Junho 2021/2022

ISCTE IUL
Instituto Universitário de Lisboa

ESTRUTURA DE DADOS E ALGORITMOS

Utilização de Grafos para estudo de comunidades na rede social Facebook

Docente:

Prof.ª Maria Pinto-Albuquerque

Grupo 21:

Diogo Freitas (104841); daafs@iscte-iul.pt

Inês Silva (104912); imcsa1@iscte-iul.pt

Pedro Machado (98601); pbmoo@iscte-iul.pt

EDA – Grafos

Introdução

O presente trabalho foi realizado no âmbito da Unidade Curricular Estrutura de Dados e Algoritmos e tem como objetivo relacionar uma das estruturas de dados mais comuns, os **Grafos**, com a análise de redes sociais, nomeadamente o **Facebook**.

Neste trabalho irá ser necessário, não só implementar um TAD Grafo, como também, usar o **algoritmo de Kruskal**, a fim de se obter uma árvore de cobertura mínima.

De seguida, iremos aplicar esse mesmo TAD Grafo a fim de estudar as comunidades, que se podem obter de um conjunto de dados, sobre *profiles* e *circles* (lista de amigos) no Facebook, para tal, iremos ter que:

- Desenvolver um algoritmo de agrupamento hierárquico (*clustering*);
- Criar os (sub)grafos para cada cluster;
- Usar o NetworkX para visionar os grafos criados e retirar conclusões.

1- Grafos

Informalmente, um grafo pode ser visto como um conjunto de pontos, intitulados de vértices. Estes vértices podem ter conexões com outros vértices chamadas de arestas, cada uma liga um par de pontos (extremidades) que a determina. A representação usual é feita por linhas (arestas) a ligar pontos do plano (vértices).

Os pontos dos grafos possuem alguma característica que os torna únicos, cada um guardando os seus respetivos dados; os vértices podem estar ou enumerados, ou possuírem algum nome, ou outra característica que os diferencie.

As arestas possuem um "peso" diferente para cada ligação, demonstrando que, existem tipos de ligação diferentes, com vértices, tanto finais como iniciais diferentes.

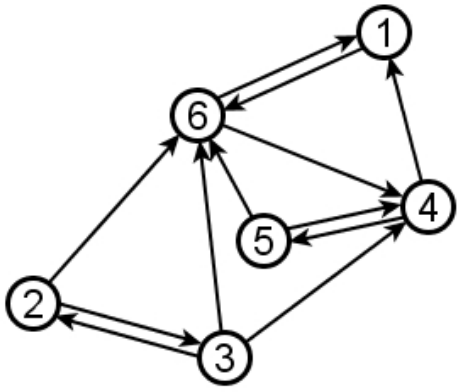


Figura 1 - Exemplo de um Grafo não pesado

Em relação á representação computacional, neste trabalho foi usado/criado grafos tendo em conta os **Mapas de adjacência**.

Esta implementação tem como base de que, no construtor, cada vértice está dentro de um dicionário, sendo este a key() do respetivo dicionário; cada um, possui o seu próprio dicionário como values(), podendo colocar nele, o respetivo vértice final a que está ligado, como key(), e o peso da Edge (ligação) como values().

Em baixo, poderemos ver uma imagem que representa o que foi dito anteriormente:

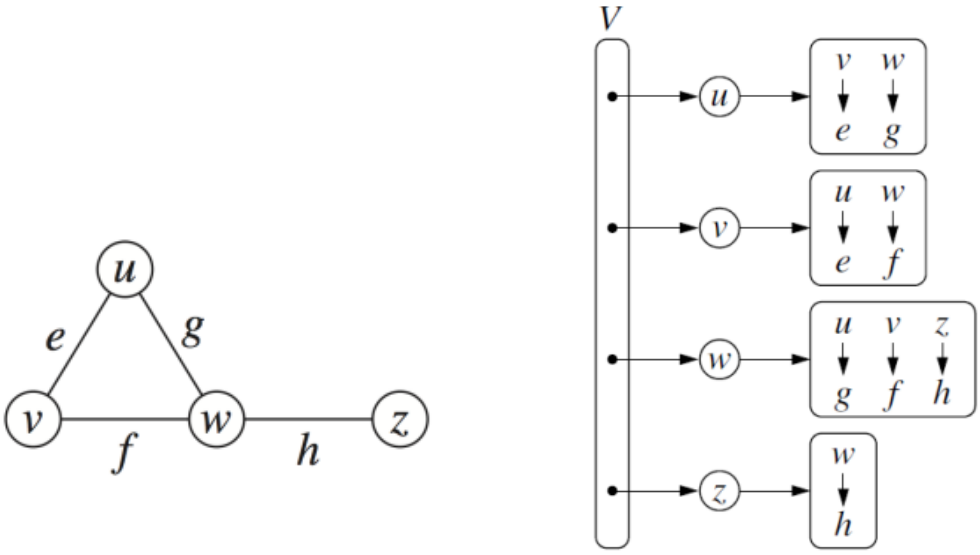


Figura 2 – Implementação do Grafo com um Mapa de Adjacências

Fase 1 – TAD Grafo e algoritmo de Kruskal

a) Implementação do TAD Grafo

Para a realização do Grafo, é necessário criar 2 classes anteriores, a fim de dar suporte na criação deste, as classes necessárias são:

1. A classe Vertex (A fim de criar os vértices)
2. A classe *Edge* (Com o objetivo de criar as arestas)

Classe Vertex

```
class Vertex:


    def __init__(self, x):
        self._elemento = x

    def __hash__(self):
        return hash(self._elemento)

    def __str__(self):
        return '{0}'.format(self._elemento)

    def __eq__(self, other):
        if not isinstance(other, type(self)):
            return NotImplemented
        return self._elemento == other._elemento

    @property
    def elemento(self):
        return self._elemento
```

Acima, podemos visionar a **class Vertex**. Esta possui um construtor, o **__init__**, que recebe o parâmetro do elemento (guardando assim o valor do vértice); a função **__hash__** retorna o valor de hash de um objeto, caso este tenha um, possibilitando a comparação das diferentes keys() de um dicionário; a função **__str__** permite que se faça prints de um objeto desta classe; a função **__eq__** permite que se compare objetos da mesma classe; a função elemento devolve o valor do vértice.

