Par de pontos mais próximo

Gustavo Lopes Rodrigues , Homenique Vieira Martins , Rafael Amauri Diniz Augusto

7 de novembro de 2021

Objetivos

- Criar uma solução $O(n^2)$
- Criar uma solução $O(n * log(n)^2)$
- Criar uma solução $O(n * log(n))$

1 Algoritmos

1.1 Força bruta

Complexidade: $O(n^2)$
Código fonte: $src/algorithms/brute\_force.py$

1.1.1 Passos

1. Escolha um ponto
2. Escolha outro ponto qualquer
3. Calcule a distância entre os dois pontos
4. Verifique se a distância entre os dois é a menor (armazenar menor distância em variável auxiliar)
5. Repetir processo até que todos os pontos sejam comparados entre-si

1.2 Divisão e conquista

Complexidade: $O(n * log(n)^2)$
Código fonte: *src/algorithms/divide_and_conquer.py*

1.2.1 Passos

1. Encontre o ponto médio na matriz classificada, podemos tomar P $[\frac{n}{2}]$ como ponto médio.
2. Divida a matriz dada em duas metades. O primeiro subarray contém pontos de P [0] a P $[\frac{n}{2}]$. O segundo subarray contém pontos de P $[\frac{n}{2} + 1]$ a P [n-1].
3. Encontre recursivamente as menores distâncias em ambos os subarrays. Sejam as distâncias dl e dr. Encontre o mínimo de dl e dr. Seja o mínimo d.
4. Ordene a faixa da matriz de acordo com as coordenadas y. Esta etapa é O (n*log(n)). Ele pode ser otimizado para O (n) classificando e mesclando recursivamente.
5. A partir das 3 etapas acima, temos um limite superior d de distância mínima. Agora precisamos considerar os pares de forma que um ponto do par venha da metade esquerda e o outro da metade direita. Considere a linha vertical passando por P $[\frac{n}{2}]$ e encontre todos os pontos cuja coordenada x está mais próxima do que d da linha vertical do meio. Construa um array strip de todos esses pontos.
6. Encontre a menor distância na faixa. Isso é complicado. À primeira vista, parece ser uma etapa O (n^2), mas na verdade é O (n). Pode-se provar geometricamente que, para cada ponto da faixa, só precisamos verificar no máximo 7 pontos após ele (observe que a faixa é classificada de acordo com a coordenada Y).
7. Por fim, retorne o mínimo de d e a distância calculada na etapa acima (etapa 6)

1.3 Divisão e conquista otimizado

Complexidade: $O(n * log(n))$
Código fonte: $src/algorithms/dc_optimized.py$

1.3.1 Otimização

Este algoritmo possui a mesma base que o algoritmo anterior, com uma única diferença, os pontos já estão ordenados a partir do eixo y, fazendo com que, durante a execução do passo 5 no algoritmo, o resultado possa ser encontrado em um tempo O(n).

2 Complexidade dos algoritmos

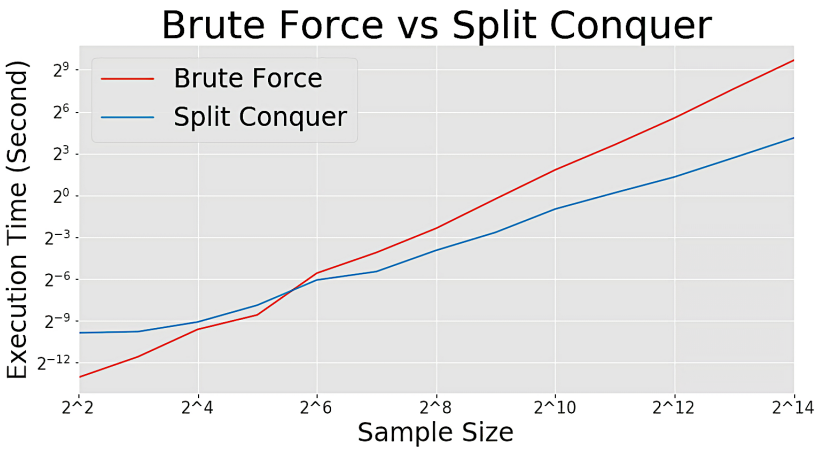

Figura 1: Fonte: Basic Algorithms — Finding the Closest Pair