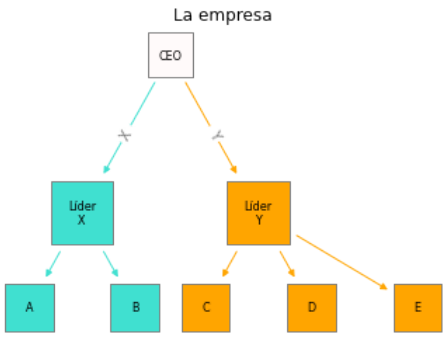
## Práctica de grafos con Networkx

1. Supongamos un organigrama que muestra la estructura jerárquica de una organización y las relaciones entre los empleados en los diferentes niveles y/o áreas dentro de ella. Consideremos que la empresa tiene 2 equipos X e Y trabajando dentro de ella. En total hay ocho empleados en la empresa: un CEO, 2 líderes de equipo, 2 empleados en el equipo X y 3 empleados en el equipo Y. Si cada nodo tuviera como máximo 2 nodos secundarios, habría sido un árbol binario, en este caso es ternario. Construimos el organigrama:

In [1]:
```
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

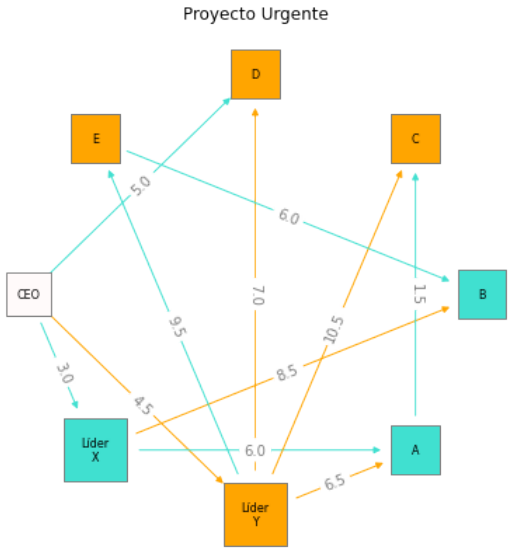
In [2]:
```
G = nx.DiGraph()
nodos = np.arange(0, 8).tolist()

G.add_nodes_from(nodos)
G.add_edges_from([(0,1),(0,2),(1,3),(1,4),(2,5),(2,6),(2,7)])
pos={0:(10.0,10.0),1:(7.5,7.5),2:(12.5,7.5),3:(6.0,6.0),4:(9.0,6.0),5:(11.0,6.0),6:(14.0,6.0),7:(17.0,6.0)}
labels={0:"CEO",1:"Líder\nX",2:"Líder\nY",3:"A", 4:"B",5:"C",6:"D",7:"E"}

colors = ["snow","turquoise","orange","turquoise","turquoise","orange","orange","orange"]

edge_colors = ["turquoise","orange","turquoise","turquoise","orange","orange","orange"]

sizes = [1000, 2000, 2000, 1200, 1200, 1200, 1200, 1200]

nx.draw_networkx(G, pos=pos, labels=labels, arrows=True,
                 node_shape = "s", node_size = sizes,
                 node_color = colors,
                 edge_color = edge_colors,
                 edgecolors = "gray",
                 font_size=8)

plt.rcParams["figure.figsize"]=[7,7]
nx.draw_networkx_edge_labels(G, pos = pos, edge_labels={(0,1):'X', (0,2):'Y'}, font_color='gray')
plt.title("La empresa")
plt.axis('off')
plt.show()
```




Luego, a raíz de una entrega urgente de proyecto, todos se involucran en el mismo con una carga de horas de trabajo estimada, coordinando la dirección de entrega de cada parte a otra, según una planificación:

In [11]:

```
G = nx.DiGraph()
nodos = np.arange(0, 8).tolist()

