Version: 3-23-2022 10:00 PM

CS 2223 D22 Term. Homework 2

Homework Instructions

- This homework is to be completed individually. If you have any questions as to what constitutes improper behavior, review the examples as I have posted online http://web.cs.wpi.edu/~heineman/html/teaching_/cs2223/d22/#policies
- Due Date for this assignment is **6 PM April 4th**. Homework submissions received after 6PM receive a 25% late penalty. Extension is good for 48 hours.
- Submit your assignments electronically using the canvas site for CS2223. Login to canvas.wpi.edu and locate HW2. You must submit a single ZIP file that contains all of your code as well as the written answers to the assignment.
- All of your Java classes must be defined in a package, such as USERID.hw2, where USERID is your login name

Questions

1. Mathematical Analysis [20 points] (1 bonus point)
2. Working with linked lists [20 points]
3. Data Type Exercise [60 points] (4 bonus point)

Getting Started

Copy the following files into your USERID.hw2 package:

- `algs.hw2.Analysis`
- `algs.hw2.RangeList`
- `algs.hw2.SpecialQueue`
- `algs.hw2.PerformanceRangeList` ← Use to validate you achieve expected performance
- `algs.hw2.PerformanceSpecialQueue` ← Use to validate you achieve expected performance

If you are attempting the bonus questions, then write your results in a writtenQuestions.txt file

Please do your best not to procrastinate on this assignment, since it will truly test your debugging skills.

Version: 3-23-2022 10:00 PM

Q1. Mathematical Analysis [20 points]

This question is a more complicated version of what you will see on the midterm exam. You can find this code in the **`algs.hw2.Analysis`** class. Copy this class into your **`USERID.hw2`** package and modify it based on the requirements below.

Given the following `proc` function, let S(N) be the number of times `power(base, exp)` is invoked when calling `proc(A, 0, n-1)` on an array, **`A`**, of length **`n`** containing integer values from **`0`** to **`n-1`**.

```java
static long power(int base, int exp) {
  return (long) Math.pow(base, exp);
}

public static long proc(int[] A, int lo, int hi) {
  long v = power(A[lo], 2);
  if (lo == hi) {
    return v + power(A[hi],2);
  }

  int m = (lo + hi) / 2;
  long total = proc(A, lo, m) + proc(A, m+1, hi);
  while (hi > lo) {
    total += power(A[hi], 2);
    lo += 2;
  }
  return total;
}
```

For this assignment, develop the recurrence relationship for S(N) and compute its closed-form formula. Then modify the `Analysis` class to output an updated table that shows the computed counts and the result of your model.

Question 1.1 (8 pts.)

Identify the Base Case for S() and the Recursive Case for S(n). Refer back to lecture for the format of this question.

Question 1.2 (12 pts.)

Derive an exact solution to the recurrence for S(N) when N is a power of 2. Be sure to show your work either in the `WrittenQuestions.txt` file or by a picture that you upload. Validate that your formula is correct by modifying the `model(int n)` implementation so it prints proper result when run.

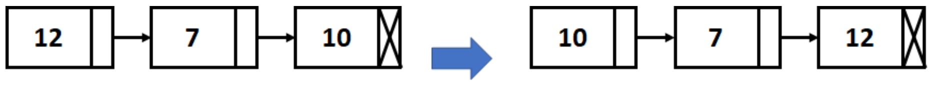
Bonus Question 1.3 (1pt.)

Derive a formula that predicts the `Value` printed for `proc(A, 0, A.length-1)` when A contains the integers from 0 to N-1 and N is a power of 2. *[Note: I am still working on this one! Might take several hours]*

Q2. Working with Linked Lists (20 pts)

In class I have described how to implement the Queue data type using linked lists. For this assignment, you are going to create a `SpecialQueue` that uses a linked list of Node objects to provide the following API. In addition to the expected `enqueue()`, `dequeue()`, `size()` and `isEmpty()` methods, you must implement two more:

- `swapEndPoints()` must swap the locations of the oldest and youngest values within the queue. You must abide by the restriction that **the value stored in each node cannot be changed.** For example, given `SpecialQueue` on left, calling `swapEndPoints()` results in:



Before *After*

- `dequeueLargest()` must remove from the queue the node containing the largest value and return that value.

