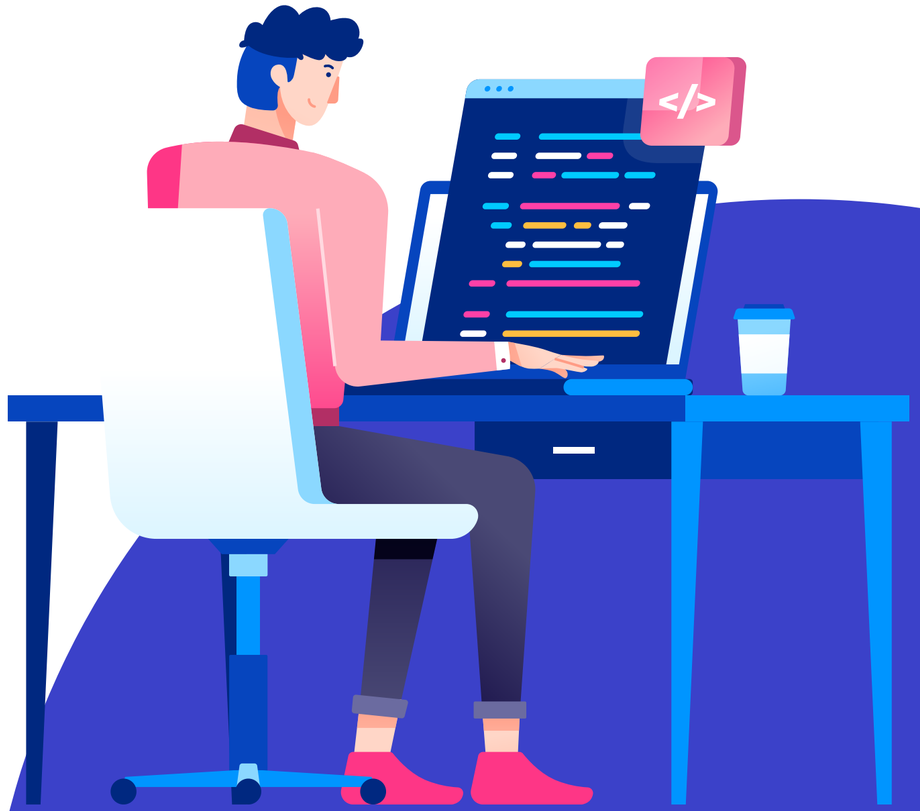
MINIMUM SPANNING TREE

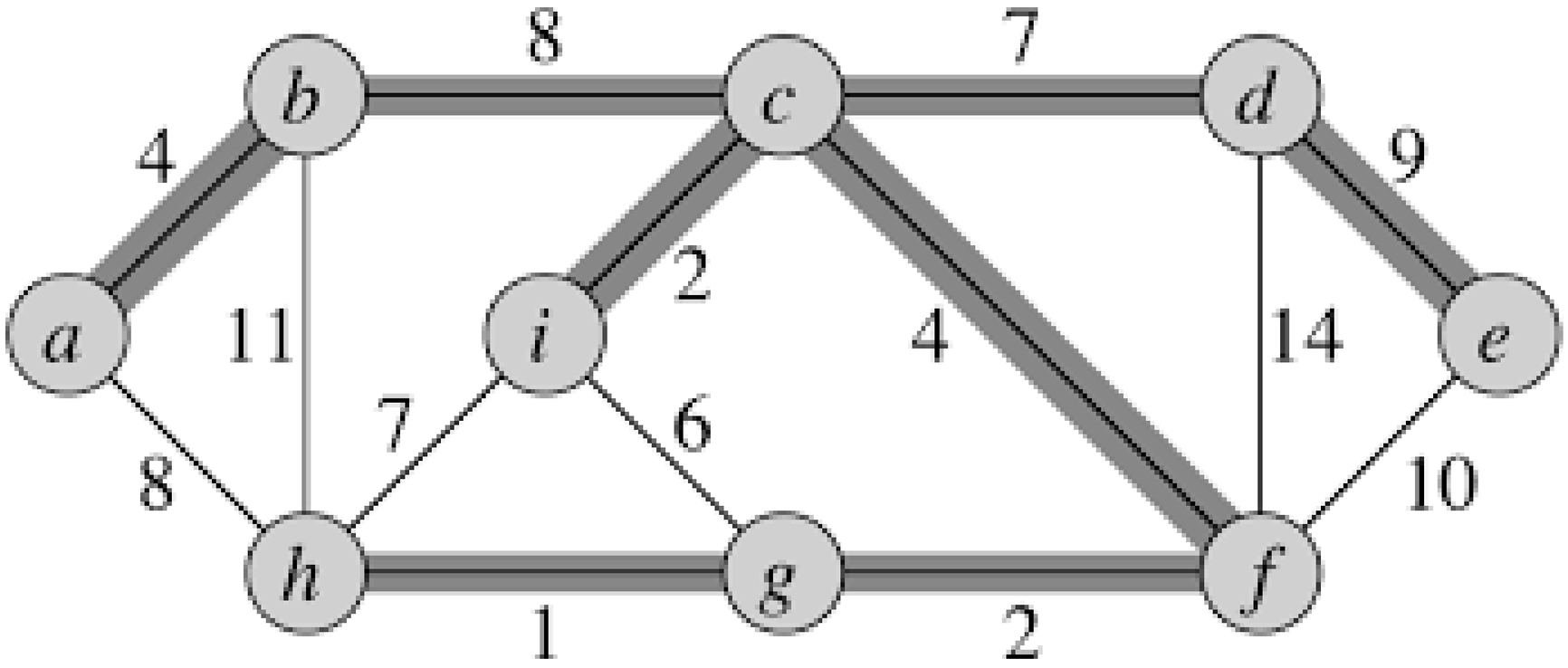
PRIM'S ALGORITHM

Juan E. Aguilera



REGLAS GENERALES

¿QUE ES UN MST?



¿APLICACIONES?

TRAVELING SALESMAN PROBLEM

DISEÑO DE REDES Y CIRCUITOS

LOGISTICA Y DISTRIBUCION

```python
class Graph():
    def __init__(self, vertices):
        self.V = vertices
        #inicializa un grafo con el numero de vertices ingresado pero sin ninguna conexion entre ellos
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]


    def printMST(self, parent):
    #Funcion para imprimir el arbol minimo de expansion.
    # "\t" es usado para insetar un espacio de tabulacion
        total = 0
        print("Edge \tWeight")
        for i in range(1, self.V):
            print(parent[i], "-", i, "\t", self.graph[i][parent[i]])
            total += self.graph[i][parent[i]]
        print("Total weight is: ", "\t", str(total))

    def minKey(self, key, mstSet):

        # Inicializa el minimo valor con infinito (sys.maxsize)
        # Esta funcion encuentra el par de vertices con la minima distancia. Ojo, solo de los vertices que no han sido añad
        min = sys.maxsize
        for v in range(self.V):
            if key[v] < min and mstSet[v] == False:
                min = key[v]
                min_index = v
```

```python
def primMST(self):

    key = [sys.maxsize] * self.V #inicializa todas las distancias con el valor de infinito.
    parent = [None] * self.V  # Crea una lista con n veces None. n es el numero de vertices. Esta lista es para almacenar los e
