Projeto e Análise de Algoritmos

# Universidade Federal do Pampa

Nome: Felipe Antunes Quirino

Matrícula: 1701560422

Analise a complexidade dos algoritmos abaixo.

1. Dado um conjunto de elementos inteiros V com tamanho N, o algoritmo abaixo insere e verifica se o elemento V[i] pertence a uma segunda estrutura de dados Z.

int algoritmo1(int V[], int N){

for(int i = 0;i <= N;i++) {

inserir(V[i], Z);

}

for(int i = 0; i <= N; i++)

{

buscar(V[i], Z);

}

}

* 1. Qual a complexidade de pior caso do algoritmo se **Z** é um vetor não ordenado?

O (N^2), por causa das N buscas em que cada uma demora um tempo N

* 1. Qual a complexidade de pior caso do algoritmo se **Z** é uma árvore binária? E de melhor caso?

O pior é O (N^2), o melhor é Nlog(N)

* 1. Qual a complexidade do algoritmo se **Z** é uma árvore AVL?

Nlog(N)

* 1. Qual a complexidade de pior caso do algoritmo se **Z** é uma Hash?

Depende do hash. Considerando o set do python como em <https://wiki.python.org/moin/TimeComplexity>

O pior caso é O(N^2), pois o pior caso de busca é O(N).

1. Dado um conjunto de elementos inteiros V com tamanho N, o algoritmo abaixo imprimi em ordem crescente os elementos de V. Qual a complexidade de pior caso do algoritmo?

O(Nlog(N)), na hora de ordenar o vetor, caso aplicado o merge sort, por exemplo. Ou (max\_element(V), caso seja aplicado a ordenação em tempo linear.

int algoritmo2(int V[], int N){

ordenar(V);

for(int i = 0; i <= N; i++)

{

printf("%d”, V[i]);

}

}

1. Qual a complexidade de pior caso e melhor caso do algoritmo abaixo? Considere que **Z** é uma árvore binária.

A complexidade de pior caso é O(N^4) e melhor caso de O(N^3\*log(N))

int algoritmo3(int V, int N){ int i = N, j =0, k = 0; while(i >= 0){

k = obterValorMinimo(V); inserir(k, Z);

i--;

}

for(i = 0; i <= N; i++){

for(j = 0; j <= N; j++){

for(k = 0; k <= N; k++){

printf(“%d ”, buscar(Z, V[i]));

}

}

}

}