APS Project Progress Report

Splay Tree Implementation and Comparison with BST

Manojit Chakraborty Shubham Das

31 October 2018

1 Title

Splay Tree Implementation and Performance Comparison with BST

2 Team Members

- **Manojit Chakraborty** - M.Tech CSIS - 2018201032
- **Shubham Das** - M.Tech CSIS - 2018202004

3 Deliverables

- Study the analysis of Splay Tree
- Implement Splay Tree
- Implement Binary Search Tree
- Compare performance of the Splay Tree with that of BST for different scenarios.
- Make a detailed report of all the benchmarking and case studies.

4 Project Delivery plan

- Study about how BST works and how insert, delete, search operations can be implemented. [October 20-21]
- Implement a BST Class using Python [October 21]
- Study thoroughly about Splay Tree and all the cases for its insert, delete and search operations from the online resources and books. [October 22-25]

- Implement a Splay Tree class by extending the BST class methods of insert, search and delete. [October 25-26]
- Compare the performance of the Splay Tree with that of BST for these different scenarios (Initial Thought) -
  - Insert in increasing order, Search in increasing order, Delete in increasing order.
  - Insert in increasing order, Search in decreasing order, Delete in decreasing order
  - Insert in random order, Search in same random order, Delete in same random order.
  - Insert in random order, search in different random order.
  - Insert in random order, Search same 4 elements randomly.
  - Insert in random order, search same 10 elements randomly.
  - Insert in random order, Delete 1000 elements randomly.

  Here, the tree size will vary from 5000 to 1000000 approximately.
- Calculate all these performance measures for a number of times, generate the average values for each type of measures for both the trees, with varying tree size. [October 27-30]
- Create plots for the performance measure values in each test case for Splay Tree and BST using Matplotlib and/or Seaborn in Jupyter Notebook and generate an overall Performance Comparison Report. [October 31-November 3]
- Find an application where Splay Tree is used extensively and test the application with our own implementation and collect performance measures and results. [November 3-7]

5 Technologies to be Used

- Python
- Jupyter Notebook
- Visualization tools like Matplotlib, Seaborn

6 Online Resources

- https://en.wikipedia.org/wiki/Splay_tree
- https://www.geeksforgeeks.org/splay-tree-set-1-insert/
- https://www.geeksforgeeks.org/splay-tree-set-2-insert-delete/
- https://www.cs.usfca.edu/ galles/visualization/SplayTree.html

7 Repository where work is being committed

Github Repository : https://github.com/manojit32/APS_Project_Splay_Tree

8 Plan for Testing

We will test the performance of the Splay Tree with that of BST for these different scenarios (Initial Thought) -

- Insert in increasing order, Search in increasing order, Delete in increasing order.
- Insert in increasing order, Search in decreasing order, Delete in decreasing order
- Insert in random order, Search in same random order, Delete in same random order.
- Insert in random order, search in different random order.
- Insert in random order, Search same 4 elements randomly.
- Insert in random order, search same 10 elements randomly.
- Insert in random order, Delete 1000 elements randomly.

Here, the tree size will vary from 5000 to 1000000 approximately. We will take records of Time taken, Number of Comparisons, Space taken for each of these test cases by running them a number of times and then calculate average of each value for each of the test cases.

9 End user Document

9.1 Splay Tree

- Create a Splay Tree by using these following commands in Python Shell-

  import splay
  splaytree = splay.Splay()

- For insertion into the Splay Tree, write -

  splaytree.insert(value)

- To search a value within the Splay Tree, write -

  splaytree.search(value)

  which will return True if value exists, False otherwise.

- For deletion of a value from the Splay Tree, write -

  splaytree.remove(value)

  which will successfully delete the value if it exists in the tree. If the value is not present in the tree, it will raise an error.

9.2 Binary Search Tree

- Create a Binary Search Tree by using these following commands in Python Shell-

  import bst
  bstree = bst.BST()

- For insertion into the Binary Search Tree, write -

  bstree.insert(value)

- To search a value within the Binary Search Tree, write -

  bstree.search(value)

  which will return True if value exists, False otherwise.

- For deletion of a value from the Binary Search Tree, write -

  bstree.remove(value)

  which will successfully delete the value if it exists in the tree. If the value is not present in the tree, it will raise an error.

9.3 Performance Testing

To run an overall performance testing of Splay Tree against BST, run the following command in the terminal -

$ *python3 test_suite.py*