ASSIGNMENT 5

Introduction

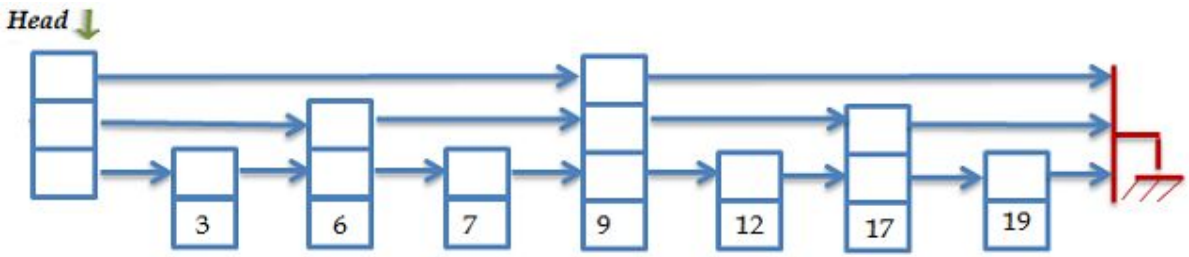
Many a times a complex problem becomes easier to solve if you have the appropriate data structure at hand. More so when the input size involved is very large. Skip List is one such simple data structure which will help you efficiently solve problems where ordering is required or dictionaries are required.

Objective

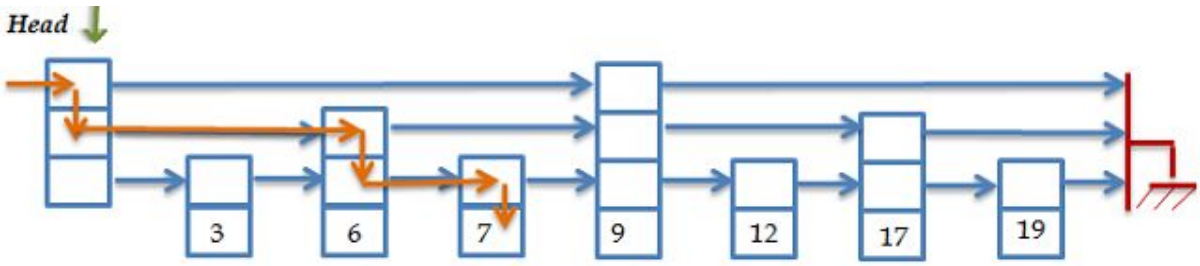
The aim of this assignment is to introduce you to the Skip List data structure including implementation of various operations on it.

Description

Skip Lists are an alternative representation of balanced trees that provides approximate O(log n) access times with the advantage that the implementation is straightforward and the storage requirement is less than a standard balanced tree structure. A distinct difference from most search structures is that the behaviour of skip lists is probabilistic without dependence on the order of insertion of elements.



Each node in a skip list has an array of several forward pointers that link to nodes next in the list. The number of forward pointers (node height) in a particular node is determined by a probability function that is applied during insertion. The distribution of the node heights is random.



To search for a key in the skip list, start at the root and work left to right. For the current node, work down from the top of the array of forward pointers. Find the first forward pointer that links to another node. Compare the search key against the key of the forward node. If the keys match, you are done. If the search key is less than the forward node, then the value is on the left side of the forward node, otherwise it is on the right side. If it is on the left side, stay on the current node and go down one and the right one to the next forward node and repeat. If the key is on the right side, move to the node that is linked by the forward node, and start over.

This way, the entire list need not be traversed and the entire search operation can be carried out in O(log n) instead of O(n) time.

Task

You are required to implement a dictionary using a skip list with a max level of 16 which allows basic operations like insertion in the list, deletion from the list, searching and displaying the entire list. There are two parts of the assignment.

Part 1

1. The input will be given through a file and will be as follows:-
   (a) The first line of the input file will be a number **N** indicating how many entries are there in the file.

   (b) Subsequent **N** lines will be in the form of "***key,value,height***"

2. The output will be a GraphViz dot file as described in the class.
   (a) GraphViz dot is a specification for representing graphs, state machines and transition diagrams. A sample file for skip list representation is uploaded in moodle.

Part 2

The input will be given through two input files and will be as follows:-

1. The first file will be defining the dictionary as follows:-
   (a) The first line of this input file will be a number **N** indicating how many entries are there in the file.

   (b) Subsequent **N** lines will be in the form of "***key,value***"

2. The second input file will be as follows:-
   (a) The first line will be a number **M** indicating the number of search keys.
   (b) Subsequent **M** lines will be in the form of "***search key***".

3. For the same input, you will vary the value of the probability ***p*** from the set $\{\frac{1}{2}, \frac{1}{e}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$

4. For every value of ***p*** you will search your skip list for the search keys from the input file and note the total search time. Finally, using ***gnuplot***, make a graph of **`p`** `Vs` **`search times.`**

Skip List Operations

The operations are described below:-

(a) `insertp(SkipList *, int key, char * val, float p)`
Insert a node into the skip list with the key and value as given using ***p*** as the probability for calculating node height. If the key already exists, then just update the value with this value.

(b) `inserth(SkipList *, int key, char * val, int h)`
Insert a node into the skip list with the key and value as given using ***h*** as the node height. If the key already exists, then just update the value with this value.

(c) `delete(SkipList *, int key)`
Delete the node from the SkipList with the given key

(d) `find(SkipList *, int key)`
Find in the SkipList the value corresponding to the key provided.

(e) `display(SkipList *)`
Output the skip list as a GraphViz DOT document.

Deliverables

You will submit your C/C++ program along with the makefile, and doxy file as a single **rollno_a5.tar.gz** file in the moodle submission link. It is mandatory that your code should follow proper indentation and commenting style. There will be deductions in the awarded marks, if you fail to do so. If you are using C++, the use of STL is not permitted.