A Data-driven approach to find links between communities using Tally's API

In [7]:
```python
# import libraries

import requests
import json
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```python
# Tally API and Api-key

url = 'https://api.tally.xyz/query'
headers = {"Api-key": 'a6a335a19f8b493c976e78566f7e1795dee68409b0e7dbd2ef98247edaa
```

In [4]:
```python
# Function taking as input
# 1) lim: first: representing the first indexed proposal
# 2) off: skip: represennting the skipped indexed proposal
# Returns the query to be run

def query_(lim,off):
    query = """query Governors {
      governors (

        chainIds: "eip155:1",
        pagination: {
          limit:""" + str(lim) + """,
          offset:""" + str(off) + """
        }
      )
      {
        name
        delegates {
          account {
            address
          }
        }
      }
    }"""
    return query
```

In [10]:
```
# for loop querying all on-chain proposals of 5 DAOs per loop

df = pd.DataFrame()

for x in range(5, 300, 5):

    r = requests.post(url, json={'query': query_(x,x-5)}, headers=headers)
    json_data = json.loads(r.text)

    space_lst = []
    member_lst = []

    index = 0

    for i in range(len(json_data['data']['governors'])):
        for j in range(len(json_data['data']['governors'][i]['delegates'])):
            space_lst.append(json_data['data']['governors'][i]['name'])
            member_lst.append(json_data['data']['governors'][i]['delegates'][j]['a

    df = df.append(pd.DataFrame([space_lst, member_lst]).T)

df.columns = ['DAO', 'Contributors']
df = df.drop_duplicates()
```

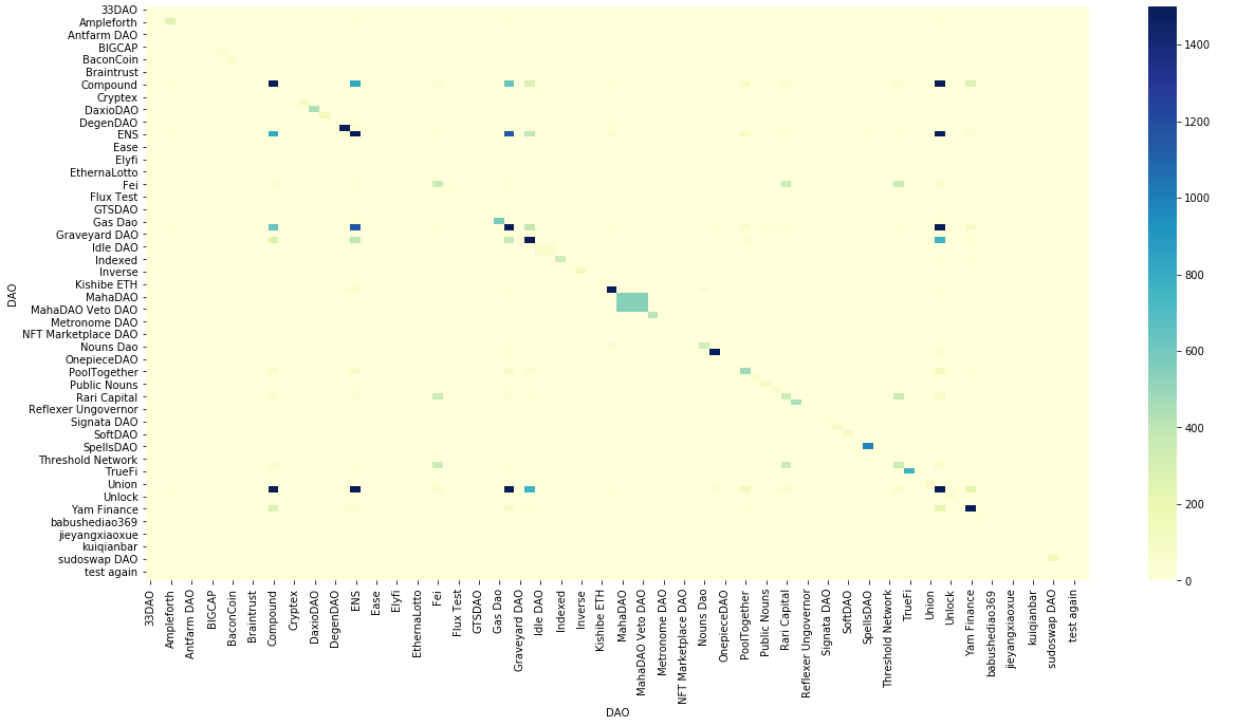
In [11]:
```
# dataframe containing contributors addresses to their DAO

df.head()
```

