Disciplina: TIN0114 – Estruturas de Dados 1
Professor: Pedro Moura
Data: 14/07/2024

Trabalho Final

Conforme apresentado em sala de aula, as Filas de Prioridade são utilizadas em contextos em que se deseja representar uma fila de elementos segundo um determinado critério de prioridade. Algumas estruturas de dados para Filas de Prioridade oferecem métodos eficientes, isto é, de baixa complexidade, para realizar operações cruciais nesse contexto, tais como remover o menor elemento, inserir um elemento ou alterar a prioridade de um dado elemento. Uma das principais formas de se implementar uma Fila de Prioridade é por meio da estrutura de dados *Heap* Binária. Neste trabalho, será proposto então utilizar as estruturas de dados *Heap* Binária e Árvore Binária para implementar um algoritmo de **Clusterização Hierárquica**.

Em um problema de **clusterização** (agrupamento), tem-se um conjunto de pontos *P* em um determinado espaço, com uma noção de distância definida entre pontos desse conjunto, e deseja-se agrupar tais pontos em *clusters* (agrupamentos), de tal forma que: (i) membros de um *cluster* sejam próximos/similares aos demais do mesmo grupo; (ii) membros de *clusters* diferentes sejam distantes/dissimilares entre si. Percebe-se, pois, que um conceito chave para se realizar a clusterização de um conjunto de pontos é o de **métrica de distância**, tais como euclidiana, cosseno, Jaccard, etc. Ademais, tais pontos normalmente estão localizados em um espaço vetorial de alta dimensão.

Quando esses pontos representam vetores em um espaço não-euclidiano, a distância usualmente utilizada é a distância cosseno, que considera o cosseno do ângulo entre esses dois vetores. Por sua vez, quando tais pontos representam conjuntos, a distância Jaccard, calculada em termos das operações de união e interseção, é a mais usual. Finalmente, quando tais pontos representam vetores em um espaço euclidiano, a distância mais comumente utilizada é a euclidiana. Esta distância é aquela que será utilizada como métrica de distância neste trabalho e sua fórmula, dados dois pontos p e q em um espaço bidimensional, é dada por

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

Dentre inúmeras aplicações, é possível citar como exemplo de um problema de clusterização o cenário de uma loja de varejo on-line em que cada produto é representado por um ponto de coordenadas (x_1, x_2, ..., x_n), em que $x_i = 1$ se, e somente se, o i-ésimo consumidor comprou tal produto. Cada dimensão desse espaço é, então, um valor binário 0 ou 1. Deseja-se, assim, obter *clusters* de produtos similares para, por exemplo, propor promoções combinando produtos de um mesmo grupo.

Outro exemplo de aplicação está no contexto de documentos (textos) em linguagem natural. Dessa forma, é possível representar cada documento por um vetor (x_1, x_2, ..., x_k), em que $x_i = 1$ se, e somente se, a i-ésima palavra (em alguma ordem) aparece nesse documento. A dimensão k desse espaço vetorial corresponde ao tamanho do léxico (número de palavras diferentes) que aparece nos documentos. Tem-se, pois, que documentos representados por conjuntos similares de palavras tendem a ficar no mesmo *cluster*, denotando então que tratam de um mesmo tópico (esporte, política, economia, etc.).

Os algoritmos de clusterização podem se dividir em duas categorias:

(i) **Aglomerativos** são aqueles em que cada ponto começa em seu próprio *cluster*. Esses *clusters* são então combinados com base em sua "proximidade", de acordo com uma determinada métrica de distância/similaridade. Tais junções de *clusters* continuam até que um critério de parada seja atingido, tal como um número pré-determinado de *clusters*; e

(ii) **Atribuição de Pontos** que são aqueles que possuem uma fase inicial de estimação dos *clusters* e, após isso, os pontos são considerados em alguma ordem e cada um é atribuído ao *cluster* em que ele se encaixa melhor. O conceito de encaixe também é regido segundo uma métrica de distância/similaridade.