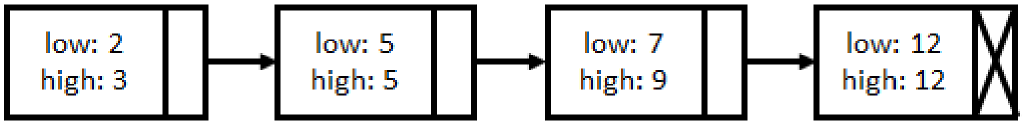
```
package USERID.hw2;
public class SpecialQueue {

  class Node {
    final int   value;     // once created, cannot change
    Node        next;

    public Node (int val) { this.value = val; }
  }

  // operations are independent of N, the # of elements in the queue.
  public void enqueue(int val) { ... }
  public int dequeue() { ... }
  public int size() { ... }
  public boolean isEmpty() { ... }
  public void swapEndPoints() { ... }

  // operations are dependent on N, the # of elements in the queue.
  public int dequeueLargest() { ... }
}
```

The `dequeueLargest()` method executes in time O(N), in other words, directly proportional to the number of values in `SpecialQueue`. All other methods can be implemented to perform in constant O(1) time.

If you cannot make `swapEndPoints()` execute in O(1) time, then you must at least ensure a correct implementation.

Q3. Range List Data Type Exercise (76 pts)

This assignment gives you a chance to demonstrate your ability to program with Linked Lists. You are to implement the following data type *RangeList* which represents a collection of consecutive integer ranges using a linked list of ranges. A *range* of consecutive integers is defined by a low integer and a high integer. For example, you can represent the sequence of consecutive integers 2, 3, 4, and 5 by the range [2,5]. You can represent a collection of unique positive integers as a collection of ranges. For example, the values 2, 3, 5, 7, 8, 9, 12 can be represented by the linked list of ranges below:



As you can see from this example, a range can consist of just a single value. The ranges in a RangeList must satisfy the following two properties:

- **Ascending property** – the ranges appear in ascending order, such that all integers in a range are smaller than the ranges that come after it, and larger than the ranges that appear before it.
- **Gap property** – Given two consecutive ranges, R1 and R2, where R1.next = R2, there must exist an integer N such that R1.high < N and N < R2.low.

The example RangeList above does not contain the integers 4, 6 or 10. The **Gap property** ensures that a RangeList uses the least amount of storage. A RangeList can be dynamically created by adding and removing integer values. For example, given the RangeList containing a single range [1,3], when you remove the integer 2 from the RangeList, the result is a RangeList consisting of two ranges, [1,1] → [3,3]. Similarly, if to an empty RangeList, add the integers 1 and 3, the resulting structure contains two ranges [1,1] → [3,3], which satisfies the **Ascending property**. If you then add the integer 2 to this RangeList, the resulting structure must be [1,3] to satisfy the **Gap property**.

This question will prove to be a challenging programming assignment. Do not procrastinate!

```
package USERID.hw2;
public class RangeList {

  class Node {
    int     low;
    int     hi;
    Node    next;

    public Node (int lo, int hi)
  }

  // operations that are independent of the size of the Composite
  public int numberValues() { ... }

  // operations that are dependent (in some way) on N. See documentation
  public int numberRanges() { ... }
  public boolean equals(Object o) { ... }
  public boolean subsetOf(RangeList rl) { ... }
  public boolean contains(int val) { ... }
  public boolean add(int val) { ... }
  public boolean remove(int val) { ... }
  public String toString() { ... }

  // bonus methods only
  public int random() { ... }
  public Iterator<Integer> iterator() { ... }
```

Copy **`algs.hw2.RangeList`** into `USERID.hw2` and complete its implementation, which must conform to performance specifications that are included in the sample code. More documentation is found in the sample file. In he performance specifications, N refers to the number of integers in the `RangeList`.

We will validate the output against the set of test cases in **TestRangeList** that we develop for the grading. Individual breakdown of points is found on the rubric. There is a **PerformanceTest** included which reveals the total time to compute $(n!)^2$ for standard values of n (to run this performance test, copy it to your `USERID.hw2` package and execute).

This assignment is challenging if you have not programmed extensively with linked lists. Do not wait. Get started on this assignment as soon as you can.

Bonus Question 3.1. (1 pt) Implement `random()` method that returns a random integer uniformly selected among the integers in the range list.

Bonus Question 3.2. (1 pt) Implement `iterator()` that produces an Iterator object that returns the integers contained in the `RangeList` in ascending order.

