UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

JOSÉ MARIA CLEMENTINO JUNIOR **11357281**

Projeto 2– Agrupamento Algoritmos Gulosos

São Carlos

2019

Execução:

Para executar o código, após a instalação do NodeJS, basta executar o comando "**node main.js < data.txt**" no prompt de comando.

O tempo de execução de cada tarefa é, em média, para o arquivo "dados.txt":

- Criação do Grafo a partir das coordenadas lidas: 130 ms
- Execução do algoritmo de Kruskal: 370 ms
- Execução do algoritmo de Prim: 130 ms

Para visualização do grafo gerado, basta exportar o arquivo **./testOctave.txt,** no **Octave Online**(deve fazer o login) ou executá-lo na versão Desktop .

Implementação: Conforme sugerido na descrição do projeto, a implementação foi feita de forma modular e dividida nos seguintes arquivos:

- ***./main.js :*** Arquivo principal que lê pares de coordenadas um a um do stdin, os armazena em um array e chama as funções necessárias para gerar os clusters, que são salvos em arquivos em ".txt" no mesmo formato do arquivo "./classes.txt";
- **./node_modules/graph.js:** Este arquivo define a classe "graph". Seu construtor gera um grafo a partir de coordenadas;
- ***./node_modules/kruskal.js:*** Contém a função para clusterização utilizando o algoritmo de Kruskal. O algoritmo recebe um grafo como parâmetro e retorna um array contendo os clusteres. Conforme sugerido, o algoritmo agrupa os nós do grafo utilizando uma estrutura Union-Find até obter o total de k grupos;
- ***./node_modules/prim.js:*** Contém a implementação da clusterização utilizando o algoritmo de Prim. Este algoritmo utiliza fila de prioridades com heap binária cuja implementação não possui a função "decrease priority". Por esse motivo a implementação do algoritmo de Prim foi feita com a variação sugerida em classe, onde a cada nó N inserido em S se adiciona todos as arestas relacionadas a N na fila de prioridade. Após a obtenção da Árvore Geradora Mínima, é aplicada uma rotina de clusterização os k-1 maiores nós são removidos. Para classificar os nós e obter os clusters foi utilizada uma estrutura de Union-Find.
- ***./node_modules/union-find.js:*** Contém a implementação da estrutura Union-Find.

- ***./node_modules/utils.js:*** Contém funções úteis como: *saveToFile,* que salva um array de clusters em um arquivo ".txt" ***;*** *formatClusterIndexes,* que formata os índices dos clusteres para ficarem entre 1 e k;
- **./node_modules/priority-queue.js:** Implementação de fila de prioridades. Fonte: https://www.npmjs.com/package/js-priority-queue

Análise dos Resultados e Conclusões

Tanto o algoritmo de Prim quanto o algoritmo de Kruskal obtiveram os mesmos resultados para a tarefa de dividir as entradas em clusters, sendo que estes resultados diferem da solução proposta.

Pode-se observar na Figura 1: ver na imagem abaixo que nas regiões 3 e 4 – regiões que ligam agrupamentos - a distância entre os nós adjacentes é pequena, sendo menor que a distância entre os nós das regiões 1 e 2 e seus nós adjacentes mais próximos. Desta forma, ao se remover/desconsiderar os 6 maiores nós para gerar os clusteres, as regiões 1 e 2 foram desconectadas indevidamente e as regiões 3 e 4, que deveriam ser desconectadas, não foram.

