BUSQUEDA EN PROFUNDIDAD EN UNA LISTA DE ADYACENCIA

Función DFS para recorrer una LA

In [ ]:
```
import graphviz as gv
```

In [ ]:
```
def dfs(G, s):
  n = len(G)
  path = [-1]*n
  visited = [False]*n

  def _dfs(u):
    visited[u] = True
    for v in G[u]:
      if not visited[v]:
        path[v] = u
        _dfs(v)

  _dfs(s)
  return path
```

Generamos una LA segun el siguiente texto:

In [ ]:
```
%%file 03a.la
1 4
0 3 5 6
4 5 6
1 7
0 2 6 7
1 2
1 2 4
3 4
```

```
Writing 03a.la
```

Cargamos la LA generada desde el archivo 03a.la a un arreglo tipo numpy en la variable G1

In [ ]:
```
G1 = []
with open("03a.la") as f:
  for line in f:
    if line == "-\n":
      G1.append([])
    else:
      G1.append([int(x) for x in line.split()])

print(G1)
```

```
[[1, 4], [0, 3, 5, 6], [4, 5, 6], [1, 7], [0, 2, 6, 7], [1, 2], [1, 2, 4], [3, 4]]
```

In [ ]:
```
path = dfs(G1, 5)
path
```

Out[ ]:
```
[1, 5, 4, 7, 0, -1, 2, 4]
```

Creamos una función para dibujar el grafo utilizando la libreria gv, indicando si ser un grafo dirigido o no. Si le pasamos una lista con una ruta, debera colorear dicha ruta (path).

In [ ]:
```
def drawG_al(G, directed=False, path=[]):
  graph = gv.Digraph("di-anyname") if directed else gv.Graph("anyname")
  n = len(G)
  added = set()
  for v, u in enumerate(path):
    if u != -1:
      graph.edge(str(u), str(v), dir="forward", penwidth="2", color="orange")
      added.add(f"{u},{v}")
      added.add(f"{v},{u}")
  for u in range(n):
    for v in G[u]:
      if not directed and not f"{u},{v}" in added:
        added.add(f"{u},{v}")
        added.add(f"{v},{u}")
        graph.edge(str(u), str(v))
      elif directed:
        graph.edge(str(u), str(v))
  return graph
```

Visualizamos el grafo no dirigido del arreglo G1

In [ ]:
```
drawG_al(G1, path=path)
```

Out[ ]: