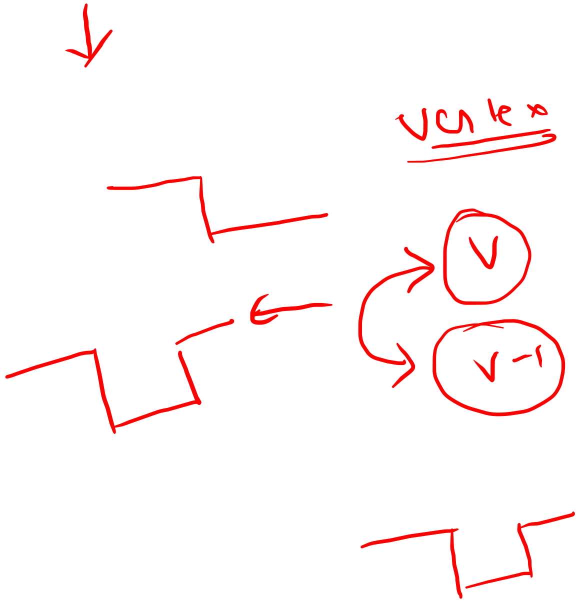
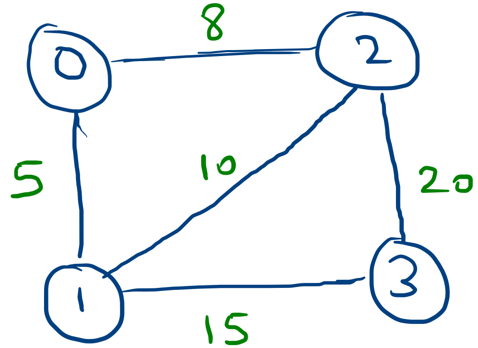
Prim's Algorithm (C++ Implementation)

→



G =



Prim's Algorithm (C++ Implementation)

Quick Revision of concept

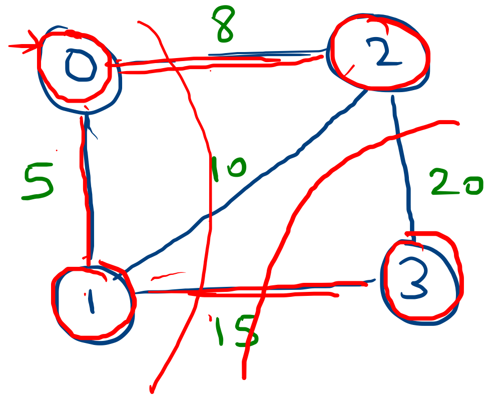
min

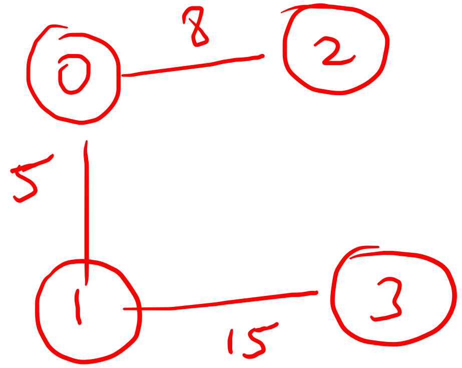
Path sum

res = 0 + 5 + 8 + 15 } ⇒ 28

mset = {0, 1, 2, 3}

G =





C++ Program

↳ we have 2 challenges

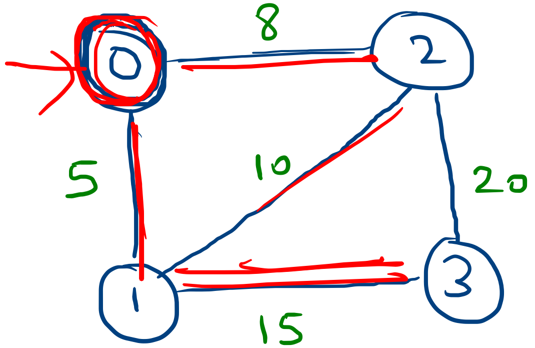
1) Maintain mset { }

[create a vector<int> (V, false)
boolean]

