Name: Reid Roberts

Algorithm Choice: Dijkstra's Algorithm

1. Dijkstra's algorithm starts from a single source vertex and finds the shortest paths to all other vertices in a graph. It can be thought of as a "wave" that starts from the source vertex and expands outward by choosing to extend the path that has the current smallest total distance from A. Once we know the shortest path to a vertex u, we can use that vertex to see if its neighbors can then be improved with a shorter distance.

   Key Constraints: all edge weights must be nonnegative

   Source: ⍰ Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 3rd ed., MIT Press.

   ⍰

2. Pseudocode for Dijkstra's algorithm

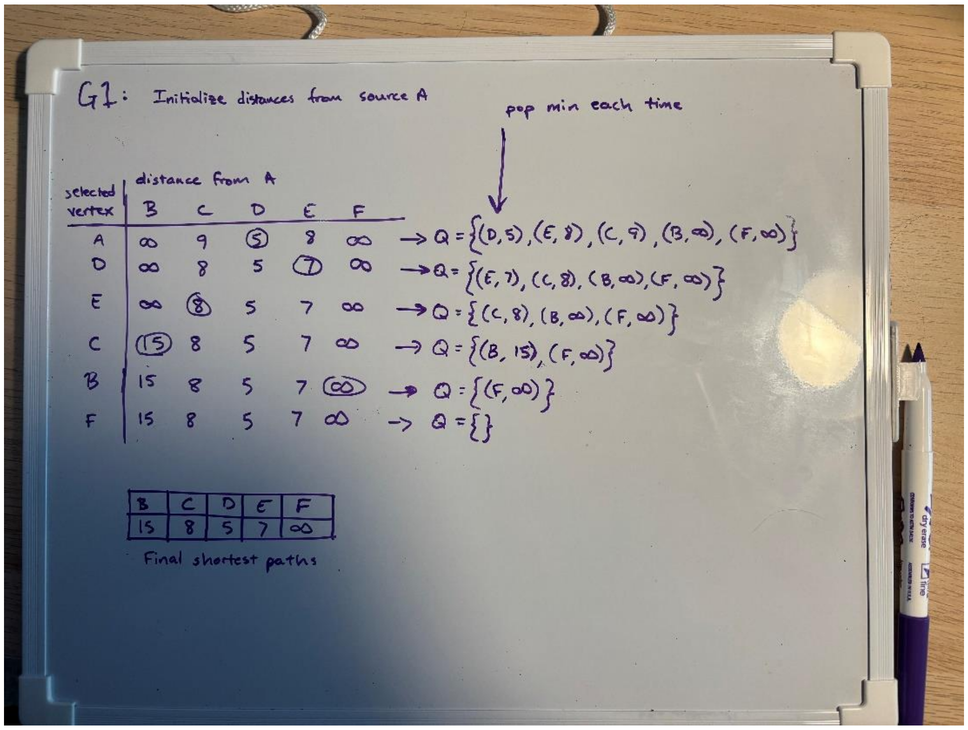
   DIJKSTRA(Graph, source):

```
    INITIALIZE-SINGLE-SOURCE (Graph, s)
    S = set()      # initialize a set to store vertices whose shortest-path distances are finalized
    Q = []         # min-heap priority queue to evaluate the minimum distance vertex next
    For each vertex u in Graph:
        INSERT u into min-priority queue Q
    while Q is not empty:
        u = EXTRACT-MIN(Q)    #evaluate min element from Q
        S = S ∪ {u}           #add finalized vertex to set S
        for each neighbor v of u:
                RELAX(u, v, w)  #compare path distances
                if distance from source to v changes:
                        DECREASE-KEY to update min-priority queue
```

   Source: ⍰ Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 3rd ed., MIT Press.