Tais algoritmos de clusterização também podem ser diferenciados quanto ao fato de assumirem um espaço euclidiano ou trabalharem com alguma outra métrica de distância. No caso euclidiano, que utilizada a distância euclidiana, é possível representar um agrupamento de pontos pelo seu ***centroide***, que é a média dos pontos contidos nesse *cluster*. Esta é a estratégia que será adotada no algoritmo a ser implementado neste trabalho.

Um algoritmo de **Clusterização Hieráquica** começa com cada ponto no seu próprio *cluster*. A cada iteração, *clusters* maiores são construídos combinando-se dois *clusters* menores. Isso é executado até que um determinado critério de parada seja atingido. Pode-se então definir, em linhas gerais, o pseudocódigo do algoritmo da seguinte forma:

```
ENQUANTO não atingiu o critério de parada FAÇA
   Pegue os dois melhores clusters para combinar;
   Combine esses dois clusters em um só cluster;
FIM-ENQUANTO
```

Note-se que algumas questões não estão especificadas no algoritmo acima e, quando se vai aplicá-lo a um determinado domínio, é necessário que sejam definidas. A primeira delas se refere à representação dos *clusters*. Assume-se que o espaço é euclidiano, de modo que é possível representar um *cluster* pelo seu *centroide*, que é a média dos pontos desse *cluster*. Mais especificamente, o valor de uma coordenada do *centroide* é obtido pela média dos valores da mesma coordenada para os pontos do *cluster*. Note que, em um grupo de um único ponto, esse corresponde ao *centroide* e tal fato torna a inicialização dos *clusters* bem simples.

A segunda questão a ser abordada é qual regra seguir ao selecionar *clusters* para combinar em cada iteração do algoritmo. Neste trabalho, será adotada a regra de que **a distância entre dois *clusters* corresponde à distância euclidiana de seus *centroides***, de tal forma que se selecionam os dois *clusters* mais próximos segundo essa distância. Existem, entretanto, outras formas de escolher qual par de *clusters* selecionar em cada iteração, tal como considerar aquele par que contenha a menor distância entre quaisquer dois pontos, sendo um ponto de cada *cluster*.

Finalmente, deve-se definir o critério de parada a ser adotado. Pode-se utilizar, por exemplo, um número predefinido de *clusters* como objetivo, para que o algoritmo pare a sua execução ao chegar a esse número. Pode-se, ainda, escolher parar quando se chega a um estágio em que a melhor combinação de dois *clusters* existentes leva a um *cluster* inadequado.

Neste trabalho, o algoritmo será executado até que reste apenas um único *cluster*, agrupando os demais *subclusters* dentro de si. Retorna-se, assim, a árvore representando a forma como os *clusters* foram combinados. Essa forma de resposta faz sentido em algumas aplicações como, por exemplo, aquela em que os pontos são genomas de diferentes espécies e a distância entre eles reflete a diferença no genoma. Assim, nesse caso, a árvore representa a evolução dessas espécies, isto é, a ordem provável em que duas espécies derivaram de um ancestral comum.

Seja considerado então como exemplo o conjunto de pontos no plano cartesiano exibido na Figura 1, em que cada ponto é designado segundo as suas coordenadas (x, y).

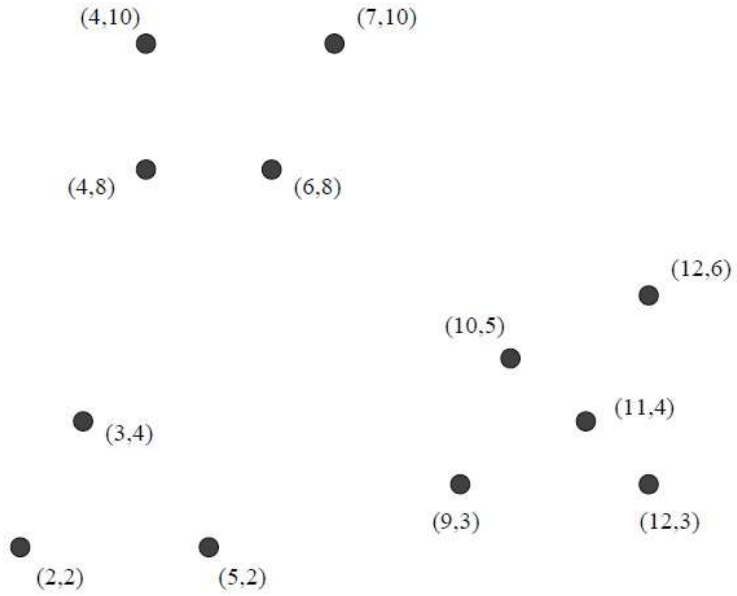

Figura 1: Conjunto de doze pontos no plano cartesiano para serem clusterizados hierarquicamente.

