*Se realizó un grafo con 11 lugares dentro de México, se realizaron todas las combinaciones en las que podemos recorrer nuestros lugares tomando como referencia el tiempo y estos fueron algunos resultados*

**Grafo con puntos geográficos**

**Lugar Estado**

Monterrey Nuevo León Mty

Xilitla San Luis Potosí X

Bacalar Quintana Roo Bac

Islas Marietas Nayarit IM

Palenque Chiapas Pque

Bosque de Chapultepec Cuidad de México Ch

Pirámides de Teotihuacán Estado de México Teo

Mérida Yucatán MY

El Malecón de Mazatlán Sinaloa Maz

La Paz Baja California Sur LP

Papantla Veracruz Pp

Combinaciones posibles entre todos los lugares dentro de nuestro grafo con las horas correspondientes en su recorrido:

g.conecta('Mty', 'X', 8)  
g.conecta('Mty', 'Bac', 25)  
g.conecta('Mty', 'IM', 13)  
g.conecta('Mty', 'Pque', 19)  
g.conecta('Mty', 'Ch', 9)  
g.conecta('Mty', 'Teo', 10)  
g.conecta('Mty', 'MY', 25)  
g.conecta('Mty', 'Maz', 9)  
g.conecta('Mty', 'LP', 23)  
g.conecta('Mty', 'PpV', 10)

g.conecta('LP', 'Maz', 14)  
g.conecta('LP', 'IM', 21)  
g.conecta('LP', 'X', 26)  
g.conecta('LP', 'PpV', 27)  
g.conecta('LP', 'Ch', 24)  
g.conecta('LP', 'Teo', 24)  
g.conecta('LP', 'Pque', 35)  
g.conecta('LP', 'MY', 40)  
g.conecta('LP', 'Bac', 40)

g.conecta('Maz', 'IM', 6)  
g.conecta('Maz', 'X', 13)  
g.conecta('Maz', 'PpV', 14)  
g.conecta('Maz', 'Ch', 11)  
g.conecta('Maz', 'Teo', 11)  
g.conecta('Maz', 'Pque', 21)  
g.conecta('Maz', 'MY', 26)  
g.conecta('Maz', 'Bac', 27)

g.conecta('IM', 'X', 17)  
g.conecta('IM', 'PpV', 15)  
g.conecta('IM', 'Ch', 15)  
g.conecta('IM', 'Teo', 9)  
g.conecta('IM', 'Pque', 28)  
g.conecta('IM', 'MY', 35)  
g.conecta('IM', 'Bac', 34)

g.conecta('X', 'PpV', 6)  
g.conecta('X', 'Ch', 6)  
g.conecta('X', 'Teo', 6)  
g.conecta('X', 'Pque', 17)  
g.conecta('X', 'MY', 22)  
g.conecta('X', 'Bac', 22)

g.conecta('PpV', 'Ch', 4)  
g.conecta('PpV', 'Teo', 3)  
g.conecta('PpV', 'Pque', 11)  
g.conecta('PpV', 'MY', 16)  
g.conecta('PpV', 'Bac', 16)

g.conecta('Ch', 'Teo', 1)  
g.conecta('Ch', 'Pque', 11)  
g.conecta('Ch', 'MY', 17)  
g.conecta('Ch', 'Bac', 16)

g.conecta('Teo', 'Pque', 11)  
g.conecta('Teo', 'MY', 16)  
g.conecta('Teo', 'Bac', 16)

g.conecta('Pque', 'MY', 7)  
g.conecta('Pque', 'Bac', 6)

g.conecta('MY', 'Bac', 4)

