CSCE 5150 Analysis of Computer Algorithms.

Assignment - 03

Ganesh Gundekarla
11700551.

Question-01 :- Use Strassen's Algo to Compute the matrix product $C = AB$ where $A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix}$

$B = \begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix}$

soln :- In order to compute the Matrix Product $C = AB$, using the Strassen's Algo, we need to be performing several Matrix Operations and some Recursive Calls. This Algo uses Matrix Partitioning and Arthemetic Operations in order to reduce the number of required number of Multiplications. So, Instead of Recursive Multiplication, in this method of a $n/2 \times n/2$ Matrix it only performs 7 Operations. So, with this we will abe to Reduce some Complexity of Time than using a Traditional Method.

step ① :- Here, we divide the Matrices A, B into $n/2 + n/2$ sub Matrices which takes the TC of $\theta(1)$.

step ② :- Then, we Compute 10 Matrixes by $S_1, S_2 \ldots S_{10}$ by either + or - operations only $\theta(n^2)$

step ③ :- Using the Above 10 Matrices, we go on to find 7 Matrices $P_1, P_2 \ldots P_7$ by Multiplicn $\longrightarrow$ TC :- $T(n/2) \cdot 7$

step ④ Add and Substract A to construct subMatrices C_{ij} of C $\longrightarrow \theta(n^2)$

$$T_n = 7T(n/2) + \theta(n^2).$$

$$a = 7, b = 2, f(n) = \theta(n^2)$$

$$\log_b a = \log_2 7 \quad T_n = \theta(n \log_2 7) \quad = \theta(n^{2.8074})$$

So, given $A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix}$ $B = \begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix}$

step ① $A_{11}=1$, $A_{12}=3$ $B_{11}=6$ $B_{12}=8$

$A_{21}=7$ $A_2=5$ $B_{21}=4$ $B_{22}=2$

step ② $S_1 = B_{12}-B_{22} = 8-2=6$

$S_2 = A_{11}+A_{12} = 1+3=4$

$S_3 = A_{21}+A_{22} = 7+5=12$

$S_4 = B_{21}-B_{11} = 4-6=-2$

$S_5 = A_{12}+A_{22} = 1+5=6$

$S_6 = B_{11}+B_{22} = 6+2=8$

$S_7 = A_{12}-A_{22} = 3-5=-2$

$S_8 = B_{21}+B_{22} = 4+2=6$

$S_9 = A_{11}-A_{21} = 1-7=-6$

$S_{10} = B_{11}+B_{22} = 6+8=14$

step ③

$P_1 = A_{11}\cdot S_1 = 11\cdot 6 = 66$

$P_2 = S_2\cdot B_{22} = 4\cdot 2 = 8$

$P_3 = S_3\cdot B_{11} = 12\cdot 6 = 72$

$P_4 = A_{22}\cdot S_4 = 5\cdot -2 = -10$

$P_5 = S_5\cdot S_6 = 6\cdot 8 = +48$

$P_6 = S_7\cdot S_8 = -2\cdot 6 = -12$

$P_7 = S_9\cdot S_{10} = -6\cdot 14 = -84$

step ④

$C_{11} = P_5+P_4-P_2+P_6$

$= 48-10-8-12 = 18$

$C_{12} = P_1+P_2 = 6+8 = 14$

$C_{21} = P_3+P_4 = 72+(-10) = 62$

$C_{22} = P_5+P_1-P_3-P_7 = 48+6-72-(-84) = 66$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} 18 & 14 \\ 62 & 66 \end{bmatrix}$$

Question-3.

