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 que vezes que a função $fibonacci()$ é chamada no código a seguir.

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

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

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

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."

```
// Exemplo. Entrada: 2 5 7 6 8 9. Saída: 2, 5, 6, 7, 8, 9.
void ordenaComUmaTroca(int v[]) {
  int x = -1, y = -1;
  int prev = v[0];

  for (int i = 1; i < v.length; i++) {
    if (prev > v[i]) {
      if (x == -1) {
              x = i - 1; y = i; // achou 1 fora do lugar
      } else  y = i; // achou o outro
    }
    prev = v[i];
  }
  int aux = v[x]; v[x] = v[y]; v[y] = aux;
}
```

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".

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

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.

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

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.

```
void quicksort(int[] a, int lo, int hi) {
  if (hi <= lo) return;

  int i = lo-1, j = hi;
  int t, v = a[hi];

  while (true)  {
    while (a[++i] < v) ;
    while (v < a[--j]) if (j == lo) break;

    if (i >= j) break;
    t = a[i]; a[i] = a[j]; a[j] = t;
  }
  t = a[i]; a[i] = a[hi]; a[hi] = t;

  quicksort(a, lo, i-1);
  quicksort(a, i+1, hi);
}
```