Version: Apr 6th 2020 9:00 AM

CS 2223 D20 Term. Homework 2

Homework Instructions

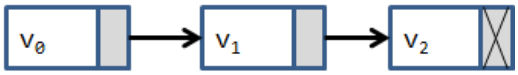
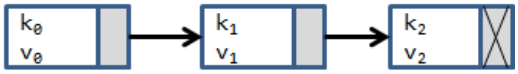
- This homework is to be completed individually. If you have any questions as to what constitutes improper behavior, review the examples as I have posted online http://web.cs.wpi.edu/~heineman/html/teaching_/cs2223/d20/#policies.
- Due Date for this assignment is 2PM April 16th. Submissions received after 2PM receive a 25% late penalty. Submissions received after 6PM will receive zero credit.
- 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 packager USERID where USERID is your CCC user id.

Homework Context

This homework is concerned with how memory is allocated to store information. In Java (and most programming languages) you have two possible choices:

- Contiguous memory that stores an array of values
- Fragmented nodes with pointers to other nodes

Homework1 was concerned exclusively with arrays. Now we are ready to explore more dynamic structures.

Fragmented	*Linked List of values*  *Linked List of (k,v) pairs* 

Version: Apr 6th 2020 9:00 AM

Getting Started

Copy the following files into your **`USERID.hw2`** package:

- `algs.hw2.TaleOfTwoCitiesExtractor`
- `algs.hw2.Q1`
- `algs.hw2.Q2`
- `algs.hw2.WordList`
- `algs.hw2.WordSymbolTable`
- `algs.hw2.Q3`
- `algs.hw2.Selection`

Because I cannot be entirely sure **how** you have checked out the code for HW2, I will post a video that tells you whether you have completed the above steps properly.

In particular, once you have copied `TaleOfTwoCitiesExtractor`, you should execute it within your project. It should run and tell you that it is "OK." If this doesn't happen, contact me or a TA/SA for help.

Version: Apr 6th 2020 9:00 AM

Q1. Working With Linked Lists [40 pts.]

Often you will need to construct a **set** of items and support search capability (i.e., to determine if a target value is in the set of not). A mathematical set contains unique items, so no element can be duplicated in the set. Use the following class, which you will copy into your **USERID.hw2** area and modify to work properly. It exists to maintain a set of unique Strings. You can add to the set, return its size, check whether the set contains a given element, or remove an element from the set.

```
public class WordList {

  class Node {
    String     word;
    Node       next;

    Node(String w) {
      this.word = w;
    }
  }

  /**
   * If the given element doesn't exist in the set then update
   * the set and return true; otherwise return false. This means that
   * adding a duplicate element to a set must return false.
   * @param elt      element to be added.
   */
  boolean add(String elt) { … }

  /**
   * Returns true if the element exists within the set.
   * @param elt      target element to be searched.
   */
  boolean contains(String elt) { … }

  /** Returns the number of elements in the set. */
  public int size() { … }

  /**
   * Returns true if the given element was in the set (and was removed) or
   * false if the given element did not belong to the set.
   * @param elt      element to be removed.
   */
  public boolean remove (String elt) { … }

  /** For debugging, return comma-separated string of elements in the set. */
  public String elements() { … }
}
```

Your task for question 1 is to use this data type (as implemented using Linked Lists) to answer three questions about the Tale Of Two Cities by Charles Dickens. There are 45 chapters in the book, which I have extracted into separate files. I will admit that the transcription is quite awkward. For example, everything has been converted to lower case, and all punctuation marks have been removed. Some words are subdivided improperly, but this is what we have to work with!

Version: Apr 6th 2020 9:00 AM

I am providing a helper class, `TaleOfTwoCitiesExtractor`, which you can use to extract each of the words, in order, from a given chapter. I recommend that you copy this class into your **`USERID.hw2`** package just in case you have to edit it to locate the chapters from the Tale of Two Cities

Despite there being 45 chapters, your program should be able to answer all three questions in less than 1-2 minutes. For this entire question, you can rely on `TaleOfTwoCitiesExtractor` to return each of the words one at a time. Simply accept that this class is responsible for determining the words one at a time.

Q1.1 [10 pts.] Which chapter contains the most number of words in total, out of the 45 chapters?

Q1.2 [10 pts.] Which chapter (of the 45) contains the fewest number of unique words? Be sure to report how many unique words occur in the chapter you identify.