Write a PseudoCode for the Brute Force Method of solving the Max Sub Array problem which should be running in $\Theta(n^2)$ time.

solution:- The pseudo code of the Brute Force Method for solving the Max-Sub-Array Problem is given here

→

```
Brute_Force Maximum Subarray (arr)
{
    n = length(arr);
    max_sum = - infinity;
    left_index = 0;
    right_index = 0;
    for (i=0 ; i<=n-1 ; i++)
    {
        current_sum = 0;
        for( j=i ; j <n-1; j++)
        {
            current_sum = current_sum + arr[j] ;
            if( current_sum > max_sum)
            {
                max_sum = current_sum;
                left_index = i;
                right_index = j;
            }
        }
    }
    return (left_index, right_index, max_sum);
}
```

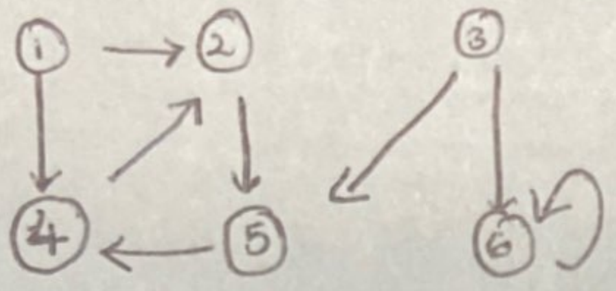
TC of above pseudoCode = $\theta(n^2)$

here,

n = length of input Array.

Question - (4)

show the Result of Running BFS on following directed Graph, using vertex 3 as source. Assume that the vertices are considered in an increasing order.

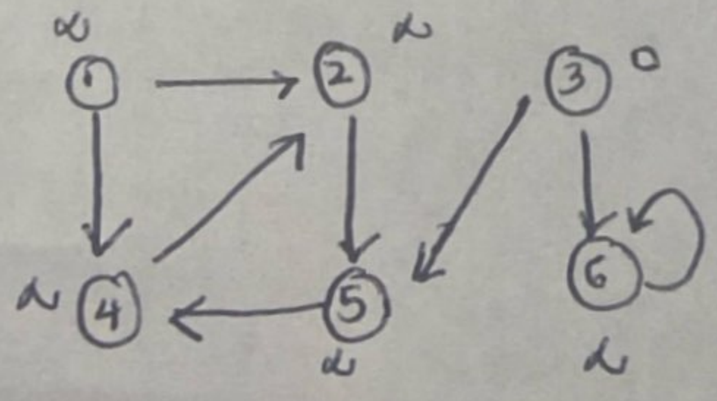


solution.

Pseudocode

```
BFS(G,s)
for each vertex u in V-{s}
    d[u] = +infty
end for
 d[s] = 0
 Q = Empty  // Create a FIFO
Enqueue (Q,s)
while Q. not empty
      u = Dequeue (Q)
 for each v in Adj[u]
     if d[v] = +infty
        d[v] = d[u]+1
        enqueue (Q,v)
     endif
 end for
endwhile
return d
```

- Initially assigning all the distances as ∞, and source distance as '0'
  Here, given source vertex is 3. Also Q is Empty



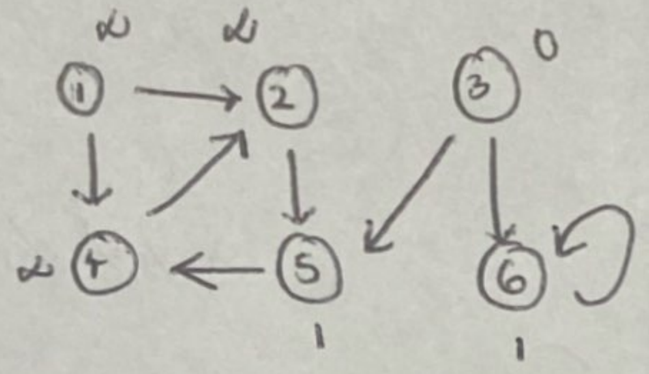
Enqueue (Q,3) ⇒ Q = {3}

Now, Q is not Empty, so we will Dequeue from FIFO Queue and
→ initialize it to u.

$u = 3$

Now, we will check for adjacent nodes of u i.e (3) and if it's distance is ∞, then we will increment the distance of it's parent node with '1' and Assign it to it's Adjacent Node. Then we will Enqueue that Adjacent node into Q. We will repeat these steps untill our Q is not Empty.