Classe Edge

```
class Edge:
    def __init__(self, origem, destino, peso):
        self._origem = origem
        self._destino = destino
        self._peso = peso

    @property
    def peso(self):
        return self._peso

    def __hash__(self):
        return hash((self._origem, self._destino))

    def __str__(self):
        return f'{self._origem} -> {self._destino} = {self._peso}'

    def endpoints(self):
        return (self._origem, self._destino)

    def opposite(self, vertice):
        return self._destino if vertice is self._origem else self._origem

    def mostra_edge(self):
        print(f'({self._origem}, {self._destino}')

    def __eq__(self, other):
        if not isinstance(other, type(self)):
            return NotImplemented
        return self._origem == other._origem and self._destino ==
other._destino
```

Acima, podemos visionar a **class Edge**. Esta possui um construtor, o **__init__**, que recebe como parâmetro o vértice de origem (origem), o vértice de destino (destino) e o peso da ligação (peso); a função peso devolve o peso da ligação; esta classe também possui a função **__hash__** e **__str__** e **__eq__** já referidos anteriormente;

Classe Graph

Método (assinatura)	Descrição
vertex_count()	Devolve a quantidade de vértices do grafo
vertices()	Devolve um iterador sobre os vértices
edge_count()	Devolve a quantidade de arestas do grafo
edges()	Devolve um iterador sobre as arestas do grafo
get_edge(u, v)	Devolve a aresta entre o vértice u e v, se existir
degree(v, out= True)	Para um grafo não orientado, devolve o grau do nó. Num digrafo, devolve o grau saída, se out=True, ou de entrada, se out=False
incident_edges(v, out=True)	Devolve um iterador sobre todas as arestas incidentes em v. Num digrafo, devolve um iterador sobre arestas que entram ou que saiem, consoante out
insert_vertex(x = None)	Insere no grafo e devolve um objeto vértice com x armazenado
insert_edge(u, v)	Cria no grafo e devolve o objeto aresta entre o vértice u e v (identificadores de vértices do grafo)
remove_vertex(v)	Remove o vértice que armazena v e actualiza arestas incidentes
remove_edge(e)	Remove a aresta pedida, se existir

Tabela 1 – Respetivas funções necessárias para o TAD Grafo

Na tabela 1 podemos visionar as funções necessárias para que a **class Graph** esteja realmente completa. Para que possamos adicionar os vértices, criados com a **class Vertex**, ao graph utilizamos a função **insert_vertex**, onde iremos dar o elemento á função; para adicionarmos os edges, utilizamos a função **insert_edge**, onde damos dois vértices, o inicial e o destino; as funções **remove_vertex** e **remove_edge** fazem o oposto das referidas anteriormente, respetivamente.

b) Algoritmo de Kruskal

O algoritmo de Kruskal é um algoritmo de árvore geradora mínima que recebe um grafo como entrada e encontra o subconjunto das arestas desse grafo, formando então uma árvore que incluí todos os vértices e os ordena por ordem crescente do seu valor, tentando, desta forma, realizar a menor soma possível dos seus *edges*. Este algoritmo só funciona caso os *Edges* do grafo possuam peso.

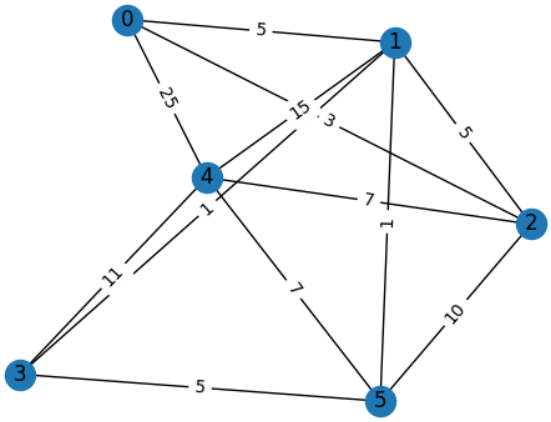


Figura 3 – Grafo Pesado

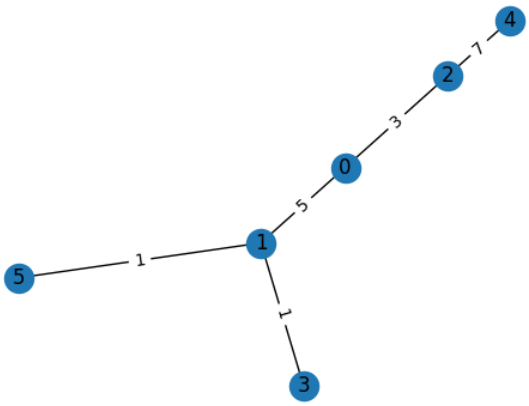


Figura 4 – Grafo Pesado com o algoritmo de Kruskal aplicado

Após aplicarmos o algoritmo de Kruskal no grafo Pesado (Figura 3), obtivemos o Grafo apresentado na Figura 4. Repara-se que este possui todos os vértices conectados, sem haver nenhum ciclo, com as *edges* de menor peso, tal e qual referido na definição do algoritmo de Kruskal. Ao somar-mos o peso das *Edges*, é notório a diferença das mesmas, antes de se aplicar o algoritmo de Kruskal, a soma das *Edges* era de 95 (Figura 3) e após o algoritmo de Kruskal, este valor passou para 17 (Figura 4).

Fase 2 – K-Clustering e as comunidades

a) Implementação do algoritmo de agrupamento (*clustering*) hierárquico.

Antes de prosseguir para explicação e implementação do algoritmo, é importante referir que, o nosso grupo, interpretou o peso das *edges* do ficheiro **facebook_netwrok.csv** como sendo a **distância entre os vértices**, ou seja, quanto menor o peso das *edges*, maior será a proximidade dos usuários (*Vertex*). Por termos interpretado desta maneira, na hora da realização do Algoritmo de K-Clustering, o grupo optou por eliminar os k *edges* mais pesado.

O K-Clustering consiste em, dado um grafo, neste caso, após aplicar o algoritmo de Kruskal, este faz com que os conjuntos se separem, agrupando os objetos em diferentes grupos, as chamadas comunidades, de modo a maximizar as distâncias entre os agrupamentos. Estas "distâncias" medem a falta de semelhança entre os dois objetos e está diretamente relacionada com o peso das *Edges*. Devido á nossa interpretação, quanto maior o peso das ligações, menor a semelhança, por outras palavras, menor será a proximidades dos mesmos. O "K" simboliza o número de ligações, com menor semelhança, que irão ser eliminadas.