G.add_nodes_from(nodos)
G.add_edges_from([(0,1),(0,2),(1,3),(1,4),(2,5),(2,6),(2,7),(3,5),(7,4),(0,6),(2,3)])

labels={0:"CEO",1:"Líder\nX",2:"Líder\nY",3:"A", 4:"B",5:"C",6:"D",7:"E"}

colors = ["snow","turquoise","orange","turquoise","turquoise","orange","orange","orange"]

edge_colors = ["turquoise","orange","turquoise","turquoise","orange","orange","orange"]

sizes = [1000, 2000, 2000, 1200, 1200, 1200, 1200, 1200]

nx.draw_networkx(G, pos=nx.shell_layout(G), labels=labels, arrows=True,
                 node_shape = "s", node_size = sizes,
                 node_color = colors,
                 edge_color = edge_colors,
                 edgecolors = "gray",
                 font_size=8)

plt.rcParams["figure.figsize"]=[7,7]
nx.draw_networkx_edge_labels(G, pos=nx.shell_layout(G), edge_labels={(0,1):'3.0', (0,2):'4.5', (1,3):'6.0',(1,4):'8.5',
                                                                     (2,5):'10.5',(2,6):'7.0',(2,7):'9.5',(3,5):'1.5',
                                                                     (7,4):'6.0',(0,6):'5.0',(2,3):'6.5'}, font_color='gray')
plt.title("Proyecto Urgente")
plt.axis('off')
plt.show()
```




La disposición del grafo determinaría la siguiente tabla:

| | CEO | Líder X | Líder Y | A | B | C | D | E |
|---|---|---|---|---|---|---|---|---|
| CEO | | 3.0 | 4.5 | | | | 5.0 | |
| Líder X | | | | 6.0 | 8.5 | | | |
| Líder Y | | | | 6.5 | | 10.5 | 7.0 | 9.5 |
| A | | | | | | 1.5 | | |
| B | | | | | | | | |
| C | | | | | | | | |
| D | | | | | | | | |
| E | | | | | 6.0 | | | |

Calculando la centralidad

El grado de centralidad de un nodo es la parte del total de nodos a los que está conectado.
Un nodo con un alto grado de centralidad generalmente se considera altamente activo.
En G, el nodo 2 (Líder Y), tiene el grado más alto de centralidad ya que está conectado
a otros cuatro nodos.

In [4]:

```
nx.degree_centrality(G)
```

Out[4]:

```
{0: 0.42857142857142855,
 1: 0.42857142857142855,
 2: 0.7142857142857142,
 3: 0.42857142857142855,
 4: 0.2857142857142857,
 5: 0.2857142857142857,
 6: 0.2857142857142857,
 7: 0.2857142857142857}
```

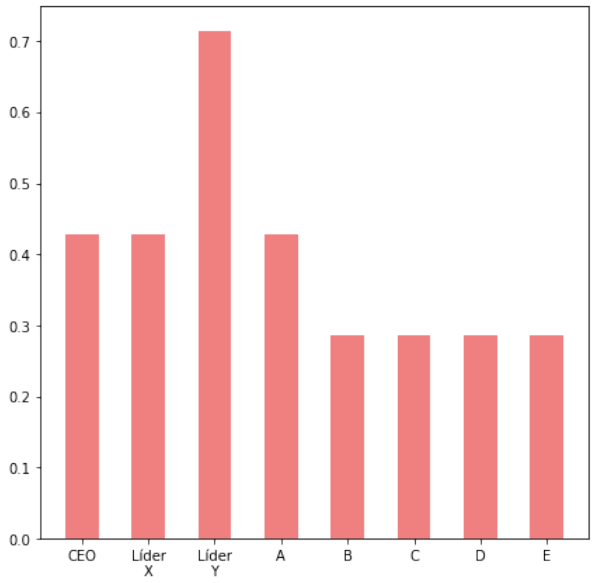
In [5]:

```
def valores(c):
    lista = []
    for elemento in c.values():
        lista.append(elemento)
    return lista

def etiquetas(c):
    lista = []
    for elemento in c.values():
        lista.append(elemento)
    return lista

val = valores(nx.degree_centrality(G))
eti = etiquetas(labels)
```

In [6]:

