Assignment 4:

Samer Kakoore | ske19003
Rami Abukhader|Rar19002

1. optimal values of all cities given that the destination is city E

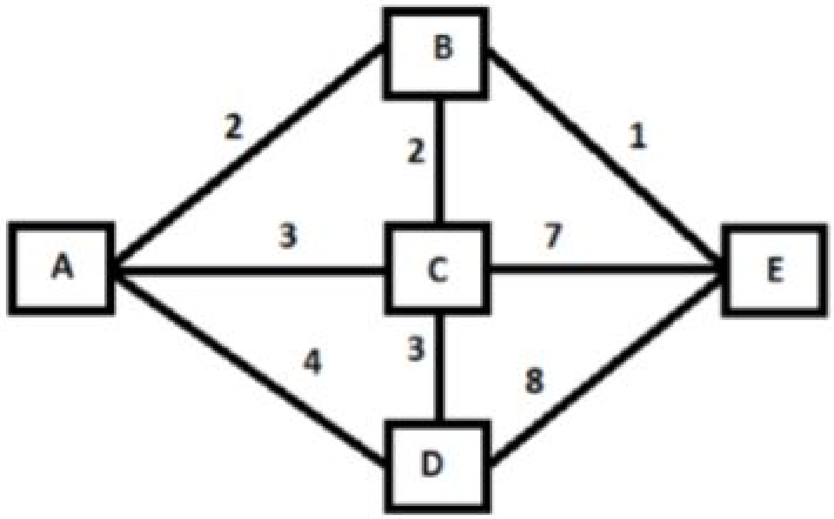

Figure 1

Regarding to figure 1 the destination between every neighbor nodes will be like this:
(A, B)= 2, (A, C)= 3, (A, D)= 4, (B, C)= 2, (B, E)= 1, (C, D)= 3, (C, E)= 7, (D, E)= 8
Regarding to the assignment we assume the destination node is **E**
We assume W(nodes) is the best (smaller) distance value to node E.

W(A)	W(B)	W(C)	W(D)	W(E)
∞	∞	∞	∞	0

Start discover the neighbor of node E:

W(B) = min(W(E) + (B,E) , W(B)) = min(1 , ∞) = 1
W(C) = min(W(E) + (C,E) , W(C)) = min(7 , ∞) = 7
W(D) = min(W(E) + (D,E) , W(D)) = min(8 , ∞) = 8

Start discover the neighbor for last nodes discovered before:

W(A)	W(B)	W(C)	W(D)	W(E)
∞	1	7	8	0

Neighbor B:
W(A) = min(W(B) + (B,A) , W(A)) = min(3 , ∞) = 3
W(C) = min(W(B) + (B,C) , W(C)) = min(3 , 7) = 3

W(A)	W(B)	W(C)	W(D)	W(E)
3	1	3	8	0

Neighbor C:
W(A) = min(W(C) + (C,A) , W(A)) = min(6 , 3) = 3
W(B) = min(W(C) + (C,B) , W(B)) = min(5 , 1) = 1
W(D) = min(W(C) + (C,D) , W(D)) = min(6 , 8) = 6

W(A)	W(B)	W(C)	W(D)	W(E)
3	1	3	6	0

Neighbor D:
W(A) = min(W(D) + (D,A) , W(A)) = min(10 , 3) = 3
W(C) = min(W(D) + (D,C) , W(C)) = min(9 , 3) = 3

Start discover the neighbor for last nodes discovered before:

W(A)	W(B)	W(C)	W(D)	W(E)
3	1	3	6	0

Neighbor A:
W(B) = min(W(A) + (A,B) , W(B)) = min(5 , 1) = 1
W(C) = min(W(A) + (A,C) , W(C)) = min(6 , 3) = 3
W(D) = min(W(A) + (A,D) , W(D)) = min(7 , 6) = 6

Neighbor B:
W(A) = min(W(B) + (B,A) , W(A)) = min(3 , 3) = 3
W(C) = min(W(B) + (B,C) , W(C)) = min(3 , 3) = 3

Neighbor C:
W(A) = min(W(C) + (C,A) , W(A)) = min(6 , 3) = 3
W(B) = min(W(C) + (C,B) , W(B)) = min(5 , 1) = 1
W(D) = min(W(C) + (C,D) , W(D)) = min(6 , 6) = 6
Neighbor D:

$W(A) = \min(W(D) + (D,A)\ ,\ W(A)) = \min(10\ ,\ 3) = 3$
$W(C) = \min(W(D) + (D,C)\ ,\ W(C)) = \min(9\ ,\ 3) = 3$

W(A)	W(B)	W(C)	W(D)	W(E)
3	1	3	6	0

2. Find the shortest path from each city to E.

Bellman-Ford is an algorithm help us to find the shortest path from each city to E in the graph. The algorithm works by overestimating the length of the path from E to all vertices. Then it iteratively relaxes the estimates by finding new shorter paths than the previously overestimated paths. We keep doing this repeatedly for all vertices to be able to guarantee that the end result is optimized.

Start Vertix E.
First iteration:

E	B	C	D	A
0	∞	∞	∞	∞

Second iteration:

E	B	C	D	A
0	1	7	8	∞

.
.
.
.

Final iteration:

E	B	C	D	A
0	1	3	6	3