Entre todas as distâncias entre pares de pontos, existe um empate da distância medida entre os pares de pontos (*10*, *5*) e (*11*, *4*), assim como (*11*, *4*) e (*12*, *3*), em que ambos os pares distam $\sqrt{2}$ entre si. Arbitrariamente, escolhe-se o par (*11*, *4*) e (*12*, *3*) para realizar a combinação. O resultado está expresso na Figura 2, que mostra o *centroide* criado para representar o novo *cluster*, designado como o par ordenado (*11.5*, *3.5*) referente à média dos dois pontos do agrupamento.

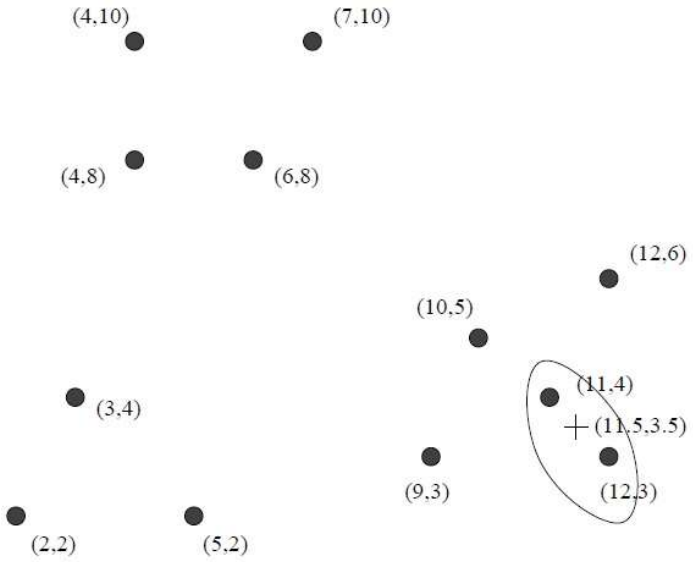

Figura 2: Combinação dos dois primeiros pontos em um *cluster*.

Nesta etapa, os *clusters* mais próximos são aqueles que representam os pontos (*4*,*10*) e (*4*, *8*) que são combinados e resultam em um *cluster* de *centroide* (*4*, *9*). Agora os *centroides* mais próximos são aqueles de (*10*, *5*) e (*11.5*, *3.5*), que são combinados. Isso resulta em um *cluster* de três pontos cujo centroide é (*11*, *4*), que coincidentemente é um dos pontos do *cluster*. O estado dos *clusters* está mostrado na Figura 3.

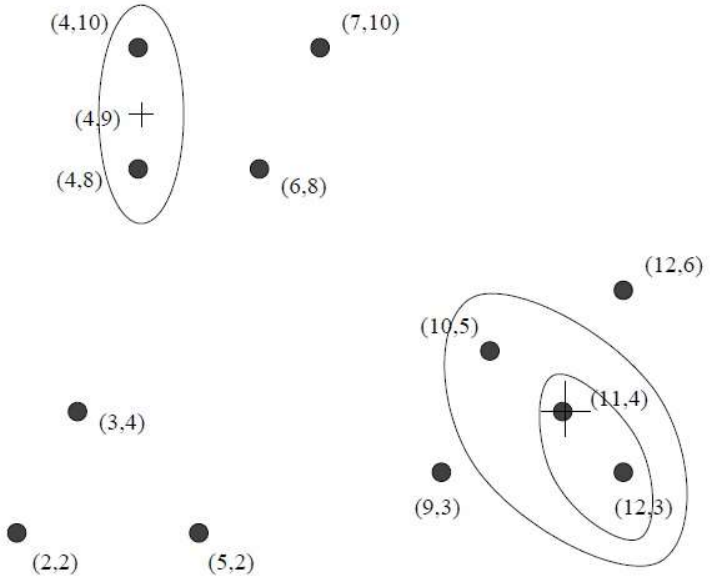

Figura 3: Clusterização após duas iterações do algoritmo.

