Lista 1 – Projeto e Análise de Algoritmos – 2017s2

1. Para comparação de problemas com custos conhecidos em função de n, pede-se:

   (a) Suponha que estamos comparando implementações dos algoritmos de ordenação *insertion sort* e *merge sort* em uma mesma máquina. Para entradas de tamanho n, o *insertion sort* executa com custo total de $8n^2$, enquanto o *merge sort* executa com custo $64n \lg n$. Para quais valores de n o *insertion sort* supera o *merge sort*?

   (b) Qual é o menor valor de n para o qual um algoritmo, cujo tempo de execução é $100n^2$, executa mais rápido que um algoritmo cujo tempo de execução é 2^n? *(revisado em 06-set)*

2. Em relação a crescimento de funções, pede-se:

   (a) Para cada função $f(n)$ e tempo t na tabela seguinte, determine o maior tamanho n de um problema que pode ser resolvido no tempo t, assumindo que o algoritmo para resolver o problema leve tempo $f(n)$ nanossegundos (10^{-9} segundos).

$f(n)$ nanossegundos \ Tempo t	1 segundo	1 minuto	1 hora	1 dia	1 mês	1 ano	1 século
$\lg n$							
$\sqrt{n}$							
n							
$n \lg n$							
n^2							
n^3							
2^n							
$n!$							

   (b) Considerando a seguinte lista de funções, ordene-as (da menor para a maior) em relação a taxa de crescimento, de modo que, se a função $g(n)$ segue imediatamente a função $f(n)$ em sua lista, então deveria ser o caso de $f(n) \in O(g(n))$.

   $f_1(n) = n^{2,5}$
   $f_2(n) = \sqrt{2n}$
   $f_3(n) = n + 10$
   $f_4(n) = 10^n$
   $f_5(n) = 100^n$
   $f_6(n) = n^2 \log n$

   (c) Em cada um dos itens abaixo, indique se $f(n) \in O(g(n))$ ou $f(n) \in \Theta(g(n))$ ou $f(n) \in \Omega(g(n))$. *(revisado em 19-set)*

	$f(n)$	$g(n)$
(a)	$n - 100$	$n - 200$
(b)	$n^{\frac{1}{2}}$	$n^{\frac{2}{3}}$
(c)	$100n + \log n$	$n + (\log n)^2$
(d)	$n \log n$	$10n \log 10n$
(e)	$\log 2n$	$\log 3n$
(f)	$10 \log n$	$\log(n^2)$
(g)	$\frac{n^2}{\log n}$	$n(\log n)^2$
(h)	$n^{0,1}$	$(\log n)^{10}$

	$f(n)$	$g(n)$
(i)	$(\log n)^{\log n}$	$\frac{n}{\log n}$
(j)	$\sqrt{n}$	$(\log n)^3$
(k)	$n^{\frac{1}{2}}$	$5^{\lg n}$
(l)	$n2^n$	3^n
(m)	2^n	2^{n+1}
(n)	$n!$	2^n
(o)	$(\log n)^{\log n}$	$2^{(\lg n)^2}$

3. Prove as seguintes séries por indução matemática: *(revisado em 11-set)*

   (a) $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

   (b) $\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

   (c) $\sum_{i=1}^{n} (2i-1) = n^2$

   (d) $\sum_{i=0}^{n} i^3 = \frac{n^2(n+1)^2}{4}$, $n \geq 1$

4. Resolva as seguintes recorrências por meio do **método de iteração** ou **expansão telescópica**, e faça a prova, por indução, da fórmula fechada:

   (a) $T(1) = 0 \quad ; \quad T(n) = T(n-1) + c \quad ; \quad c$ constante e $n > 1$

   (b) $T(1) = 0 \quad ; \quad T(n) = T(n-1) + 2^n$

   (c) $T(1) = k \quad ; \quad T(n) = cT(n-1) \quad ; \quad c, k$ constantes e $n > 0$

   (d) $T(1) = 1 \quad ; \quad T(n) = 3T(n/2) + n \quad ;$ para $n > 1$

   (e) $T(1) = 1 \quad ; \quad T(n) = T(\sqrt{n}) + \log_2 n \quad ;$ para $n \geq 1$