    key[0] = 0 #Asigna 0 a la distancia del vertice cero para que empiece por ahi el MST
    mstSet = [False] * self.V  # Crea una lista con n veces False. n es el numero de vertices. Esta lista se usa para la compar

    parent[0] = -1

    for cout in range(self.V):

        # Esta linea escoje el minima distancia entre los vertices que no han sido procesados aun.
        u = self.minKey(key, mstSet)

        #Esta linea agrega el vertice con la menor distancia al MST.
        mstSet[u] = True


        # Actualiza los valores de la matriz de adyacencia (grafo). Recordemos que fueron inicializados con un valor infinito
        # Luego de actualizarlos y no contener infinitos, actualiza solo aquellos que pueden tener una distancia menor y aun no
        for v in range(self.V):
```

```
            # Luego de actualizarlos y no contener infinitos, actualiza solo aquellos que pueden tener una distancia menor y aun n
            for v in range(self.V):

                #El for primero analiza si self.graph[u][v]>0 para saber si son vertices adyacentes, luego verifica mstSet[v] == F
                #que no pertenece aun al MST. y luego valida que el valor actual de la distancia sea mayor al que està en el grafo
                if self.graph[u][v] > 0 and mstSet[v] == False \
                and key[v] > self.graph[u][v]:

                    key[v] = self.graph[u][v]
                    parent[v] = u

        self.printMST(parent)



if __name__ == '__main__':
    g = Graph(6)
    g.graph = [
        [0,1,7,6,0,0],
        [1,0,0,4,6,0],
        [7,0,0,0,0,9],
        [6,4,0,0,0,7],
        [0,6,0,0,0,5],
        [0,0,9,7,5,0]

]
```