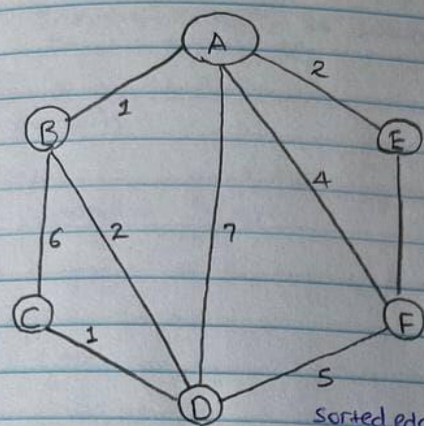
LAB 11 B

problem 1: Kruskal's Algorithm

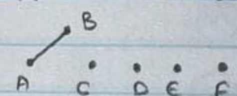


Sorted edges = {AB, CD, AE, BD, EF, AF, DF, BC, AD}.

T = { }

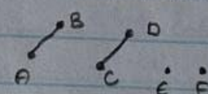
Step 1: initialization

Step2: $C(A) \neq C(B)$



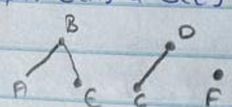
T = {AB}

Step3: $C(C) \neq C(D)$



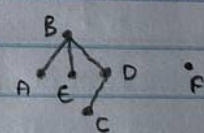
T = {AB, CD}

Step4: $C(A) \neq C(E)$



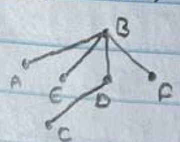
T = {AB, CD, AE}

Step 5: $C(B) \neq C(D)$



T = {AB, CD, AE, BD}

Step6: $C(E) \neq C(F)$



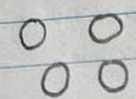
T = {AB, CD, AE, BD, EF}

problem 2: suppose $G:(V,E)$ is undirected (un weighted) simple graph. A subset U of V is called a base for G in every edge for G ~~is every~~ has at least one endpoint in U. Do the ff:

A) Given $G=(V,E)$. is it true that V itself is a base for G? explain.

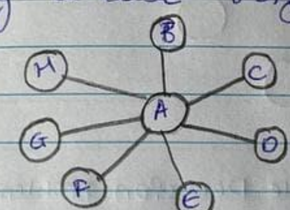
Yes, by definition of E, every e in E has an endpoint in V.

B) Is there a graph G having a base that is the empty set? If so, give an example.



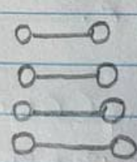
Yes, any graph with one or more vertices and no edges is an example.

C) Given an example of a graph G having 8 vertices and having a base of size 1.



if $U=\{A\}$ and every edge has one end point in U.

D) Given an example of a graph G having $2n$ vertices with the property that every base for G has size at least n.



if n of the given edges are in U then the each node have one end point in the graph.

e) Devise an algorithm to solve Smallest Base Decision problem.

```
P ← obtain set of all subsets of V

currentMin ← |V|
currentBase ← V

for U in P do
    for e in E do
        a ← left endpoint of e
        b ← right endpoint of e
        if a in U or b in U then
```

currentMin ← |U|

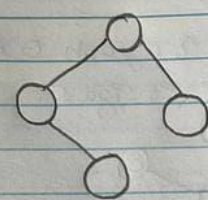
currentBase ← U

return U.

the running time is $O(m2^n)$

problem 3: suppose $G = (V, E)$ is an undirected graph (unweighted) simple graph. A spanning cycle for G is a simple cycle in G that contains every vertex of G.

A) give an example of connected graph having three or more vertices which has no spanning cycle.



B) The spanning cycle Decision problem is

Given a graph $G = (V, E)$, does G contain a spanning cycle?

A graph G itself a simple cycle iff G is connected and every vertex in G has degree 2.

Algorith: Check for Spanning (G(V, E))

Input: edges and vertices of G

output: True if spanning and false otherwise

take all edges & build ~~a~~ adjacency matrix

1 → if edge exists 0 if it doesn't exist.

while(there is unvisited edge)

{ check if there is connection from last to the first node

check if graph is connected

~~check if each vertex = dev(v) = 2~~ ?.