LAB 11 A

problem 1 : Must every dense graph be connected ?

NO, it doesn't ~~necessarily~~ need to be connected to be dense.

proof : Let G is a graph with |V| vertices and |E| edges and is disconnected.

G has components $G_1, G_2, G_3 \dots, G_k$ each has v,e.

since, it is possible that all those components be complete graphs

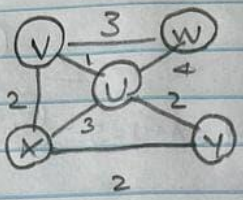
$$m_G = m_1 + m_2 + m_3 \dots m_k$$

$$= \left(\frac{n_1 - 1}{2}\right) + \left(\frac{n_2 - 1}{2}\right) \dots \left(\frac{n_k - 1}{2}\right)$$

$$= \sum_{i=1}^{k} \frac{n_i (n_i - 1)}{2} = O(n^2)$$

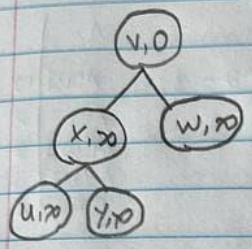
So the graph G has $O(n^2)$ no of edges which is equal with the amount expected by dense graphs.

problem 2: carry out fast Dijkstra's Algorithm



A

A[v]	0
A[x]	∞
A[w]	∞
A[u]	∞
A[y]	∞



$X = \{\}$

$M = \{v, x, w, u, y\}$

Step 1 remove min (v,0)

vertice adjacent to v: x, w, u

process x:

greedylen = A[v] + wt(v,x) = 0+2 = 2

greedylen < A[x]? yes

old = A[x] A[x] = 2

Q.update key((x,old), (x,2))

process w:

greedylen = A[v] + wt(v,w) = 0+3

Greedlen < A[w]? Yes

Oldlen ≠ A[w] A[w] = 3

Q.update((w,old),(w,3))

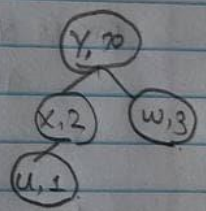
process u:

greedylen = A[v] + wt(v,u) = 0+1

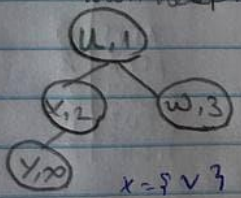
greedylen < A[u]? Yes

Old = A[u] A[u] = 1

Q.update((u,old), (u,1))



down heap x 3



$X = \{v\}$

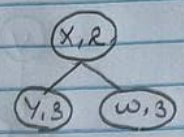
$M = \{x, w, u, y\}$

A

A[v]	0
A[x]	2
A[w]	3
A[u]	1
A[y]	∞

Step 2: remove min (u)
M.remove (u)
Vertices adjacent to u: x, w, y

Process x:
greedyLen = A[u] + wt(u,x) = 1+3 = 4
greedyLen < A[x]? NO



process w:
greedyLen = A[u] + wt(u,w) = 1+4 = 5
greedyLen < A[w]? NO

process y:
greedyLen = A[u] + wt(u,y) = 1+2 = 3
greedyLen < A[y]? yes
old = A[y] A[y] = 3
Q.updatekey ((x,old),(y,3)

A	
A[v]	0
A[x]	2
A[w]	3
A[u]	1
A[y]	3

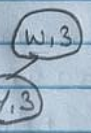
x = {v, u}
M = {x, y, w}

Step 3: removemin (x)
M.remove (x)
Vertices adjacent x: y

process y:
greedyLen = A[x] + wt(x,y) = 2+2 = 4.
greedyLen < A[y]? NO

x = {v, u, x}
M = {y, w}

A	
A[v]	0
A[x]	2
A[w]	3
A[u]	1
A[y]	3



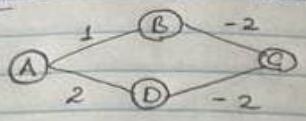
Step 4: removemin (w)
M.remove (w)
vertices adjacent to w: NO
x = {v, u, x, w}
M = {y}

remove (y)
x = {v, u, x, w, y}
M = { }

A	
A[v]	0
A[x]	2
A[w]	3
A[u]	1
A[y]	3

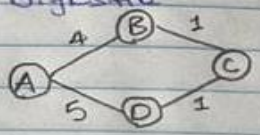
problem 3: Shortest Path from A to C

A)



there is no algorithm that truely computes the shortest path on undirected negative weight edge graphs

Let's add 3 to all the edges and work it out with Dijkstra



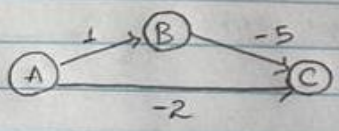
the shorted path from A to C is A-B-C which is 5

$5 - 4 = -\frac{1}{2}$

However, by following the path A-B-C-B-C which is -7

By following that path we can compute smaller and smaller super path length.

B)



1- what goes wrong by applying Dijkstras Algorithm it will compute A[C] = -2 because of the greedy length choice.

2- Dynamic programming.

D[A] = 0

D[B] = min {D[x] + wt(x,v) | (x,v) ∈ E}

= D[A] + wt(A,B) = 0 + 1 = 1

D[C] = min { D[A] + wt(A,C) = 0 - 2 = -2
D[B] + wt(B,C) = 1 - 5 = -4

D[C] = -4 // correct.