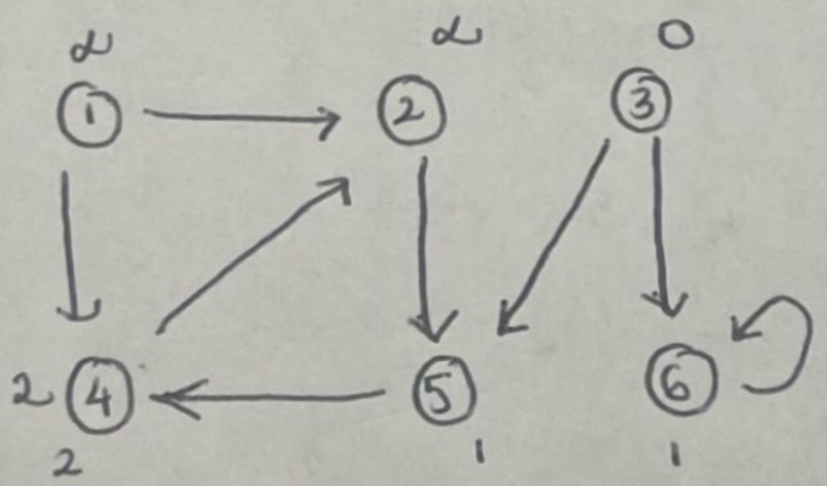
→ Here, for node 3, we have 5 and 6 as their Adjacent nodes & distance of 3 is 0, and distance of 5, 6 is ∞, so increment distance by 1 and Assign to 5, 6 and Enqueue 5, 6 into Q.



$Q = \{\cancel{3}, 5, 6\}$

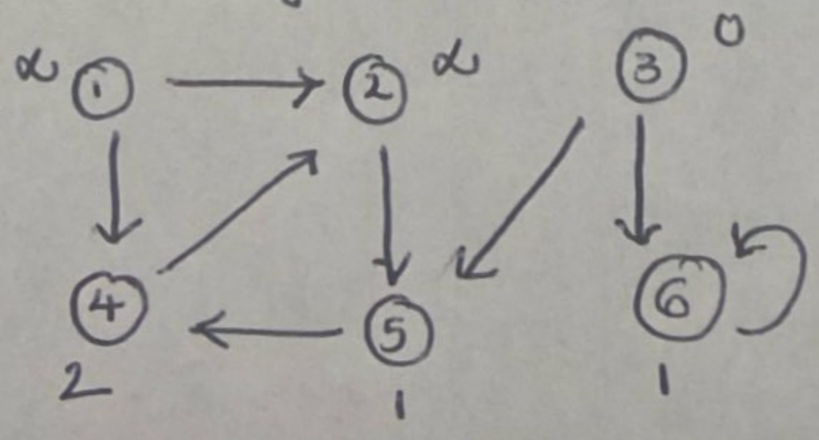
- Q is Not Empty, and Pop node 5, and Adjacent to 5 are only 4 node. It's the distance is ∞, so increment distance with 1 i.e $1 + 1 = 2$

and Enqueue 4 into Q.



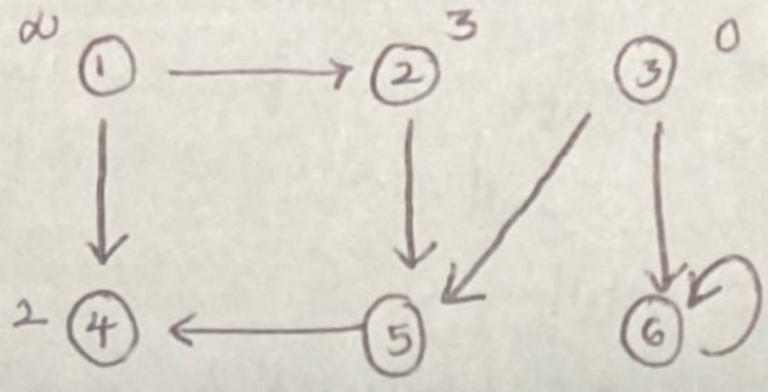
$Q = \{\cancel{3}, \cancel{5}, 6, 4\}$

Q is Not Empty, and Pop node 6, and adjacent to (6) is Node (6) itself as it has self Node and it's distance is 1, but not infinity, so there is No change in it's next Step.



