CSE 373: Homework 2

Heaps and Big O

Due: April 12th, 11:59 PM to Canvas

Introduction

In this assignment, you will implement the Priority Queue ADT (represented by the PriorityQueue.java interface). You will do this by creating your own BinaryHeap class that implements PriorityQueue. Again, you will also create a simple testing suite for PriorityQueues. This time, however, you will not have access to any .class files. For this portion, try and make your tests as complete as possible. Test all applicable functions.

Importing into Eclipse

Since there are no .class files, importing into eclipse will be very easy. Download the HW2Eclipse.zip file from the website. In eclipse, select File -¿ Import -¿ Existing Projects into Workspace. There, choose Select Archive File and select the .zip file. Make sure HW2 is selected and click finish.

If you choose not to use eclipse, the relevant java files are available in HW2StarterCode.zip.

1 Implementing a Priority Queue

In this part of the assignment, you will implement a Priority Queue. We have given you an interface, PriorityQueue.java and some starter code for your implementation in BinaryHeap.java. *Do not modify the PriorityQueue interface.* We have given you the private class, HeapData, that holds data and priority together. You may add additional functionality to this class as you see fit. For simplicity, you may assume that Strings inserted into the heap are unique, but the heap should support duplicate priorities. You may create helper functions as needed. When implementing the BinaryHeap, we recommend that you use indexing that starts at 1 to make parent/child calculations easier.

There are eight functions that we expect you to implement in BinaryHeap:

- `BinaryHeap()`: The default constructor for BinaryHeap. It should instantialize an empty Heap.

- `BinaryHeap(int startArray)`: The default constructor for BinaryHeap. It should instantialize an empty Heap where the array is of size startArray
- `boolean isEmpty()`: This function returns true if the heap is empty and false otherwise.
- `int size()`: This function returns the current number of items in the heap.
- `String findMin()`: This function returns the item of smallest priority in the heap. It should not remove that item from the heap. If the heap is empty, it should return null.
- `void insert(String data, int priority)`: This function adds the String, priority pair into the heap. Utilize the private class HeapData to store this pair. If the array is full when an insert is called, you must resize the array to accommodate the new data.
- `String deleteMin()`: This function should return and remove the item of smallest priority in the heap. Objects of duplicate priority may dequeue in any order. If the heap is empty, it should return null.
- `void makeEmpty()`: This function should remove all elements from the heap.
- `boolean changePriority(String data, int newPri)`: This element should change the priority of a String in the heap. You may assume that the Strings input into the heap are unique. This function returns true if successful, and false if it is unable to find the String in the heap.

Your grade for this code will be based on the ability of your implementation to pass our tests.

2 Testing the Priority Queue

Again, we will expect you to provide tests for the BinaryHeap data structure. Do this in the BinaryHeapTest.java class. In this assignment, you will not be given any test implementations. Because of this, make sure your tests are as thorough as possible. At the start of each test, assume that toTest is an empty heap. For testEmpty and testOne, only test functions that are applicable for those sizes. testMany should test all functions. For this, helper tests may be incredibly helpful. You do not have to pass empty BinaryHeaps to your helper functions, use them entirely as you see fit. These three given tests should all return true when passed and false if the implementation fails the test. Use the `public static`

`void main()` to run your tests and test your implementation. Only the three test functions (and whatever helper functions they may use) will be graded. Your grade from this portion will be based on the ability of tests to recognize correct and incorrect implementations of the PriorityQueue.

3 Writeup

The writeup will contain a couple questions about your implementation and test suites and theoretical questions about heaps in general.

1. For each of the functions that you implemented given by the PriorityQueue.java interface, give the worst-case bigO notation for their runtime. Give the notation in terms of `n`, which should be the number of elements in the heap. This should be the tightest upper bound you can provide and should be a bigTheta bound if possible. Explain briefly how you came to each answer.

2. From your testMany function, describe any helper tests that you may have created. Additionally, identify the function you found most difficult to test. Explain why this function was most difficult. Finally, explain why BinaryHeap was easier or more difficult to test than the Queue from last week.

3. Given the following segments of code, determine their asymptotic (bigTheta) runtimes in terms of `n`. For part c, only give the runtime of `mysteryThree(int n)`

   (a)
```
public void mysteryOne(int n) {
  int sum = 0;
  for (int i = n; i >= 0; i--) {
    if ((i % 5) == 0) {
      break;
    } else {
      for (int j = 1; j < n; j*=2) {
        sum++;
      }
    }
  }
}
```

   (b)
```
public void mysteryTwo(int n) {
```

```
  int x = 0;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < (n * (n + 1) / 3); j++) {
      x += j;
    }
  }
}
```

(c)
```
public void mysteryThree(int n) {
  for (int i = 0; i < n; i++) {
    printCats(n);
  }
}

public void printCats(int n) {
  for (int i = 0; i < n; i++) {
    System.out.println("catsmoop");
  }
}
```

Deliverables

For this assignment, there will be three submissions on Canvas.

- Part 1 consists of only your BinaryHeap.java implementation. Submit this file only to H2P1 on Canvas.
- Part 2 consists of your file BinaryHeapTest.java. Submit this file only to H2P2 on Canvas.
- Part 3 consists of the writeup for the problem. Submit a .pdf of typed or neatly written solutions to H2P3 on Canvas.