Em baixo, é possível visualizar o algoritom K-Clustering, com k igual 2, aplicado ao grafo apresentado na Figura 3

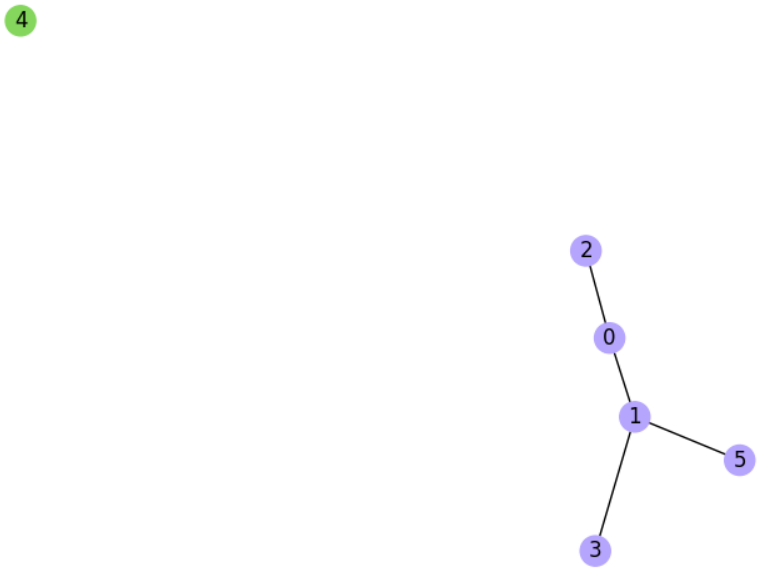
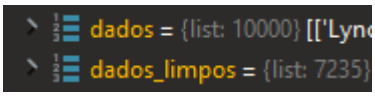


Figura 5 – Grafo Pesado com o algoritmo de K-Cluster, com k igual a 2

Ao analisar-mos o grafo da Figura 5, é possível verificar que existem duas "comunidades" diferentes, uma formada pelos vértices 5, 1, 3, 0, 2; e outra formada apenas pelo vértice 4. É notório que o edge que foi eliminado foi o que conectava o vértice 4 ao 2, tendo este um peso de sete, o maior peso quando comparado com os outros edges (Comparar com a Figura 4).

b) Criação dos (sub) grafos para o *ficheiro facebook_netwrok.csv*

Após verificarmos que o algoritmo K-Cluster estava a funcionar corretamente, fomos então para a aplicação do mesmo para o ficheiro *facebook_netwrok.csv*. Antes de começar a aplicar o algoritmo, seria necessário verificar os dados para ver se estes tinham de ser limpos ou não. Após uma breve visualização dos mesmos, foi possível reparar que existiam 10000 e alguns deles se repetiam e, para isso, seria necessário limpar os dados. O nosso grupo optou por, caso os 2 vértices tivessem ligados com *edges* diferentes, os pesos das *edges* seriam todos somados, e os vértices ficariam com apenas uma ligação. Após a limpeza dos dados, passamos de 10000 *edges* para 7235 *edges* diferentes.



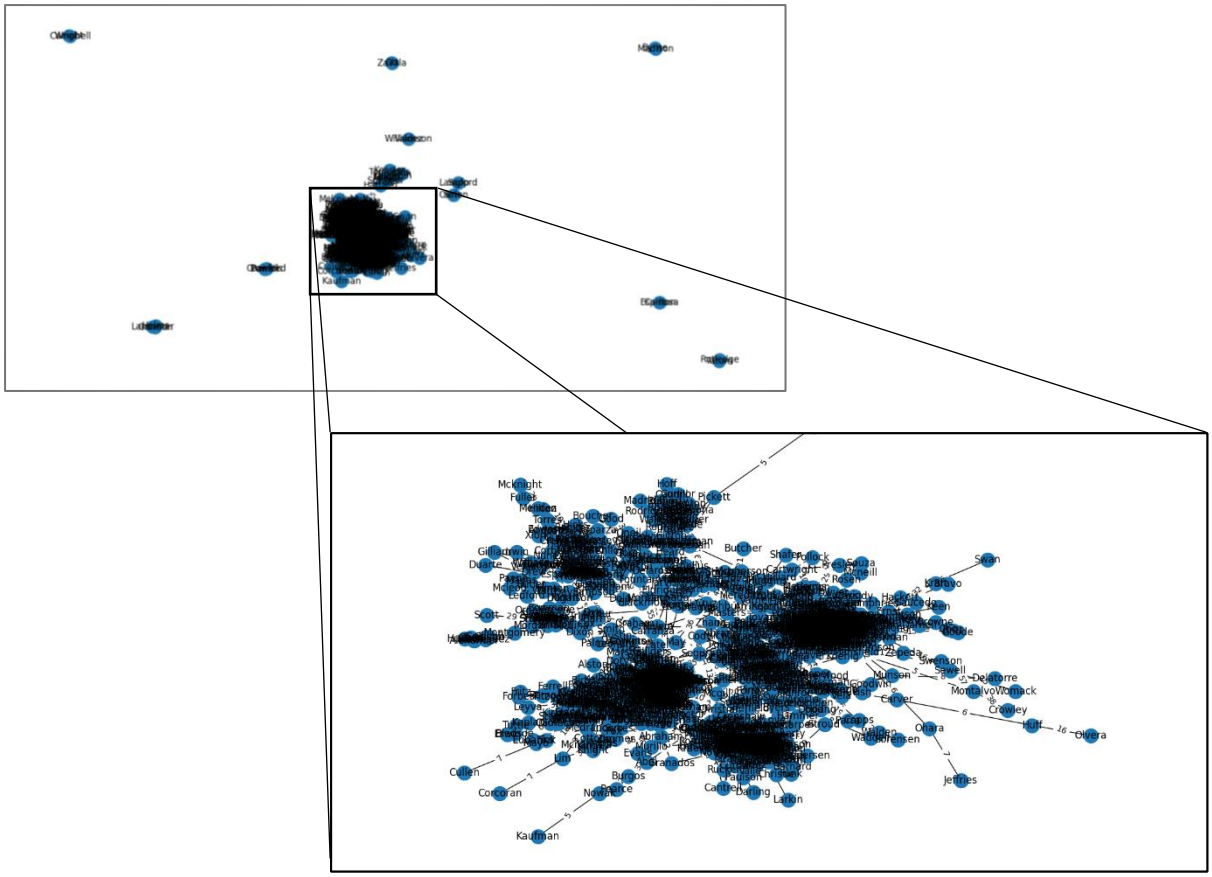


Figura 6 – Grafo Pesado com os dados do ficheiro facebook_netwrok.csv

Na figura 6 podemos visualizar o grafo com os dados do ficheiro *facebook_netwrok.csv*, tendo já sido aplicado o algoritmo de Kruskal. É notório a existência de "comunidades" diferentes logo á partida, sendo a maior a que se encontra um pouco abaixo do centro, que possui um elevado número de vértices conectados quando comparado com as outras "comunidades".

