1. (1,5 pontos) Prove por indução forte que todo número inteiro positivo pode ser escrito como a soma de potências de 2 distintas. Explicite **claramente** a base, hipótese e passo de sua prova.

2. (1 ponto) Complete o enunciado do Teorema Master:
Sejam $a \geq 1$ e $b > 1$ constantes, seja f(n) uma função e seja T(n) definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + f(n).$$

Então T(n) pode ser limitada assintoticamente da seguinte maneira:
(a) Se $f(n) \in O(n^{\log_b a - E})$ para alguma constante $E > 0$, então $T(n) \in \theta(\quad)$
(b) Se $f(n) \in \theta(n^{\log_b a})$, então $T(n) \in \theta(\quad)$
(c) Se $f(n) \in \Omega(n^{\log_b a + E})$, para algum $E > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \theta(\quad)$

3. (1,5 pontos) Encontre uma fórmula fechada para as seguintes relações de recorrência, expressando sua solução em termos da classe θ. Justifique cada resposta indicando a aplicabilidade do Teorema Master ou provando sua substituição (usando indução). Indique as condições iniciais convenientes assim como quaisquer restrições em que n sejam necessárias para a validade de sua solução (por exemplo: T(1) = 17 e assumimos que n é potência de 13).

(a) $T(n) = 8T(n/4) + n^{\sqrt{2}}$
(b) $T(n) = 8T(n/4) + n^2$
(c) $T(n) = 8T(n/4) + n^{3/2}$
(d) $T(n) = T(n-1) + n/2$
(e) $T(n) = T(\lfloor \sqrt{n} \rfloor) + \log_2 n$

4. (1,2 pontos) Em que casos os seguintes algoritmos atingem complexidade O(nlog n)? Não precisa justificar.

(a) Selection Sort
(b) Insertion Sort
(c) Merge Sort
(d) Heap Sort
(e) Quick Sort
(f) Radix Sort

5. (2,0 pontos) Seja A um vetor de n números inteiros (de tamanhos arbitrários) que está quase completamente ordenado em ordem crescente. Sabendo que há exatamente log n elementos em posições incorretas com respeito a esta ordem, sua tarefa é descrever um algoritmo eficiente para re-ordenar o vetor. Sua descrição deve ser em linguagem natural ou em pseudo-código mas **não** em linguagem de programação. Você pode utilizar qualquer algoritmo ou estrutura de dados conhecidos, fazendo referência a eles apenas por nome ou projetar seu próprio algoritmo. Se sua descrição for suficientemente simples para que se possa facilmente inferir sua corretude, não precisa provar que está correto. Caso contrário, inclua uma breve prova de corretude. A pontuação dessa questão será dependente da eficiência do algoritmo proposto: 0 se o algoritmo for $\Omega(n^2)$, 0.5 se for $\theta(n \log n)$, 2.0 se for $O(n \log n)$. O modelo computacional considerado é o modelo de árvore binária de decisões.

6. (2,0 pontos) Dados um vetor **ordenado** A de n números reais, $n \geq 1$, e um número real x, queremos determinar se existem A[i] e A[j], $1 \leq i,j \leq n$ tais que $x = A[i] + A[j]$. Projete **por indução** um algoritmo de complexidade O(n) para este problema. Escreva a relação de recorrência T(n) de

seu algoritmo e prove que o resultado da recorrência é de fato O(n).
Dica: O que podemos concluir da comparação de A[1] + A[n] com x?

7. (1,0 ponto)
(a) Qual a complexidade de pior caso do algoritmo de seleção do elemento mediano de um vetor de n elementos utilizando-se o algoritmo da seleção do QuickSort? Por quê?
(b) Qual a complexidade de caso médio do algoritmo de seleção do elemento mediano de um vetor de n elementos utilizando-se o algoritmo da seleção do QuickSort? (Não precisa justificar.)