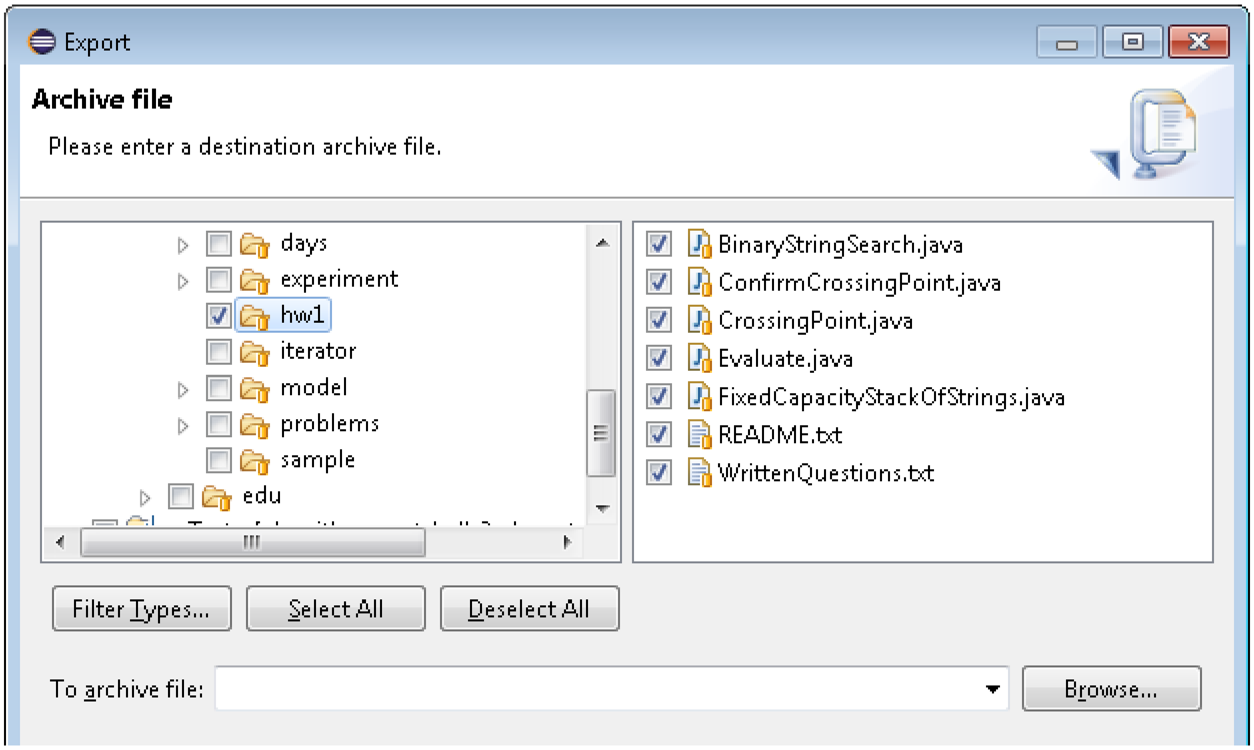
Bonus Question 3.3. (2 pt) Let's say you want to uniformly randomly choose N integers from among the integers 1 to N. What is the expected number of unique integers that result? You can use `RangeList` to empirically answer this question. You can find online evidence that the expected number of unique integers chosen in this way is $N * (1 - \frac{1}{e})$. Write a program to confirm this result empirically using values of N doubling from 128 to 65536. Now go further and empirically determine the expected number of unique ranges in the `RangeList` empirically and develop a formula that predicts this value. **HINT:** It looks something like the formula for the expected number of unique integers.

Version: 3-23-2022 10:00 PM

Submission Details

Each student is to submit a single ZIP file that will contain the implementations. In addition, there is a file "WrittenQuestions.txt" in which you are to complete the short answer problems on the homework.

The best way to prepare your ZIP file is to export your entire **`USERID.hw2`** package to a ZIP file using Eclipse. Select your package and then choose menu item "**Export…**" which will bring up the Export wizard. Expand the **General** folder and select **Archive File** then click **Next**.



You will see something like the above. Make sure that the entire "hw2" package is selected and all of the files within it will also be selected. Then click on **Browse…** to place the exported file on disk and call it USERID-HW2.zip or something like that. Then you will submit this single zip file in canvas.wpi.edu as your homework2 submission.

Addendum

If you discover anything materially wrong with these questions, be sure to contact the professor or TA/SAs posting to the discussion forum for HW2 on piazza;

When I make changes to the questions, I enter my changes in red colored text as shown here.

Change Log

1. Fixed the linked li