Logo de seguida, foi aplicado o algoritmo do K-Cluster como k = 2 para verificar o que ocorreria. Para uma melhor visualização e análise do grafo, foram colocadas cores diferentes para as "comunidades" diferentes e foram retiradas as descrições dos edges e dos vértices, obtendo o seguinte:

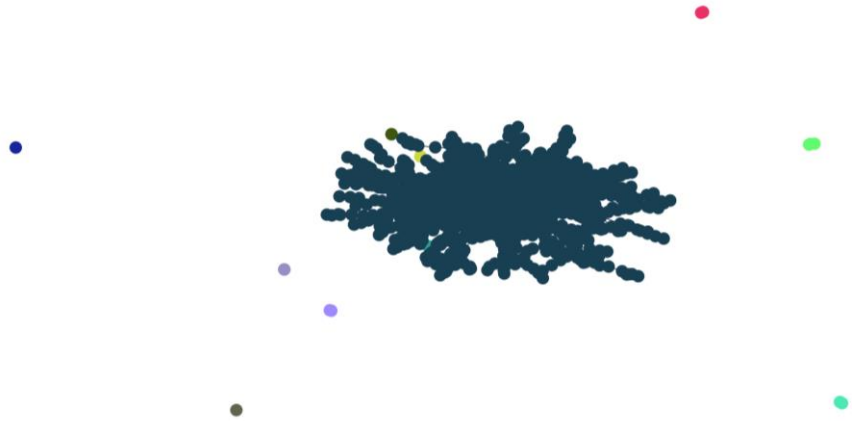


Figura 7 – Grafo ficheiro com o algoritmo de K-Cluster, com k = 2

A diferença deste grafo para o anterior, quando comparado com as "comunidades" existentes em cada um, é exatamente a mesma, demonstrando que com o k = 2, o algoritmo não irá ter nenhum efeito no grafo, pois, irá apenas eliminar o Edge com maior peso, retirando a ligação com menor proximidade entre os usuários presentes nela.

Após a visualização do grafo anterior, começamos a aumentar o valo de k pouco a pouco e reparámos que, a partir de k = 10, já existiam mudanças visíveis e, de alguma forma, significativas nas "comunidades" existentes, como é possível verificar no grafo abaixo:

Figura 8 – Grafo ficheiro com o algoritmo de K-Cluster, com k igual a 10

De seguida, decidimos aumentar o k-clustering para k = 300, um valor muito elevado, com o objetivo de visualizar, não só comportamento do grafo, como também o quanto este algoritmo iria impactar e desenvolver o número de "comunidades" do grafo.

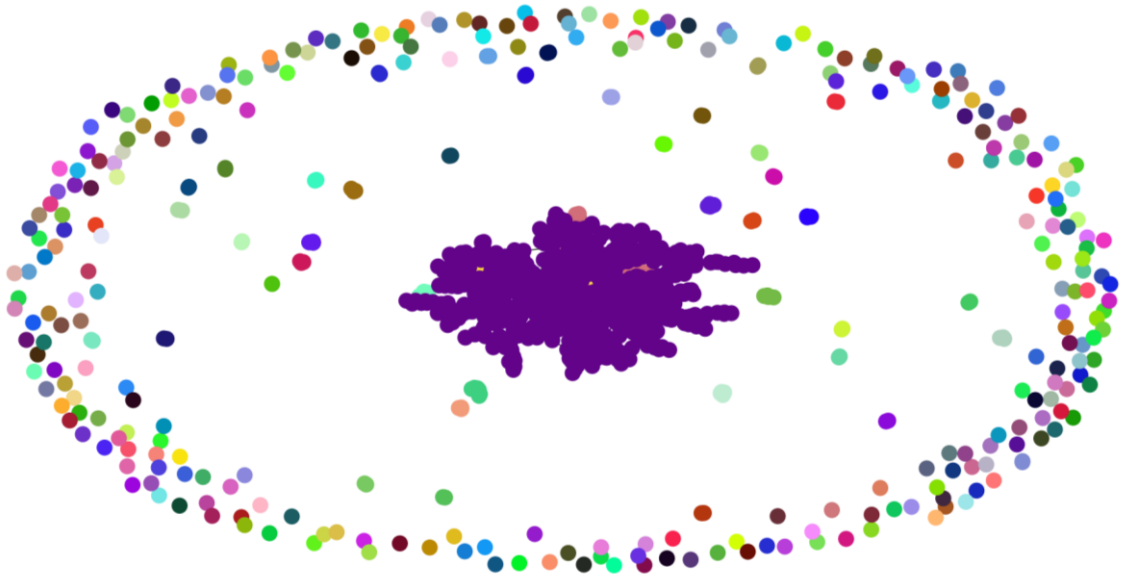


Figura 9 – Grafo ficheiro com o algoritmo de K-Cluster, com k igual a 100

Este procedimento teve um grande impacto, quando comparado com o grafo inicial, fazendo com que o número de comunidades tivesse um aumento muito elevado (Figura 8). Por terem sido retiradas as 300 ligações mais pesadas (ou com a interpretação que o nosso grupo fez, das interações mais fracas)

Podemos assim entender que, caso o grafo possua um elevado número de vértices e de edges, é necessário que o k do clustering seja mais elevado, para que exista alguma mudança relativa e importante no número de "comunidades" do grafo. É importante referir que é preciso ter cuidado com o número selecionado, pois caso sejam retirados muitos edges, não será possivel retirar conclusões sólidas olhando para o grafo, já que as comunidades estarão muito fragmentadas.

c) Uso do NetworkX para a visualização e estudo das sub-comunidades em cada comunidade

1- k_clique_communities(G, k, cliques=None)

Este algoritmo é a união de todos os cliques de tamanho k que podem ser alcançados através de k-cliques adjacentes (compartilhando k-1 nós), ou seja, ele irá formar todos os subgrafos possíveis com, no mínimo, **K** elementos. Neste algoritmo não necessitamos de aplicar, nem o Kruskal, nem o k-cluster (este último iria diminuir as possibilidades de criar sub-grafos, pois, iria eliminar as ligações mais pesadas)

Ao aplicar-mos este algoritmo com **K** = 10 obtivemos o seguinte resultado:

[frozenset({'Steele', 'Clark', 'Reyes', 'Barnett', 'Russell', 'Thornton', 'Fowler', 'Berry', 'Stephens', 'Jenkins', 'Gilbert', 'Elliott', 'Burton', 'Stewart', 'Hoffman', 'Lambert', 'Dean', 'Coleman', 'Tran'}),

frozenset({'Green', 'Rios', 'Weber', 'Cunningham', 'Olson', 'Jennings', 'Rhodes', 'Delgado', 'Harrison', 'Dunn'}),

frozenset({'Carr', 'Jacobs', 'Garrett', 'James', 'Spencer', 'Schmidt', 'Moreno', 'Davidson', 'Carlson', 'Bryant', 'Barnes', 'Howell', 'Ruiz', 'Peterson', 'Armstrong', 'Stanley', 'Salazar', 'Powers', 'Schneider', 'Walters', 'Cook', 'Gibson', 'Thomas', 'Ford', 'Bradley', 'Guerrero', 'Pierce', 'Pearson', 'Little', 'Henry', 'Arnold', 'Norris', 'Jordan', 'Mcdaniel', 'Brown', 'Carroll', 'Ferguson', 'Perkins', 'Estrada', 'Maldonado', 'Hill', 'Lynch', 'Douglas', 'Lewis', 'Bell', 'Griffin', 'Watts', 'Nguyen', 'Larson', 'Stone', 'Morris', 'Gomez', 'Hughes', 'Vasquez', 'Frazier', 'Lee', 'Castillo', 'Wilson', 'Murray', 'Neal', 'Chambers', 'Horton', 'Robinson', 'Wong', 'Gonzales', 'Diaz', 'Park', 'Newman', 'Banks'}),

frozenset({'Jacobs', 'Salazar', 'Ferguson', 'Schneider', 'Contreras', 'Hill', 'Robinson', 'Bradley', 'Nguyen', 'Diaz', 'Aguilar'})]

É possível verificar que foram criados 4 subgrafos diferentes, cada um respeitando a regra de que têm de ter no mínimo 10 elementos. Um desses subgrafos é muito maior quando comparado com os outros, possuindo 69 elementos.

De seguida, colocamos o **K** = 15 para visualizar as mudanças que poderiam ocorrer, e obtivemos o seguinte resultado:

[frozenset({'Garrett', 'Lewis', 'Gomez', 'Schmidt', 'Watts', 'Wilson', 'Murray', 'Mcdaniel', 'Salazar', 'Lee', 'Nguyen', 'Carroll', 'Arnold', 'Spencer', 'Gonzales', 'Vasquez', 'Schneider', 'Robinson', 'Banks', 'Armstrong', 'Douglas', 'Jordan'}),

frozenset({'Garrett', 'Lewis', 'Gomez', 'Murray', 'Salazar', 'Barnes', 'Lee', 'Nguyen', 'Carroll', 'Arnold', 'Griffin', 'Norris', 'Gonzales', 'Schneider', 'Robinson', 'Banks', 'Armstrong', 'Jordan'})]

É notória o número reduzido de subgrafos, quando comparado com o anterior, quando aplicado um **K** = 15

2- Kernighan_lin_bisection (G, partition=None, max_iter=10, weight='weight', seed=None)

É feita a divisão de um grafo em dois blocos usando o algoritmo de **Kernighan–Lin**. Este algoritmo particiona uma rede em dois conjuntos trocando iterativamente os pares de nós para reduzir o corte do edge entre os dois conjuntos. Os pares são escolhidos de acordo com uma forma modificada de Kernighan-Lin, que move os nós individualmente, alternando entre os lados para manter a bissecção equilibrada.

1ª subgrupo - [{'Drew', 'Cramer', 'Mead', 'Truong', 'Marcus', 'Fitch', 'Shipley', 'Clifford', 'Bateman', 'Montgomery', 'Bonner', 'Pike', 'Rodriquez', 'Doss', 'Rizzo', 'Pritchett', 'Givens', 'Mayes', 'Esquivel', 'Farris', 'Leal', 'Bird', 'Ratliff', 'Grimm', 'Duong', 'Dunham', 'Mansfield', 'Caballero', 'Hoyt', 'Palacios', 'Childress', 'Stacy', 'Michel', 'Robison', 'Hollis', 'Thorpe', 'Adair', 'Mcknight', 'Masters', 'Steiner', 'Kidd', 'Richter', 'Block', 'Mobley', 'Smallwood', 'Buckner', 'Segura', 'Wesley', 'Plummer', 'Rosen', 'Knowles', 'Quintana', 'Compton', 'Ventura', 'Kiser', 'Montes', 'Harden', 'Mcqueen', 'Timmons', 'Fink', 'Mcelroy', 'Stern', 'Lott', 'Odell', 'Amaya', 'Camp', 'Humphries', 'Cassidy', 'Diamond', 'Bray', 'Leslie', 'Latham', 'Maloney', 'Stubbs', 'Tillman', 'Mata', 'Dotson', 'Hutton', 'Osborn', 'Ahmed', 'Neely', 'Rosa', 'Conklin', 'Brewster', 'Hudson', 'Hatcher', 'Brennan', 'Fields', 'Paredes', 'Ernst', 'Spicer', 'Sutherland', 'Fontenot', 'Caudill', 'Villegas', 'Tolbert', 'North', 'Pelletier', 'Hurd', 'Peralta', 'Yee', 'Ash', 'Guevara', 'Kearney', 'Mcpherson', 'Stanford', 'Sharpe', 'Harper', 'Raines', 'Dick', 'Blackmon', 'Kirkland', 'Steward', 'Silver', 'Lay', 'Norwood', 'Albright', 'Baird', 'Toth', 'Sinclair', 'Andersen', 'Field', 'Scott', 'Spence', 'Morin', 'Locke', 'Pack', 'Colvin', 'Whalen', 'Elkins', 'Putnam', 'Swenson', 'Post', 'Hinton', 'Hanley', 'Lucero', 'Whaley', 'Waller', 'Delaney', 'Beltran', 'Tackett', 'Ricks', 'Benitez', 'Pierre', 'Medeiros', 'Winkler', 'Clifton', 'Hastings', 'Delong', 'Varela', 'Swift', 'Souza', 'Alston', 'Kaur', 'Walsh', 'Trejo', 'Barnhart', 'Madden', 'Meredith', 'Eddy', 'Gagnon', 'Land', 'Presley', 'Davila', 'Mccauley', 'Sweet', 'Zapata', 'Arrington', 'Beach', 'Daugherty', 'Duke', 'Messer', 'Rowland', 'Tovar', 'Gustafson', 'Major', 'Ferrell', 'Alexander', 'Lackey', 'Finch', 'Riddle', 'Gary', 'Crum', 'Ogden', 'Hay', 'Langston', 'Cummins', 'Mckinley', 'Winn', 'Greene', 'Swartz', 'Street', 'Lehman', 'Willard', 'Foreman', 'Elmore', 'Renteria', 'Shearer', 'Hinojosa', 'Nix', 'Medrano', 'Holliday', 'Mcleod', 'Rosas', 'Hutchison', 'Cagle', 'Cody', 'Bloom', 'Montano', 'Dempsey', 'Dodge', 'Kilgore', 'Mccollum', 'Landis', 'Sheridan', 'Lord', 'Chaney', 'Pryor', 'Fountain', 'Koehler', 'Kenney', 'Bunch', 'Sierra', 'Blount', 'Ma', 'Napier', 'Amos', 'Quezada', 'Coyle', 'Rudolph', 'Childers', 'Reece', 'Castro', 'Ellis', 'Brandon', 'Machado', 'Eubanks', 'Smiley', 'Gabriel', 'Sullivan', 'Branch', 'Hilliard', 'Rodrigues', 'Kay', 'Hurtado', 'Connor', 'Capps', 'Triplett', 'Hargrove', 'Herndon', 'Redmond', 'Lugo', 'Saldana', 'Mcmanus', 'Guy', 'Metz', 'Edmonds', 'Butcher', 'Story', 'Coulter', 'Elder', 'Gallo', 'Schmitt', 'Beyer', 'Denton', 'Cullen', 'Vinson', 'Zimmer', 'Gomes', 'Muniz', 'Rhoades', 'Cuevas', 'Hadley', 'Kowalski', 'Urban', 'Dailey', 'Varner', 'Kuhn', 'Ornelas', 'Diehl', 'Haas', 'Payton', 'Bergeron', 'Darby', 'Peacock', 'Vernon', 'Carrasco', 'Hendrickson', 'Gunter', 'Fraser', 'Hoff', 'Gamble', 'Oliver', 'Travis', 'Lindsay', 'Huerta', 'Coates', 'Mcneill', 'Alvarez', 'Dugan', 'Hager', 'Schafer', 'Haines', 'Cho', 'Sprague', 'Langford', 'Epps', 'Crow', 'Winter', 'Dickens', 'Bingham', 'Dewitt', 'Langley', 'Phipps', 'Dye', 'Person', 'Bolden', 'England', 'Connell', 'Goldberg', 'Bright', 'Cotton', 'Sapp', 'Spangler', 'Lutz', 'Spivey', 'Trotter', 'Chen', 'Holley', 'Zhang', 'Carney', 'Snell', 'Daley', 'Dobbs', 'Posey', 'Pollock', 'Mcclellan', 'Tripp', 'Bergman', 'Teague', 'Crouch', 'Simms', 'Berman', 'Esposito', 'Hopper', 'Browne', 'Polk', 'Gore', 'Thurman', 'Milligan', 'Goldstein', 'Rollins', 'Romo', 'Winston', 'Tomlinson', 'Jarrett', 'Trent', 'Petty', 'Hand', 'Hong', 'Dolan', 'Baez', 'Doty',

