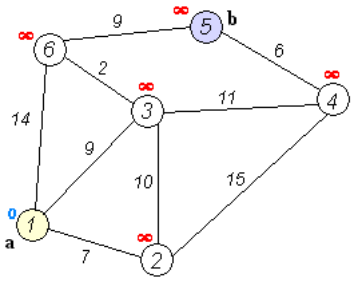
Assignment 5 Part 2

For this assignment you'll finally implement Dijkstra's algorithm for finding the shortest path between nodes in a graph.



In the above graph the shortest path from node a (1) to node b (5) costs 20. It is achieved by traveling from 1 → 3 (costing 9), then from 3 → 6 (costing an additional 2), and finally from 6 → 5 (costing an additional 9). Your program will output the final answer, 20 in this case, and the best possible path from the starting node to every other node in the graph (see sample output below).

Using your `Graph` and your `BetterPriorityQueue` from part 1, implement Dijkstra's algorithm. Fill in the code in the `dijkstra()` function in `Dijkstra.cpp`

Consider adding more tests! Seriously, it's a good idea.

Add a comment at the top of `dijkstra()` that explains the Big-O time-complexity of the algorithm. Be careful about your answer. It isn't straightforward as it depends on how you implemented the Dijkstra algorithm, but also it depends on your `Graph` and `BetterPriorityQueue` implementations!

Sample Output:

```
user@machine$ ./dijkstra
1 | [(1:0)->(2:0) w:7], [(1:0)->(3:0) w:9], [(1:0)->(6:0) w:14]
2 | [(2:0)->(1:0) w:7], [(2:0)->(3:0) w:10], [(2:0)->(4:0) w:15]
3 | [(3:0)->(1:0) w:9], [(3:0)->(2:0) w:10], [(3:0)->(6:0) w:2], [(3:0)->(4:0)
w:11]
4 | [(4:0)->(2:0) w:15], [(4:0)->(3:0) w:11], [(4:0)->(5:0) w:6]
5 | [(5:0)->(6:0) w:9], [(5:0)->(4:0) w:6]
6 | [(6:0)->(1:0) w:14], [(6:0)->(3:0) w:2], [(6:0)->(5:0) w:9]

(1: 0)
(2: 7)
(3: 9)
(6: 11)
(4: 20)
(5: 20)
ans: 20
```

When you're done you should consider the place you've arrived at. Could someone in CS1 do this? NOT A CHANCE! They don't have access to this PDF.

Submission: Your program will be submitted using canvas and git + github.

1) You should already have a private "HW5" repo with `fmresearchnovak` as a collaborator.

2) Make a new commit to the repo before the submission deadline for part 2.

3) On canvas you should upload a link to this github repository
`https://github.com/fmresearchnovak/HW5.git`

I will grade the final commit made before the due-date. This means that you can actually submit on canvas at any time before the deadline, and still make changes (new commits) to the code.