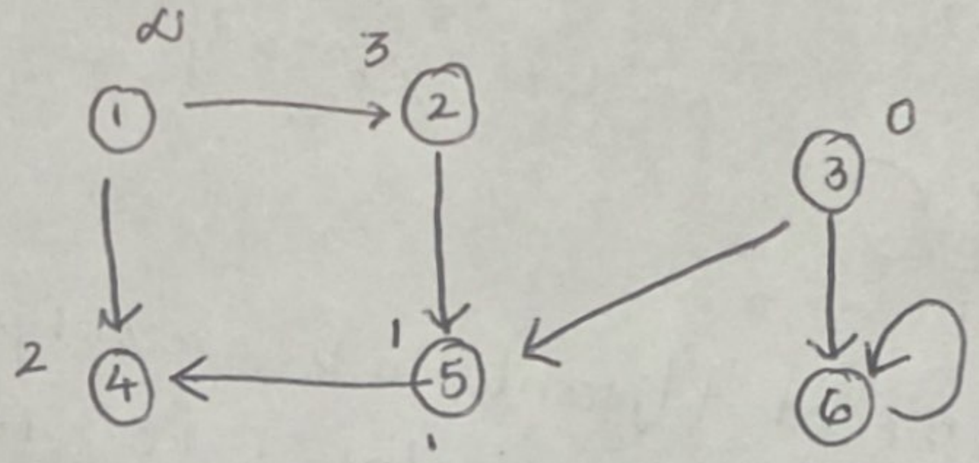
$Q = \{\cancel{3}, \cancel{5}, \cancel{6}, 4\}$

• Q is Not Empty and pop node 4, and the adjacent Nodes to 4 are only 2. Its distance is ∞, and so the incremental distance (4) with 1 i.e 2+1=3 and assign it to (2). Enqueue (2) into Q.



Q = { 3, 5, 6, 4, 2 }

Q is Not Empty, pop Node 2 and it's Adjacent Nodes are only node (5), But (5) distance is ∞. So we will Not make Any changes.



Q = { 5, 6, 4, 2 }

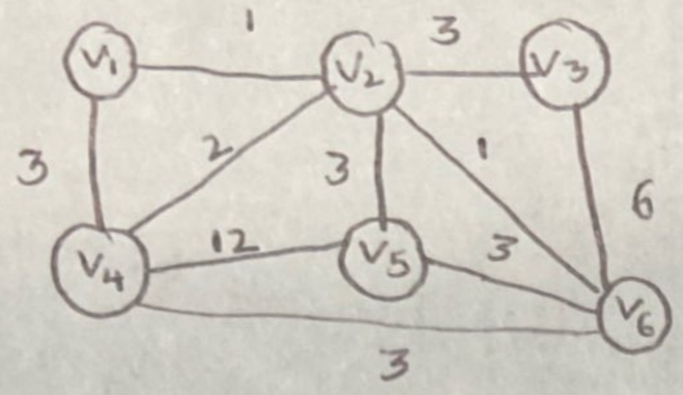
Now Q is Empty. and so will return the distance S.

Below are the distances from the Source Node to each other node/vertex to Reach them. here vertex (1) distance is still ∞, because we don't have possible path from (3) to reach (1) in given Graph.

Vertex	1	2	3	4	5	6
distance	∞	3	0	2	1	1

Question - 5.

Run Prim's Algo with the Root Vertex v_4 for finding the Min Spanning Tree on the following graph.

