Data Structures Final Project: Fibonacci Heap

Suyog Joshi
December 21, 2022

1 Introduction

- I have implemented fibonacci heaps as my final project. A Fibonacci heap is a collection of trees which follow the min or max heap property. I have made a Fibonacci heap following the min heap property. The main operations that can be performed are: **insert, delete, extractMin, mergeHeaps, and decreaseKey**. There is an 'interface' to perform all of these operations. It allows you to choose the number of heaps you want to create, and stores them in a list of heaps. Then, you can choose a heap to modify, and are given the choice between the above operations. Once the operation is done, you can again choose an operation to do, or can choose a different heap to modify. These are the possible operations:
  1. *Insert* - asks for an input and inserts a node with that value into the heap.
  2. *Delete* - asks for an input and deletes the node with that value from the heap.
  3. *Extract Min* - removes the minimum element from the heap, and rearranges the heap to follow the min heap property, if needed. Prints the value of the extracted node.
  4. *Merge Heaps* - asks for the index of the heap that you want to merge with the one you are modifying. Merges them into the same index as you are currently in.
  5. *Decrease Key* - asks for two inputs. The first input is the value that you want to change, and the second is the value that you want to change it to. Changes the value, and rearranges the heap if necessary.
  6. *Print Heap* - prints the heap.
  7. *Print Min* - prints the value of the min node.
  8. *Exit* - exits the operation chooser. Goes to the heap chooser.

2 Testing

- To test the program, I wrote a function called **tester()**, which generates n random integers, and inserts them into the list. This function can be modified to call any of the above operations. To use this function, you have to change the value of *bool test* in the main function to true. In this function, I tested all of the operations with various quantities of heaps and nodes.
- To test non-integer inputs, I manually inputted values in the interface.
  - *Strings*: if the string begins with a non-digit character, the input is rejected and the program asks for a valid input. If the string begins with a digit, all digits before the first non-digit character are counted as a single integer, and that value is used as the input.
  - *Floats*: the value before the decimal point is taken.
  - *Errors*: any input greater than 16 characters breaks the program.

  After each operation, I had to manually check if the result was correct. All of the operations work as expected.

3 How to Use

- To use the program, you can use the interface or the tester function. Setting the value of *bool test* to true will ignore the interface, and call the **tester()** function.The interface first asks how many heaps you want to create. The heaps are indexed from 1 to n (the input). Then, you can choose which heap you want to modify. Finally, you are given the choice between the 8 operations listed above.