MCTA003-17 - ANÁLISE DE ALGORITMOS
CENTRO DE MATEMÁTICA, COMPUTAÇÃO E COGNIÇÃO
UNIVERSIDADE FEDERAL DO ABC
PROF. MAYCON SAMBINELLI

Algoritmos Recursivos: projeto e correção

1. Faça um algoritmo recursivo para calcular $\binom{n}{k}$, sem usar a fórmula fechada para isso, e prove a sua correção. Por definição, $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ para $1 \leq k < n$ e $\binom{n}{n} = \binom{n}{0} = 1$.

2. O *Quicksort* é outro algoritmo de divisão e conquista para o problema da ordenação. Ele é muito utilizado na prática, pois seu tempo esperado de execução é $\Theta(n \log n)$ e as constantes escondidas pela notação assintóticas são bem pequenas.

Seja $A[1..n]$ um vetor com n elementos. Dizemos que A está *particionado* com relação a um elemento, chamado *pivô*, se os elementos que são menores do que o pivô estão à esquerda dele e os outros elementos (maiores ou iguais) estão à direita dele. Note que o pivô está em sua posição correta final com relação ao vetor ordenado. A ideia do *Quicksort* é particionar o vetor e recursivamente ordenar as partes à direita e à esquerda do pivô, desconsiderando-o. Seu pseudocódigo é dado no Algoritmo 1.

Algorithm 1 Algoritmo *Quicksort* para ordenação.

```
1: Função QUICKSORT(A, inicio, fim)
2:     Se inicio < fim então
3:         p = ESCOLHEPIVO(A, inicio, fim)
4:         troque A[p] com A[fim]
5:         x = PARTICIONA(A, inicio, fim)
6:         QUICKSORT(A, inicio, x − 1)
7:         QUICKSORT(A, x + 1, fim)
```

Considere que a função ESCOLHEPIVO sempre devolve a posição fim, executando, portanto, em tempo $\Theta(1)$. A função PARTICIONA recebe o vetor A e as posições $inicio$ e fim e devolve a posição final do pivô após a partição. Ela executa em tempo $\Theta(n)$. Responda:

(a) Assumindo que PARTICIONA está correto, prove que QUICKSORT está correto.

3. Escreva uma versão recursiva da busca binária e prove a correção do seu algoritmo (você deve fornecer um pseudo-código).

4. Em um conjunto de S de n pessoas, uma **celebridade** é alguém que é conhecido por todas as pessoas de S mas que não conhece ninguém. Para representar a informação sobre esse grupo de pessoas usamos uma matriz binária. Dada uma matriz binária M de dimensões $n \times n$, definimos que $M[i, j] = 1$ se a pessoa i conhece a pessoa j e $M[i, j] = 0$ caso contrário. Dado um conjunto de n pessoas e a matriz M associada,

(a) Escreva um algoritmo recursivo que determine se existe uma celebridade S. Em caso afirmativo, i.e., S contém uma celebridade α, o seu algoritmo deve retornar α, caso contrário, o seu algoritmo de retornar NIL.

(b) Demonstre a correção do seu algoritmo

5. O seguinte algoritmo afirma ser capaz de fazer calcular a^n.

```
1: Função EXPONENCIACAO(a, n)
```

```
    Se n = 0 então
        Devolve 1
    Senão
        y ← Exponenciacao(a, ⌊n/2⌋)
        y ← y × y
        Se n ≡ 1 (mod 2) então
            y ← a × y
    Devolve y
```

Prove ou disprove a afirmação anterior.

6. No ensino fundamental, aprendemos um algoritmo para multiplicar dois números "grandes". Esse algoritmo tem tempo de execução $O(n^2)$ quando os números têm n dígitos[1].

O algoritmo de Karatsuba é um algoritmo de divisão e conquista que promete multiplicar dois números inteiros x e y que possuem n dígitos de uma forma mais eficiente. Sua ideia principal baseia-se em reescrever um número em partes menores com aproximadamente metade do número de dígitos cada. Por exemplo, 3147869 pode ser escrito como $10^4 \cdot 314 + 7869$.

O algoritmo é apresentado a seguir. Ele usa a função IGUALATAM(x, y), que deixa os números x e y com o mesmo número de dígitos (igual ao número de dígitos do maior deles) colocando zeros à esquerda se necessário. Ela devolve o número de dígitos (agora igual) desses números.

```
Função KARATSUBA(x, y)
    n = IGUALATAM(x, y)
    Se n ≤ 2 então
        Devolve xy
    seja x = 10^⌈n/2⌉ a + b e y = 10^⌈n/2⌉ c + d, onde a e c têm ⌊n/2⌋ dígitos cada e b e d têm ⌈n/2⌉ dígitos cada
    p1 = KARATSUBA(a, c)
    p2 = KARATSUBA(b, d)
    p3 = KARATSUBA(a + b, c + d)
    Devolve 10^(2⌈n/2⌉) p1 + 10^⌈n/2⌉ (p3 − p1 − p2) + p2
```

Prove a correção do algoritmo de Karatsuba.

[1] É um bom exercício entender por que isso vale.