'Braun', 'Phillips', 'Ng', 'Crabtree', 'Westbrook', 'Hoskins', 'Dickson', 'Stapleton', 'Tyson', 'Tobin', 'Oneil', 'Hutchins', 'Zepeda', 'Francisco', 'Salgado', 'Cates', 'Lujan', 'Wilhelm', 'Rouse', 'Tang', 'Otto', 'Shapiro', 'Vogel', 'Pate', 'Crews', 'Emery', 'Kauffman', 'Arthur', 'Rubin', 'Tipton', 'Fernandez', 'Workman', 'Mayo', 'Xiong', 'Costello', 'Kimball', 'Caruso', 'Egan', 'Proctor', 'Cornelius', 'Bragg', 'Justice', 'Aldridge', 'Bland', 'Madrid', 'Boyce', 'Krause', 'Christopher', 'Roe', 'Parks', 'Murdock', 'Courtney', 'Engel', 'Kirkpatrick', 'Ferris', 'Wolff', 'Thomson', 'Wade', 'Aviles', 'Ludwig', 'Forbes', 'Villalobos', 'Good', 'Fonseca', 'Sherwood', 'Stover', 'Ring', 'Camacho', 'Tracy', 'Keen', 'Cherry', 'Corcoran', 'Bruno', 'Peoples', 'Garland', 'John', 'Waite', 'Toney', 'Singer', 'Laird', 'Mckenna', 'Shoemaker', 'Gaston', 'Saenz', 'Irwin', 'Aquino', 'Puckett', 'Louis', 'Rainey', 'Painter', 'Mcneil', 'Casillas', 'Pritchard', 'Flaherty', 'Abraham', 'Lim', 'Hathaway', 'Espinosa', 'Bolton', 'Ackerman', 'Wooten', 'Riggs', 'Greenwood', 'Carranza', 'Ziegler', 'Joyce', 'Crocker', 'Snyder', 'Dillard', 'Escobedo', 'Mooney', 'Gillis', 'Gay', 'Devine', 'Meade', 'Hilton', 'Schmitz', 'Simons', 'Regan', 'Akers', 'Stratton', 'Jung', 'Odom', 'Hightower', 'Magee', 'Pettit', 'Kincaid', 'Wynn', 'Angel', 'Lyon', 'Magana', 'Santos', 'Rock', 'Bowling', 'Stahl', 'Starr', 'Stroud', 'Cahill', 'Manley', 'Kraft', 'Thorne', 'Purcell', 'Manuel', 'Cope', 'Blue', 'Covington', 'Fisher', 'Ramey', 'Eldridge', 'Ritchie', 'Rodriguez', 'Snider', 'Layton', 'Daly', 'Medina', 'Esparza', 'Gilliam', 'Dill', 'Marino', 'Fitzgerald', 'Washburn', 'Farr', 'Metzger', 'Hinson', 'Roth', 'Gold', 'Morgan', 'Chu', 'Carlton', 'Uribe', 'Samuel', 'Hackett', 'Saucedo', 'Goins', 'Yanez', 'Shirley', 'Hagen', 'Levine', 'Denny', 'Goss', 'Prater', 'Hays', 'Chappell', 'Heard', 'Leyva', 'Downing', 'Jacob', 'Ledford', 'Holden', 'Zavala', 'Meeks', 'Fritz', 'Burt', 'Finn', 'Holbrook', 'Newsome', 'Schumacher', 'Cano', 'Vu', 'Simmons', 'Voss', 'Flanagan', 'Butts', 'Merrill', 'Barlow', 'Franks', 'Starks', 'Quintero', 'Maurer', 'Staley', 'Bermudez', 'Ott', 'Boucher', 'Tan', 'Herron', 'Paige', 'Dahl', 'Abrams', 'Hagan', 'Brand', 'Mackey', 'Sadler', 'Haywood', 'Pickett', 'Nava', 'Sheldon', 'Quiroz', 'Kern', 'Queen', 'Guidry', 'Booker', 'Dwyer', 'Rico', 'Burkett', 'Couch', 'Britton', 'Galindo', 'Kraus', 'Foster', 'Piper', 'Childs', 'Cooley', 'Groves', 'Bernal', 'Whitfield', 'Melvin', 'Hooper', 'Portillo', 'Mcallister', 'Grace', 'Vickers', 'Bullard', 'Ocampo', 'Sargent', 'Healy', 'Carlisle', 'Osorio', 'Bustamante', 'Mcnamara', 'Kyle', 'Babcock', 'Henley', 'Donnelly', 'Davison', 'Connors', 'Grady', 'Mcdonough', 'Reyna', 'Cartwright', 'Jarvis', 'Sykes', 'Kelly', 'Overton', 'Biggs', 'Sheehan', 'Holder', 'Hartley', 'Alonso', 'Cooke', 'Fair'},

