PROVA TEÓRICA

AB1
Prof. Rodrigo Paes.
Universidade Federal de Alagoas

1 ENCONTRE A COMPLEXIDADE (3.5 PONTOS).

O algoritmo de ordenação Counting Sort pode ser utilizado em situações específicas com resultados melhores que o Merge Sort, por exemplo. Mas em alguns casos, o resultado pode ser desastroso.

Suponha as duas instâncias de entradas abaixo e encontre a complexidade do Counting Sort nos dois casos. Você pode manipular os valores da entrada antes e depois que utilizar a ordenação. Explique claramente a sua ideia, passo-a-passo.

a) São dados n inteiros, onde cada valor pertence ao intervalo $[1, n^2]$
b) São dados n inteiros, onde cada valor pertence ao intervalo $[n^2, n^2+n]$

2 APLIQUE EM CADA ITEM OS MÉTODOS TEOREMA MESTRE, ÁRVORE E SUBSTITUIÇÃO PARA RESOLVER AS SEGUINTES RECORRÊNCIAS. (3 PONTOS)

a) $T(n) = 9T\left(\frac{n}{3}\right) + n$
b) $T(n) = 1 + T(\frac{n}{2})$

3 REDUÇÃO (3.5 PONTOS).

Muitas vezes quando nos deparamos com um problema em que não temos a solução, buscamos modelar este problema de forma vê-lo como um outro problema em que sabemos resolver. Por exemplo, suponha que você se deparou com o problema de encontrar o caminho mais curto em um grafo direcionado sem pesos e você não conhece nenhum algoritmo para fazer isso. Mas você conhece o BFS. Então, você pode enxergar esse problema do caminho mais curto como um problema de busca em largura e usar o BFS para resolvê-lo.

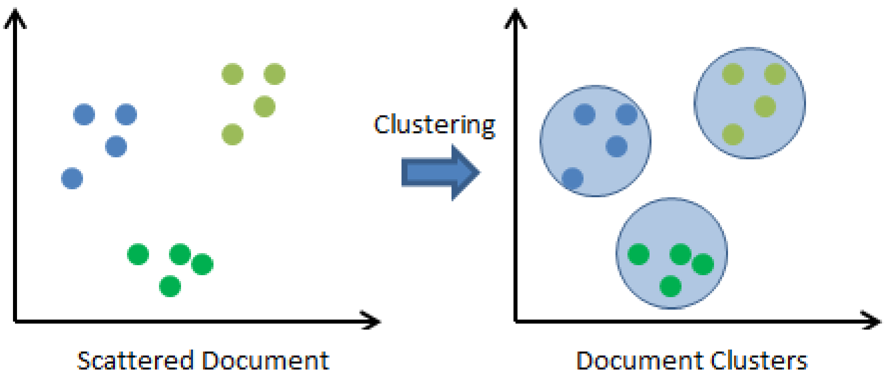
Nesta questão, temos a mesma situação. O problema a seguir pode ser resolvido utilizando um algoritmo de grafos que já estudamos. O objetivo desta questão é representar o problema abaixo como sendo esse outro problema de grafos visto em sala e mostrar como a solução é encontrada.

Descrição do problema:

Dado um conjunto de pontos em um espaço 2D e um inteiro K, divida os postos em K conjuntos de forma que eles fiquem os mais próximos possível. Outra forma de ver o problema é dividir os pontos em K subconjuntos de tal forma a maximizar a distância entre qualquer par de conjuntos:

$dist(S_1, S_2) = \min \{ dist(x, y) \mid x \text{ in } S_1, y \text{ in } S_2 \}$

Exemplo, para k = 3:



Algoritmo:

```
Seja C = {{v1}, {v2},. . ., {vn}};

while |C| > K

     Seja e = (u, v), com u in Ci e v in Cj, a aresta de menor
     custo entre dois conjuntos distintos em C

     Substitua Ci e Cj por Ci U Cj
```

Observação: v1, v2 , ... são vértices.