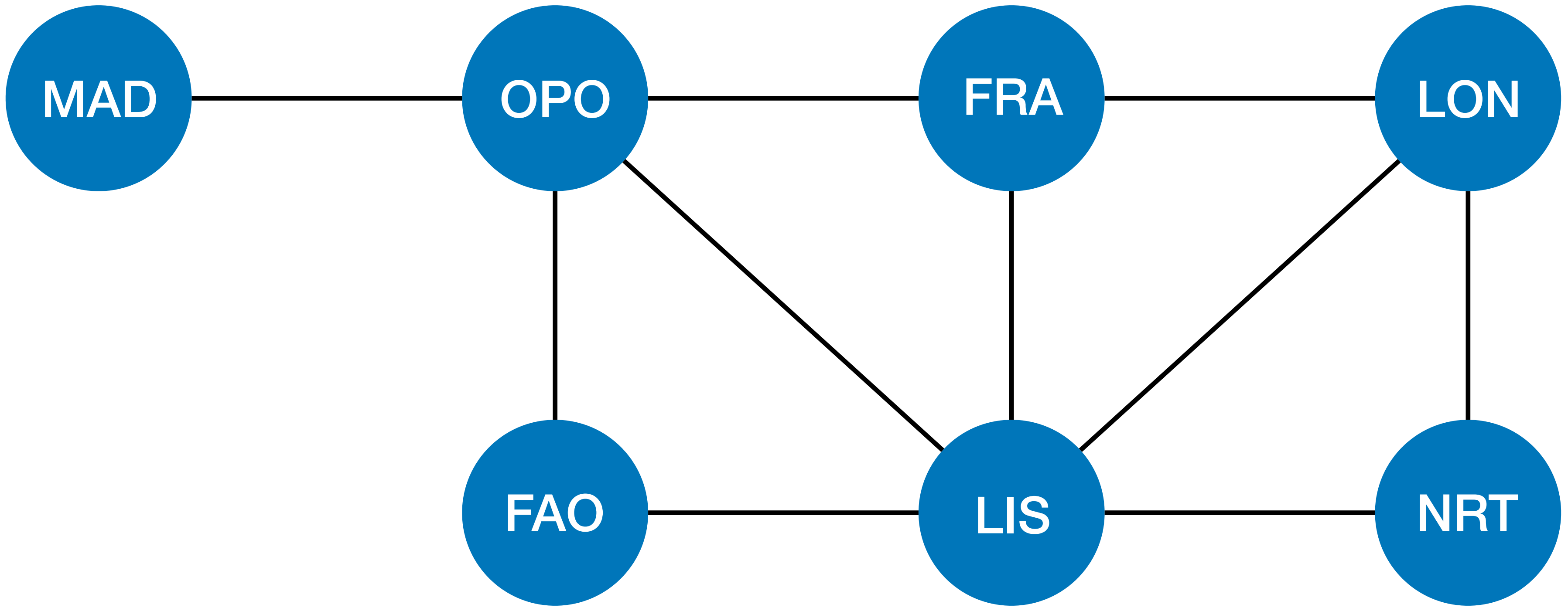
Laboratórios de Algoritmia II

Algoritmos de grafos

Grafos não pesados

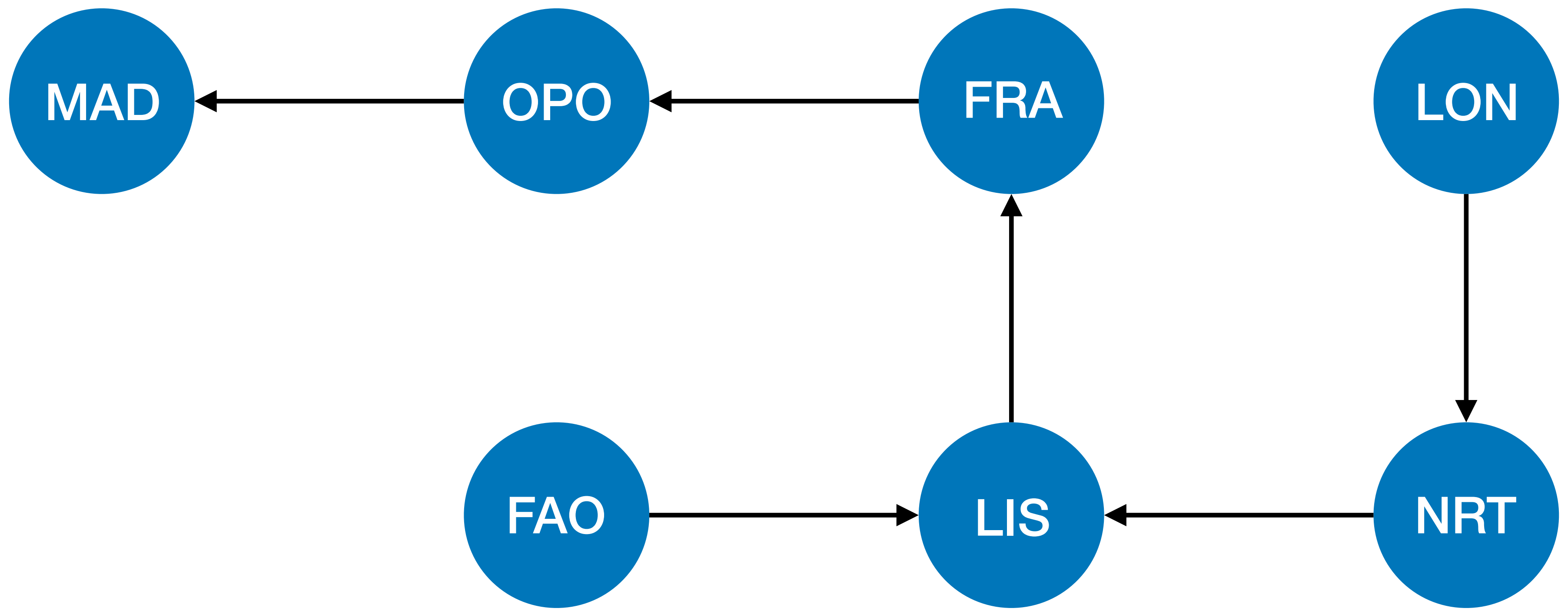

MAD
OPO
FRA
LON
FAO
LIS
NRT

Representação

```
exemplo = [("OPO","LIS"),("OPO","FAO"),
           ("LIS","FAO"),("MAD","OPO"),
           ("LIS","LON"),("FRA","OPO"),
           ("LIS","NRT"),("LON","NRT"),
           ("LON","FRA"),("LIS","FRA")]

def build(arestas):
    adj = {}
    for o,d in arestas:
        if o not in adj:
            adj[o] = set()
        if d not in adj:
            adj[d] = set()
        adj[o].add(d)
        adj[d].add(o)
    return adj
```

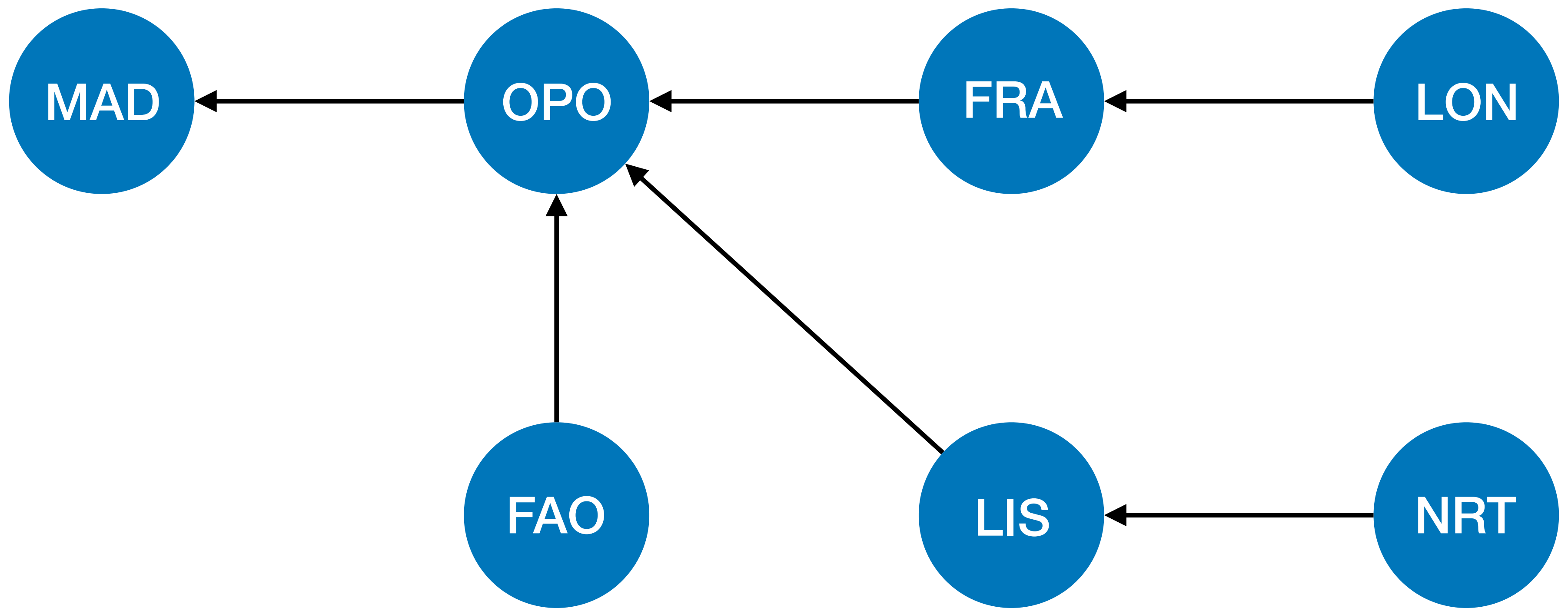
Travessia em profundidade

```
def dfs(adj,o):
    return dfs_aux(adj,o,set(),{})

def dfs_aux(adj,o,vis,pai):
    vis.add(o)
    for d in adj[o]:
        if d not in vis:
            pai[d] = o
            dfs_aux(adj,d,vis,pai)
    return pai

dfs(build(exemplo),”MAD")
```