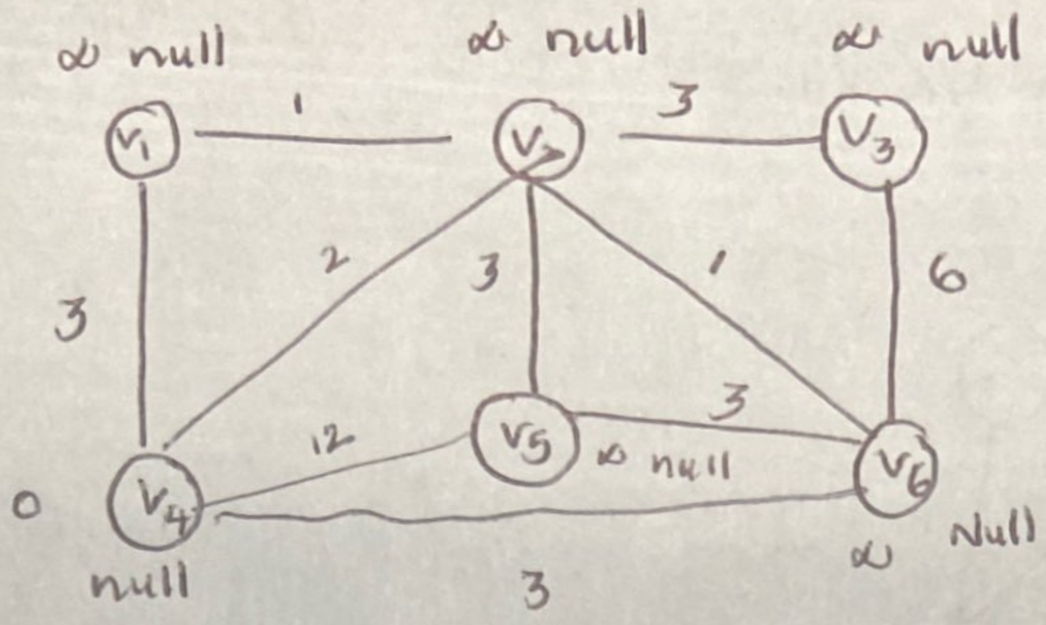


solution:-

```
MST_Prim(G,w,r)
Q = Empty
for each vertex u in V
    key[u] = infty  // Min-weight of Any Edge (w,u) & w in A
    pi[u]=nill      // Parent of u.
    Insert(Q,u)
end for
Decrease_key(Q, r, 0)
while Q not Empty
      u= Extract_min(Q)
      for each v in Adj[Q]
          if (v in Q) and (w(u,v) <key[v])
              Decrease_key(Q,v, w(u,v))
              pi[v]=u   // parent of v
          end if
      end for
    end while
    return A = {(v, Pi[v]) : v in V-{r}}   MST
```

→ Initially Assign all the vertices with their Edge weight as ∞ and it's parent as Null, Decrease Soure vertex Edge to 0. Given Source vertex is V_4 and insert all Vertices into Queue Q.

$$Q = \{ V_1, V_2, V_3, V_4, V_5, V_6 \}$$



As Q is Not Empty, we have to Extract_min(Q) Here as Weight V_4 Edge Weight is 0 and is ~~Maximum~~ Minimum, we extract V_4 initially

