UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 1

Release: Monday 22 March 2021, 18:00
Deadline: Friday 26 March 2021, 18:00

PLAGIARISM POLICY

UNIVERSITY OF PRETORIA

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.library.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the *Library* quick link, and then choose the *Plagiarism* option under the *Services* menu). If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding. Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

Objectives

The aim of this practical is to learn how to implement and use skip lists.

Instructions

Complete the task below. Certain classes have been provided for you alongside this specification in the *Student files* folder. You have been given a main file which will test some code functionality, but it is by no means intended to provide extensive test coverage. You are encouraged to edit this file and test your code more thoroughly. Remember to test boundary cases. Submission instructions are given at the end of this document.

Task 1: Skip Lists [30]

A skip list is a variant of the ordered linked list that makes a non-sequential search possible. This data structure was proposed by William Pugh in his paper "*Skip Lists: A Probabilistic Alternative to Balanced Trees*". It allows *O(log n)* search complexity as well as *O(log n)* insertion and deletion complexity within an ordered sequence of `n` elements. This is achieved by building a linked list with multiple layers or levels. The bottom level is an ordinary linked list, while each higher level is a sparser subset list of the level below it.

You have been given a partially implemented skip list class and a skip list node class to use. Your task is to implement the following methods in the skip list class according to the given specification:

```
boolean isEmpty()
```

This function should determine whether the skip list is empty or not. It returns true if the skip list is empty and false otherwise.

```
void insert(T key)
```

This function should insert the given `key` in the skip list. `key` should be inserted in the correct position in the skip list to maintain the increasing order. `key` should also be added to all the levels as determined by the `chooseLevel()` method. Duplicate keys should not be added into the skip list.

```
boolean delete(T key)
```

This function should delete the given `key` from the skip list. It returns true if `key` is removed successfully and false otherwise. `key` should be removed from the skip list from all the levels it is referenced in.

`T first()`

This function should determine the first key in the skip list. It returns the first key value in the skip list. Return `null` if the skip list is empty.

`T search(T key)`

This function should find the given `key` in the skip list. If `key` is found in the skip list it should return the key value and otherwise it should return `null`.

`String getPathToLastNode()`

This method should return a string representing the path to the **last node** in the skip list by taking the **fewest number of skips**. The shortest route to the last node in the skip list involves traversing the skip list in **descending order** of level. Return a string with each **key** in the path in square brackets appended together with no whitespace at all. If the list is empty return an empty string. Please consult Figure 1 and 2 for examples of precisely how the output string should look like.

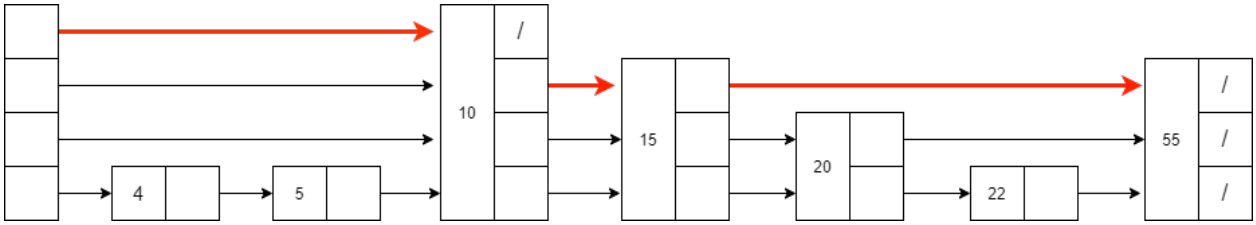

Figure 1: Example of the path with the fewest number of skips in order to reach the last node. In this case return the string: **[10][15][55]**

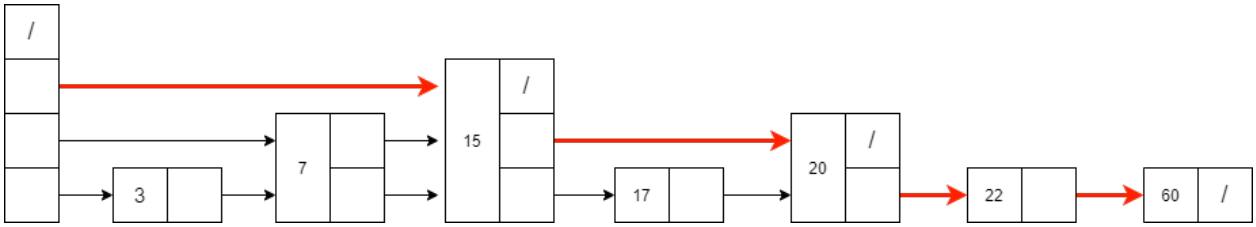

Figure 2: Second example of traversing the skip list to the last node. In this case return the string: **[15][20][22][60]**

Only implement the methods listed above. Do not modify any of the other code that you were given for this task.

Submission

You need to submit your source files on the Fitch Fork website (https://ff.cs.up.ac.za/). All methods need to be implemented (or at least stubbed) before submission. Place your **SkipList.java** and **SkipListNode.java** file in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number. There is no need to include any other files in your submission. You have 4 days to complete this practical, regardless of which practical session you attend. You have 5 submissions and your best mark will be your final mark. Upload your archive to the *Practical 1* slot on the Fitch Fork website. Submit your work before the deadline. **No late submissions will be accepted!**