```
plt.bar(np.array(eti), np.array(val), width = 0.5 ,color='lightcoral')
```

Out[6]: <BarContainer object of 8 artists>




La centralidad de intermediación es una medida de cuántas veces un nodo en particular se encuentra en el camino más corto entre todos los pares de nodos en un gráfico. En G, el líder Y tiene la centralidad de intermediación más alta, seguido del líder X. Esto implica que los líderes de equipo actúan como puentes entre el CEO y el personal. Luego le siguen A y E. El CEO tiene una centralidad de intermediación cero porque no se encuentran entre dos nodos.

In [7]:

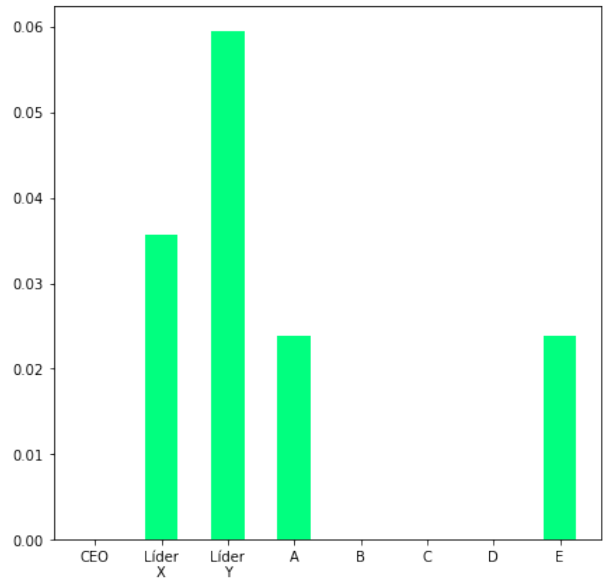
```
nx.betweenness_centrality(G)
```

Out[7]:

