Due : Dec. 17 (Mon.), 13:00
Late submission is allowed until Dec. 19, 13:00 with deduction of 20%

General Notes

- *Read this homework guideline carefully.* If you do not follow the guidelines, you may receive a 0 regardless of whether your code works or not.
- Do not use any IDEs (Eclipse, IntelliJ IDEA, etc.)
  - We recommend Sublime Text (Linux/Mac/Windows), Atom (Linux /Mac/Windows), Notepad++ (Windows), or TextWrangler (Mac).
  - IDEs often create a "package" of your code, which breaks the autograder.
  - If you know how to fix the package problem, you can use any IDE you want. However, we will not answer any questions related to this problem since we have already recommended a solution.
- Do not change any method or class signatures. If your code changes any class or method names or signatures, you will receive an automatic 0.
- Make sure your code compiles. Non-compiling code will automatically receive a 0. If you have a problem that is causing you to not be able to compile, it may be better to just comment out the incorrect code and return a dummy value (something like null or -1) so the rest can compile.
- Make sure that your code does not print out anything (there should be no System.out.println in your code). You will receive an automatic 0 if your code outputs something to STDOUT during the tests.
- To ensure that your code will be accepted by the autograder, you should submit your code on YSCEC, download it again, recompile it and check the provided test suite. This way, you know that the file you are submitting is the correct one.

Overview

This assignment consists of two parts, implementing an algorithm to answer a substring query and determining whether or not a graph is bipartite.

1 Substring Search

You will implement a structure that *efficiently* counts a pattern occurrence over a string.

This structure has the following operation:

`countPattern(`p`)` counts the number of occurrences of the pattern p from the string.

The string will be given through the constructor of the class.

For instance, given the string "mississippi", the results of `countPattern` calls according to each pattern are:

Pattern	Result	Note
`i`	4	`mississippi`
`issi`	2	`mississippi` (They are overlapped.)
`ip`	1	`mississippi`
`sss`	0	

The characters of string as well as patterns are in ASCII range (from 1 to 127; 0 is the null byte). For each testcase, you will be given one string that might be very long. There will be a lot of pattern to be tested and you will count the occurrences of a pattern per one `countPattern` call.

2 Bipartite Graph

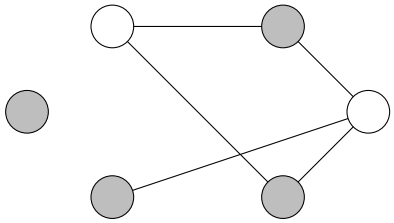
An undirected graph $G = (V, E)$ is *bipartite* if there is a partition (L, R) of V such that

- $L \subseteq V$, $R \subseteq V$, $L \cup R = V$, $L \cap R = \emptyset$ (partition),
- Every edge $e \in E$ is $\{l, r\}$, where $l \in L$, $r \in R$.

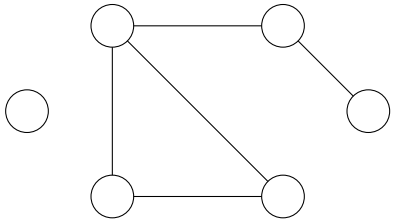
You will design an algorithm that determines whether a graph is bipartite or not. You will implement the following operations:

`insertVertex(`v`)` adds a vertex v into the graph,
`insertEdge(`u`, `v`)` adds an edge whose endpoints are u and v into the graph,
`removeVertex(`v`)` removes a vertex v and its every connected edges from the graph,
`removeEdge(`u`, `v`)` removes an edge whose endpoints are u and v from the graph,
`isBipartite()` determines whether the graph is bipartite or not.

For example, this graph is bipartite (note that the vertices are partitioned into two groups):



but this is not:



You are free to implement any algorithm that you wish, but you must code it yourself (for example, you cannot just call Java's sort on your array or copy a solution from the internet). We will only be testing that your code produces a correct result and terminates within a reasonable amount of time.

General Directions

- Write your name and student ID number in the comment at the top of the all source files you will hand in.
- Implement all of the required functions in the appropriate class.
- You should not import anything that is not already included in the file.
- Pay careful attention to the required return types and edge cases.
- If you have a question, first check Q&A board on YSCEC course page and write an open question. If it contains private things, send us an e-mail. We will not answer to questions about Java language itself.

Grading Policy

You will receive a 0 if you did not follow the guidelines or if your code does not compile. After the due date, there is a two day grace period during which you can still submit your homework. Homework submitted during this time will receive a 20% deduction.

Submission Procedure

To submit your homework, simply drag and drop each files individually into the attachment box on the YSCEC assignment page. You *must not* zip them or place them in any folder. For this assignment, you should submit only the following files:

- Substr.java,
- Bipartite.java,
- (student_id).txt (not including the parentheses).

Inside of the (student_id).txt file, you should include the following text with your name at the bottom.

In completing this assignment, I pledge that I have not given nor received any unauthorized assistance.
[WRITE YOUR NAME IN ENGLISH]

If this file is missing, you will get a 0 on the assignment. It should be named *exactly* your student id, with no other text. For example, *2018123456.txt* is correct while something like *2018123456_hw1.txt* will receive a 0.

Compiling

You can compile your Java code using the following command.:

`% javac -encoding ISO-8859-1 -cp ".:junit-4.10.jar" [Java source].java` (on Linux/Mac)

`> javac -encoding ISO-8859-1 -cp ".;junit-4.10.jar" [Java source].java` (on Windows)

You can also test your implementation using the following command.:

`% java -cp ".:junit-4.10.jar" [TestRunner]` (on Linux/Mac)

`> java -cp ".;junit-4.10.jar" [TestRunner]` (on Windows)

Testing

We have provided a small test suite for you to check your code. You can test your code by compiling and running the tester program. If any of the test cases fail, you will be notified.

Note that the test suite we will use to grade your code will be much more rigorous than the one provided here (and not necessarily a superset of the provided tests). You should consider making your own test cases to check your code more thoroughly.