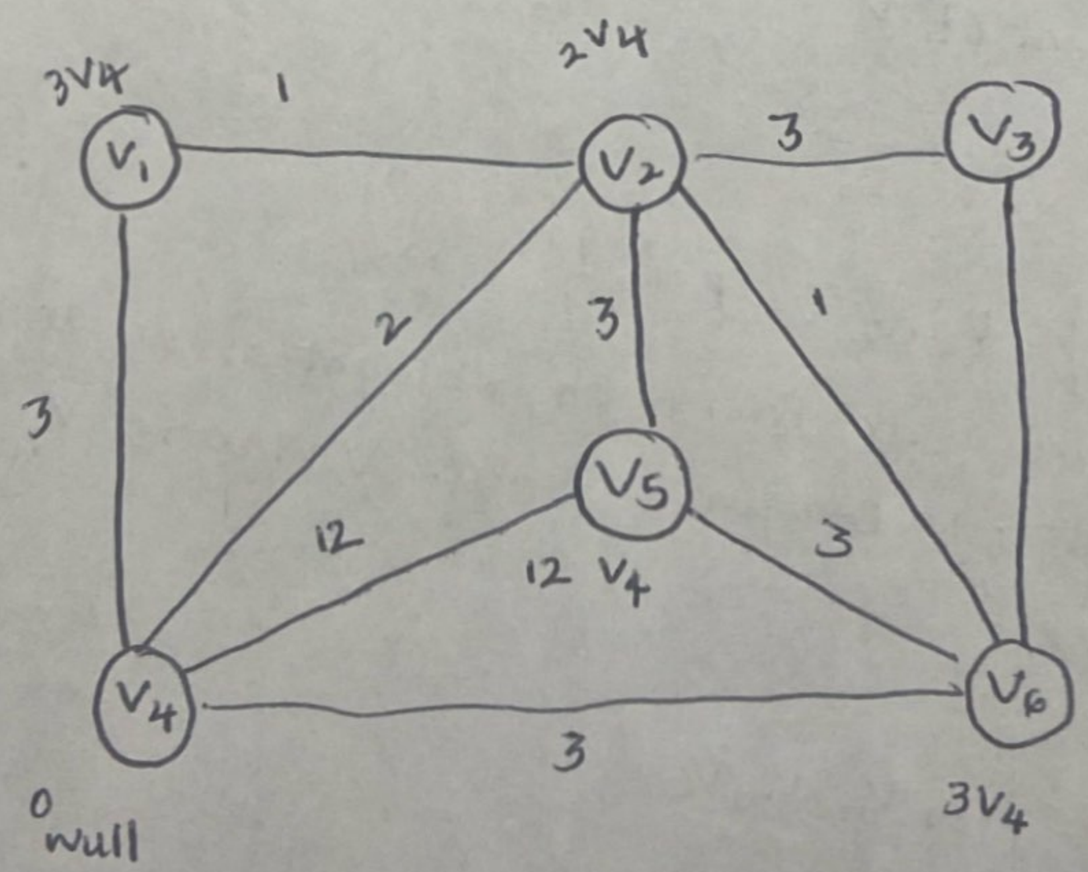
$u = V_4$, $adj[V_4] = V_1, V_2, V_5, V_6$

$Q = \{ V_1, V_2, V_3, V_5, V_6 \}$

$= V_1, V_2, V_5, V_6$ are Present in Q

$(V_4, V_1) = 3$ ($3 < \infty$) $\quad (V_4, V_2) = 2$ ($2 < \infty$) $\quad (V_4, V_5) = 12$ ($12 < \infty$) $\quad (V_4, V_6) = 3$ ($3 < \infty$)

- Now, Update Edge Weight of Each Vertex with Smaller values. Update it's Parent Node Information too



$V_1 \rightarrow 3$

$V_2 \rightarrow 2$ ✓ → Min.

$V_3 \rightarrow \infty$

$V_5 \rightarrow 12$

$V_6 \rightarrow 3$

- As v_2 holds smaller / Minimum Edge weight, Extract v_2 from Q $\Rightarrow$ $u = v_2$

$adj[v_2] = v_1, v_3, v_4, v_5, v_6$

$Q = \{v_1, v_3, v_5, v_6\}$

adj Vertices v_1, v_3, v_5, v_6 are Present.