```
{0: 0.0,
 1: 0.03571428571428571,
 2: 0.05952380952380952,
 3: 0.023809523809523808,
 4: 0.0,
 5: 0.0,
 6: 0.0,
 7: 0.023809523809523808}
```

In [8]:
```
val = valores(nx.betweenness_centrality(G))
plt.bar(np.array(eti), np.array(val), width = 0.5 ,color='springgreen')
```

Out[8]: <BarContainer object of 8 artists>



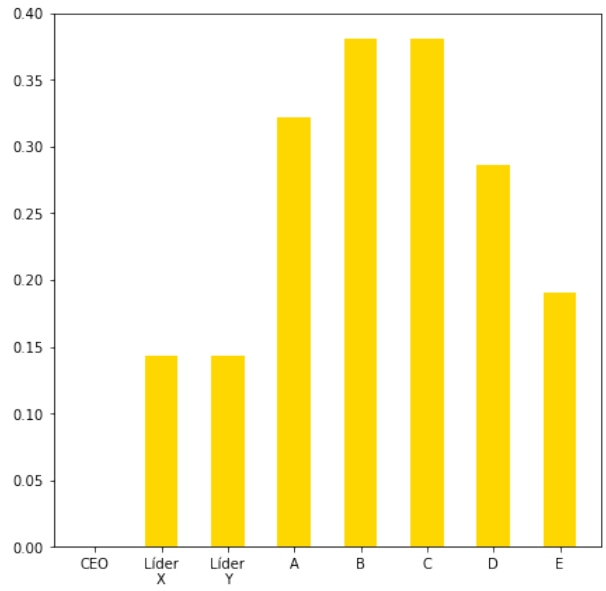

La centralidad de cercanía es una medida de la proximidad de un nodo a otros nodos. Se calcula como el promedio de la longitud de la ruta más corta desde el nodo hasta todos los demás nodos de la red. En el caso de la centralidad de cercanía, los nodos con valores más bajos tienen una centralidad más alta. Esto implica que el CEO y los líderes de equipo tienen más centralidad de cercanía en comparación con el personal.

In [9]:
```
nx.closeness_centrality(G)
```

Out[9]:
```
{0: 0.0,
 1: 0.14285714285714285,
 2: 0.14285714285714285,
 3: 0.3214285714285714,
 4: 0.38095238095238093,
 5: 0.38095238095238093,
 6: 0.2857142857142857,
 7: 0.19047619047619047}
```

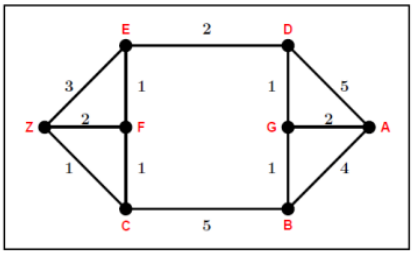
In [10]:
```
val = valores(nx.closeness_centrality(G))
plt.bar(np.array(eti), np.array(val), width = 0.5 ,color='gold')
```

Out[10]: <BarContainer object of 8 artists>

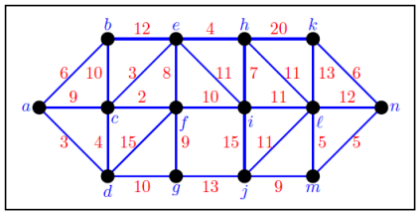



Cabe señalar que los valores de centralidad de intermediación y centralidad de cercanía cambiarían si G fuera un gráfico no dirigido, mientras que el grado de centralidad permanecería igual.

2 . Utilizar el algoritmo de Dijkstra para determinar en el grafo ponderado siguiente un camino de longitud mínima entre los vértices Z y A. Construir el grafo.




3. El plano muestra los puntos de conexión y las posibles líneas telefónicas en una urbanización. La zona quedará comunicada cuando dos puntos cualesquiera estén conectados. En rojo está indicado el precio de cada línea en miles de dólares. Calcular el diseño de la red más barata que conecte la zona.




Construir el grafo y calcular: Radio, diámetro, excentricidad, centro, periferia y densidad.

4. Sea G = (V,A) el grafo ponderado de 8 vértices, los cuales etiquetamos de A a H que representan poblaciones entre las cuales se implementará un medio de transporte. Los valores representan cantidad de combustible a utilizar. Determinar qué cantidad de combustible se necesitará utilizar en una ruta que conecte las poblaciones A y H, corresponde encontrar el camino más corto de A a H.

| A | B | C | D | E | F | G | H | Vértice | Arista |
|---|---|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - | A | - |
| - | 4 | - | - | 5 | - | - | - | B | AB |
| - | - | 8 | - | 5 | - | - | - | E | AE |
| - | - | 8 | - | - | 6 | 9 | - | F | EF |
| - | - | 8 | 7 | - | - | 8 | 11 | D | FD |
| - | - | 8 | - | - | - | 8 | 11 | C | BC |
| - | - | - | - | - | - | 8 | 11 | G | FG |
| - | - | - | - | - | - | - | 11 | H | FH |

5. Se considera el grafo ponderado G definido por la siguiente tabla, donde los vértices representan ciudades y las aristas representan rutas existentes entre las poblaciones. Los pesos indican longitudes en Kms.

| | *A* | *B* | *C* | *D* | *E* | *F* | *G* | *H* | *I* |
|---|---|---|---|---|---|---|---|---|---|
| *A* | | 20 | | | | 34 | | | 45 |
| *B* | | | 20 | | | 10 | | | 26 |
| *C* | | | | 28 | | | | | 22 |
| *D* | | | | | | | 18 | 19 | 13 |
| *E* | | | | | | 22 | 12 | 25 | |
| *F* | | | | | | | 30 | | 12 |
| *G* | | | | | | | | 16 | 14 |
| *H* | | | | | | | | | 32 |

a. Usando el algoritmo apropiado, calcular la distancia más cortas desde A a las restantes poblaciones y especificar cuáles son dichas distancias.

b. Se ha construido una nueva ruta entre las poblaciones B y G de forma que ahora, la distancia entre A y H es de 68 Kms. Determinar cuál es el peso que le corresponde a la arista {B,G}.

6. Dos amigos, Ana y Pedro, quieren visitar la ciudad de Praga, en la que los lugares turísticos de mayor importancia, las rutas entre los mismos y las distancias en kilómetros, vienen dados por la siguiente tabla:

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| A | | 12 | | 6 | | 5 | | 4 | |
| B | 12 | | 7 | | | | 8 | | 2 |
| C | | 7 | | 7 | | | | 5 | |
| D | 6 | | 7 | | 2 | | | | 1 |
| E | | | | 2 | | 3 | | | |
| F | 5 | | | | 3 | | 6 | | 15 |
| G | | 8 | | | | 6 | | 3 | |
| H | 4 | | 5 | | | | 3 | | 5 |
| I | | 2 | | 1 | | 15 | | 5 | |

Dibujar el grafo asociado al problema y responder: Cuál es la ciudad más lejana?

7. En la siguiente tabla se muestran las conexiones entre las computadoras de los 12 empleados de una oficina (en cada cuadro se muestra la longitud del cable que une las correspondientes computadoras)

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | 2 | 6 | | | | | | | | 7 | |
| B | | | | 7 | | 11 | 13 | | | | | |
| C | | | | 2 | | | 12 | | | | | |
| D | | | | | | | | | 14 | 12 | | |
| E | | | | | | 4 | | | | | | 14 |
| F | | | | | | | | | | 10 | | |
| G | | | | | | | | | 8 | | | |
| H | | | | | | | | | 3 | | | |
| J | | | | | | | | | | | | 9 |

a. El jefe decide que deben trabajar entre pares, formados de la siguiente manera: A con B, C con D, etc.  Es posible?

b. Obtener, la mínima distancia en metros de cable entre las computadoras A y J y el camino mínimo entre ellas.

8. En todos los casos dados, obtener la matriz de adyacencia e incidencia.