Tenemos en total 55 combinaciones posibles

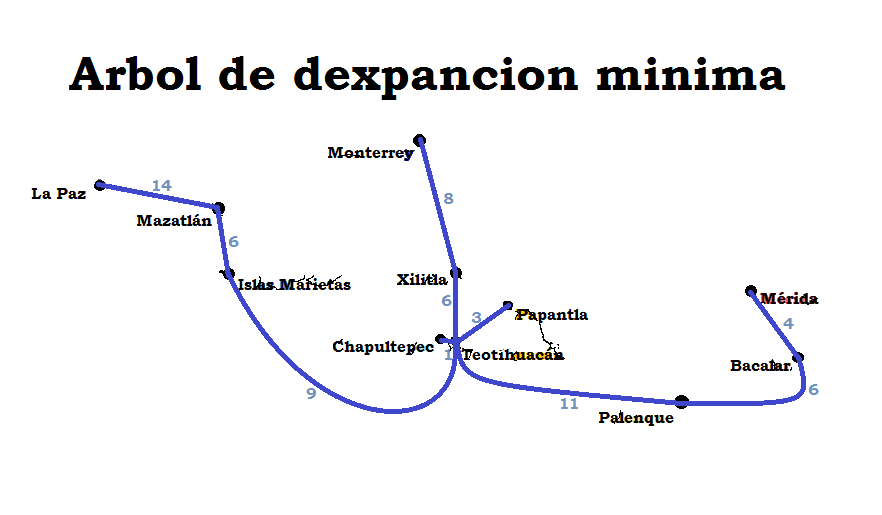
**Árbol de Expansión Mínima**

Con la función Kruskal podemos encontrar las aristas más pequeñas entre todos nuestros lugares

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

MST con peso 68 : {'Mty', 'LP', 'Maz', 'Ch', 'Teo', 'Pque', 'MY', 'Bac, 'X', 'IM', 'PpV'}

('Teo', 'Ch') 1 ('Teo', 'PpV') 3 ('Bac', 'MY') 4 ('Bac', 'Pque') 6   
('Teo', 'X') 6 ('IM', 'Maz') 6 ('X', 'Mty') 8 ('Mty', 'X') 8  
('Teo', 'IM') 9 ('IM', 'Teo') 9 ('Pque', 'Teo') 11 ('Teo', 'Pque') 11  
('Maz', 'LP') 14 ('LP', 'Maz') 14

Gráficamente este sería nuestro árbol de expansión mínima

Probamos 11 veces un recorrido empezando de un punto en nuestro grafo y terminando en el mismo y estos fueron los resultados:

['Mty', 'X', 'Teo', 'PpV', 'IM', 'Maz', 'LP', 'Pque', 'Bac', 'MY', 'Ch']

Mty X 8

X Teo 6

Teo PpV 3

PpV IM 15

IM Maz 6

Maz LP 14

LP Pque 35

Pque Bac 6

Bac MY 4

MY Ch 17

Ch Mty 9

Tiempo total: 114

['Teo', 'PpV', 'IM', 'Maz', 'LP', 'X', 'Mty', 'Pque', 'Bac', 'MY', 'Ch']

Teo PpV 3

PpV IM 15

IM Maz 6

Maz LP 14

LP X 26

X Mty 8

Mty Pque 19

Pque Bac 6

Bac MY 4

MY Ch 17

Ch Teo 1

Tiempo total: 118

['Pque', 'Teo', 'PpV', 'IM', 'Maz', 'LP', 'X', 'Mty', 'Ch', 'Bac', 'MY']

Pque Teo 11

Teo PpV 3

PpV IM 15

IM Maz 6

Maz LP 14

LP X 26

X Mty 8

Mty Ch 9

Ch Bac 16

Bac MY 4

MY Pque 7

tiempo total: 112

['Ch', 'Teo', 'PpV', 'IM', 'Maz', 'LP', 'X', 'Mty', 'Pque', 'Bac', 'MY']

Ch Teo 1

Teo PpV 3

PpV IM 15

IM Maz 6

Maz LP 14

LP X 26

X Mty 8

Mty Pque 19

Pque Bac 6

Bac MY 4

MY Ch 17

Tiempo total: 102

['IM', 'Maz', 'LP', 'Teo', 'PpV', 'X', 'Mty', 'Pque', 'Bac', 'MY', 'Ch']

