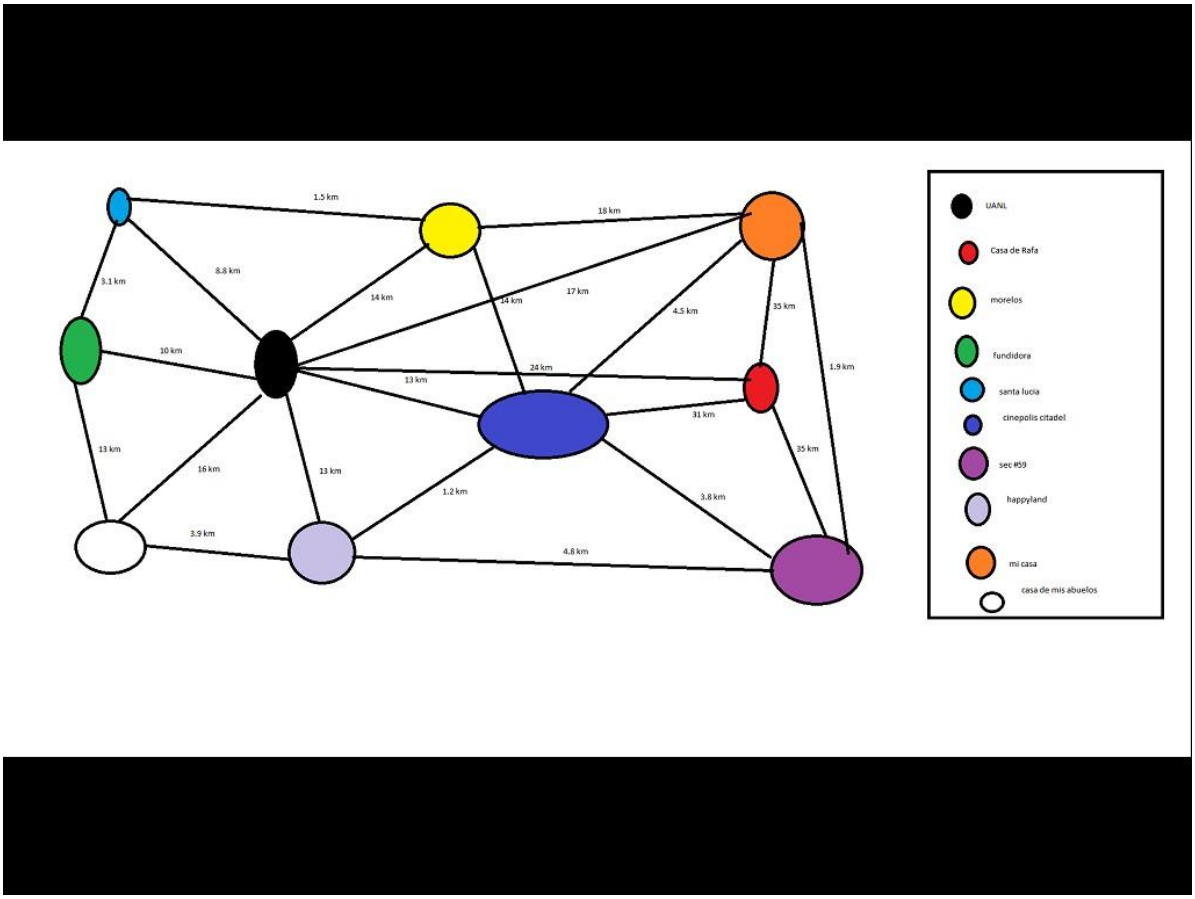
# UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
# FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

## MATEMATIZAS COMPUTACIONALES
## PIA


Evelyn Esmeralda Guerra Obregón
**Alumna:**

**Martes 22 de mayo de 2018**

Para iniciar con este trabajo primero estableceré el grafo con el que se va a trabajar representado por lugares que visito regularmente (vértices) y la distancia que existe entre dichos lugares (aristas).




| Letra | Lugar |
|---|---|
| **N** | UANL(FCFM) |
| **R** | Casa de Rafa (mi novio) |
| **Af** | Cinepolis Citadel |
| **V** | Fundidora |
| **B** | Casa de mis abuelos |
| **Na** | Mi casa |
| **A** | Calle Morelos |
| **G** | Happyland |
| **Ac** | Santa Lucia |
| **M** | Secundaria #59 |

***El nodo más cercano***

Consiste en tomar un lugar al azar y visitar a su ciudad vecina más cercana pero que no haya sido visitada anteriormente y así hasta visitar todos los lugres de manera que todas las opciones sean las más cercanas al punto donde se encuentra.

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

Resultados:

| El mejor camino fue: | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| R | Na | R | R | Na | R | Na | Na | R | Na |
| AF | M | AF | AF | M | AF | M | M | AF | M |
| G | Af | G | G | Af | G | Af | Af | G | Af |
| B | G | B | B | G | B | G | G | B | G |
| V | B | V | V | B | V | B | B | V | B |
| Ac | V | Ac | Ac | V | Ac | V | V | Ac | V |
| A | Ac | A | A | Ac | A | Ac | Ac | A | Ac |
| N | A | N | N | A | N | A | A | N | A |
| Na | N | Na | Na | N | Na | N | N | Na | N |
| M | R | M | M | R | M | R | R | M | R |

***El problema del agente viajero***

En el problema del agente viajero el objetivo es encontrar un recorrido completo que conecte todos los nodos de una red, visitándolos tan solo una vez y volviendo al punto de partida, y que además minimice la distancia total de la ruta, o el tiempo total del recorrido.

Utilizando el algoritmo de kruskal, el cual se encarga de encontrar el valor de la suma de todas las distancias entre las ciudades de nuestro grafo.

```
k=g.kruskal()
print(k)
mejor=-1
camino=[]
for r in range (10)
        va=random.choise(list(k.v))
        dfs=k.DFS(va)
        peso=0
        for i in range(lend(dfs)-1):
                peso+-g.E[(dfs[i],dfs[i+1])]
        peso+-g.E[(dfs[-1],dfs[0])]
        print("nodo inicial",va)
        for i in range(len(dfs)-1):
                print("De",dfs[i],"\ta",dfs[i+1],"\tla distancia es",g.E[(dfs[i],dfs[i+1])])
        print("De",dfs[-1],"\ta",dfs[0],"tla distancia es",g.E[(dfs[-1],dfs[0])])

        print("Distancia total:,peso,"\n")
        if mejor==-1 or
                mejor>peso:
                camino=dfs
print("la mejor ruta es:")
for k in camino:
        print(k,'->')
print(camino[0])
print("nCon distancia de", mejor)
```

Resultado

R -> Af -> G -> B -> V -> Ac -> A -> N -> Na -> M -> R

Con una distancia total de 121.6 km