SSC0503 - Introdução à Ciência de Computação II
Respostas da 6ª Lista
Professor: Claudio Fabiano Motta Toledo (`claudio@icmc.usp.br`)
Estagiário PAE: Jesimar da Silva Arantes (`jesimar.arantes@usp.br`)

Resposta Pergunta 4: Descreva sobre quais contextos é mais adequado o uso a busca binária do que a busca sequencial e vice-versa.

A utilização da busca binária é mais adequada quando se tem dados que já se encontram ordenados. Caso os dados não estejam ordenados, no entanto, deseja-se efetuar muitas buscas sobre tais dados, então pode ser interessante ordenar os dados e então aplicar a busca binária. A seguinte expressão descreve a partir de quantas buscas efetuadas torna-se interessante ordenar os dados e então aplicar a busca binária ao invés de apenas buscar pela busca sequencial. Seja x o número de buscas efetuadas e n o tamanho do vetor de dados. Nesse exemplo assume-se que o método de ordenação aplicado seja o merge sort com complexidade $O(n \cdot lg\ n)$.

$x \cdot ComplexBuscaSequencial \geq ComplexOrdenar + x \cdot ComplexBuscaBinaria$
$x \cdot n \geq n \cdot lg\ n + x \cdot lg\ n$
$x \cdot n - x \cdot lg\ n \geq n \cdot lg\ n$
$x \cdot (n - lg\ n) \geq n \cdot lg\ n$
Assumindo que $(n - lg\ n) \approx n$, para valores grandes de n.
$x \cdot n \geq n \cdot lg\ n$
$x \geq \frac{n \cdot lg\ n}{n}$
$x \geq lg\ n$

Dessa forma, quando desejar efetuar mais do que $lg\ n$ buscas então vale a pena ordenar os dados e aplicar a busca binária. Nesse sentido, para um vetor de $n = 1.000.000$, e caso deseja-se fazer mais do que 20 buscas ($lg\ 1.000.000 = 20$) vale a pena ordenar os dados.

Já a utilização da busca sequencial é mais adequada quando os dados não estão ordenados.

Resposta Pergunta 5: Desenvolva um programa em C que faça a busca sequencial iterativa sobre um vetor de tamanho N. Em seguida, diga qual a análise de complexidade no melhor caso, pior caso e caso médio.

```c
int busca_sequencial_iterativa(int vet[], int key, int size) {
    int i = 0;
    while ((i < size) && (vet[i] != key)) {
        i++;
    }
    if (i >= size) {
        return -1;
    } else if (vet[i] == key) {
        return i + 1;
    }
    return -1;
}
```

Listing 1: Resposta do exercício 5 codificado na linguagem C

Análise de Pior Caso: $O(n)$
Análise de Melhor Caso: $O(1)$
Análise de Caso Médio: $O(n/2) = O(n)$

Resposta Pergunta 6: Desenvolva um programa em C que faça a busca sequencial recursiva sobre um vetor de tamanho N. Em seguida, diga qual a análise de complexidade no melhor caso, pior caso e caso médio.

```
int busca_sequencial_recursiva(int vet[], int key, int size, int i) {
    if (i >= size) {
        return -1;
    } else if (vet[i] == key) {
        return i + 1;
    } else {
        return busca_sequencial_recursiva(vet, key, size, i+1);
    }
}
```

Listing 2: Resposta do exercício 6 codificado na linguagem C

Análise de Pior Caso: $O(n)$
Análise de Melhor Caso: $O(1)$
Análise de Caso Médio: $O(n/2) = O(n)$

Resposta Pergunta 8: Desenvolva um programa em C que faça a busca binária recursiva sobre um vetor de tamanho N. Em seguida, diga qual a análise de complexidade no melhor caso, pior caso e caso médio.

```
int busca_binaria_recursiva(int vet[], int key, int start, int end) {
    int middle = (start+end)/2;
    if (start > end) {
        return -1;
    } else if (vet[middle] == key) {
        return middle + 1;
    } else if (vet[middle] > key) {
        return busca_binaria_recursiva(vet, key, start, middle-1);
    } else {
        return busca_binaria_recursiva(vet, key, middle+1, end);
    }
}
```

