A common comment was ".equals to compare vertices", meaning that you used `.equals()` to compare vertices. But should you? Remember the difference between `==` and `.equals()` and why `.equals()` is used for `String` — `==` compares "what's in the box", which, for objects, is the address of the object in memory while `.equals()` invokes a method to do the comparison. `==` provides a definition of "equals" that is "same object" i.e. the same memory location. `.equals()` allows for a definition of "equals" that is "equivalent" — these two objects are considered to be indistinguishable. In the case of `String`, we usually mean "equivalent" — two strings are the same if they have the same sequence of characters, immaterial of the particular object. In the case of vertices, it's really the object that matters — if you think of drawing the graph on the page, two different circles are two different vertices even if they have the same label and are connected to the same other vertices. `==` is thus the most appropriate comparison in this case, though by default `.equals()` does the same comparison as `==` so it works out this time.

What does it mean in `run()` if the vertex removed from the priority queue has a distance of infinity? Do you need to handle this possibility for correctness? Can you improve efficiency?