1) Projete um algoritmo para pesquisa de um elemento *x* em um vetor ordenado A[1..n] modificando a estratégia de divisão que empregamos na Busca Binária, isto é, a divisão deverá separar em dois conjuntos com n/3 e 2n/3 elementos. Projete essa "Busca Ternária", analise sua complexidade e correção e depois a compare com a Busca Binária.

---

2) Três "pesquisadores" propuseram o seguinte algoritmo de ordenação "elegante":

```
  STOOGE-SORT(A, i, j)
  1 se A[i] > A[j]
  2   então troca(A[i], A[j])
  3 se i + 1 ≥ j
  4   então "finaliza processamento"
  5 k ← ⌊(j-i+1)/3⌋              (Arredonda para menos)
  6 STOOGE-SORT(A, i, j-k)       (primeiros dois terços)
  7 STOOGE-SORT(A, i+k, j)       (últimos dois terços)
  8 STOOGE-SORT(A, i, j-k)       (primeiros dois terços
novamente)
```

a) Mostre que para n elementos, **STOOGE-SORT(A,1, n)** ordena corretamente o arranjo de entrada A[1..n].

b) Forneça uma recorrência para o tempo de execução no pior caso de **STOOGE-SORT** e um limite assintótico restrito (notação Θ) sobre o tempo de execução no pior caso.

---

3) Em um vetor A [1.. ∞], de "tamanho infinito", as primeiras n células contem números inteiros ordenados de forma crescente e o restante das células está preenchido com ∞. Projete um algoritmo que recebe o vetor A e uma chave x para pesquisar e devolve a posição de x em tempo proporcional a O(lgn). Prove que funciona corretamente e consome o tempo proposto.

Obs.: você não tem a informação do valor de n nos dados de entrada.

---

4) Considere dois conjuntos de números inteiros A e B com *m* e *n* elementos, respectivamente. Eles não estão necessariamente ordenados, e assuma também que $m \leq n$. Utilizando a estratégia de divisão e conquista, mostre como podemos computar $A \cup B$ e $A \cap B$ em um tempo proporcional a *O(nlgm)*.
