Report

1.

Cities.mincost = Integer.MAX\_VALUE

Cities.pi = null

Start.mincost = 0

minheap <- Cities

do while minheap.size > 0

city = minheap.extractmin()

for each neighbor of city

do if neighbor.mincost > city.mincost + weight

neighbor.pi = city

minheap.changekey(neighbor, city.mincost + weight)

endif

endfor

endwhile

Runtime Analysis-

0 – 1) O(n)

2) O(1)

3) O(n)

4) O(n)

5) O(log(n))

6) O(n)

7 – 8) O(1)

9) O(log(n))

Therefore, runtime is O(n2 \*log(n)), but can be reduced to O(m\*log(n)) where m is the number of edges because lines 4 + 6 will be at most m. N is the number of vertices.

2.

Cities.mincost = Integer.MAX\_VALUE

Cities[0].mincost = 0

minheap <- Cities

existence <- 1

Do while minheap.size > 0

city = minheap.extractMin()

existence[city] = 0

for each neighbor of city

do if neighbor in minheap AND weight < neighbor.mincost

minheap.changekey(neighbor, weight)

endif

endfor

endwhile

0) O(n)

1) O(1)

2 – 4) O(n)

5) O(log(n))

6) O(1)

7) O(n)

8) O(1)

9) O(log(n))

Therefore, runtime is O(n2 \* log(n)) = O(m \* log(n)), because lines 4 + 7 will be at most m since it cannot be larger than the total number of edges where m is the number of edges and n is the number of vertices.

3.

An adjacency list is better from both a time and space complexity perspective. There will be at most O(V + E) space with adjacency lists whereas a matrix will have O(V^2). In addition, we do not care about O(1) query for whether there exists an edge since we are iterating over all neighboring cities of a city.