Implementing Dijkstra's shortest-path algoritm

1. Preamble

In this assignment, you use your solution to A5 (or ours) in implementing the shortest-path algorithm. Your solution to A7 (or ours) will then be used in the final project A8. We have cut this assignment to the minimum while still giving you the invaluable experience of implementing the algorithm. Our solution is 40 lines long, counting the method specification and comments and blank lines. Further, we give you a JUnit testing class. If your method passes all the tests, it should be correct.

Look at the grading rubric in the pinned Piazza post Assignment A8; **20 points ride on using good style!**

Keep track of how much time you spend on A7; we will ask for it upon submission.

Read this whole document and the pinned Piazza post Assignment A8 before beginning to code.

Doing this assignment before prelim 2 will help you fully understand the algorithm.

Collaboration policy and academic integrity

You may do this assignment with one other person. Both members of the group should get on the CMS and do what is required to form a group *well before the assignment due date*. Both must do something to form the group: one proposes, the other accepts.

People in a group must *work together*. It is against the rules for one person to do some programming on this assignment without the other person sitting nearby and helping. Take turns "driving" —using the keyboard and mouse.

With the exception of your CMS-registered group partner, you may not look at anyone else's code, in any form, or show your code to anyone else (except the course staff), in any form. You may not show or give your code to another student in the class.

Getting help

If you don't know where to start, if you don't understand testing, if you are lost, etc., please SEE SOMEONE IMMEDIATELY —an instructor, a TA, a consultant. Do not wait. A little in-person help can do wonders.

2. The release code

Download file a7release.zip from Piazza note Assignment A7 and unzip it. It has two directories, `src` and `data`.

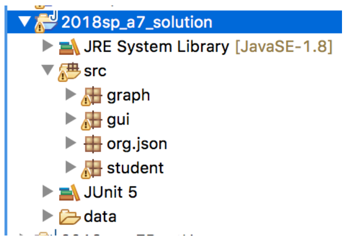
Start a new Eclipse project called, say, A7 (you can name it whatever you want).

Drag directory `src` over the project in the Package Explorer pane. You will be asked whether directory src should be overwritten. Yes, it should be.

Drag directory `data` over the project in the Package Explorer pane. File PastTester.java will now show errors. That is because JUnit 5 is not on the build path.

Add JUnit 5 to the build path. One way to do this is to create a new JUnit testing class (Menu item File -> New -> Junit Test Case), then delete it.

The project should be similar to the diagram on the right (your JRE System Libraries may be different). The project contains our solution Heap.java. Replace it by yours if you want.



3. Running the program

Class graph.Main contains method main. To run the program, open class Main in the Eclipse editor, select class Main in the Package Explorer pane, and choose menu item Run -> Run. A GUI will open with a graph. You can get a new randomly-generated graph using menu item Graph -> New Random Map. You can drag the nodes of the graph around to make it easier to see a part of it. The text at the bottom of the window tells you what to do: Click a start node, click an end node, and you will see in red the shortest from start to end (once you complete the assignment).

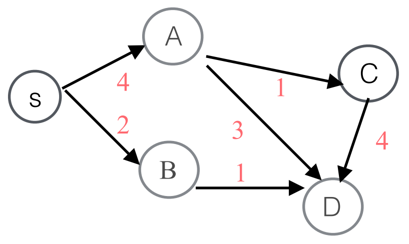
4. What to do for this assignment

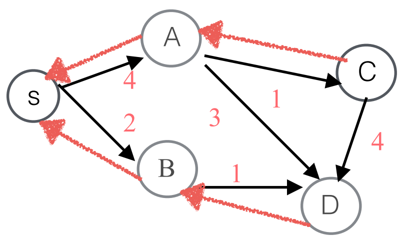
Your job is to implement method Paths.minPath. It is marked “TODO”. We give you everything else, including our Heap.java. The comment in the body of minPath directs you to the course Piazza, pinned note Assignment A7. It contains important directions and guidelines. You are responsible for it. File Paths.java is the only file you must change and submit.