Listing 3: Resposta do exercício 8 codificado na linguagem C

Análise de Pior Caso: $O(lg\ n)$
Análise de Melhor Caso: $O(1)$
Análise de Caso Médio: $O(lg\ n)$

Resposta Pergunta 9: Desenvolva um programa em C que faça a busca ternária recursiva sobre um vetor de tamanho N. Em seguida, diga qual a análise de complexidade no melhor caso, pior caso e caso médio.

```
int busca_ternaria_recursiva(int vet[], int key, int start, int end) {
```

```
    int middle1 = start + (end - start)/3;
    int middle2 = start + 2*((end - start)/3);
    if (start > end) {
       return -1;
    } else if (vet[middle1] == key) {
          return middle1 + 1;
    } else if (vet[middle2] == key) {
          return middle2 + 1;
    } else if (vet[middle1] > key) {
          return busca_ternaria_recursiva(vet, key, start, middle1 - 1);
    } else if (vet[middle2] < key ) {
          return busca_ternaria_recursiva(vet, key, middle2 + 1, end);
    } else {
          return busca_ternaria_recursiva(vet, key, middle1 + 1, middle2 - 1);
    }
}
```

Listing 4: Resposta do exercício 9 codificado na linguagem C

Análise de Pior Caso: $O(log\ n)$
Análise de Melhor Caso: $O(1)$
Análise de Caso Médio: $O(log\ n)$

Resposta Pergunta 11: Desenvolva um programa em C que faça a busca por interpolação recursiva sobre um vetor de tamanho N. Em seguida, diga qual a análise de complexidade no melhor caso, pior caso e caso médio.

```
int busca_interpolacao_recursiva(int vet[], int key, int start, int end) {
 if (vet[end] == vet[start]) {
       return -1;
    }
    int middle = (int)(start + (end - start) * (1.0*key - vet[start])/(vet[end
   ]-vet[start]));
    if (middle > end || start > end) {
       return -1;
    } else if (vet[middle] == key) {
       return middle + 1;
    } else if (vet[middle] > key) {
       return busca_interpolacao_recursiva(vet, key, start, middle-1);
    } else {
       return busca_interpolacao_recursiva(vet, key, middle+1, end);
    }
}
```

Listing 5: Resposta do exercício 11 codificado na linguagem C

Análise de Pior Caso: $O(n)$
Análise de Melhor Caso: $O(1)$
Análise de Caso Médio: $O(log\ log\ n)$

Resposta Pergunta 14: Qual a expressão que descreve a relação de recorrência da busca binária. Prove usando o método mestre a complexidade da busca binária.

$T(n) = T(n/2) + 1$
Aplicando o teorema mestre:
$T(n) = aT(n/b) + f(n)$
Nesse sentido temos:
$a = 1$
$b = 2$
$f(n) = 1$
Como $a \geq 1$, $b > 1$ e $f(n)$ é assintoticamente positivo, logo podemos aplicar o método mestre.
Verificando cada caso temos:
Caso 1:
$f(n) = O(n^{log_b a - \epsilon})$
$1 = O(n^{log_2 1 - \epsilon})$
$1 = O(n^{0 - \epsilon})$
$1 = O(n^{-\epsilon})$
Falso
Caso 2:
$f(n) = \Theta(n^{log_b a})$
$1 = \Theta(n^{log_2 1})$
$1 = \Theta(n^0)$
$1 = \Theta(1)$
Verdade
Logo, $T(n) = \Theta(n^{log_b a} \cdot lg\ n) = \Theta(n^0 \cdot lg\ n) = \Theta(1 \cdot lg\ n) = \Theta(lg\ n)$
Caso 3:
$f(n) = \Omega(n^{log_b a + \epsilon})$
$1 = \Omega(n^{log_2 1 + \epsilon})$
$1 = \Omega(n^{0 + \epsilon})$
$1 = \Omega(n^{\epsilon})$
Falso

Resposta Pergunta 17: Suponha que tenhamos números entre 1 e 1000 em uma árvore binária de busca e que desejamos buscar pelo número 363. Qual das seguintes sequências não poderia ser a sequência de nós examinados?

- (E) 925, 202, 911, 240, **912**, 245, 363.

O nó 912 viola a condição de formação da árvore binária de busca.