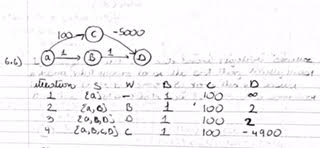
Problem 6.6



In the above case and in accordance to the constraints placed by dijkstra’s algorithm, at each iteration, an assumption is made that all previous iterations are correct and when we add a node to the set and set the distance to dist[v], we must infer this distance to be optimal. If this distance is not optimal, there must be some other path to the vertex V that is of shorter length.

Now in the above case, when we added D to the set after its distance computed was 2, we assumed that the distance was the most minimal distance to D. When we add node C to the set, dist[D through C] < dist[D through A and b] even though the already shortest cost to D was already determined. Thus, this condition violates Dijkstra’s arguments.

Problem 7.1

Insert(G, I, J):

C = G[I];

While(C.next != null)

C = C.next

C.next = J

C = G[J]

While(C.next != null)

C = C.next

C.next = I

Delete(adjlist, I, J)

For key in adjlist

If key is I or key is J

Set.add(getindex(key))

Ptr1 = adjlist(set[0])

Ptr2 = adjlist(set[1])

While(1):

If Ptr1.next is I or J

If Ptr1.next.next == null

Ptr1.next == null

Else

Ptr1.next = Ptr1.next.next

If Ptr1 != null

Ptr1 = Ptr1.next

If Ptr2.next is I or J

If Ptr2.next.next == null

Ptr2.next == null

Else

Ptr2.next = Ptr2.next.next

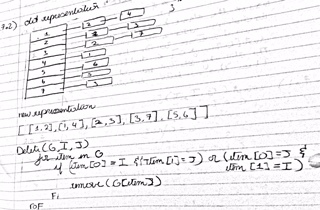
If Ptr2!= null

Ptr2= Ptr1.next

If Ptr1 and Ptr2 are null

Break out of while loop

7.2



7.3

