Universidade Estadual de Maringá

Centro de Tecnologia - Departamento de informática

Curso de Bacharelado em Informática

Projeto e Análise de Algoritmos

Profº. Dr. Rodrigo Calvo

Acadêmicos	RA
Luiz Flávio Pereira	91706
Marcos Vinícius Peres	94594

Maringá - PR

2018

Configurações do computador:

Para o desenvolvimento deste programa, foi utilizado o sistema operacional Windows 10, 64 bits Pt-br. O programa foi desenvolvido na linguagem de programação C, usando a Ide DEV C++, tendo como o compilador TDM-GCC 4.9.2 64 bits Release.

Relatório do programa:

Para encontrar o K-ésimo termo foi utilizado o método de Busca Binária Recursiva e posteriormente um método de ordenação chamado ***Counting Sort***.

O define TAM_MAX será o tamanho do vetor original.

Dentro do código main, é criado um vetor de tamanho TAM_MAX e é esperado do usuário que ele interaja com menu de opções que o programa fornece. A primeira opção do menu é a de carregar o vetor de TAM_MAX posições, que faz com que o usuário alimente o vetor com números inteiros **maiores que zero**. A segunda opção é o acha o K-esimo em T(n) = O(Log n), ou seja, achar o maior valor do vetor obedecendo a seguinte propriedade: A1 < A2 < ... < Ak > Ak+1 > Ak+2 > ... >. Tendo a terceira opção do nosso menu, achar o K-esimo em tempo de T(n) = O(n), vale ressaltar que essa opção é adicional e não a solicitada pela descrição do trabalho. Nossa quarta opção é a de ordenação, depois que o usuário preencheu o vetor, ele tem a opção de ordenar o vetor que foi preenchido. No programa usamos um algoritmo para ordenar os elementos em tempo O(n) chamado de ***Counting Sort***, um algoritmo que usa da contagem para sua ordenação.

Vale ressaltar que as opção 2, 3 e 4 do menu só serão executadas se o vetor estiver completamente preenchido com valores fornecido pelo usuário.

Lista de tratamentos de exceções do programa:

- Se inserir valor menor que 1, então encerra-se o carregamento do vetor e o programa voltará ao menu inicial;
- O programa não permite ordenar ou achar k-ésimo se o vetor não estiver complemente carregado;
- Se o usuário tentar inserir elementos mesmo que o vetor já estava totalmente cheio, retornará mensagem de erro e voltará ao menu inicial;
- Ao buscar o k-ésimo o programa valida se o vetor está inicia decrescente, se ele possui um trecho do programa com valores repetidos consecutivamente, verifica

Árvore de Recursão da Busca Binaria Recursiva: $T(n) = T\left(\frac{n}{2}\right) + \theta(1)$

$$T(n)$$

$$|$$

$$T\left(\frac{n}{2}\right)$$

$$|$$

$$T\left(\frac{n}{4}\right)$$

$$|$$

$$T\left(\frac{n}{8}\right)$$

$$\vdots$$

$$T(1) = \frac{n}{2^H} = 1$$

Deste modo a propriedade logarítmica nos diz que H = $log\, n$ e, portanto, temos que $a^H = n^{\log 1} = n^0 = 1$ número de folhas.

Todos os níveis possui custo constante, deste modo, a chamaremos de C.

Altura = H = $log\, n$

Custo total = C. $log\, n$, porém, podemos ignorar a constante C, porque tempos constantes não são levados em consideração na análise assintótica de algoritmos.

Deste modo, podemos afirmar que a busca binária recursiva possui $T(n) \in \theta(log\, n)$.

Método da substituição : $T(n) = T\left(\frac{n}{2}\right) + \theta(1)$

Palpite: θ(log n)

Para n = 1, temos:

$n^{\log 1} = 1$, verdadeiro.

Para um n > 1, temos:

$$1.\left(\log\frac{n}{2}\right)+1$$

$$1(log\,n-log\,2)+1$$

$$log\,n-1+1$$

$$log\,n$$

Assim comprovamos que o palpite da busca binaria recursiva é verdadeiro ao aplicá-lo no método de substituição, portanto, provou-se por definitivo que a busca binária recursiva possui $T(n) \in \theta(log\,n)$.