description of the data structure you created (i.e. a theoretical definition of the data structure and the actual data arrangement in the classes).

I have created a Skiplist. I have used a vector of lists to implement the Skiplist. In skiplist each Node has one iterator to iterator down and value. In Binary Search Tree, each Node to the left of a given Node has to be less than the given Node and each Node on the right is greater than the given Node. In skiplist, flip coins decide that how many lists will have a specific value. Having said that, each level contains ½ the elements than the level below it.

1. A description of how you implemented the calculation of (a) insert cost (b) search cost (c) delete cost.
   1. I have calculated the insert cost for each node at the time of insertion. It is equal to the number of times while loop, inside the insert functions, executions. Because while loop terminates after the 0th level.
   2. I have calculated the search cost the same way as insertion. The only difference is that it should be equal or less than insert cost. Because if the value exists on any higher level than the algorithm does not need to go any lower in the vector, whereas in the insert function we have to go all the way till 0th list in the vector.
   3. Deletion cost will be same as search cost. Once we get to the desired node in the list, we do not need to compare any further. We simply need to delete the nodes down to that one.
2. Best case, worst case, and average case theoretical runtimes (Big-O) for the insert, search, and delete functions.

All functions have the same worst, average and best scenarios and the Big-Oh notations. Therefore, I have only described the Insert Function. The other two have a similar explanation to insert because they share a common algorithm for most of the time.

* 1. Insert Function
     1. Best case: it is the case when we can insert a node in the highest level which only needs one or two comparisons depending on your implementation. Therefore, the Big-Oh will be **.**
     2. Average Case: when the flips are randomized, and we assume that the probability of getting heads or tails is 0.5 which makes it a randomized skiplist. Therefore, the list should have . Therefore, the Big-Oh notation will be **.**
     3. Worst Case: When the probability of getting tails is zero or near zero, the skiplist will eventually turn into a linked list and we will have to parse n elements to insert a Node. Therefore, the Big-Oh notation will be **.**

1. Additionally, answer the following questions:
   1. How likely is that an item will be inserted into the level of the skip list?

To insert a node into the level, we need heads in flips. The probability of getting n-1 heads in n flips is equal to .

* 1. If you were to increase the probability of getting a “head” (positive result, keep flipping the “coin”), what would this do to the average runtime of insert, search and delete?

In both the cases, increasing or decreasing the probability of the getting a head, will result in an imperfect skiplist. Which might have more or less number of nodes in a given level than it supposed to have with the probability .

* 1. How does the order of the data (sorted, reverse sorted, random) affect the number of comparisons?

The order of the input data does not affect the search and delete functions because the skiplist will be the same for all three types of data after inserting them. The insert function will have a smaller number of comparisons if the data is sorted than random. Because to go down one level after inserting a node, sometimes there are more elements between two numbers already present in the lower level than in the higher level. The Big-Oh notations should remain same.

* 1. How does the runtime compare to a Binary Search Tree for the insert, search and delete operations?

If we assume that the coin is a fair coin than it is better to use the skiplist over a regular binary search tree for sorted input. Because a regular binary search tree will turn into a linked list if the input data is sorted. Whereas it isn’t the case for the skiplist.

* 1. In what cases might a Binary Search Tree be more efficient than a skip list? In what cases in might be less efficient?

If the coin is a biased one and the data is random than it is better to use a binary search tree over a skiplist. If the coin is not a biased one and the data is sorted than it is better to use a skiplist over a regular binary search tree.

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

“*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*”

Your Name: Pratik Patel Date: 04/13/2019