Questões mais elaboradas envolvendo algoritmos.

1. Que estratégia usada em algum algoritmo de ordenação por comparação você utilizaria para encontrar o menor elemento de um array? Implemente esse algoritmo.
   `public int selecionaMenor(int[] array)`
2. Baseado na implementação anterior, implemente um outro algoritmo que seleciona o menor elemento de um array que seja maior que um outro elemento existente no array.
   `public int selecionaMenor(int[] array, int existente)`
3. A k-esima estatística de ordem de um conjunto de dados é o k-esimo menor elemento desse conjunto de dados. Implemente um algoritmo que calcula a k-esima estatística de ordem de um array (assuma que k está dentro do tamanho do array). Procure usar suas ideias implementadas em 1 e 2.
   `public int orderStatistics(int[] array, int k)`
4. Implemente um algoritmo recursivo para a exponenciação, onde pow(a,b) onde a é a base e b é o expoente. Calcule o tempo de execução de seu algoritmo (você deve obrigatoriamente encontrar uma relação de recorrência porque seu algoritmo é recursivo).
5. Dado um array ordenado de numeros encontrar o par de números no array cuja soma é o mais próximo possível de um número dado x. Dica: o par de números mais próximo pode ser encontrado varrendo-se o array da esquerda para a direita e da direita para a esquerda, pegando números cuja soma seja a mínima possivel e se aproxime de determinado valor. Com isso,ao final da execução, você deve guardar os valores dos índices desses números. Essa estratégia é explicada a seguir:
   a. Inicialize uma variável “diff” com o máximo valor possível (infinito)
   b. Inicialize duas variáveis de índices left e right sendo 0 e array.length-1 respectivamente
   c. Itere em loop enquanto left < right
      i. Se abs(array[right]-array[left] – x < diff) entao atualiza diff e os resultados (indices) de left e right
      ii. Senão se abs(array[right]-array[left] – x > diff) entao left++
      iii. Senao right--
6. Imagine que você deseja ordenar um array de forma tal que, cada elemento indexado com par do array é maior que o elemento à sua esquerda e o elemento à sua direita. Sua resposta deve ser em um outro array. Seu algoritmo deve ser O(n.log n). Dica: procure enxergar a solução com um array ordenado. Exemplo:
   ```
   Input:  {1, 2, 3, 4, 5, 6, 7}
   Output: {1, 7, 2, 6, 3, 5, 4}
   ```
7. Dado um array de inteiros ordenado, floor(x) é o elemento do array que é menor (e mais próximo possível de x) ou igual a x (podendo x pertencer ou não ao array). Analogamente, ceil(x) seria o elemento do array que é maior (e mais próximo possível de x) ou igual a x. Implemente algoritmos que calculam floor e ceil usando a estratégia de busca linear.