Comp 251: Assignment 4

Instructor: Jérôme Waldispühl

Due on December 7th at 11:55:00 PM
Test runs on December 3, 9PM, and December 7, 11:55PM. No submission accepted after December 8th, 11:55PM

General instructions (Read carefully!)

- Your solution must be submitted electronically on MyCourses.
- To some extent, collaborations are allowed. These collaborations should not go as far as sharing code or giving away the answer. You must indicate on your assignments (i.e. as a comment at the beginning of your java source file) the names of the people with whom you collaborated or di aff). If you did not collabora d be able to orally explain y
- This assignment is due on December 3^{rd} at 11h55 ithout penalty until December 7th, 11:55PM and then with 20% penalty on responsibility to guarantee that your assignment is subm
- This assignment will be submitted in two components. The programming component, worth 60 points, will be submitted in the HW4 - Programming submission folder. **Submit a zip file**.The long answer component, worth 40 points, will be submitted in the HW4 - Long Answer submission folder. Make sure you submit in the right folder.
- Multiple submissions are allowed before the deadline. We will only grade the last submitted file. Therefore, we encourage you to submit as early as possible a preliminary version of your solution to avoid any last minute issue.
- Late submissions can be submitted for 24 hours after the deadline, and will receive a flat penalty of 20%. We will not accept any submission more than 24 hours after the deadline. The submission site will be closed, and there will be no exceptions, except medical.
- Violation of any of the rules above may result in penalties or even absence of grading. If anything is unclear, it is up to you to clarify it by asking either directly the course staff during office hours, by email at (`cs251@cs.mcgill.ca`) or on the discussion board on Reddit

(recommended). Please, note that we reserve the right to make specific/targeted announcements affecting/extending these rules in class and/or on the website. It is your responsibility to monitor the course website and MyCourses for announcements.

Programming component

- Add your code only where you are instructed to do so. You can add some helper methods in the classes you are asked to modify. Do not modify the code in any other way and in particular, do not change the methods or constructors that are already given to you, do not import extra code and do not touch the method headers. **Any failure to comply with these rules will result in an automatic 0.**
- The starter code includes a tester class. If your code fails those tests, it means that there is a mistake somewhere. We will grade your code with a more challenging set of examples. We therefore highly encourage you to modify that tester class, expand it and share it with other students on the discussion board. Do not include it in your submission.
- Your code should be properly commented and indented.
- **Do not change or alter the name of the files you must submit**. Files with the wrong name will not be graded. Make sure you are not changing file names by duplicating them. For example, main(2).java will not be graded. Make sure to double-check your zip file.
- In order to receive a grade for the programming part of the assignment, you must submit all components of th ons would be met with a grade 0.

Long Answer com

- The long answer is divided into several components. Please in your solution. Presentation points will be granted to full so sented. A solution is considered correctly presented when it is:

  1. Written with a text editing software with equation formatting like LaTeX. Unformatted equations like (3(x+5)+5)/(3x+4) will not be graded.
  2. If written by hand, scanned with a correctly functioning scanner with sufficient contrast to be easily readable. Pictures of assignments will not be graded.
  3. Clean looking, with no crossed out regions or ink/eraser smearing.

- Be brief and to the point in the solution to the long answer questions. You will be provided with the maximum number of words (not including equations) your solution should contain. You can skip all the information that is already provided in the task description and start immediately with the first step of your solution. Show your work.

1. (30 points) **Saving the Earth from aliens**

In 2246 aliens declared war on humans. To save humanity, we need to disrupt the alien communication network while breaking the minimum possible number of links. You are tasked with writing a set of utility methods to determine the global minimum cut of the network using a randomized algorithm (contraction algorithm).

Here are the technical details about the network: the network is an instance of the class Graph.java. Some examples of what the network could look like are described in the files `network_1.txt`, `network_2.txt`, where the first line is the expected size of the min cut and the following lines describe the edges, i.e. the line "a b" means that there is an edge between node a and node b. To test your code, run: `java GlobalMinCutTester network_1.txt` and `java GlobalMinCutTester network_2.txt`.

