Comp 251: Assignment 4

Answers must be submitted online by December 6th (11:59 pm), 2021.

General instructions (Read carefully!)

- **Important:** All of the work you submit must be done by only you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. For Comp251, we will use software that compares programs for evidence of similar code. This software is very effective and it is able to identify similarities in the code even if you change the name of your variables and the position of your functions. The time that you would spend modifying your code to avoid being caught would be better invested in creating an original solution.

  Please don't copy. We want you to succeed and are here to help. Here are a couple of general guidelines to help you avoid plagiarism:

  Never look at another assignment solution, whether it is on paper or on the computer screen. Never share your assignment solution with another student. This applies to all drafts of a solution and to incomplete solutions. If you find code on the web, or get code from a private tutor, that solves part or all of an assignment, do not use or submit any part of it! A large percentage of the academic offenses in CS involve students who have never met, and who just happened to find the same solution online, or work with the same tutor. If you find a solution, someone else will too. The easiest way to avoid plagiarism is to only discuss a piece of work with the Comp251 TAs, the CS Help Centre TAs, or the COMP 251 instructors.

- Your solution must be submitted electronically on codePost. Here is a short **tutorial** to help you understand how the platform works. You should already be on the platform since this is the second assignment.

- 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 discussed your assignments (including members of the course staff). If you did not collaborate with anyone, write "No collaborators". If asked, you should be able to orally explain your solution to a member of the course staff.**

- This assignment is due on December 6^{th} at 11:59 pm. It is your responsibility to guarantee that your assignment is submitted on time. We do not cover technical issues or unexpected difficulties you may encounter. Last minute submissions are at your own risk.

- 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.
- In exceptional circumstances, we can grant a small extension of the deadline (e.g. 24h) for medical reasons only. However, such request must be submitted before the deadline, and justified by a medical note from a doctor, which must also be submitted to the McGill administration.
- 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 Ed (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 Ed for announcements.
- The course staff will answer questions about the assignment during office hours or in the online forum. We urge you to ask your questions as early as possible. We cannot guarantee that questions asked less than 24h before the submission deadline will be answered in time. In particular, we will not answer individual emails about the assignment that are sent sent the day of the deadline.

Technical considerations

- You are provided some starter code that you should fill in as requested. Add your code only where you are instructed to do so. You can add some helper methods. 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. The format that you see on the provided code is the only format accepted for programming questions. **Any failure to comply with these rules will result in an automatic 0.**
- Public tests cases are available on codePost. You can run them on your code at any time. If your code fails those tests, it means that there is a mistake somewhere. Even if your code passes those tests, it may still contain some errors. We will grade your code with a more challenging, private set of test cases. We therefore highly encourage you to modify that tester class. You can discuss test cases with fellow students, but do not share files. List all the students you collaborated with on the top of your source code.
- Note that your code will be evaluated with Java 8 on a linux environment (similar to ubuntu). It is your responsibility to make sure your code compiles and runs correctly when executed from command line in this type of environment. Codepost has a limit of 30 seconds
- Your code should be properly commented and indented.
- **Do not change or alter the name of the files you must submit, or the method headers in these files**. 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. Do not add packages or change the import statements.
- Submit your files individually on codePost (no zips)
- **You will automatically get 0 if the files you submitted on codePost do not compile, since you can ensure yourself that they do.**
- **Note that public test cases do not cover every situation and your code may encounter runtime errors when tested on private tests. This is why you need to add your own test cases and compile and run your code from command line on a linux system (see the relevant tutorial on MyCourses).**

Exercise 1 ***Saving the Earth from aliens (50 points)*** 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 for that)
  - Remove edges that had one end equal to u and other equal to v. Heads up: while looping through edges and removing them, you may run into `ConcurrentModificationException` depending on how you try doing it. Thankfully Google and Stack Overflow still exist in 2246 and you can find multiple 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.

Exercise 2 ***Rescuing Anatoly (50 points)*** 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 the naive multiplication algorithm in `naive(int size, int x, int y)`
- Implement the Karatsuba algorithm in `karatsuba(int size, int x, int y)`
- Evaluate the number of arithmetic operation of each method by keeping track of their 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 classes on codepost**, 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.