MAD
OPO
FRA
LON
FAO
LIS
NRT

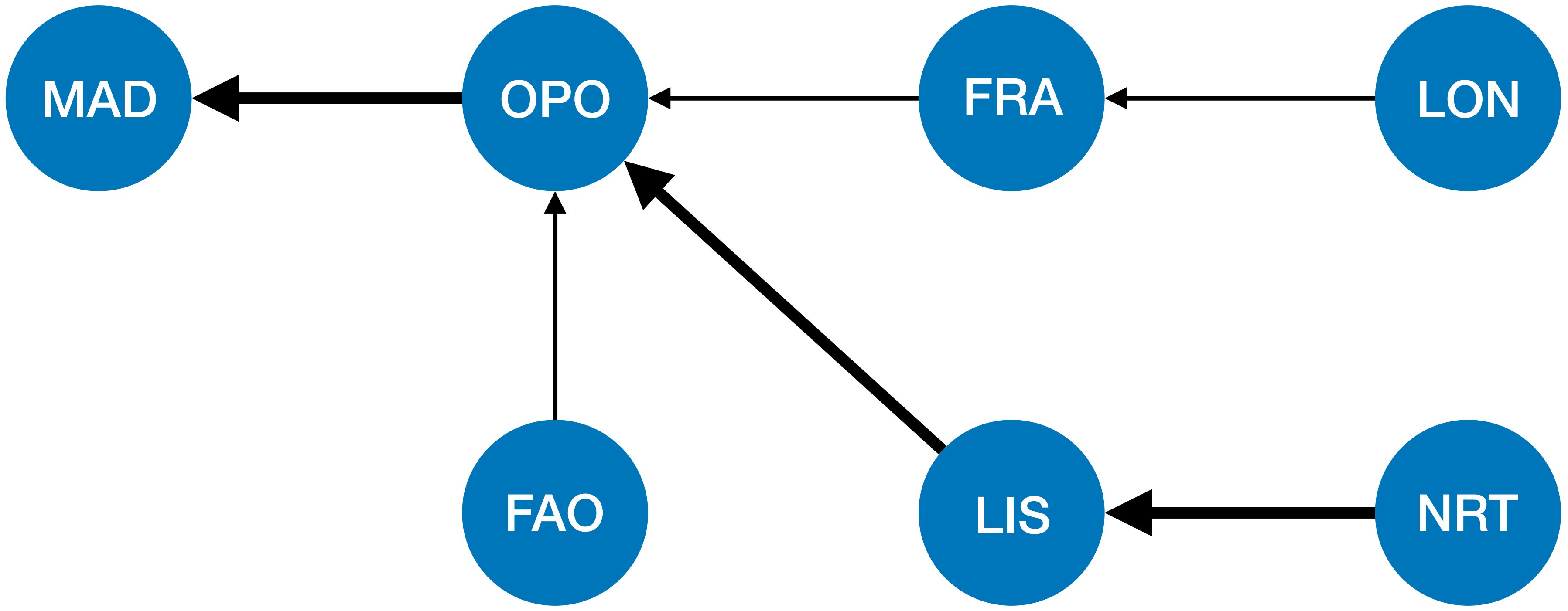
Travessia em largura

```
def bfs(adj,o):
    pai = {}
    vis = {o}
    queue = [o]
    while queue:
        v = queue.pop(0)
        for d in adj[v]:
            if d not in vis:
                vis.add(d)
                pai[d] = v
                queue.append(d)
    return pai

bfs(build(exemplo),”MAD")
```

