CSCI 3104 Fall 2021
Instructor: Profs. Grochow and Waggoner

Quiz- Standard 3

Due Date .... TODO
Name .... **John Blackburn**
Student ID **jobl2177**

Contents

1 Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You **may not collaborate with other students**. **Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Discord, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

Problem 1.

- My submission is in my own words and reflects my understanding of the material.
- I have not collaborated with any other person.
- I have not posted to external services including, but not limited to Chegg, Discord, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

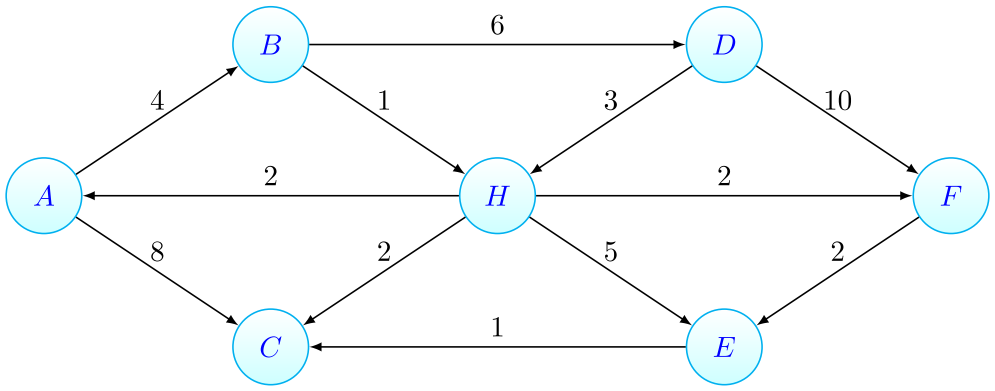
Agreed (John Blackburn). □

3 Standard Dijkstra

3.1 Problem 2

Problem 2. Consider the weighted graph $G(V, E, w)$ pictured below. Work through Dijkstra's algorithm on the following graph, using the source vertex A.

- Clearly include the contents of the priority queue, as well as the distance from A to each vertex at each iteration.
- If you use a table to store the distances, clearly label the keys according to the vertex names rather than numeric indices (i.e., `dist['B']` is more descriptive than `dist['1']`).
- You do **not** need to draw the graph at each iteration, though you are welcome to do so. [This may be helpful scratch work, which you do not need to include.]



Answer. Our source node is A. we begin by setting the distances for each node other than A to ∞. We also set A as processed.

Next we examine A's neighbors, updating their found distances and placing them in the priority queue. The distance table is now:

$Q = [(B, 4), (C, 8)]$

We now have B first in our queue with a distance of 4.

The found distances from A are now:

$(B, 4), (C, 8), (D, \infty), (E, \infty), (F, \infty), (H, \infty)$

Our next iteration brings us to vertex B, we mark it as processed, and begin to examine B's unprocessed neighbors. Them being H and D. Now we update our queue to add these new neighbors. H will go first because it has the lowest distance from A.

$Q = [(H, 5), (C, 8), (D, 10)]$

This brings our distances to:

$(B, 4), (C, 8), (D, 10), (E, \infty), (F, \infty), (H, 5)$

Next we remove H from the queue mark it as processed and examine the unprocessed neighbors. We examine C and find that we can decrease the distance to 7 due to a new shorter path being found. Then we examine F and mark it's distance as 7. And finally we examine E, and mark it's distance to 10. We then rework the queue to hold the new vertex's. Our new queue is:

$Q = [(F, 7), (C, 7), (D, 10), (E, 10)]$

Our new distances are now:

$(B, 4), (C, 7), (D, 10), (E, 10), (F, 7), (H, 5)$

Our next iteration brings us to vertex F. we process F. We visit all the unprocessed neighbors of F, them being only E. We update the distance of E to be 9 because a new path has less distance than the previously found. Our new queue is:

$Q = [(C, 7), (E, 9), (D, 10)]$

Distances:

$(B, 4), (C, 7), (D, 10), (E, 9), (F, 7), (H, 5)$

Our next iteration is on vertex C. We process C. We check all unprocessed neighbors, and we see that C has no neighbors. SO we move to the next iteration. Queue:

$Q = [(E, 9), (D, 10)]$

Distances:

$(B, 4), (C, 7), (D, 10), (E, 9), (F, 7), (H, 5)$

Our next iteration is on vertex E. we mark it as processed. We check all unprocessed neighbors, and find we don't have any. We move on. Queue:

$Q = [(D, 10)]$

Distances:

$(B, 4), (C, 7), (D, 10), (E, 9), (F, 7), (H, 5)$

Our next iteration is on vertex D. we mark it as processed. check unprocessed neighbors, we find all it's neighbors have been processed. We now longer have any unprocessed vertices and nothing left in our queue. Therefore we have found all minimum distances to each node from vertex$A$. Final Distances:

$(B, 4), (C, 7), (D, 10), (E, 9), (F, 7), (H, 5)$ □