Q1.3 [10 pts.] What is the total number of unique words in the entire book (all 45 chapters)?

Q1.4 [10 pts.] Which two distinct chapters (of the 45) share the most words in common? You should be able to use `WordList` to answer this question as well.

Version: Apr 6th 2020 9:00 AM

Q2. Working with Symbol Tables [30 pts]

A common scenario is to implement a symbol table of keys and accumulate counts of repetitions for these keys. Copy this class into your **`USERID.hw2`** area and modify to work properly.

```
public class WordSymbolTable {

  class Node {
    String     word;
    int        count;
    Node       next;

    Node(String w, int count) {
      this.word = w;
      this.count = count;
    }
  }

 /** Increase the count for given key. Note: this might need to add the word in the first
   * place. Returns TRUE if the word was newly added, otherwise FALSE
   *
   * @param key       key whose count has increased by 1. */
  public boolean increment(String key) { … }

  /** Decrease the count for given key. Note: this might need to remove the key once the
   * count reaches zero. Returns TRUE if the word was removed, otherwise FALSE
   *
   * @param key       key whose count is to decrease by 1. */
  public boolean decrement(String key) { … }

  /** Returns the number of keys in the symbol table. */
  public int size() { … }

  /** Return the accumulated counts of all keys in the word table. */
  public int totalCounts() { … }

  /** Returns true if the given key was in the word table (and was removed) or
   * false if the given key did not belong to the word table.
   * @param key      key to be removed. */
  public boolean remove (String key) { … }

  /** Returns count of the key, if it exists in the word table; 0 otherwise. */
  int count(String key);

  /** Returns true if the key exists in the word table; false otherwise. */
  boolean contains(String key);

  /** Returns key whose repetition count is equal to the maximum in the Symbol table
   * Note that there may be multiple words that have the maximum count, so this method
   * only needs to return one of them. */
  public String mostFrequent() { … }

  /** Output semicolon-separated (k,v) pairs in collection to the console. */
  String pairs();
}
```

You can continue to use the helper `TaleOfTwoCitiesExtractor` class for this question.

Q2.1 [10 pts.] What is the most frequently used word in The Tale Of Two Cities? How many times is it used?

Q2.2 [10 pts.] What are the top-ten most frequently used words in The Tale Of Two Cities? How many times is it used?

Q2.3 [10 pts.] How many words occur exactly once in The Tale Of Two Cities?

Version: Apr 6th 2020 9:00 AM

Q3. Mathematical Analysis [20 pts.]

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.Q3` class. Copy this class into `USERID.hw2.Q3` 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`.