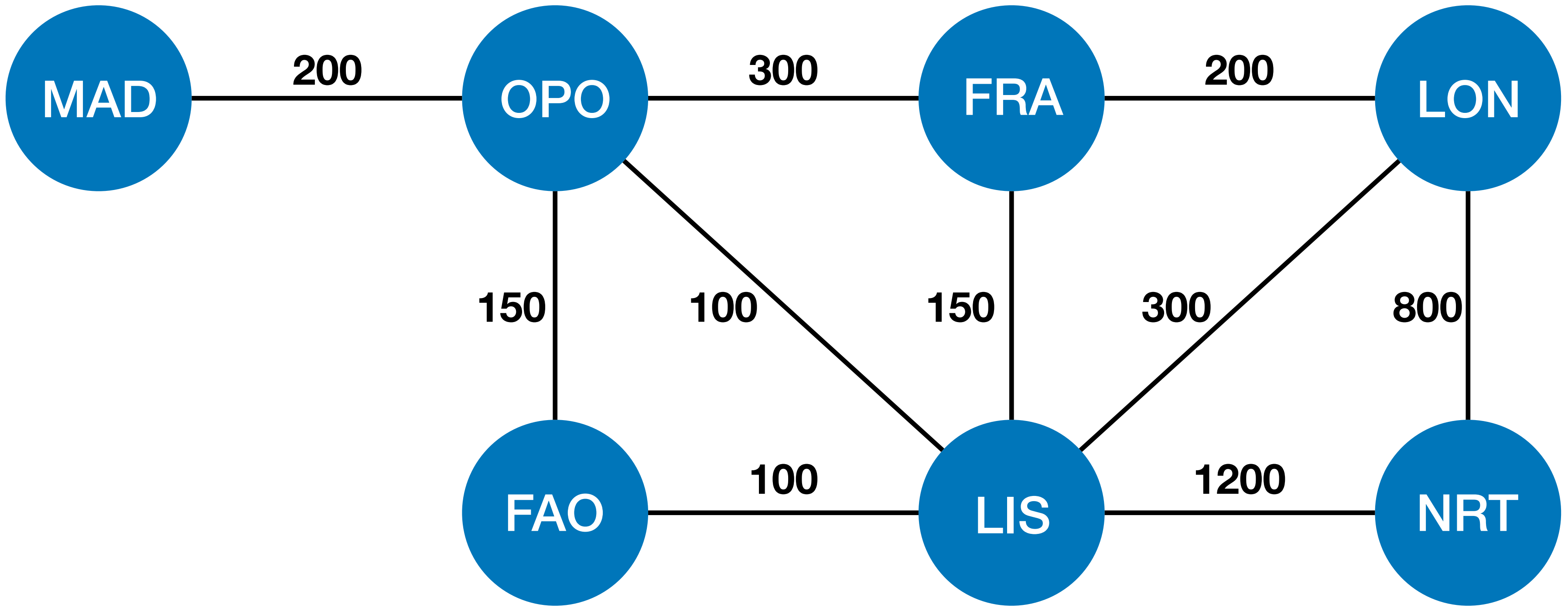

MAD
OPO
FRA
LON
FAO
LIS
NRT

Caminho mais curto

```
def caminho(adj,o,d):
    pai = bfs(adj,o)
    caminho = [d]
    while d in pai:
        d = pai[d]
        caminho.insert(0,d)
    return caminho

caminho(build(exemplo),”MAD","NRT")
```


