PROYECTO FINAL

MELINA PLATAS SANCHEZ

1740967

22-MAYO-2018

INTRODUCCION:

En este trabajo se realizara un programa en donde se explique la ruta más corta y en cuanto tiempo se realiza en 10 lugares que son municipios del estado de monterrey. No solamente usaremos Kruskal si no también ocuparemos de un programa llamado vecino más cercano.

Y para este proyecto se necesita esencialmente conocer acerca de grafos para poder realizar las rutas más cortas.

DEFINICIONES QUE OCUPAREMOS:

GRAFO:

Conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permite relaciones binarias entre elementos de un conjunto.

ALGORITMO:

Es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite llevar a cabo una actividad mediante pasos sucesivos.

ARBOL:

Es una gráfica el cual no existen ciclos.

ALGORITMO KRUSKAL:

Es un algoritmo para encontrar un árbol de expansión mínima en un grafo ponderado.

ALGORITMO VECINO MÁS CERCANO:

También llamado algoritmo voraz (greedy) permite al vigilante elegir la ciudad no visitada más cercana como próximo movimiento.

PROGRAMAS EN PYTHON

KRUSKAL

```
def shortest(self, v): # Dijkstra's algorithm
    q = [(0, v, ())] # arreglo "q" de las "Tuplas" de lo que se va a almacenar dondo 0 es la distancia
    dist = dict() #diccionario de distancias
    visited = set() #Conjunto de visitados
    while len(q) > 0: #mientras exista un nodo pendiente
        (l, u, p) = heappop(q) # Se toma la tupla con la distancia menor
        if u not in visited: # si no lo hemos visitado
            visited.add(u) #se agrega a visitados
            dist[u] = (l,u,list(flatten(p))[::-1] + [u])    #agrega al diccionario
        p = (u, p) #Tupla del nodo y el camino
        for n in self.vecinos[u]: #Para cada hijo del nodo actual
            if n not in visited: #si no lo hemos visitado
                el = self.E[(u,n)] #se toma la distancia del nodo acutal hacia el nodo hijo
                heappush(q, (l + el, n, p)) #Se agrega al arreglo "q" la distancia actual mas la ditan
    return dist #regresa el diccionario de distancias

def kruskal(self):
    e = deepcopy(self.E)
    arbol = Grafo()
    peso = 0
    comp = dict()
    t = sorted(e.keys(), key = lambda k: e[k], reverse=True)
    nuevo = set()
    while len(t) > 0 and len(nuevo) < len(self.V):
        #print(len(t))
        arista = t.pop()
        w = e[arista]
        del e[arista]
        (u,v) = arista
        c = comp.get(v, {v})
        if u not in c:
            #print('u ',u, 'v ',v ,'c ', c)
            arbol.conecta(u,v,w)
            peso += w
            nuevo = c.union(comp.get(u,{u}))
            for i in nuevo:
                comp[i]= nuevo
    print('MST con peso', peso, ':', nuevo, '\n', arbol.E)
    return arbol
```

VECINO MÁS CERCANO

```
def vecinoMasCercano(self):
    ni = random.choice(list(self.V))
    result=[ni]
    while len(result) < len(self.V):
        ln = set(self.vecinos[ni])
        le = dict()
        res =(ln-set(result))
        for nv in res:
            le[nv]=self.E[(ni,nv)]
        menor = min(le, key=le.get)
        result.append(menor)
        ni=menor
    return result
```