Análise de Algoritmos e Técnicas: Ciclos em Grafos, *Backtracking* e *Branch and Bound*

Henrique Augusto Rodrigues[1]

[1]Instituto de Ciências Exatas e Informática - Departamento de Ciência da Computação
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

Resumo: *Divide and Conquer* é um paradigma de design de algoritmo, quebrando um problema maior em dois, ou, mais subproblemas, preferencialmente de tamanhos iguais, tornando-os simples de serem resolvidos diretamente, entrando na parte da conquista onde, concatena todos o resultados obtidos em um resultado do problema original.

Abstract: *Divide and Conquer it is a paradigm of algorithm design, breaking a bigger problem in two, or, more subproblems, preferably of equal sizes, making them simple to be solved directly, entering in the part of the conquest where, it concatenates all the obtained results in a result of the original problem.*

1. Visão Geral

Na disciplina de Projeto e Análise de Algoritmos, entender diferentes resoluções para um mesmo problema e diferentes técnicas para dadas soluções é de suma importância. Neste trabalho será feito uma análise do problema dos pares de pontos mais próximos, utilizando a abordage divisão e conquista. Para checar o código, vá a branch 'Closest Pair of Points' *Github*.

2. Arquitetura da máquina

Chip: Apple M1 Memory: 8 GB macOS: Ventura 13.01

2.1. Divisão e Conquista

Divisão e Conquista é uma técnica que pode ser dividida em três partes fundamentais: dividir um problema maior recursivamente em problemas menores(Dividir), resolver todos os sub-problemas (Conquistar) e, por fim, a solução do problema inicial é dada através da combinação dos resultados de todos os problemas menores computados(Combinar). Problemas que usam divisão e conquista são característiscos em possuir um princípio chamado Overlapping subproblems, ou seja, o problema pode ser divido em subproblemas que são reutilizados várias vezes não gerando novos subproblemas. Exemplos de problemas conhecidos que usam desta estratégia são o problema de ordenação interna(usando quicksort ou mergesort) e o problema dos pares de pontos próximos, esse último abordado no trabalho que, se encontra na subsecão a seguir.

2.2. Pares de Pontos mais Próximos

Dado um conjunto de N pontos em um espaço, é preciso encontrar os dois pontos do conjunto que possuem a menor distância um do outro. Este problema possui mais de uma implementação, porém uma das mais eficientes é utilizando divisão e conquista. Que consiste em, dividir o plano ao meio [n/2] ¡ i ¡ [n/2], ao se aproximar de i (meio do plano)

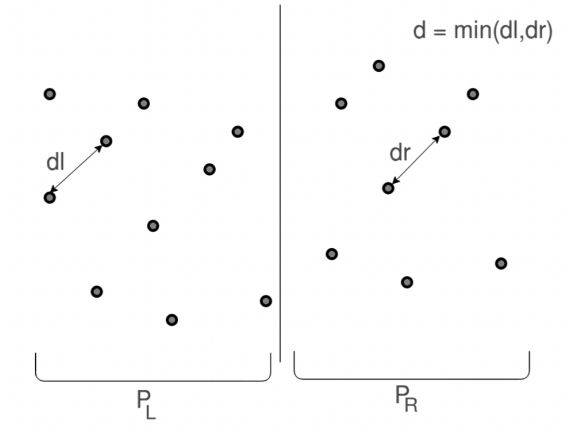

Figure 1. Representação menor distância entre dois pontos

analisaremos a distância entre os pontos mais próximos da esquerda para a direita e, seu correlato até o problema ser resolvido.

A entrada para o problema é um conjunto de pontos com um dos eixos já ordenado, encontramos o ponto do meio, separamos o conjunto ao meio, e então recursivamente descobrimos as menores distâncias entre os conjuntos separados, para então achar a solução geral que queremos.

3. Resultados

Utilizando 6 pares de pontos no plano cartesiano (2D), e fazendo o comparativo com as instâncias dadas no enunciado do problema.

3.1. Programa Original

0.00s user 0.00s system 0/100 cpu 0.590 total

3.2. Programa com 1000 instâncias

0.00s user 0.00s system 58/100 cpu 0.008 total

3.3. Programa com 100000 instâncias

0.00s user 0.00s system 51/100 cpu 0.002 total

3.4. Programa com 1000000 instâncias

0.00s user 0.00s system 56/100 cpu 0.003 total

4. Conclusão

O paradigma da divisão e conquista, resolve inúmeros problemas maiores em um tempo da execução satisfatório contendo poucas instâncias, a partir do momento em que esse número sobe, começa a exigir mais do sistema e tais resultados começam a variar de tal forma que, é preciso uma análise mais aprofundada para afirmar se, o aumento do número de instâncias são computadas em sua totalidade, ou, chega em um ponto que causa uma falha no sistema e não segue adiante na procura dos pares de pontos mais próximos.

5. Contribuições

O autor (Henrique Augusto Rodrigues), construiu, compilou, executou e analisou os resultados do programa e também, construiu o texto.