MAD
OPO
FRA
LON
FAO
LIS
NRT

Grafos pesados

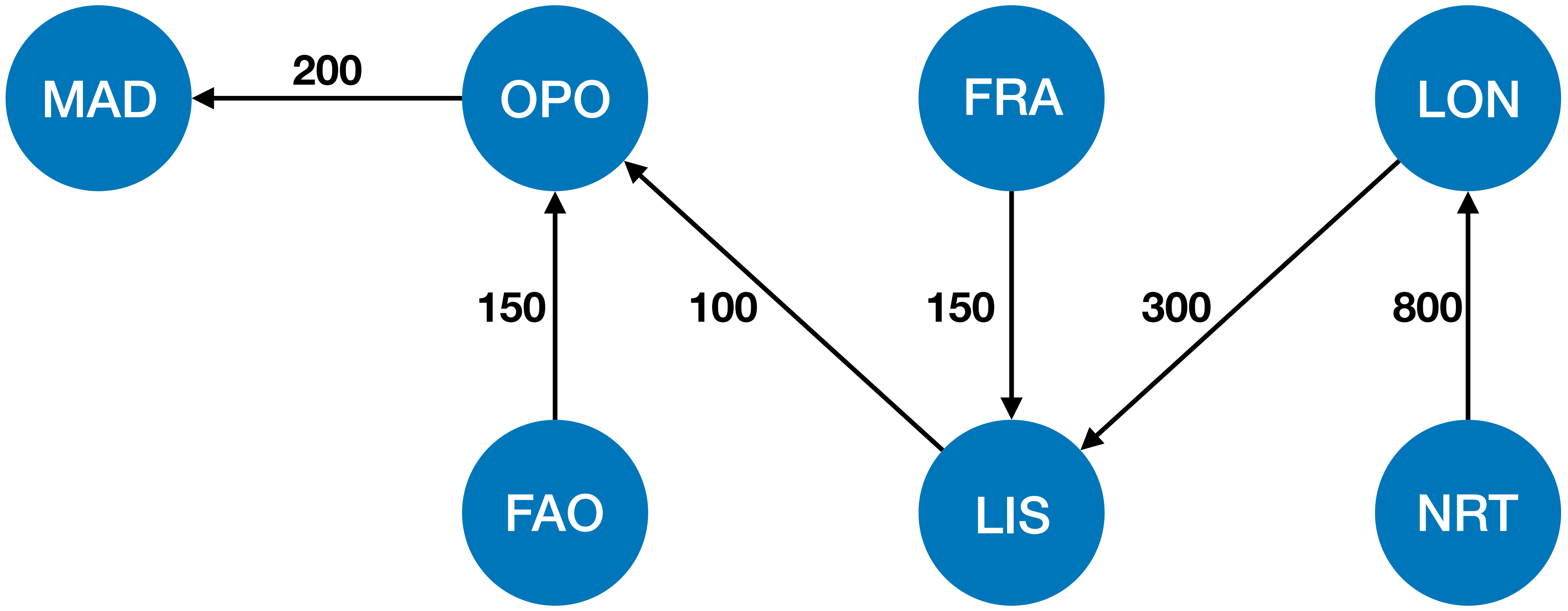

MAD
OPO
FRA
LON
FAO
LIS
NRT
200
300
200
150
100
150
300
800
100
1200

Representação

```
exemplo = [("OPO","LIS",100), ("OPO","FAO",150),
           ("LIS","FAO",100), ("MAD","OPO",200),
           ("LIS","LON",300), ("FRA","OPO",300),
           ("LIS","NRT",1200),("LON","NRT",800),
           ("LON","FRA",200), ("LIS","FRA",150)]

def build(arestas):
    adj = {}
    for o,d,p in arestas:
        if o not in adj:
            adj[o] = {}
        if d not in adj:
            adj[d] = {}
        adj[o][d] = p
        adj[d][o] = p
    return adj
```