Your tasks:

- Implement `Graph.contractEdge()`. When contracting an edge e = (u, v), instead of replacing u and v by single new supernode w as in the course slides, you will merge the node u into v. More specifically, you will:
  - Remove the edge e = (u, v), and remove the node u (use the `removeNode()` method
  - Remov loopin thr `ConcurrentModificationExce` ou try doing it. Thankfully Google and Stack Overflow still ex tiple ways to fix it.
  - For all remaining edges, replace all occurrences of u with v
- Implement `GlobalMinCut.global_min_cut()`, which must return two lists of nodes (characters) corresponding to the two partitions of the cut.
  - For each node v, you will record the list S(v) of nodes that have been contracted into v
  - Initially S(v) = v for each v
  - Run the contraction algorithm. After each contraction of an edge e = (u, v) you must update S(v), i.e. all the nodes that were in the supernode u must now be added to the supernode v, because we just merged u into v
  - Once you have only two nodes u and v left, return S(u) and S(v)
  - For your convenience, the Graph class has several utility methods: `getNbEdges()`, `getNodes()`, `getEdge()`, `getExpectedMinCutSize()` and others.
- Implement `GlobalMinCut.global_min_cut_repeated()`. This method is already almost done. Given a graph, this method should call `global_min_cut()` until it either obtains the minimum cut or exceeds a large number of iterations, in which case

you'll know you did something wrong. More specifically, it has an int parameter maxIterations, we expect the algorithm to have found the min cut before then with high probability, it is used as a sanity check and to avoid infinite loops. `global_min_cut_repeated()` also takes a Random object as a parameter, don't touch it and don't worry about it, we only use it for grading so we can use seeds.

- You need to add a call to `global_min_cut()`
- Since `global_min_cut()` modifies the graph, you need to create copies of the original graph at each iteration and pass the copy. Use the copy constructor `Graph(graph)` provided.

2. (30 points) **Rescuing Anatoly**
The aliens are also interested in multiplication methods, especially the Karatsuba algorithm, which is so fast they fear it could thwart the success of their invasion. They have gone back in time to kidnap Anatoly Alexeyevich Karatsuba to force him to reveal his secrets. Your task is to implement his algorithm to show aliens any human can be replaced with a small piece of code and convince them to return him to his family.

To make a convincing point, you will need to compare the naive and Karatsuba divide-and-conquer methods to multiply two integers `x` and `y`. You will implement a recursive version of both algorithms in `Multiply.java`.

Your tasks:

- Implement th `ze, int x, int y)`
- Implement the Karatsuba algorithm in `kar nt y)`

- Evaluate the number of arithmetic operation of each me efficiency (or cost).
  - The variable `size` is the size of the integers `x` and `y`, and is defined as the number of bits used to encode them (Note: we assume that `x` and `y` have the same size). **We define the size as the number of bits starting from the right that are used in the product.**
  - We define the cost as the number of brute force arithmetic operations of the (addition, subtraction, or multiplication) executed by the algorithm multiplied by the size (in bits) of the integers involved in this operation (Note: We ignore the multiplication by powers of 2 which can be executed using a bit shift. Of course, this is a crude approximation).
  - In particular, for the base case (i.e. when the size of the integers is 1 bit), this cost will be 1 (brute force multiplication of two integers of size 1). In the induction case, the naive method executes 3 arithmetic operations of integer of size m (i.e. cost is $3 \cdot m$), in addition of the number of operations executed by each recursive call to the function. By contrast, the Karatsuba algorithm requires 6 arithmetic operations of size m on the top of the cost of the recursion.

  - Each method (i.e. `naive` and `karatsuba`) will return an integer array `result` that stores the value of the multiplication in the first entry of the array (i.e. `result[0]`), and the cost of this computation in the second entry (i.e. `result[1]`)

- The output of your program will print a list of numbers such that the first number of each row is the size of the integers that have been multiplied, the second number is the cost of the naive method, and the third number the cost of the Karatsuba method.

If you do not submit your two java programs in a single zip file on MyCourses, the aliens will take over the world (as well as university administration), and your assignment will have to be sent to Mars for evaluation, causing a delay of several years in the grading process.

Part 2: Long Answer question The aliens are upset about our resistance! They changed the physic rules of our world to render inefficient our computers. They corrupted the binary counter (See Lecture 20) in such way that flipping a bit at index k costs now 2^k instead of 1. Basic operations are now expensive and our computers became very slow...

3. (20 points) Using the **aggregating method**, show that the amortized cost of the fonction `increment` seen in class is now $O(\log n)$. Give the cost of a sequence of n operations `increment` as a sum with accurate upper and lower bounds for the indices of the sum operator, then solve it.

4. (20 points) Fortunately [illegible] lt back. The determined the smallest i k 2 $<< n$ he binary counter to 0 (i.e. the counter is automatically reset when its value reac ote that resetting all bits to zero has a cost. Use the **accounting m** ese conditions the amortized cost of the operations `incremen` termine the cost of a `reset` operation, indicate the amortized costs of each operation, and prove that the credit never goes negative. Then, conclude.

5. (5 points) Congratulations! The aliens gave up and you saved the planet. *Starfleet* has decided to award you 5 bonus points and wish you a well deserved end-of-the-year break![1]

[1] After the final examination on December 16^{th} though...