2ª subgrupo - {'Greenberg', 'Guillen', 'Miles', 'Mann', 'Abel', 'Shaw', 'Diaz', 'Watkins', 'Curran', 'Avalos', 'Mccray', 'Dejesus', 'Stanley', 'Burks', 'Dickerson', 'Weber', 'Haynes', 'Neff', 'Pearson', 'Sumner', 'Byrd', 'Stone', 'Parra', 'Benavides', 'Morris', 'Hannah', 'Carter', 'Neal', 'Vazquez', 'Vang', 'Allen', 'Guthrie', 'Franklin', 'Hammer', 'Velasco', 'Woods', 'Hopkins', 'Henry', 'Johnston', 'Madrigal', 'Larson', 'Beaver', 'Sorensen', 'Belcher', 'Geiger', 'Martinez', 'Carmona', 'Yarbrough', 'Hyde', 'Nowak', 'Pinto', 'Henderson', 'Keys', 'Woody', 'Powell', 'Goodwin', 'Jones', 'Lovett', 'Byrne', 'Sizemore', 'Gray', 'Werner', 'Turner', 'Guerrero', 'Delarosa', 'Freeman', 'Webb', 'Price', 'Cash', 'Castillo', 'Castellanos', 'Adams', 'Vargas', 'Heller', 'Bravo', 'Robertson', 'Palmer', 'Walden', 'Parker', 'Beatty', 'Nicholas', 'Rogers', 'Delgado', 'Hanson', 'Sewell', 'Barnard', 'Mcclendon', 'Miller', 'Kim', 'Fox', 'Bell', 'Koenig', 'Patterson', 'Washington', 'Burton', 'Kendall', 'Douglas', 'Rice', 'Robbins', 'Felix', 'Burris', 'Ruiz', 'Pedersen', 'Jeffries', 'Hall', 'Ivey', 'Ramirez', 'Rutherford', 'Hart', 'Link', 'Nunez', 'Noel', 'Collins', 'Cornell', 'Lynch', 'Lancaster', 'Davidson', 'Nguyen', 'Burns', 'Vera', 'Hayes', 'Cunningham', 'Tompkins', 'Cole', 'Richard', 'Kessler', 'Powers', 'Ohara', 'Dean', 'Page', 'Battle', 'Schaffer', 'Key', 'Hyatt', 'Dubois', 'Owens', 'Yeager', 'Porter', 'Ly', 'Leonard', 'Reed', 'Holman', 'Aragon', 'Hicks', 'Wagner', 'Penn', 'Griggs', 'Payne', 'Ortega', 'Bassett', 'Larkin', 'Griffin', 'Kurtz', 'Wallace', 'Mcgrath', 'Cooper', 'Castle', 'Rios', 'Gibbons', 'Huggins', 'Cote', 'Anaya', 'Bradley', 'Haney', 'Locklear', 'Patel', 'Cordero', 'Bryant', 'Berry', 'Funk', 'Austin', 'Beck', 'Lu', 'Romano', 'Torres', 'Coker', 'Stevens', 'Keenan', 'May', 'Cordova', 'Hurt', 'Mendez', 'Hewitt', 'Mayfield', 'Herbert', 'Blanton', 'Simpson', 'Lowe', 'Carr', 'Judd',