Name: Reid Roberts

3. G1: http://csc.columbusstate.edu/carroll/6109/shortestPaths/digraph_weighted_cyclesA_G.svg



G1: Initialize distances from source A

pop min each time

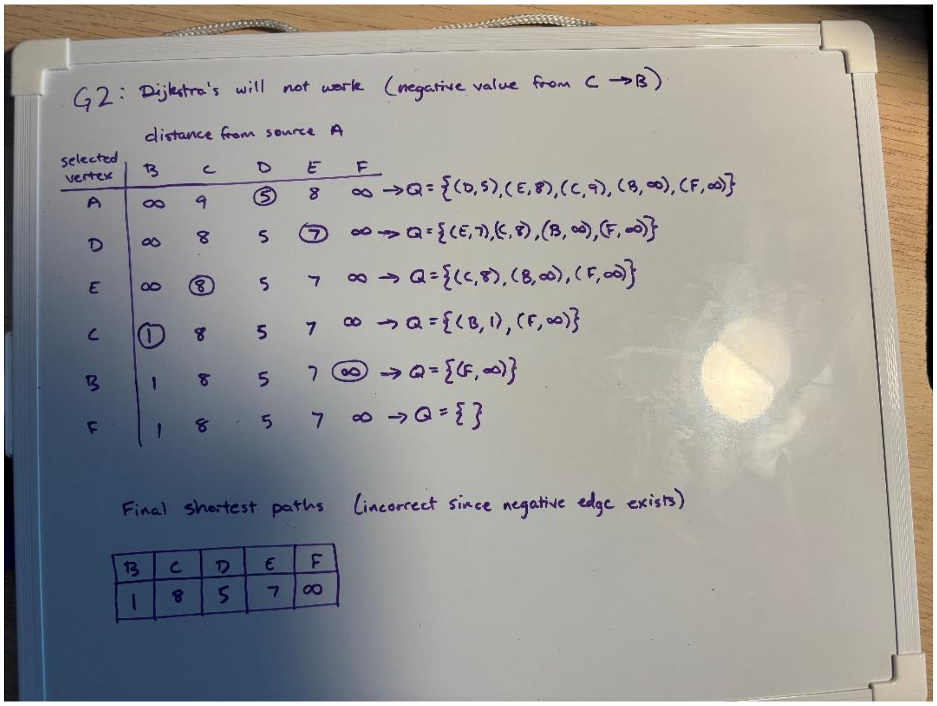
selected vertex	distance from A B	C	D	E	F	
A	∞	9	$\textcircled{5}$	8	∞	→ Q = {(D,5), (E,8), (C,9), (B,∞), (F,∞)}
D	∞	8	5	$\textcircled{7}$	∞	→ Q = {(E,7), (C,8), (B,∞), (F,∞)}
E	∞	$\textcircled{8}$	5	7	∞	→ Q = {(C,8), (B,∞), (F,∞)}
C	$\textcircled{15}$	8	5	7	∞	→ Q = {(B, 15), (F,∞)}
B	15	8	5	7	$\textcircled{∞}$	→ Q = {(F,∞)}
F	15	8	5	7	∞	→ Q = {}

B	C	D	E	F
15	8	5	7	∞

Final shortest paths

G2: http://csc.columbusstate.edu/carroll/6109/shortestPaths/digraph_weighted_negcyclesA_G.svg

This graph will not work with Dijkstra's algorithm because there is a negative weight value from vertex C to B



G2: Dijkstra's will not work (negative value from C → B)