5. Backpointers

The basic shortest-path algorithm calculates the length of the shortest path from a start node to an end node. Here, we show, in the context of A7, how to extend the algorithm to also calculate the shortest path itself. Often in programming, we write a basic algorithm and then extend it to produce more information. It’s a standard practice/technique.

It is difficult to maintain the shortest path from a start node S to every other node. For example, look at the diagram to the left below and ask yourself: How, in start node S, would you store the shortest paths from the start to all nodes? You would need to store *four* paths, from S to A, to B, to C, and to D. If the graph had 1,000 nodes, you would be storing information for 1,000 paths in S! There *must* be a better way.





We do something else. Look at the diagram to the right. The shortest path from S to D is (S, B, D). Therefore, in node D, store the *back-pointer* on this path, i.e. the *previous* node on this path: B. We show it with a squiggly red arrow. Similarly, the shortest path from S to B is (S, B), so node B contains a *back-pointer* to S.

As one more example, the shortest path from S to C is (S, A, C), so node C contains *back-pointer* A, A contains *back-pointer* S, and S contains null as its back-pointer.

That’s it! With only one extra piece of info in each node, a *back-pointer*, we can store all the information needed to give us the path from S to any node, but in reverse. To find the shortest path from S to some node n, use the back-pointers beginning in node n to construct the path. That takes time proportional to the length of the path. Not bad!

6. Making your code readable and efficient —20 points.

Your code will be tested for correctness by our grading program. But we will also grade your submission for readability and efficiency, for some of the issues that we have been talking about throughout this course. Specifically, you will lose points for the items listed the Piazza post @1437 (get to it from the pinned Assignment A7 note). On a positive note, being aware of these items as you program can help you manage the implementation more efficiently and successfully because your code will stay simple and readable.

7. Read this list carefully

1. Implement method MinPath.minPath. It is marked with “// TODO …”. It ***must*** be an implementation of the algorithm given in Piazza note @1436. It is similar to the one on the lecture slide of lecture 21 that is titled “Final algorithm”. The algorithm should be refined to meet the specification and environment in which it is being implemented. See below for more info.
2. The final algorithm in the lecture slides stops when shortest paths from node *start* to *all* nodes have been determined. However, your algorithm should stop as soon as the shortest path from node *start* to node *end* has been determined; once that is known, the method must *not* continue to calculate shortest paths. This must be done by putting an appropriate test near the beginning of the main loop body and returning the result if the test is met. If you do not do this, you get a 15-point deduction.

3. Read carefully the grading guidelines given in Piazza post @1437.
4. We have provided function Paths.makePath, which builds the path from the back-pointers. Call it to build the shortest path once the end node is detected. Study it to see how the path is constructed.
5. Debugging/testing. When testing/debugging, you will want some small maps to work with. For these, try seeds: 7, 16, 1, 6, 19, 18. You can also change constant Graph.GraphGeneration.MAX_NODES to a small number. (GraphGeneration is a static class within class Graph.)

When testing/debugging, you may want to print out the frontier at each iteration. To do that, write a method in your class Heap to print it out in some suitable form.

6. You can use the GUI to eyeball how your program is doing. However, we have provided complete test cases in JUnit testing class PathsTester. If your method Paths.minPath passes those tests, you can consider the method to be correct.

What to do submit

In class Paths, in the comment at the top, put the hours hh and minutes mm that you spent on this assignment. Please change the hh to an integer and the mm to an integer and change nothing else on that line. Do other things and we may have to deal with his manually, which takes time.

Write a few lines about what you thought about this assignment. Submit on the CMS (only) file Paths.java. We know that your function minPath uses class Heap, but we assume you have not changed Heap's behavior by changing its public methods. We will use *our* correct Heap.java in testing your function.