IM Maz 6

Maz LP 14

LP Teo 24

Teo PpV 3

PpV X 6

X Mty 8

Mty Pque 19

Pque Bac 6

Bac MY 4

MY Ch 17

Ch IM 15

Tiempo total: 107

['Bac', 'MY', 'Pque', 'Teo', 'PpV', 'IM', 'Maz', 'LP', 'X', 'Mty', 'Ch']

Bac MY 4

MY Pque 7

Pque Teo 11

Teo PpV 3

PpV IM 15

IM Maz 6

Maz LP 14

LP X 26

X Mty 8

Mty Ch 9

Ch Bac 16

Tiempo total: 103

['MY', 'Bac', 'Pque', 'Teo', 'PpV', 'IM', 'Maz', 'LP', 'X', 'Mty', 'Ch']

MY Bac 4

Bac Pque 6

Pque Teo 11

Teo PpV 3

PpV IM 15

IM Maz 6

Maz LP 14

LP X 26

X Mty 8

Mty Ch 9

Ch MY 17

Tiempo total: 102

['MY', 'Bac', 'Pque', 'Teo', 'PpV', 'IM', 'Maz', 'LP', 'X', 'Mty', 'Ch']

MY Bac 4

Bac Pque 6

Pque Teo 11

Teo PpV 3

PpV IM 15

IM Maz 6

Maz LP 14

LP X 26

X Mty 8

Mty Ch 9

Ch MY 17

Tiempo total: 102

['LP', 'Maz', 'IM', 'Teo', 'PpV', 'X', 'Mty', 'Pque', 'Bac', 'MY', 'Ch']

LP Maz 14

Maz IM 6

IM Teo 9

Teo PpV 3

PpV X 6

X Mty 8

Mty Pque 19

Pque Bac 6

Bac MY 4

MY Ch 17

Ch LP 24

Tiempo total: 112

['LP', 'Maz', 'IM', 'Teo', 'PpV', 'X', 'Mty', 'Pque', 'Bac', 'MY', 'Ch']

LP Maz 14

Maz IM 6

IM Teo 9

Teo PpV 3

PpV X 6

X Mty 8

Mty Pque 19

Pque Bac 6

Bac MY 4

MY Ch 17

Ch LP 24

Tiempo total: 112

['Pque', 'Teo', 'PpV', 'IM', 'Maz', 'LP', 'X', 'Mty', 'Ch', 'Bac', 'MY']

Pque Teo 11

Teo PpV 3

PpV IM 15

IM Maz 6

Maz LP 14

LP X 26

X Mty 8

Mty Ch 9

Ch Bac 16

Bac MY 4

MY Pque 7

Tiempo total: 112

Analizando los 10 casos podemos decir que aproximadamente el tiempo mínimo con el que podemos recorrer todos los lugares en el grafo es de 102 horas

Pero aún nos falta analizar a nuestro vecino más cercano así que no podemos decir que esas es la hora mínima final

**Vecino más cercano**

Para el cecino más cercano utilizamos la siguiente función la cual nos dio como resultado un camino con el cual se hacen menos el tiempo en recórrelo que en los casos que realizamos

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

['MY', 'Bac', 'Pque', 'PpV', 'Teo', 'Ch', 'X', 'Mty', 'Maz', 'IM', 'LP']

MY Bac 4

Bac Pque 6

Pque PpV 11

PpV Teo 3

Teo Ch 1

Ch X 6

X Mty 8

Mty Maz 9

Maz IM 6

IM LP 21

Tiempo total: 75

En conclusión, podemos decir que con las 10 pruebas iniciales que se realizaron con el algoritmo kruskal no son suficientes para poder calcular un recorrido que nos dé el menor tiempo posible, así que para poder encontrarlo se tuvo que realizar con el algoritmo del vecino más cercano y así encontrar el menor tiempo posible para recorrer nuestros lugares