   (f) $T(1) = 1 \quad ; \quad T(n) = 8T(n/2) + n$

   (g) $T(1) = 1 \quad ; \quad T(n) = T(n/3) + n$

   (h) $T(1) = 1 \quad ; \quad T(n) = 7T(n/4) + n$

   (i) $T(1) = 1 \quad ; \quad T(n) = T(n/4) + n^2$

   (j) $T(1) = 1 \quad ; \quad T(n) = 3T(n/4) + n^2$

   (k) $T(1) = 1 \quad ; \quad T(n) = 4T(n/2) + n^2 \lg n$

5. Use o **teorema mestre** para determinar o tempo de execução dos algoritmos expressos pelas recorrências abaixo:

   (a) $T(n) = T(n/2) + \Theta(1)$

   (b) $T(n) = 2T(n/2) + n^3$

   (c) $T(n) = T(9n/10) + n$

   (d) $T(n) = 16T(n/4) + n^2$

   (e) $T(n) = 7T(n/3) + n^2$

   (f) $T(n) = 7T(n/2) + n^2$

   (g) $T(n) = 2T(n/4) + \sqrt{n}$

   (h) $T(n) = 3T(n/2) + n \log n$

   (i) $T(n) = 4T(n/2) + n^2\sqrt{n}$

   (j) $T(n) = 27T(n/3) + n^3$

   (k) $T(n) = 64T(n/4) + n^2$

   (l) $T(n) = 4T(n/2) + n^2 \log n$

6. Use o **método de árvore de recursão** para determinar o tempo de execução dos algoritmos expressos pelas recorrências abaixo:

   (a) $T(n) = 2T(n/2) + \Theta(n)$ *(merge sort)*

   (b) $T(n) = 3T(\lfloor n/2 \rfloor) + \Theta(n)$

   (c) $T(n) = 4T(\lfloor n/2 \rfloor) + \Theta(n)$

   (d) $T(n) = 2T(n/2) + \Theta(n^2)$

   (e) $T(n) = 2T(n/3) + \Theta(n)$

7. Suponha que, para entradas de tamanho n, você tenha que escolher um dentre três algoritmos A, B e C.

   (a) Algoritmo A resolve problemas dividindo-os em cinco subproblemas de metade do tamanho, recursivamente resolve cada subproblema e então combina as soluções em tempo $O(n)$.

   (b) Algoritmo B resolve problemas dividindo-os em dois subproblemas de tamanho $n-1$, recursivamente resolve cada subproblema e então combina as soluções em tempo $O(1)$.

   (c) Algoritmo C resolve problemas dividindo-os em nove subproblemas de tamanho $n/3$, recursivamente resolve cada subproblema e então combina as soluções em tempo $O(n^2)$.

   Qual o consumo de tempo de cada um desses algoritmos? Expresse as suas respostas em termos da notação O, mas procure dar as respostas com funções para limites superiores mais próximos possíveis. Qual algoritmo é assintoticamente mais eficiente no pior caso? Justifique as suas respostas.

8. Para os algoritmos abaixo, pede-se:

   (a) Quantas linhas, em função de n, em forma de $\Theta(\cdot)$, o seguinte programa imprime? Escreva e resolva a recorrência. Considere n como uma potência de 2.

```
FUNÇÃO f(n)
1: se n > 1 então
2:     imprime linha ("ainda rodando")
3:     f(n/2)
4:     f(n/2)
```

   (b) Considere dois números inteiros n e r como argumentos, sendo $n \geq 0$ e $r \geq 0$. Qual o número de vezes que a subrotina CAIXA-PRETA é chamada pelo algoritmo ALGO? Expresse esse número como função de n e r e justifique a sua resposta.

```
ALGO(n, r)
1: se n = 0 então
2:     CAIXA-PRETA(n, r)
3:     devolve 1
4: k ← r× ALGO(n − 1, r)
5: para i ← 1 até k faça
6:     CAIXA-PRETA(n, i)
7: devolve k
```

9. A MEDIANA(A) é um algoritmo que devolve o i-ésimo menor valor, sendo $i = \lfloor \frac{n+1}{2} \rfloor$ (*mediana inferior*) ou, se preferir, $i = \lceil \frac{n+1}{2} \rceil$ (*mediana superior*), de um dado vetor A de n elementos. *(revisado em 15-out)*

   (a) Implemente duas versões deste algoritmo em qualquer linguagem de programação:
   – **versão 1:** em tempo $\Omega(n \log n)$
   – **versão 2:** em tempo médio $O(n)$

   (b) (**OPCIONAL** – *pontuação extra)* Implemente duas versões do **filtro de mediana**, considerando os dois algoritmos desenvolvidos no item (a), para matrizes bidimensionais $m \times n$ de inteiros $0 \leq f(i,j) \leq 255$, sendo $0 \leq i < m$, $0 \leq j < n$, supondo janela de filtro com vizinhança parametrizável de $p \times q$, sendo $2 \leq p < m$ e $2 \leq q < n$. A técnica, exemplos e código (em C) podem ser consultados no seguinte documento: `https://www.ime.usp.br/~reverbel/ccm118-12/eps/ep4.pdf`
   Avalie o tempo de execução real (por exemplo, em segundos) das duas versões implementadas do filtro para uma matriz (imagem) suficientemente grande ($\geq 640 \times 480$ pixels) e para diferentes escolhas de p e q (por exemplo, 3, 7, 15, ...).