



## Problema D

# Agrupamentos

*Arquivo fonte:* agrupamentos.{ c | cpp | java | py }

*Autor:* Leandro Luque

Você está trabalhando num projeto que envolve a exibição em um mapa da localização de prestadores de serviço de uma grande empresa. Entre os desafios do projeto está uma lentidão cada vez maior no carregamento do mapa, dado o grande volume de prestadores exibidos - que hoje passa dos 50 mil. Ainda, a visualização do mapa está muito poluída, pois muitos pontos são exibidos em uma área pequena. Você fez uma PoC (prova de conceito) com uma biblioteca de agrupamento (clusterização - que agrupa pontos próximos) do componente de mapas frontend que usa e percebeu que o problema de visualização é resolvido. No entanto, agrupar no frontend não resolve o problema de ter que carregar milhares de pontos a partir do backend. Portanto, você decidiu implementar um algoritmo de clusterização no backend. O algoritmo é uma variante do implementado em diversas bibliotecas de mapas disponíveis no mercado.

Em resumo, o algoritmo consiste nos seguintes passos:

1. Selecione um ponto (no nosso caso, selecionaremos o primeiro ponto na entrada);
2. Verifique qual é o cluster mais próximo:
  - Se não existir cluster mais próximo ou se o cluster mais próximo estiver a mais de  $x$  unidades de distância (considerando a distância euclidiana entre o ponto e o centro do cluster), crie um novo cluster com o ponto como seu centro;
  - Caso contrário, adicione o ponto ao cluster mais próximo e recalcule o centro do cluster para ser igual a média aritmética das posições de seus pontos constituintes. Caso um ponto esteja à mesma distância de dois ou mais clusters, ele deverá ser adicionado ao cluster criado primeiro.
3. Selecione o próximo ponto e repita (2) até passar por todos os pontos.

Considere um mapa com os seguintes pontos: A (7, 7), B (11, 7), C (9, 7), D (6, 8), E (13, 8) - Figura D.1a; e raio do cluster igual a 2.

1. Começaremos pelo primeiro ponto informado: A (7,7);
2. Como ainda não existem clusters, será criado um novo com centro em (7, 7) - Figura D.1b;
3. O próximo ponto (B) está à quatro unidades de distância do primeiro cluster e, portanto, será criado um novo cluster com (11, 7) como centro - Figura D.1c;
4. O terceiro ponto (C) está a duas unidades de distância do primeiro e do segundo clusters. Como as distâncias são iguais, deve-se incluí-lo no primeiro cluster criado (A). Após adicioná-lo, o novo centro do cluster deverá ser atualizado para  $((x_a + x_c) / 2, (y_a + y_c) / 2) = ((7 + 9) / 2, (7 + 7) / 2) = (8, 7)$  - Figura D.1d;
5. O mesmo processo segue até o fim, resultando em 4 clusters - Figura D.1e.

Seu trabalho é implementar o algoritmo explicado.

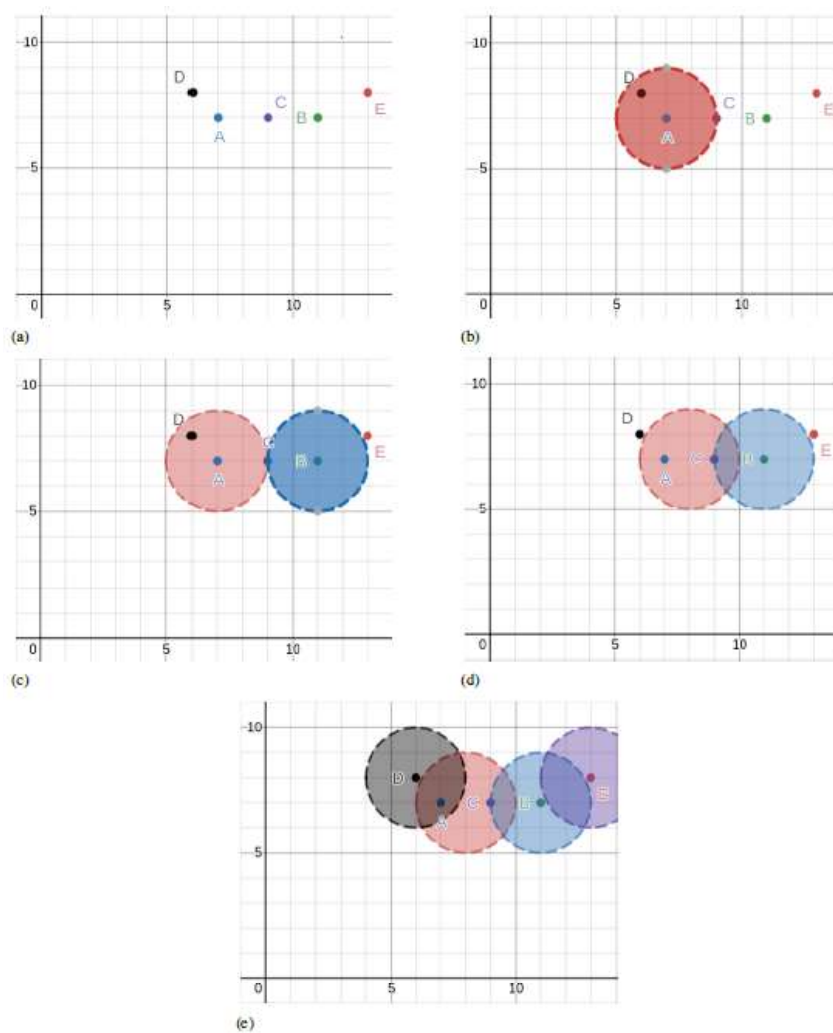


Figura D.1: Passo a passo do algoritmo

## Entrada

A entrada começa com uma linha com dois inteiros:  $N$  ( $1 \leq N \leq 100$ ) indicando o número de prestadores de serviço e  $R$  ( $1 \leq R \leq 100$ ) representando o raio que será usado para os clusters. As próximas  $N$  linhas contêm coordenadas cartesianas  $X$   $Y$  inteiras ( $-1000 \leq X; Y \leq 1000$ ) representando a localização de um prestador de serviço. A última linha de entrada termina com uma quebra de linha.

## Saída

Como saída, imprima uma linha com um número  $C$  indicando o número de clusters criados pelo algoritmo. Em seguida, imprima  $C$  linhas, cada uma com coordenadas cartesianas inteiras (truncando o resultado se necessário)  $X$   $Y$  indicando o centro de cada cluster, na ordem em que foram criados. A última linha de saída



termina com uma quebra de linha.

#### Exemplo de Entrada 1

```
5 2
7 7
11 7
9 7
6 8
13 8
```

#### Exemplo de Saída 1

```
4
8 7
11 7
6 8
13 8
```

#### Exemplo de Entrada 2

```
7 38
-966 -351
-10 899
517 524
-376 -594
-595 -696
362 -943
978 945
```

#### Exemplo de Saída 2

```
7
-966 -351
-10 899
517 524
-376 -594
-595 -696
362 -943
978 945
```