Relatório sobre o Merge Sort

Autor: Heitor Tonel Ventura (2086883)
Data: 23/05/2019

Introdução

Para a realização do trabalho 2, foi necessário pesquisar e estudar o algoritmo de ordenação MergeSort para conseguir implementá-lo e entendê-lo. Neste relatório há uma explicação com base no pesquisado de como funciona esse método de ordenação de listas.

O *Mergesort*

Informações Básicas

O *mergesort* é um método de ordenação do tipo *divide and conquer*, em que o vetor é dividido em partes e ordenado separadamente, após isso, ele é novamente único. Existem implementações desse algoritmo de maneira recursiva e iterativa, sendo a primeira a mais popular.

Vantagens

As principais vantagens são: o algoritmo é naturalmente estável; é normalmente mais veloz comparado com algoritmos mais simples como o *insertion sort;* permite fácil implementação multithread por ser um método *divide and conquer;* pior caso excelente com algoritmos velozes como o *quicksort;* combina bem com listas encadeadas.

Desvantagens

As principais desvantagens são: o custo de memória adicional par a armazenar vetores auxiliares é grande, comparado com o *quicksort* que consegue realizar tudo *in-place*; tende a ser menos veloz que o *quicksort* em média, mesmo com seu pior caso excepcional; no geral, não combina tão bem com listas sequnciais quanto o *quicksort.*

Quicksort e Mergesort

Tanto o *quicksort* quanto o *mergesort* são recursivos naturalmente, e portanto podem ter problemas de stack overflow, onde a pilha recursiva fica tão grande até o ponto de não conseguir ser armazenada, gerando o erro, mas isso só ocorrerá em uma quantidade de dados extrema, já que a quantidade de divisões recursivas varia numa função logarítimica (aproximadamente 33 divisões para uma lista com 10 bilhões de elementos).

O mergesort é significativamente superior quando o objetivo é ordenar listas encadeadas, pois neste caso, o algoritmo consegue ser feito sem necessidade de vetores auxiliares.

No geral, a escolha entre o *quicksort* e o *mergesort* tende ao lado do primeiro com modificações para evitar seu pior caso quadrático, a não ser que o objetivo seja ordenar listas encadeadas, onde o segundo consegue ser melhor caso a opção for de ordenação for implementada diretamente.

Como funciona o algoritmo

Algoritmo de Merge 2-way (A base do mergesort tradicional)

Objetivo: Dado dois vetores ordenados, unir eles de maneira que o resultado fique ordenado.

Procedimento 1: Comparar o elemento i do vetor 1 com o elemento j do vetor 2; o vencedor terá seu contador aumentado e seu valor adicionado à lista final. Isso será realizado até que um dos vetores chegue ao fim.

Procedimento 2: Verificar se alguma das listas ainda possui elementos. Caso positivo, adicioná-los ao fim da lista final.

Exemplo: merge 2-way de [1, 6] e [3, 7]

1) Compara 1 e 3 -> 1 ganha -> lista final = [1]
2) Compara 6 e 3 -> 3 ganha -> lista final = [1, 3]
3) Compara 6 e 7 -> 6 ganha -> lista final = [1, 3, 6] -> fim da primeira lista
4) Checar se a primeira lista ainda tem elementos -> não tem
5) Checar se segunda lista ainda tem elementos -> tem (7) -> adicioná-los → lista final = [1, 3, 6, 7]
6) Fim do algoritmo

Funcionamento do *mergesort*

Objetivo: Dado uma lista, dividi-la recursivamente e reconstrui-la a partir de múltiplas aplicações do algoritmo de merge.

Exemplo com a lista sequencial V = [5, 9, 3, 1, 3, 7, 8, 4], n = 8

Procedimento 1 - Dividir Recursivamente:

- Achar o meio e dividir em dois subvetores
- Isso será realizado até que se atinja um subvetor ordenado, ou seja, se só houver um elemento dentro dele.

log2 n = 3, ou seja, três divisões;

-> [5, 9, 3, 1, 3, 7, 8, 4] - inicial
-> [5, 9, 3, 1][3, 7, 8, 4] - camada 1
-> [5, 9][3, 1][3, 7][8, 4] - camada 2
-> [5][9][3][1][3][7][8][4] - camada 3

Procedimento 2 - Merging:

- Aplicar o algoritmo de merge 2-way em cada cadeia recursiva;
- A escolha de ir da esquerda para a direita é arbitrária, normalmente o algoritmo é executado da direita para a esquerda;

-> [5][9][3][1][3][7][8][4] - camada 3
    -> [5, 9][3][1][3][7][8][4]
    -> [5, 9][1, 3][3][7][8][4]
    -> [5, 9][1, 3][3, 7][8][4]
    -> [5, 9][1, 3][3, 7][4, 8]
-> [5, 9][1, 3][3, 7][4, 8] - camada 2
    -> [1, 3, 5, 9][3, 7][4, 8]
    -> [1, 3, 5, 9][3, 4, 7, 8]
-> [1, 3, 5, 9][3, 4, 7, 8] - camada 1
    -> [1, 3, 3, 4, 5, 7, 8, 9]
-> [1, 3, 3, 4, 5, 7, 8, 9] - camada inicial

Resumo Final da execução:

-> ([5] merge [9]) ([3] merge [1]) ([3] merge [7]) ([8] merge [4]) - camada 3
-> ([5, 9] merge [1, 3]) ([3, 7] merge [4, 8]) - camada 2
-> [1, 3, 5, 9] merge [3, 4, 7, 8] - camada 1
-> [1, 3, 3, 4, 5, 7, 8, 9] - camada inicial

Retorno: [1, 3, 3, 4, 5, 7, 8, 9]

Conclusão

O *mergesort* é um excelente algoritmo de ordenação, com estabilidade de resultados e eficiência estável naturalmente. Tem uma implementação simples e permite facilmente o uso de multithreading, tornando-se relativamente escalável. Possui algumas falhas críticas que o tornam inferior ao *quicksort* na maioria dos casos, mas tem seu espaço como algoritmo principal para ordenação direta de listas encadeadas. Infere-se que o *mergesort* possui utilidade tanto acadêmica e educacional quanto prática, e portanto é de importante estudo.

Referências

Em geral, o ambiente principal de pesquisa foram artigos na internet e fóruns de discussão sobre programação e desenvolvimento. Abaixo, alguns links relevantes que deram auxílio na produção deste relatório.

Informações básicas: https://pt.wikipedia.org/wiki/Merge_sort
Implementações e visualização: https://www.geeksforgeeks.org/merge-sort/
Explicação mais detalhada: https://pt.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/overview-of-merge-sort

Merge sort para listas encadeadas: https://www.geeksforgeeks.org/merge-sort-for-linked-list/
Compreensão do algoritmo de merge: https://www.youtube.com/watch?v=mB5HXBb_HY8