*UNICID – Universidade da Cidade de São Paulo*

Avaliação – Análise Complexidade

GRUPO TURMA CCA 04

Membros:

- Leonardo Couto – RGM 22741241 – 6° SEMESTRE – TURMA B

- Gustavo Gabriel de Almeida – RGM 19027265 – 6°SEMESTRE – TURMA B

- Clarice Braga Tolentino – RGM 15040666 – 6°SEMESTRE – TURMA B

- Lucas Veiga Sgrott – RGM 18897983 – 6°SEMESTRE - TURMA B

- Pedro Bastos Constantino Junior – RGM 24502995 – 8° SEMESTRE - TURMA B

São Paulo, 26 de outubro de 2020

Problema - 01

Melhor Caso – Elemento na primeira posição

R: O (1)

Caso Médio – Elemento se encontra em N / 2

R: O(N)

Pior Caso – Elemento na ultima posição do Vetor

R: O(N)

Problema – 02

Melhor Caso – Elemento se encontra em N / 2

R: O (1)

Caso Médio – Elemento na posição inicial ou no final do vetor

R: O (log n)

Pior Caso – Elemento não encontrado

R: O (log n)

Problema - 03

R1: O (1)

R2: O(n)

Problema - 04

R: O(n3)

Problema – 05

Equação de Recorrência: T(n) =1, se (N = 1)

Equação de Recorrência: T(n) = T (n - 1) + n – 1, se (N > 1)

R1: Complexidade O(n²)

R2: 49995000 comparações feitas para n=10.000.

Problema – 06

Equação de Recorrência: T(n) =1 SE (N = 1)

Equação de Recorrência: T(n) =2T (n/ 2) + n SE (N > 1)

R1: Complexidade O(c) se n = 1

R2: Divide-se o vetor V em duas metades recursivamente, cada metade é ordenadas e ambas são intercaladas criando um novo vetor ordenado, paralelamente usa um outro vetor para auxiliar a intercalação.

Complexidade: O (n log(n)) – Pior Caso

Quantidade de Ordenações: 19999 para n = 10.000

Problema – 07

R1: O(n) (Melhor Caso)

Equação de Recorrência: T(n) = 2T (n / 2) + O(n) (Melhor Caso)

Equação de Recorrência: T(n) = T (n – 1) + O(n²) (Pior caso)

Quantidade de Trocas: 20.000 para n = 10.000

R2: Na primeira chamada recursiva temos o número inicial como pivô. O processo segue em diante. Verifica-se se 0 é menor que o pivô, então troca-se i e J, Quando I ultrapassa J então troca-se o pivô por Vetor(j)

Logo mais, o pivô ocupa a posição corretamente, o que nos resta duas metades para serem ordenadas, então aplica-se o algoritmo mais uma vez recursivamente nas duas metades restantes.

R3: O melhor caso ocorre quando as duas metades são balanceadas, ou seja, cada partição com n divido por 2 elementos.

Algoritmos usados na pesquisa (C#): Merge Sort, Quick Sort e Algoritmos de ordenação linear.

Referências bibliográficas:

BLOG CYBERINI. Quicksort, análise e implementações. Disponível em: <https://www.blogcyberini.com/2018/08/quicksort-analise-e-implementacoes.html>. Acesso em: 18 de out. de 2020.

IME USP. MergeSort, análise de algoritmos. Disponível em: <https://www.ime.usp.br/~pf/analise\_de\_algoritmos/aulas/mergsrt.html>. Acesso em: 18 de out. de 2020.

WIKIPEDIA. Pesquisa binária. Disponível em: <https://pt.wikipedia.org/wiki/Pesquisa\_bin%C3%A1ria#An%C3%A1lise\_de\_Complexidade>. Acesso em: 18 de out. de 2020.

COMPLEXIDADE DE ALGORITMOS. Sequências e conjuntos. Disponível em: <http://www.faccamp.br/osvaldo/SequenciasConjuntos.pdf>. Acesso em: 19 de out. de 2020.

REPOSITÓRIO GITHUB: https://github.com/mekinus/pesquisa-complexidade