IMD0029 – ESTRUTURAS DE DADOS BÁSICAS I

PROF. EIJI ADACHI M. BARBOSA

Lista de Exercícios – Busca Binária

Questão 1. Dado um inteiro X e um array A contendo N inteiros distintos, o problema *Soma-2* é definido como o problema de determinar se existem dois índices i e j tais que A[i] + A[j] == X. Neste contexto, projete duas soluções para o problema *Soma-2* com complexidades $\Theta(n^2)$ e $\Theta(n*lg(n))$.

Questão 2: Considere um array A contendo N inteiros, com possíveis repetições, já ordenado. Neste contexto:
(A) Projete um algoritmo que recebe como entrada um inteiro K e retorna um índice i tal que A[i] == K, sendo A[i] o elemento igual a K que está mais a esquerda em A. Sua solução deverá ter complexidade Θ (lg(n)).
(B) Projete um algoritmo que recebe como entrada um inteiro K e retorna quantos elementos em A tem valor igual a K ocorrem no array A. Sua solução deverá ter complexidade Θ (lg(n)).

Questão 3: Projete um algoritmo de busca ternária, isto é, um algoritmo de busca que divide o seu espaço de busca em três. Faça duas versões da busca ternária: uma iterativa e outra recursiva. Qual a complexidade do algoritmo de busca ternária?

Questão 4: Sequências bitônicas são aquelas que possuem duas sequências, sendo uma sequência inicial crescente, seguida de uma sequência decrescente. Ou seja, os elementos de uma sequência bitônica obedecem a seguinte relação:

$$A0 < ... < Ai-1 < Ai > Ai+1 > ... > An$$

Considere que um array bitônico é um array de inteiros sem repetições cujos elementos representam uma sequência bitônica. Dado este contexto:

A) Projete um algoritmo que recebe como entrada um array de inteiros e verifica se o array é bitônico ou não.
B) Projete um algoritmo que recebe como entrada um array bitônico e retorna o índice do pico, isto é, o elemento Ai. Sua solução deverá ter complexidade Θ (lg(n)).
C) Projete um algoritmo que recebe como entrada um array bitônico A e um número inteiro K e retorna o índice i tal que A[i] == K. Sua solução deverá ter complexidade Θ (lg(n)).

Questão 5. Considere que um array inicialmente ordenado foi deslocado a direita sem querer e não se sabe quantas vezes ele foi deslocado a direita. Um deslocamento a direita de um array faz com que o último elemento do array torne-se o primeiro. Por exemplo: dado o array de entrada {1,2,3,4,5,6,7,8,9,10}, após 3 deslocamentos a direita, o

array estará da seguinte forma: {8,9,10,1,2,3,4,5,6,7}. Neste contexto:

A) Projete um algoritmo que recebe como entrada um array A que foi deslocado a direita e retorna o número de vezes que este array foi deslocado a direita. Sua solução deverá ter complexidade Θ (lg(n)).

B) Projete um algoritmo que recebe como entrada um array A que foi deslocado a direita e um inteiro K e retorna o índice i tal que A[i] == K. Sua solução deverá ter complexidade Θ (lg(n)).

Questão 6. O auto-complemento (do inglês autocomplete) é uma funcionalidade em que uma aplicação tenta "prever" o restante de uma palavra que o usuário está digitando. Nesta questão, você deverá implementar uma função que será reusada no contexto de uma aplicação com a funcionalidade de auto-complemento. A função a ser implementada baseia-se em dois parâmetros: um dicionário de palavras (Strings) e um prefixo, que é uma String representando os caracteres digitados pelo usuário até o momento. Dado um dicionário de palavras e um prefixo, a sua função deverá retornar um par de inteiros (x, y): o primeiro inteiro – x – indica o índice da primeira palavra no dicionário que começa com o prefixo passado como parâmetro e o segundo inteiro – y – indica o índice da última palavra no dicionário que começa com o prefixo. Caso nenhuma palavra no dicionário comece com um determinado prefixo, a função deverá retornar (-1, -1). Sua função deve respeitar a seguinte assinatura:

```
(int, int) SearchPrefix( String dict[N], String prefix )
```

Para a função acima, a eficiência é essencial. Desta forma, sua solução deverá obrigatoriamente ter complexidade Θ(lg(N)), em que N é o tamanho do vetor de entrada "dict".

Questão 7. Projete um algoritmo que recebe como entrada um array de inteiros v[n] com possíveis valores repetidos e retorna o número mais frequente, isto é, o valor que tem o maior número de repetições no array. Caso haja empate, isto é, caso existam dois valores com o mesmo número de repetições, este algoritmo deve retornar o valor que aparece primeiro da esquerda para direita do vetor. Esta solução deve ter complexidade Θ (n*log2n). Dica: Reuse a função da Questão 2.