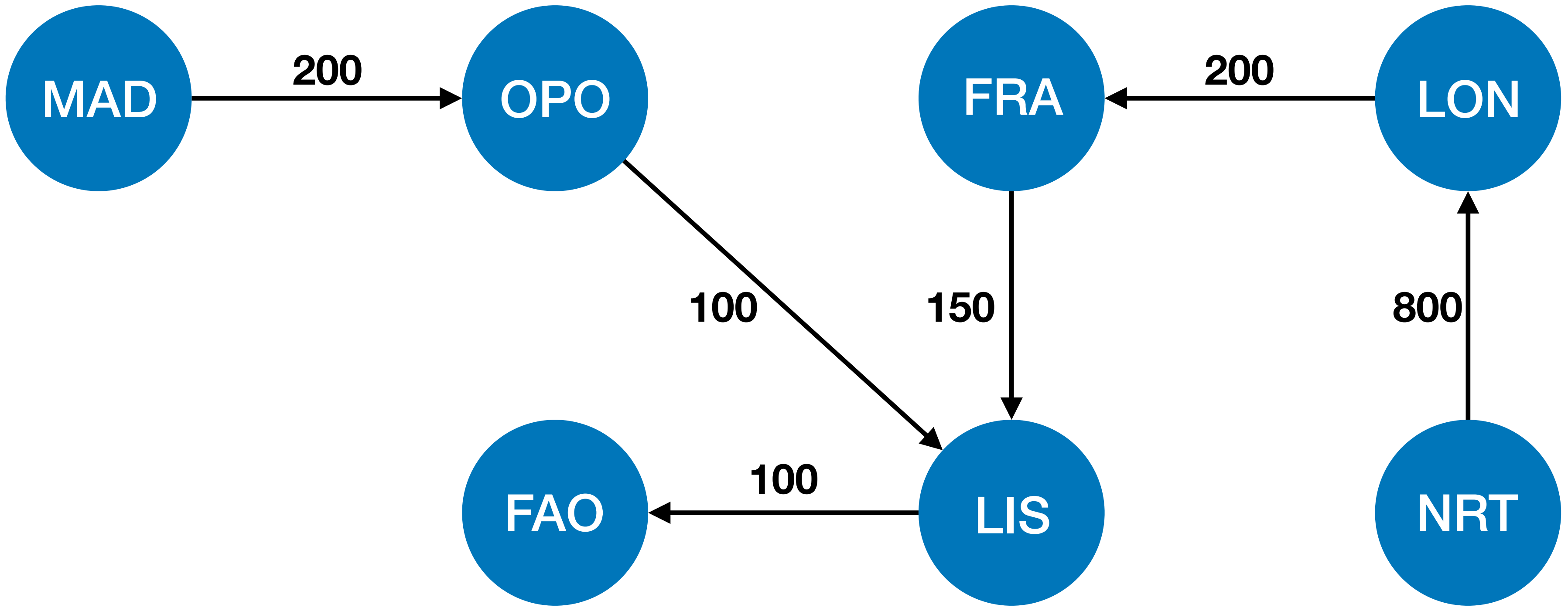
Dijkstra

```python
def dijkstra(adj,o):
    pai = {}
    dist = {}
    dist[o] = 0
    orla = {o}
    while orla:
        v = min(orla,key=lambda x:dist[x])
        orla.remove(v)
        for d in adj[v]:
            if d not in dist:
                orla.add(d)
                dist[d] = float("inf")
            if dist[v] + adj[v][d] < dist[d]:
                pai[d] = v
                dist[d] = dist[v] + adj[v][d]
    return pai

dijkstra(build(exemplo),"MAD")
```


MAD
OPO
FRA
LON
FAO
LIS
NRT
200
150
100
150
300
800

Prim

```
def prim(adjsorted):
    pai = {}
    cost = {}
    o = sorted(adj)[0]
    cost[o] = 0
    orla = {o}
    while orla:
        v = min(orla,key=lambda x:cost[x])
        orla.remove(v)
        for d in adj[v]:
            if d not in cost:
                orla.add(d)
                cost[d] = float("inf")
            if adj[v][d] < cost[d]:
                pai[d] = v
                cost[d] = adj[v][d]
    return pai

prim(build(exemplo))
```


MAD
OPO
FRA
LON
FAO
LIS
NRT
200
200
100
150
800
100

Floyd-Warshall

```
def fw(adj):
    dist = {}
    for o in adj:
        dist[o] = {}
        for d in adj:
            if o == d:
                dist[o][d] = 0
            elif d in adj[o]:
                dist[o][d] = adj[o][d]
            else:
                dist[o][d] = float("inf")
    for k in adj:
        for o in adj:
            for d in adj:
                if dist[o][k] + dist[k][d] < dist[o][d]:
                    dist[o][d] = dist[o][k] + dist[k][d]
    return dist

fw(build(exemplo))
```

	OPO	LIS	FAO	MAD	LON	FRA	NRT
OPO	0	100	150	200	400	250	1200
LIS	100	0	100	300	300	150	1100
FAO	150	100	0	350	400	250	1200
MAD	200	300	350	0	600	450	1400
LON	400	300	400	600	0	200	800
FRA	250	150	250	450	200	0	1000
NRT	1200	1100	1200	1400	800	1000	0