'Oleary', 'Vela', 'Sandoval', 'Helms', 'Mendoza', 'Ewing', 'Horner', 'Novak', 'Corbett', 'Dang', 'Pena', 'Munson', 'Dooley', 'Johnson', 'Jacobsen', 'Ryan', 'Lucas', 'Ramos', 'Godfrey', 'Christie', 'Bui', 'Lowry', 'Bennett', 'Archer', 'Donahue', 'Mchugh', 'Graham', 'Schultz', 'Hoffman', 'Darnell', 'Bonilla', 'Howell', 'Bryson', 'Soto', 'Coleman', 'Hatfield', 'Hawkins', 'Pagan', 'Dodd', 'Shea', 'Mims', 'Shafer', 'Russell', 'Barrios', 'Young', 'Weston', 'Kaplan', 'Craig', 'Maddox', 'Bacon', 'Thornton', 'Sims', 'Tucker', 'White', 'Weir', 'Self', 'Newman', 'Walters', 'Goodrich', 'Stephens', 'Sheffield', 'Nelson', 'Helton', 'Smith', 'Christian', 'Tatum', 'Benson', 'Sanders', 'Tuttle', 'Draper', 'Mcdonald', 'Hansen', 'Paz', 'Schwartz', 'Deal', 'Jackson', 'Olson', 'Kelley', 'Hess', 'Schulz', 'Alvarado', 'Spencer', 'Whitman', 'Vogt', 'Welch', 'Jewell', 'Mcdowell', 'Huff', 'Feldman', 'Talley', 'King', 'Ortiz', 'Holland', 'Craft', 'Levin', 'Fish', 'Gleason', 'Clarke', 'Barnett', 'Mcghee', 'Rucker', 'Andrews', 'Connolly', 'George', 'Samuels', 'Slater', 'Sutton', 'Schneider', 'Shelton', 'Hernandez', 'Lanier', 'Daniels', 'Bush', 'Erwin', 'Hamm', 'Weaver', 'Pina', 'Garcia', 'Lockwood', 'Newell', 'Garza', 'Gunn', 'Stanton', 'Dukes', 'Thomason', 'Morales', 'Pierce', 'Hill', 'Rhodes', 'Dixon', 'Mason', 'Barnes', 'Thacker', 'Jensen', 'Prado', 'Mcgraw', 'Granados', 'Walker', 'Woodruff', 'Whitney', 'Kruse', 'Obrien', 'Contreras', 'Keller', 'Hinkle', 'Holmes', 'Wills', 'Metcalf', 'Perry', 'Cowan', 'Matthews', 'Witt', 'Schmidt', 'Kendrick', 'Santiago', 'Cheng', 'Howard', 'Little', 'Carroll', 'Padgett', 'Cabral', 'Watson', 'Moreno', 'Gonzalez', 'Mercer', 'Youngblood', 'Dunbar', 'Grant', 'Colbert', 'Valle', 'Hoang', 'Crenshaw', 'Jamison', 'Crawford', 'Ferguson', 'Arredondo', 'Lambert', 'Frazier', 'Do', 'Marshall', 'Goldman', 'Vasquez', 'Chapman', 'Murphy', 'Murillo', 'Vaughn', 'Sears', 'Corley', 'Hatch', 'Lawrence', 'Dunn', 'Waddell', 'Hunter', 'Olvera', 'Head', 'Ward', 'Day', 'Burke', 'Knight', 'Wells', 'Baca', 'Padilla', 'Clement', 'Muller', 'Bowles', 'Boggs', 'Carver', 'Correa', 'Crockett', 'Gates', 'Harris', 'Wilson', 'Becerra', 'Alfaro', 'Rivera', 'Doherty', 'Ferreira', 'Chambers', 'Gonzales', 'Goff', 'Ervin', 'Lane', 'Han', 'Lawson', 'Dominguez', 'Peterson', 'Lee', 'Myers', 'Jimenez', 'Bauer', 'Curtis', 'Maldonado', 'Martin', 'Lewis', 'Lovell', 'Holt', 'Roper', 'Clark', 'Harrison', 'Britt', 'Mccoy', 'Duarte', 'Ray', 'Landers', 'Cortes', 'Gordon', 'Delatorre', 'Stewart', 'Jennings', 'Smart', 'Thomas', 'Hedrick', 'Jaramillo', 'Sterling', 'Carlson', 'Steele', 'Rosenberg', 'Armstrong', 'Perkins', 'Cohen', 'Miner', 'Mccabe', 'Valdez', 'Springer', 'Burgos', 'Crowley', 'Nieto', 'Campbell', 'Beal', 'Jordan', 'Davis', 'Richardson', 'Avila', 'Root', 'Black', 'Mills', 'Juarez', 'Barker', 'Cook', 'Garrett', 'Katz', 'Aguilar', 'Peters', 'Jacobs', 'Cortez', 'Poe', 'Mccann', 'Swan', 'Corona', 'Dickinson', 'Duncan', 'Driscoll', 'Brooks', 'Cardona', 'Pearce', 'Paulson', 'Robinson', 'Park', 'Brantley', 'Donaldson', 'Warren', 'Sanchez', 'Darling', 'Estrada', 'Williamson', 'Fowler', 'Mcdermott', 'Gifford', 'Guzman', 'Kennedy', 'Gomez', 'Romero', 'Montalvo', 'Chin', 'James', 'Bower', 'Minor', 'Gipson', 'Williams', 'Boswell', 'Graves', 'Murray', 'Godwin', 'Houser', 'Wheeler', 'Grossman', 'Clemons', 'Evans', 'Serna', 'Hughes', 'Spaulding', 'Gardner', 'Tran', 'Rose', 'Ross', 'Edwards', 'Butler', 'Roche', 'Roberts', 'Whitley', 'Rosado', 'Friend', 'Gorman', 'Green', 'Carmichael', 'Goode', 'Kaiser', 'Becker', 'Blanco', 'Reid', 'Kang', 'Dietz', 'Jorgensen', 'Mccain', 'Lopez', 'Bauman', 'Wright', 'Wiseman', 'Jenkins', 'Downs', 'Ybarra', 'Kaufman', 'Roland', 'Pollard', 'Richards', 'Burger', 'Nichols', 'Gibson', 'Wolfe', 'Norris', 'Benoit', 'Brown', 'Norton', 'Hendrix', 'Womack', 'Billings', 'Caldwell', 'Harrell', 'Frey', 'Gilbert', 'Wong', 'Warner', 'Wilder', 'Fuller', 'Skaggs', 'Hickey', 'Watts', 'Madison', 'Munoz', 'Cruz', 'Long', 'Webber', 'Stiles', 'Gallardo', 'Swain', 'Law', 'Flores', 'Gregory', 'Herrera', 'Aguilera', 'Reyes', 'Reynolds', 'Silva', 'Lange', 'Arnold', 'Salazar', 'Mcgowan', 'Irvin', 'Hale', 'Yost', 'Forrest', 'Siegel', 'Horton', 'Acosta', 'Dale', 'Boyd', 'Gutierrez', 'Lau', 'Coffman', 'Banks', 'Bailey', 'Cox', 'Rutledge', 'Cantrell', 'Byers', 'Colon', 'Vega', 'Harvey', 'Chavez', 'Burrell', 'Mccarthy', 'Corbin', 'Sands', 'Pereira', 'Barajas', 'Hamilton', 'Mcdaniel', 'Bowman', 'Bledsoe', 'Elliott', 'Perez', 'Fleming', 'Lyons', 'Mcgill', 'Monroe', 'Brewer', 'Kinney', 'Vigil', 'Hunt', 'Morrison', 'Ford', 'Phan', 'Meyer', 'Bishop'}]

Conclusão

Com a realização deste trabalho adquirimos vários conhecimentos associados aos grafos e a diferentes algoritmos, tais como, o Kruskal e o K-clustering.

Na primeira fase, na implementação das classes Vertex, Edge e Graph, não houve uma grande dificultando na interpretação e no entendimento no código, tendo sido feita a correção da função __hash__, da class Vertex e também, tendo sido colocado o self._number na classe Graph de forma correta, pois, este não aumentava quando se acrescentava um vértice. De forma geral, ficámos a entender como os grafos, os vértices e as edges funcionam

A implementação do algoritmo de Kruskal foi a maior dificuldade deste grupo, tendo sido perdido um dia inteiro a tentar implementá-lo, embora que teoricamente, o grupo tinha completa noção do que este algoritmo iria fazer e do seu impacto no grafo.

Na criação do algoritmo K-Clustering houvesse uma grande facilidade em recriar o mesmo. Após o debate de qual edge deveria ser eliminado, tendo-se chegado à conclusão que retirar o mais pesado era o que faria mais sentido, o grupo passou para a implementação de um código que iria colorir as diferentes "comunidades" com cores diferentes.

Em relação ao ficheiro *facebook_netwrok.csv*, não houve dificuldade na limpeza dos dados, já que o grupo criou facilmente um código que somava o peso das edges que se repetiam. O grupo teve facilidade a trabalhar com a biblioteca NetWorkX, tendo conseguido visualizar os diferentes grafos, tanto com pontos aleatórios, com os dados do *ficheiro facebook_netwrok.csv* ou também com os grafos, tendo aplicado o algoritmo de kruskal ou o K-Cluster.

No último ponto, para estudar as sub-comunidades de cada comunidade, houve uma ligeira dificuldade em entender o que era pretendido, fazendo com que o nosso grupo não conseguisse absorver muito conhecimento nem conseguisse retirar soluções muito sólidas sobre o tema.

WebGrafia

- http://xpzhang.me/teach/DS19_Fall/book.pdf
- https://networkx.org/
- https://networkx.org/documentation/stable/reference/algorithms/community.html
- https://www.programiz.com/dsa/kruskal-algorithm
- https://pypi.org/project/networkx/