2) Pick the minimum edge which
Connect the mset { } to the
other vertex

[Maintain Key[V] = { INT_MAX }
key[0] = 0]

∴ V = 4 (no. of vertex)



F | F | F | F

{ 0, 1 }

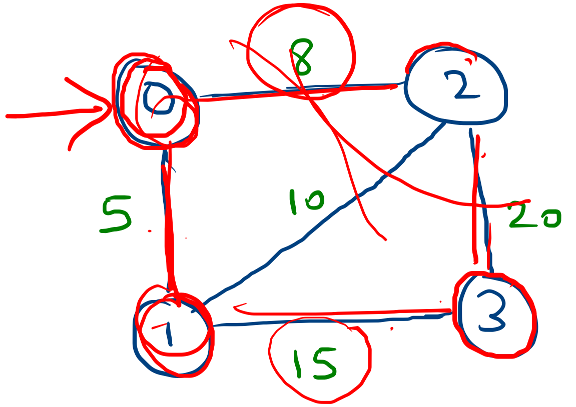
{ ∞, ∞, ∞, ∞ }

Initilisation Part

∴ V = no. of Vertex

Part 1 ⇒

```
Vector<int> key( V, INT_MAX )
key[0] = 0 ;
Vector<bool> mset( V, false)
```



Part 2 ⇒ [We traverse the Graph for finding next Vertex] →

→ We are going to find the minimum value from key[ ] and the index as Vertex 'u'

→ Now, mark the vertex 'u' as a Part of mset{ }

→ Add key to the Result.

Initially

0 1 2 3 →

key[ ] = {0, ∞, ∞, ∞}

mset[ ] = { F, F, F, F }

u

res = 0 + 5 + 8

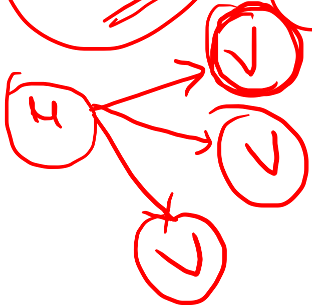
(V +

min

key

min heap

mset



```
for(int count=0 ; count < V ; count++)
{
    int u = -1;
    for(int i=0 ; i<V ; i++)
    {  if(!mset[i] && (u == -1 || key[i] < key[u]))
       { u = i; }
    }
    mset[u] = true;
    res += key[u];
    for(int V=0 ; V < V; V++)
    {  if( graph[u][V] != 0  && !mset[V])
       { key[V] = min( key[V], graph[u][V]) };
    }
}
return res;
```

0

0	∞	∞	∞

u = -1

key

O(n)

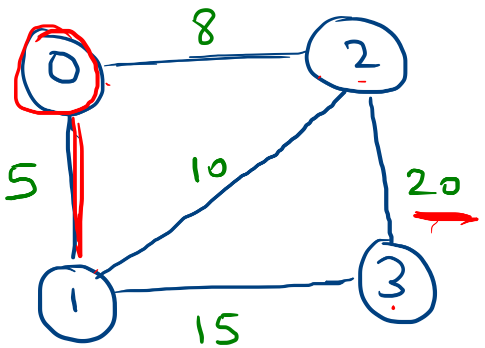
log

u = 5+8+15 = 28

3 - 0, X

2 X

u=3



count=0 ; u=0

mset[] = {T, F, F, F}

key[] = {0, 5, 8, ∞}

count=1 ; u=1

mset[] = {T, T, F, F}

key[] = {0, 5, 8, 15}

count=2 ; u=2

mset[] = {T, T, T, F}

key[] = {0, 5, 8, 15}

count=3 ; u=3

mset[] = {T, T, T, T}