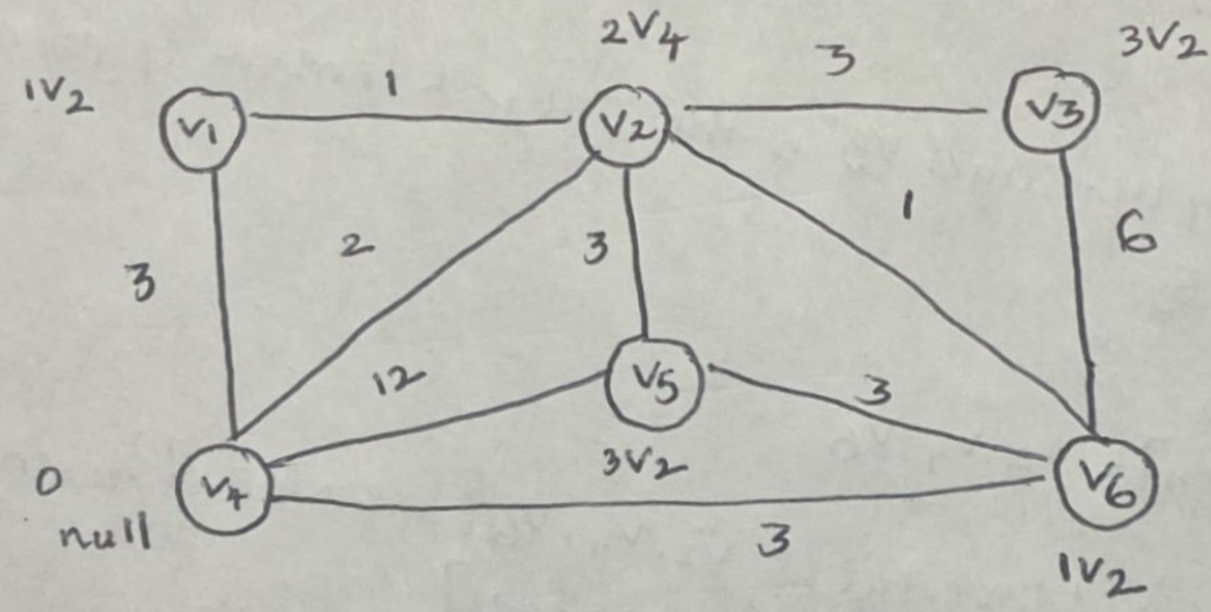
$(v_2, v_1) = 1$ → 3 → $1 < 3$ ✓

$(v_2, v_3) = 3$ → ∞ → $3 < \infty$ ✓

$(v_2, v_5) = 3$ → 12 → $3 < 12$

$(v_2, v_6) = 1$ → 3 → $1 < 3$

- Update All these 4 vertices Edges weights with Smaller vals, Update it's Parent Node info too.



$V_1 \rightarrow 1$
$V_3 \rightarrow 3$
$V_5 \rightarrow 3$
$V_6 \rightarrow 1$

- Now, v_1 and v_6 holds Min. Edge Weight, but as v_1 Comes First in Alphabitical Order, we Extract v_1 from Q

$u = v_1$, $adj[v_1] = v_2, v_4$

$Q = \{v_3, v_5, v_6\}$ = Adj Vertices v_2, v_4 both are Not Present in Q

So we Make No changes.

- As Q is still Not Empty, We look to Extract_Min from Q. As v_6 holds the Minimum Edge weight of 1, we extract v_6 from Q

$u = v_6$; $Adj[v_6] = \{v_2, v_3, v_5, v_4\}$

$Q = \{v_3, v_5\}$ ⇒ Adj-vertices v_3, v_5 are only present in Q

$(v_6, v_3) = 6$ → 3 → $6 \not< 3$

$(v_6, v_5) = 3$ → 3 → $3 \not< 3$ ×

- When we compare Edge Weights of Vertices V_3, V_5 both doesn't satisfy the Condit^n. So we doesn't make Any changes.

$V_3 \rightarrow 3$ ✓

$V_5 \rightarrow 3$ ✓

- Here, as Q is still not Empty, we look to Extract_min from Q. Now both V_3 and V_5 holds same Minimum Edge Weight of 3. But as V_3 comes first in Alphabetical Order, we extract V_3 from Q.

$u = V_3$

$Adj[V_3] = V_2, V_6$.

$Q = \{V_5\} \Rightarrow$ adj Vertices V_2, V_6 both are not present in Q. so we make No changes.

