

Análise de Algoritmos
Prof. Marcelo Keese Albertini
Faculdade de Computação - Universidade Federal de Uberlândia
Lista de exercícios 3: Escrita e análise de recorrências

1. Escreva e analise a relação de recorrência que conta o número de vezes que a função *fibonacci()* é chamada no código a seguir.

```
1 int fibonacci(int N) {  
2     if (N == 0 || N == 1) return N;  
3     return fibonacci(N-1)+fibonacci(N-2);  
4 }
```

2. Escreva e analise a relação de recorrência que conta o número de vezes que a função *moveHanoi()* é chamada no código a seguir.

```
1 // Exemplo chamada inicial: moveHanoi(10, "A", "B", "C");  
2 void moveHanoi(int n, String ini, String aux, String fim) {  
3     if (n == 1) System.out.println(ini + " -> " + fim);  
4     else {  
5         moveHanoi(n - 1, ini, fim, aux);  
6         System.out.println(ini + " -> " + fim);  
7         moveHanoi(n - 1, aux, ini, fim);  
8     }  
9 }
```

3. Escreva e analise a relação de recorrência que descreve o principal custo do pior caso da seguinte solução do problema: "Receba um vetor, quase ordenado, que pode ser ordenado com apenas uma troca. Faça essa troca."

```
1 // Exemplo. Entrada: 2 5 7 6 8 9. Saída: 2, 5, 6, 7, 8, 9.  
2 void ordenaComUmaTroca(int v[]) {  
3     int x = -1, y = -1;  
4     int prev = v[0];  
5  
6     for (int i = 1; i < v.length; i++) {  
7         if (prev > v[i]) {  
8             if (x == -1) {  
9                 x = i - 1; y = i; // achou 1 fora do lugar  
10            } else y = i; // achou o outro  
11        }  
12        prev = v[i];  
13    }  
14    int aux = v[x]; v[x] = v[y]; v[y] = aux;  
15 }
```

4. Escreva e analise a relação de recorrência que descreve o principal custo do pior caso da seguinte solução do problema: "encontrar, se houver, par de inteiros cuja soma é procurada".

```
1 // Exemplo. Entrada: v=[8,7,2,5,3,1], soma = 10. Saída: par é 8 e 2.  
2 void acharPar(int v[], int soma) {  
3     for (int i = 0; i < v.length - 1; i++)  
4         for (int j = i + 1; j < v.length; j++)  
5             if (v[i] + v[j] == soma) {  
6                 System.out.println("par é " + v[i] + " e " + v[j]);  
7                 return;  
8             }  
9 }
```

5. Escreva outra solução do problema anterior que tenha custo assintótico $o(n^2)$ (apresente código, relação de recorrência e análise). Dica: existem soluções em tempo proporcional a $n \log n$ (usando ordenação) e a n (usando tabela Hash, com custo de espaço maior).

6. Considere o problema: “Dada uma sequência ordenada de números tal que a diferença entre termos consecutivos é constante, encontre o número faltando.” Escreva uma solução que executa, no pior caso, em um número de operações $\Theta(n)$. Escreva e analise a correspondente relação de recorrência.

```
1 Exemplo. Entrada: 5,7,9,11,15. Saída: 13.
2 int numeroFaltando(int v[]) {
3     // preencha aqui..
4 }
```

7. Escreva uma solução para o problema anterior que executa, no pior caso, em um número de operações $o(n)$. Escreva e analise a correspondente relação de recorrência. Dica: considere o algoritmo de busca binária.

8. Escreva as recorrências que representam o número médio de recursões e o número médio de trocas no seguinte Quicksort.

```
1 void quicksort(int[] a, int lo, int hi) {
2     if (hi <= lo) return;
3
4     int i = lo-1, j = hi;
5     int t, v = a[hi];
6
7     while (true) {
8         while (a[++i] < v) ;
9         while (v < a[--j]) if (j == lo) break;
10
11        if (i >= j) break;
12        t = a[i]; a[i] = a[j]; a[j] = t;
13    }
14    t = a[i]; a[i] = a[hi]; a[hi] = t;
15
16    quicksort(a, lo, i-1);
17    quicksort(a, i+1, hi);
18 }
```