distance from source A

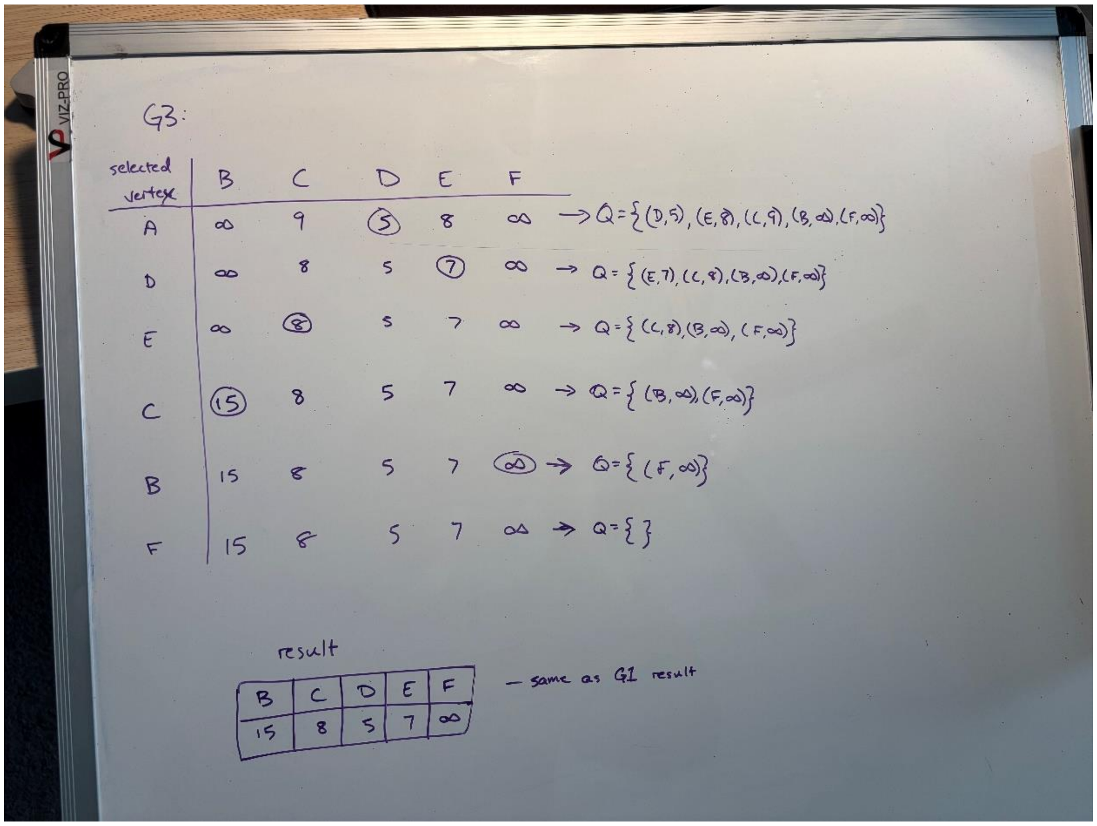
selected vertex	B	C	D	E	F	
A	∞	9	$\textcircled{5}$	8	∞	→ Q = {(D,5), (E,8), (C,9), (B,∞), (F,∞)}
D	∞	8	5	$\textcircled{7}$	∞	→ Q = {(E,7), (C,8), (B,∞), (F,∞)}
E	∞	$\textcircled{8}$	5	7	∞	→ Q = {(C,8), (B,∞), (F,∞)}
C	$\textcircled{1}$	8	5	7	∞	→ Q = {(B,1), (F,∞)}
B	1	8	5	7	$\textcircled{∞}$	→ Q = {(F,∞)}
F	1	8	5	7	∞	→ Q = {}

Final shortest paths (incorrect since negative edge exists)

B	C	D	E	F
1	8	5	7	∞

Name: Reid Roberts

G3: http://csc.columbusstate.edu/carroll/6109/shortestPaths/digraph_weighted_nocyclesA_G.svg



G3:

selected vertex	B	C	D	E	F	
A	∞	9	5	8	∞	$\rightarrow Q=\{(D,5),(E,8),(C,9),(B,\infty),(F,\infty)\}$
D	∞	8	5	7	∞	$\rightarrow Q=\{(E,7),(C,8),(B,\infty),(F,\infty)\}$
E	∞	8	5	7	∞	$\rightarrow Q=\{(C,8),(B,\infty),(F,\infty)\}$
C	15	8	5	7	∞	$\rightarrow Q=\{(B,\infty),(F,\infty)\}$
B	15	8	5	7	∞	$\rightarrow Q=\{(F,\infty)\}$
F	15	8	5	7	∞	$\rightarrow Q=\{\}$

result

B	C	D	E	F
15	8	5	7	∞

— same as G1 result

4. The running time of Dijkstra's algorithm depends on the specific implementation of the min-priority queue.
   a) With a simple implementation, the running time is $O(V^2)$:
      1. Each EXTRACT-MIN operation takes O(V) time and must be done |V| times.
      2. INSERT and DECREASE-KEY operations are O(1)
      3. Relaxation happens E times for a total of $O(V^2 + E) = O(V^2)$
   b) With a Binary min-heap, the total running time is O(E lg V)
      1. it costs O(V) time to build the heap.
      2. Each EXTRACT-MIN operation takes O(lg V) time for |V| operations
      3. Each DECREASE-KEY operation take O(lg V) and can get called up to |E| times for O((E + V) lg V) = O(E lg V) in total
   c) With a Fibonacci heap, the total running time is O(V lg V + E)
      1. The amortized cost of calling |V| EXTRACT-MIN operations is O(lg V)
      2. Each DECREASE-KEY call takes O(1) amortized time for up to |E| calls

Source: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 3rd ed., MIT Press.

Name: Reid Roberts