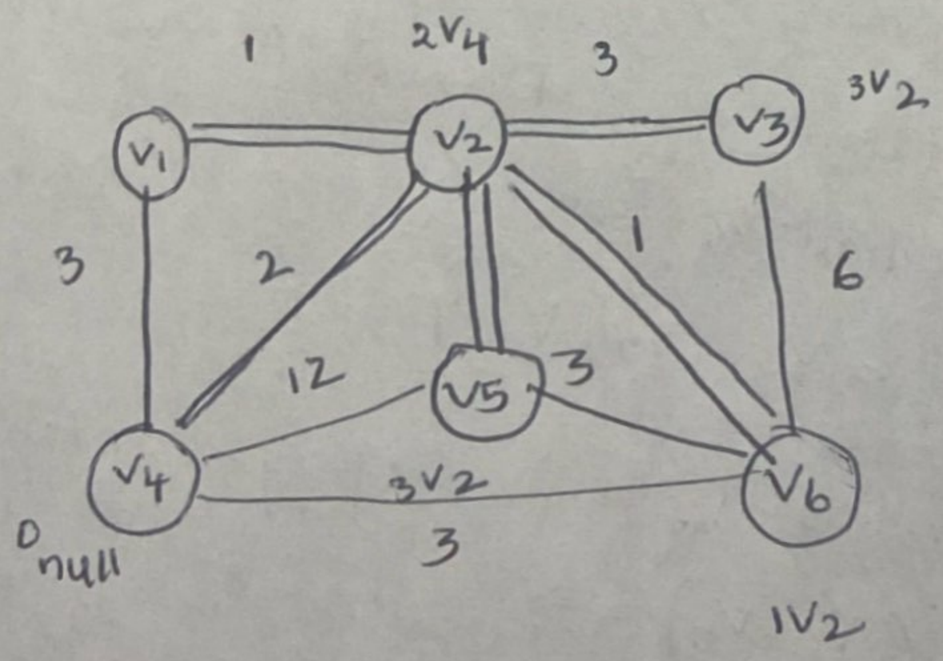
As Q is still Not Empty, and V_5 is the Only Element present in Q we extract V_5 from Q.

$u = V_5$

$adj[V_5] = V_2, V_4, V_6$

$Q = \{\}$ Adj vertices V_2, V_4, V_6 are Not Present in Q. [We Make No Changes]

Now, as Q is Empty, we stop iterating the while loop and return the Parent Nodes of each vertex in the graph. Below is the Final Min-Spanning Tree with it's Final Min Edge weights and it's Parend Nodes



vertex	V_1	V_2	V_3	V_4	V_5	V_6
Parent	V_2	V_4	V_2	Null	V_2	V_2

Question - 2.

· How Many lines does the Program Print $[n=2^d]$

```
function fun(n)
 if n>1
      print.line ('good day')
      fun(n/2)
      fun(n/2)
 end if.
```

→ The value of n is defined a 2^d, where d is exponent, lets Analyse the program's Execution for two values of d.

(a) if d=1, then $2^1=2$, the prints "good day" once and then calls itself with n/2 which is 1. The Recursion Ends here at this point, so it prints 1 Line.

(b) if d=2 then $n=2^2=4$, the Program Prints "good day" for once and then calls itself with n/2 = 2 for two Recursive Calls, the

Total Prints = 1 + 2 = 3 Lines
(Recursive)

(c) then d=3, n=8;

Prints "good day" onLe in current call and calls itself twice with n/2 which is 4, then each these print good day and perform furthur operatn

so Total =

(current Call) + 2 (first Recursive Call) + 2(second Recursive call)
+ 2(3rd Recursive Call) = 7 . Total = $2^d - 1$ lines
(n-1) lines

. we can Also solve using Recurrence Relⁿ.

T(n) = Represents No of lines Printed by Prog for given n.

$T(n) = 1$ (for a "good day") $+ \; 2T(n/2)$

$n/2$ Recursive - 2 Lines

RR is $T(n) = 2T(n/2) + 1$ where $T(1) = 0$

solving, $T(n) = 2T(n/2) + 1$

1) $T(n) = 2(2T(n/4) + 1) + 1$

$= 2^2 \cdot T(\frac{n}{4}) + 2 + 1$

2) $T(n) = 2^3 T(\frac{n}{8}) + 2^2 + 2 + 1$

$\vdots$ for K.

$T(n) = 2^K T(\frac{n}{2^K}) + 2^{K-1} + 2^{K-2} + \cdots + 2^0$

where $n = 2^K$ $T(1) = 0$

$T(n) = 0 + 2^{K-1} + 2^{K-2} + \cdots + 2^0$

$T(n) = \frac{2^K - 1}{2 - 1} = 2^K - 1$

No of lines Printed

$T(n) = n - 1$

or

$2^d - 1$ lines.