unipac.br
Barbacena

Bacharelado em Ciência da Computação

Estruturas de Dados

Material de Apoio

Parte XIII – *Método MergeSort*

Prof. Nairon Neri Silva
naironsilva@unipac.br

2º sem / 2021

Abordagem Dividir-para-Conquistar

- Método de ordenação que consiste em:
  - Dividir a entrada em conjuntos menores
  - Resolver cada instância menor de maneira recursiva
  - Reunir as soluções parciais para compor a solução do problema original.

Ordenação por Mergesort

- Princípio da Divisão e Conquista:
  - – Divida o vetor A em dois subconjuntos A1 e A2
  - – Solucione os sub-problemas associados a A1e A2. Em outras palavras, ordene cada subconjunto separadamente (chamada recursiva - recursão encerra quando atinge sub-problemas de tamanho 1)
  - – Combine as soluções de A1 e A2 em uma solução para A: intercale os dois sub-vetores A1 e A2 e obtenha o vetor ordenado
- Operação chave: intercalação.

Exemplo - Mergesort

Entrada: 47 26 33 05 99 38 64 15

1. Divide: 47 26 33 05 | 99 38 64 15
2. Resolve Recursivamente:
   - Primeira metade 47 26 33 05
     (Divide, Resolve recursivamente, Intercala
     Obtendo 05 26 33 47)
   - Segunda metade 99 38 64 15
     (Divide, Resolve recursivamente, Intercala
     Obtendo 15 38 64 99)
3. Intercala as soluções parciais:

   05 15 26 33 38 47 64 99

Algoritmo Mergesort

```
void merge(int vetor[], int comeco, int meio, int fim) {
    int com1 = comeco, com2 = meio+1, comAux = 0, tam = fim-comeco+1;
    int *vetAux;
    vetAux = (int*)malloc(tam * sizeof(int));

    while(com1 <= meio && com2 <= fim){
        if(vetor[com1] < vetor[com2]) {
            vetAux[comAux] = vetor[com1];
            com1++;
        } else {
            vetAux[comAux] = vetor[com2];
            com2++;
        }
        comAux++;
    }
     //continua no próximo slide
```

Algoritmo Mergesort

```
    while(com1 <= meio){
        //Caso ainda haja elementos na primeira metade
        vetAux[comAux] = vetor[com1];
        comAux++;
        com1++;
    }
    while(com2 <= fim) {
        //Caso ainda haja elementos na segunda metade
        vetAux[comAux] = vetor[com2];
        comAux++;
        com2++;
    }
    for(comAux = comeco; comAux <= fim; comAux++){
        //Move os elementos de volta para o vetor original
        vetor[comAux] = vetAux[comAux-comeco];
    }
    free(vetAux);
}
```

Algoritmo Mergesort

```
void MergeSort(int vetor[], int comeco, int fim){
    if (comeco < fim) {
        int meio = (fim+comeco)/2;
        MergeSort(vetor, comeco, meio);
        MergeSort(vetor, meio+1, fim);
        merge(vetor, comeco, meio, fim);
    }
}
```

Algoritmo Mergesort

- Vantagens
  - Algoritmo eficiente
  - Indicado para aplicações que exigem restrição de tempo (executa sempre em um determinado tempo para um dado n)
  - Passível de ser transformado em estável
    - Implementação de intercala
  - Fácil implementação
- Desvantagens
  - Utiliza memória auxiliar

MergeSort - Exercício

Ordene o seguinte conjunto de dados usando o algoritmo MergeSort:

V = {75, 63, 7, 84, 3, 2, 4, 0, 9}

Referências

- FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. *Lógica de Programação*. Makron books.
- GUIMARAES, Angelo de Moura; LAGES, Newton Alberto Castilho. *Algoritmos e estruturas de dados*. LTC Editora.
- FIDALGO, Robson. *Material para aulas*. UFRPE.
- NELSON, Fábio. *Material para aulas: Algoritmo e Programação.* UNIVASP.
- FEOFILOFF, P., *Algoritmos em linguagem C*, Editora Campus, 2008.
- ZIVIANI, N., *Projeto de algoritmos com Implementações em Pascal e C*, São Paulo: Pioneira, 2d, 2004.
- http://www.ime.usp.br/~pf/algoritmos/
- MELLO, Ronaldo S., *Material para aulas: Ordenação de Dados*, UFSC-CTC-INE
- MENOTTI, David, *Material para aulas: Algoritmos e Estrutura de Dados I*, DECOM-UFOP