Out[11]:

	DAO	Contributors
0	SoftDAO	0x7882caC398FFFE6E76B9c2a576F875b39e0278b1
1	SoftDAO	0xa8e6D915816d5a9F15cf1C8D19Cafe1bddc98bbA
2	SoftDAO	0x304c161c3A71FbE5867022648fBc7d9aA994A02d
3	SoftDAO	0x74FC9058e5fc25c23C628ddfa9E2d95967d3f11f
4	SoftDAO	0x55A1CD90c5FB0DD3abE513f547Ed8068D960d3B4

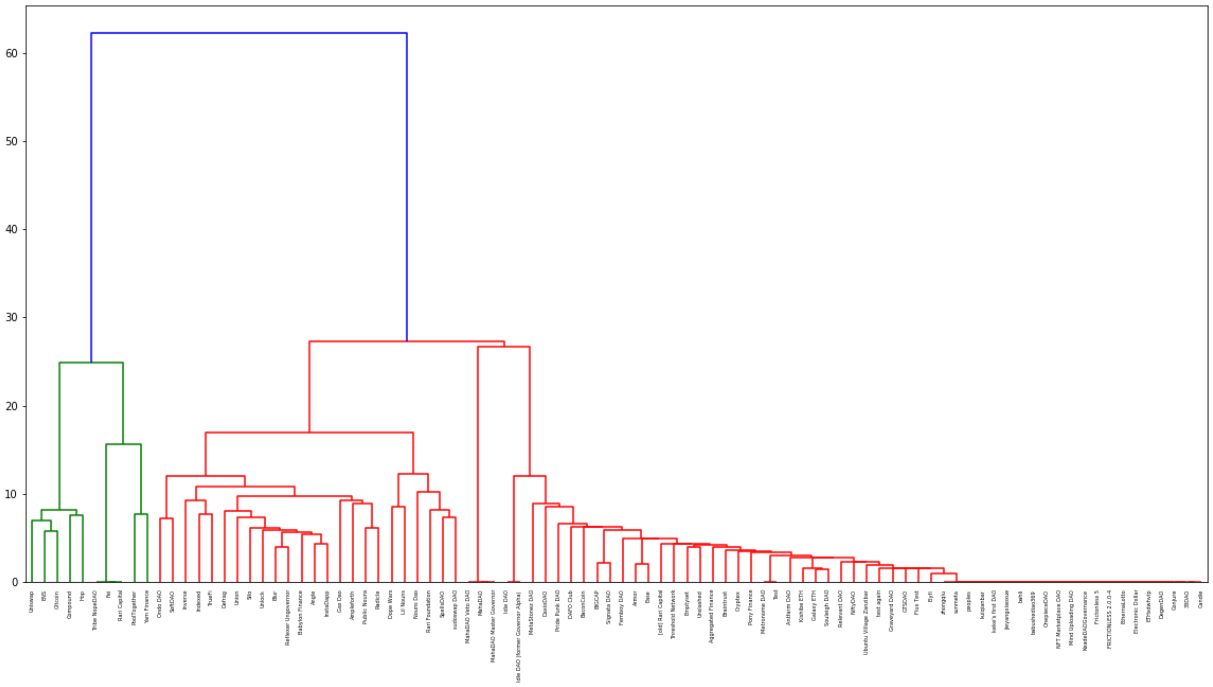
In [12]:

```
# Create a cross-tabulation table
cross_tab = pd.crosstab(index=df['DAO'], columns=df['Contributors'])

# Multiply the cross-tabulation table by its transpose to get the 2D matrix
matrix = cross_tab.dot(cross_tab.T)

# Plot
plt.figure(figsize=(20,10))

vmin = 0
vmax = 1500

# Create a heatmap using seaborn
sns.heatmap(matrix, cmap="YlGnBu", vmin=vmin, vmax=vmax)

# Show the plot
plt.show()
```



In [14]:
```
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage

z = linkage(np.log(matrix[np.log(matrix)>0]).fillna(0), method='ward', metric='euc

plt.figure(figsize=(20,10));

# Plot the dendrogram
dendrogram(z, labels=matrix.index);
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: RuntimeWarni
ng: divide by zero encountered in log
  after removing the cwd from sys.path.
```



Litterature of future work

https://www.pnas.org/doi/10.1073/pnas.122653799 (https://www.pnas.org/doi/10.1073/pnas.122653799)