Existem diversos pares de *centroides* que distam √5 e que são os centroides mais próximos. São selecionados então três desses pares: i) (*6*, 8) é combinado com o cluster de dois elementos com centroide (*4*, *9*); (*2*, *2*) é combinado com (*3*, *4*); e (*9*, *3*) é combinado com *cluster* de três elementos tendo *centroide* (*11*, *4*). Essas combinações estão ilustradas na Figura 4.

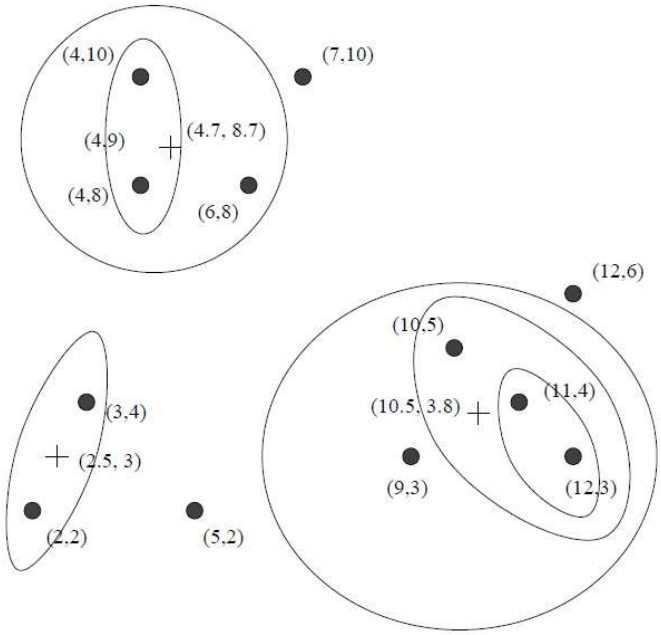

Figura 4: Mais três iterações do algoritmo de Clusterização Hierárquica.

Na continuidade, o algoritmo continua realizando combinações de *clusters* até restar apenas um único *cluster* que contenha hierarquicamente todos os demais. O resultado final do algoritmo, correspondente ao agrupamento do conjunto de pontos fornecido como entrada, está exibido na Figura 5.

Propõe-se então neste trabalho que sejam implementadas duas versões do algoritmo de Clusterização Hierárquica:

(i) *Naïve*: Uma versão que calcule a distância entre cada par de *clusters*, a fim de se obter a menor distância para realizar a combinação, a cada iteração;

(ii) *Fila de Prioridade*: Uma versão que calcula distância entre cada par de *clusters* somente no início do algoritmo e a coloca em uma **Fila de Prioridade**, de modo a ser possível obter a menor distância em *O(1)*.

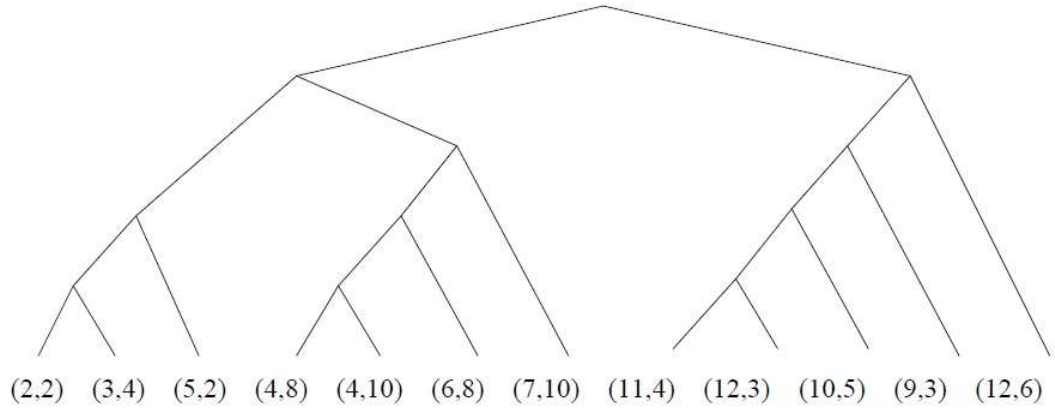

Figura 5: Árvore Binária mostrando o resultado final do algoritmo, que é o agrupamento completo do conjunto de pontos fornecido como entrada.