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

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

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

  return total;
}
```

For this assignment, develop the recurrence relationship for S(N) and compute its closed-form. Then modify the `Q3` class to output a modified table that shows the computed counts.

Question 3.1 (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 3.2 (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.

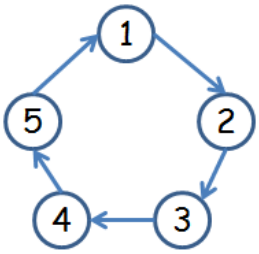
Bonus Question 3.3 (1pt.)

Can you 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.

Version: Apr 6th 2020 9:00 AM

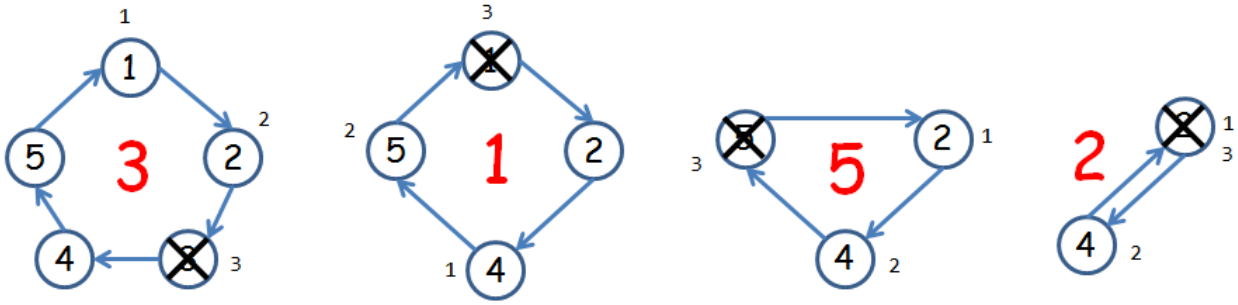
Q4. Working with Linked Lists [10 pts.]

You are in a group of N people that visit a casino and everyone wins a little bit of money. You suggest it would be better if just one person collected all the winnings and you propose the following strategy.



1. You tell everyone to stand in a circle (yourself included).
2. You ask for a volunteer to be starting person #1. In clockwise fashion, everyone in the circle is assigned an increasing number from 1 to *N*.
3. You pick a number $0 < k < N$.

Beginning with the starting person, count *k* people clockwise. The k^{th} person leaves the circle which shrinks by one in size; starting with the next person, again count *k* people clockwise and that person leaves the circle. The last one remaining collects all winnings. Visualize the situation for N=5 people with k=3 as follows. The large number in red is the person eliminated in that round (there are always N-1 rounds).



Start with person #1 and count three clockwise, so the first person eliminated is #3. Continuing clockwise around, the third person (starting at person #4) is person #1, who is then eliminated. In the third round, starting with person #2, the third remaining person is #5, who is eliminated. Finally, with two people left (#2 and #4) the last person to be eliminated is #2. The final person standing is #4.

Copy `algs.hw2.Selection` into the `USERID.hw2` package and modify it to produce the correct output. Modify the `countoff()` method to produce a `FixedCapacityQueue<Integer>` reflecting the order in which people are eliminated. The final item in the queue is the person who collects all winnings.

Your implementation MUST create a Linked List using the provided Node class. Unlike in these diagrams above, **your linked list should not have the final Node connect back to the first Node**. It must remain a true linear linked list whose final Node (the person 5 above in this example) has **`null`** as next reference.

Question 4.1 (10 pt.) When your program executes, the following two cases should work as follows

For `new Selection(5, 3).countOff()`, the resulting queue should be [3 1 5 2 4] where the last person who receives all winnings is person #4.

For `new Selection(17, 7).countOff()`, the resulting queue should be [7 14 4 12 3 13 6 17 11 9 8 10 16 5 15 1 2] where the last person who receives all winnings is person #2.

Bonus Question 4.2 (1 pt.)

If you generate the first few values of the last remaining person method, you may find some interesting relationships. Let *T(n,k)* be the last person standing from an initial circle of N people where you eliminate every kth person until only one remains. Place the results in a triangle as follows (which I call the Josephus Triangle – see www.josephustriangle.org).

```
                          Diagonal 1
                          / Diagonal 2
                         / / Diagonal 3
Row 1                   1 / / Diagonal 4
Row 2                  2 1 / /
Row 3                 3 3 2 /
Row 4                4 1 1 2
Row 5               5 3 4 1 2
Row 6              6 5 1 5 1 4
Row 7             7 7 4 2 6 3 5
```

If you know *T(n,k)* for row n and diagonal k, can you come up with a formula for *T(n+1, k)*?

Bonus Question 4.3 (1 pt.)

There is a clear pattern in diagonal 1. Can you identify the pattern that appears in Diagonal 2? And if so, can you come up with a formula that computes *T(n,2)* for any n >= 2?

Bonus Question 4.4 (1 pt.)

If you consider the right diagonal heading south east, (i.e., 1, 1, 2, 2, 2, 4, 5, …), these numbers form an unusual sequence (https://oeis.org/A007495) that seems to defy any attempt at coming up with a closed formula. There are two cases that I find interesting, however, and I have been working on them for weeks.

(a) In mathematics, Prime Numbers are studied EXHAUSTIVELY! One conjecture is that there are infinitely many twin primes, such that both *p* and *p+2* are prime numbers. With this concept in mind, I introduce the concept of twin consecutive pairs in this sequence, such that the value in the right diagonal I neighboring rows is the same. The first twin pairs are (1,1), (2,2), (2,2), and the next one occurs at (9,10) when both values are 8. Can you compute the rows for when the next twin pair occurs?

(b) The number 1, in all of my computations, only occurs as the right value in the first two rows. I am not bold enough to assert that it never occurs again. However, consider when the right value in row n is equal to n-1 (such as occurs in row two above). This seems like a fruitful place to search. This occurs ten times in the first 1000 rows. Can you compute the rows for these values?

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**.

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.