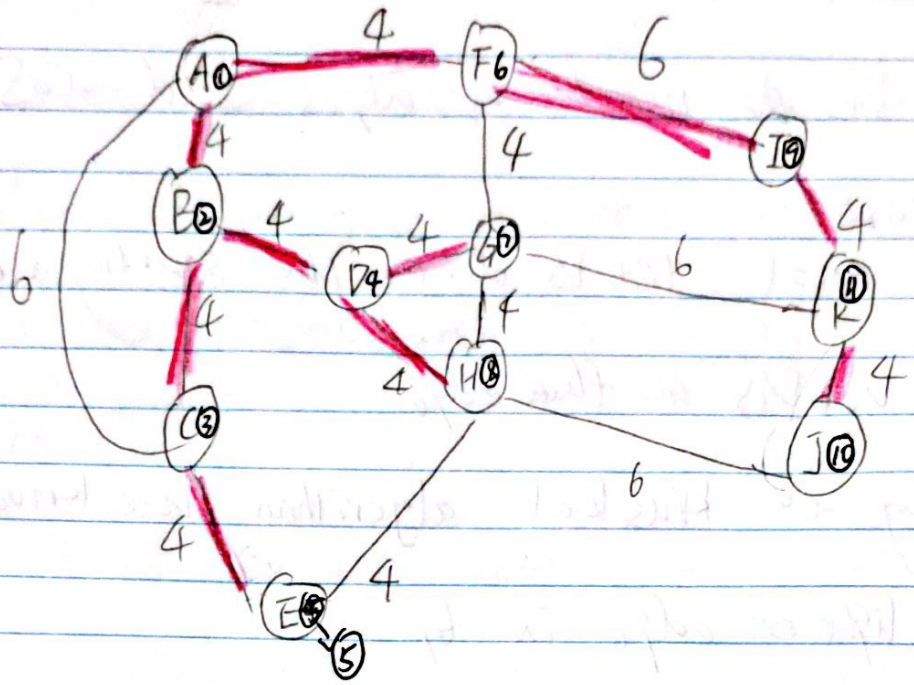
CSC 226 Assignment II

DEWEI YU
V00897211

1)

a)



The Graph is shown in Red

b) Minimum spanning Tree = (9+6+3*4+3*4)*3 = 117

2) Let T be the MST of G. Let S be the spanning tree created by Kruskal algorithm

We would like to show T == S

We will use induction on the number of edges added

to S. That is, we show that for every edge e that Kruskal adds to S, then e must be T

Let m be the member of edges add to S.

(1) Base case:

when $m=1$, let V_0 donate the starting vertex. Let $e_1 = \{V_0, V_1\}$ be that edge.

According to kruskal algorithm, we know e_1 is the lightest edge in G

Let $X=\{V_0\}$ be a cut of G, By our proof we know that minimun weight edge from X to V-X must be in the MST T. The edge is e_1.

(2) Recursive case:

Let $m=k$ and Let $S=(\{V_0, V_1 \dots, V_k\}, \{e_1, \dots e_k\})$ be the current state of tree building by kruskal algorithm after k iteration.

Assume that $e_1, \dots e_k$ are all in T, the MST for G.

Consider $m = k+1$. Run the next iteration of Kruskal, we will check an edge e_x. then e_x have 2 cases:

case1: if add e_x will create a cycle. then e_x will be discard. According to circle property. we know e_x is the heaviest edge in the cycle. e_x must contain in T

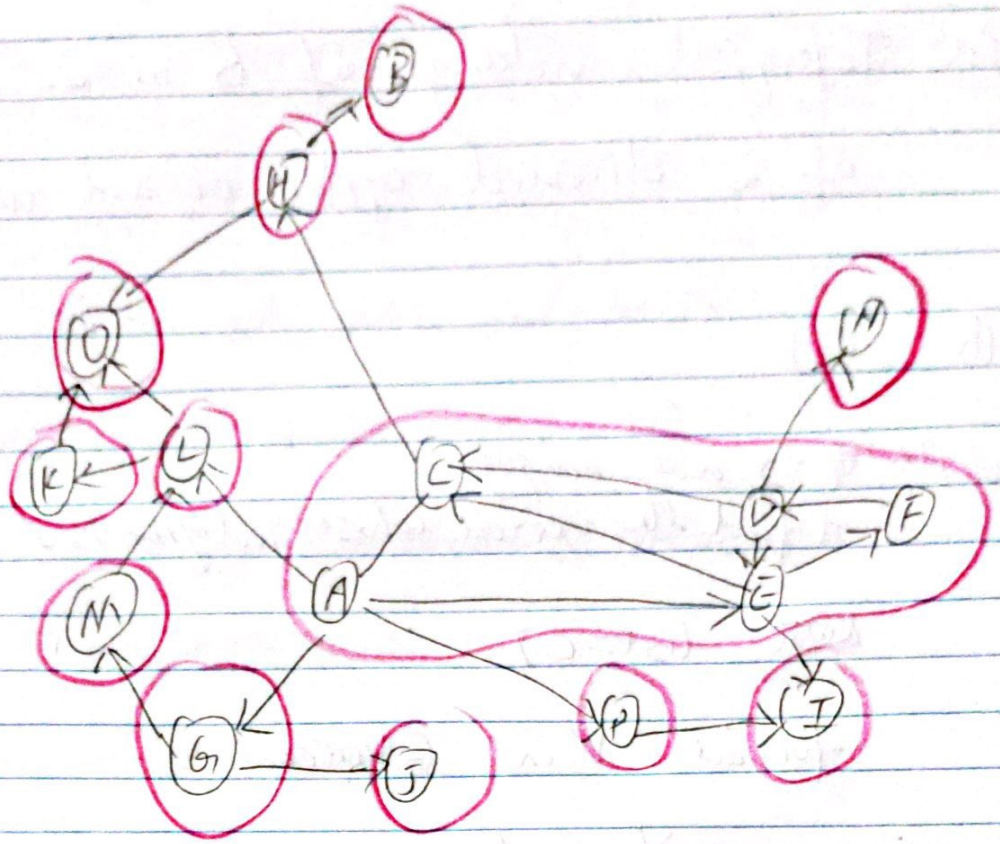
case 2: if add e_x will not create a cycle, then e_x will be added. According to cut property. Let $X = \{v_0, v_1 \dots v_k\}$ e_x must be the lightest edge between X to G-X. Then e_x must in T.

Therefore. e_x are discard or selected correctly.

~~Finally, Kruskal~~

∴ Kruskal was proven to be right.

3)



4)

a) key of pesudocode:

Each time we select the nodes whose indgree is 0, if the count of these node is larger than 1 then the topological is not unique.

pesudocode:.

Topological Sort (G):

Input: Digraph G with n-vertice

Output: Topological ordering of G or an indication of a directed cycle or not unique path

```
path = []
while G is not empty
    # find the vertexe whose indegree is 0
    list = []
    foreach V in G.Vertex
        if indegree == 0
            list.add(V)
    if len(list) == 0
        return "the graph contains cycle"
    if len(list) > 1
        return "no unique order exists"
    else
        n = list[0]
        path.add[n]
```

Remove the vertex ~~whose endpoints is n (n, u) in G. Edges~~

for each edge whose endpoint is n (n, u) in G. edges:

G. removeEdge (n, u)

~~G. remove Vertex~~ G. removeVertex (n)

return path.