Quando dois *clusters* forem selecionados e combinados, deve-se excluir todas as entradas na Fila de Prioridade que envolva qualquer um desses dois *clusters* (a distância de cada um deles para os demais *clusters*).

Consecutivamente, deve-se calcular a distância do *centroide* do novo *cluster* para os demais *clusters* correntes. Cada uma dessas distâncias calculadas deve ser inserida na Fila de Prioridade.

Em ambas as versões do algoritmo de Clusterização Hierárquica, uma estrutura de dados de Árvore Binária deve ser mantida e atualizada à medida que as combinações forem feitas. Além disso, é importante que seja estimada a complexidade, com a devida explicação, das duas implementações do algoritmo, com base na lógica e nas estruturas de dados empregadas.

Após implementar os dois algoritmos acima, você deve implementar um método que **gere um conjunto de N pontos artificiais no plano cartesiano**. Você deve então usar esse código desenvolvido para criar conjuntos de pontos dos seguintes tamanhos **N**: 10, 20, 30, 40, 50, 100, 200, 500, 1.000, 5.000, 10.000, 20.000, 50.000 e 100.000. Assim, para cada ponto *x*, as **coordenadas x_1 e x_2** de tais pontos devem ser sorteadas no **intervalo inteiro [1, N]**.

Neste ponto, você deve realizar o seguinte experimento: executar as duas versões supracitadas do algoritmo de Clusterização Hieráquica para todos os conjuntos de pontos artificialmente criados. **Para minimizar eventuais ruídos na medição de tempo, rode cada operação 10 vezes e tome o tempo médio, conforme explicado em sala de aula.**

Se, ao rodar os experimentos acima, você encontrar problemas ocasionados por limitação de poder computacional a partir de um determinado tamanho do conjunto de pontos, deve rodar os experimentos até o último tamanho possível.

Por fim, gere gráficos ilustrando o tempo de execução das operações selecionadas como uma função do tamanho de entrada, em relação às duas implementações do algoritmo *(Naïve* e usando Fila de Prioridade). Compare o gráfico de tais funções obtidas com o gráfico da complexidade teórica das duas versões do algoritmo.

Algumas considerações para o trabalho:

1. Escrever um relatório contextualizando os algoritmos, as estruturas de dados, a implementação (funcionalidade de cada método) e os experimentos realizados;
2. Você deve apresentar os tempos e gráficos obtidos, analisando-os e expondo as conclusões obtidas;
3. Não deixe de relacionar as referências utilizadas ao longo do desenvolvimento do trabalho;
4. Além da monitoria, as dúvidas do trabalho poderão ser esclarecidas nas aulas de acompanhamento. Não deixe para tirar suas dúvidas no último momento!
5. Documente qualquer problema que não conseguir resolver;
6. O programa deve ser bem testado!
7. O trabalho deve ser **realizado em trios**;
8. **O trabalho deve ser submetido na tarefa criada no Moodle e apresentado pelos grupos no dia 29/08.**

OBSERVAÇÕES:

1. O trabalho é obrigatório e vale 30% da nota N2. Todos os integrantes do grupo deverão estar presentes no dia da apresentação oral;
2. Trabalhos com alto grau de semelhança levarão nota zero; e
3. Não entregar o código, ou não entregar o relatório, ou não apresentar o trabalho implicam em nota zero no trabalho.

Referências Biliográficas

1. Capítulo 7 do livro “Mining of Massive Datasets” de Jure Leskovec, Anand Rajaraman e Jeffrey Ullman. Cambridge University Press. 2020.