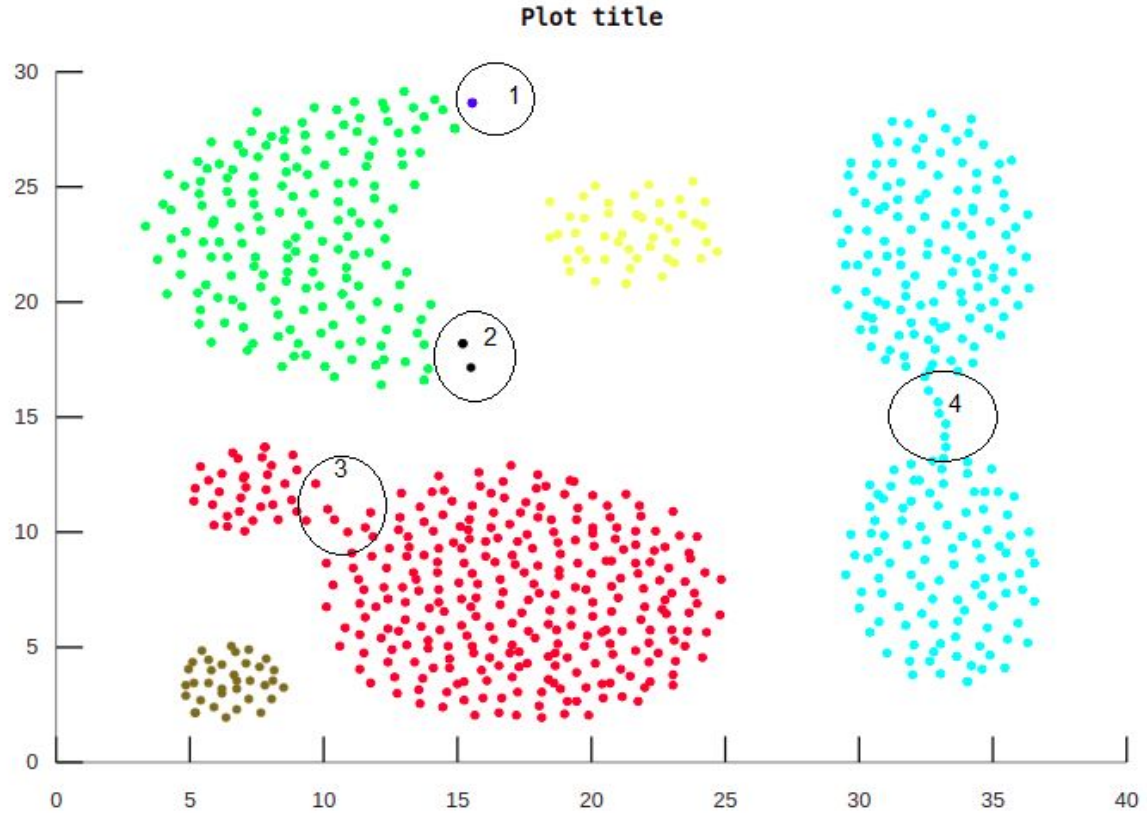

Figura 1: Agrupamento por árvore geradora mínima Link

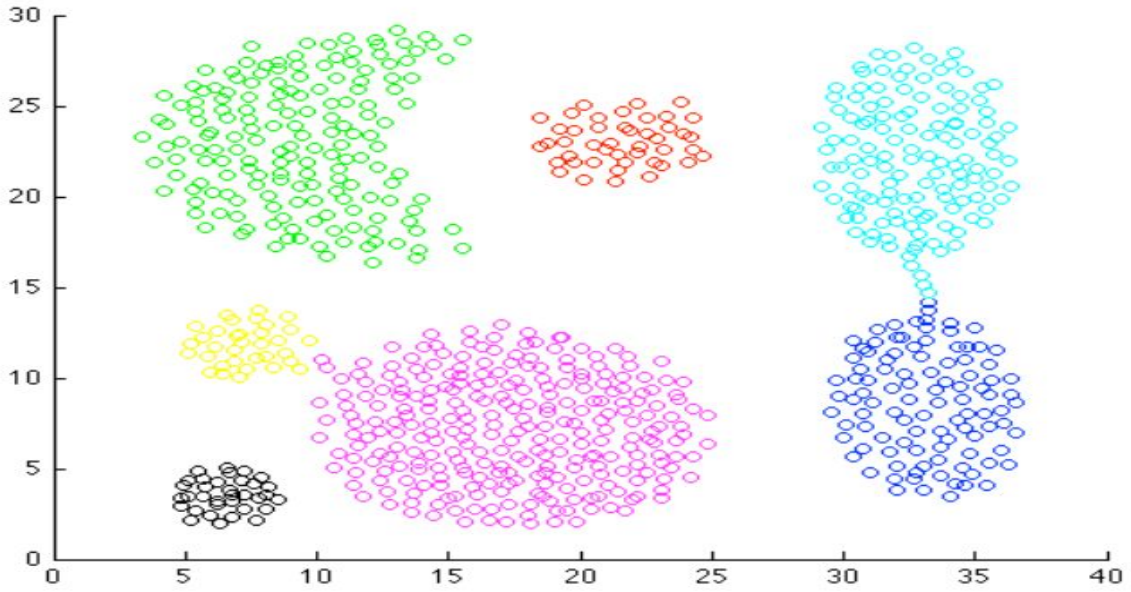

Figura 2: Agrupamento descrito pelos autores Link

Observando e fazendo uma comparação sobre o agrupamento apresentado na Figura 1 (resultado do agrupamento das árvores geradoras mínimas) e a Figura 2 (agrupamento apresentado pelos autores), é possível notar divergência entre os grupos de agrupamento. Atribui-se este comportamento, pois a árvore geradora mínima faz a verificação somente da distância entre as arestas, por exemplo: se existirem dois grupos que são conectados por poucos vértices mais bem próximos, mesmo assim a árvore geradora mínima visitará esses vértices fazendo a união indevida destes grupos. De modo que realize a separação somente dos vértices mais distantes uns dos outros , ou seja, o agrupamento sofre influência dos *vértices/outliers.* É possível observar que os grupos da Figura 2 estão melhores distribuídos, deduz-se que os autores atribuíram a distribuição dos grupos, em função da distância dos vértices associados aos pontos médios dos grupos, deste modo retirando a influência dos pontos de outliers.

Tabela 1 Tabela de Resultados (**CONTINUA** .. no Link)

Node	Data		Classes	Kruskal	Prim
0	15,55	28,65	2	1	1
1	14,9	27,55	2	2	2
2	14,45	28,35	2	2	2