1. Since n = 125 is greater than 1 start with T(n) = 7T(n/5) + 10n
   T(125) = 7T(25) + 10*125 = 7*649 + 1250 = 5793
   T(25) = 7T(5) + 10*25 = 7*57 + 250 = 399 + 250 = 649
   T(5) = 7T(1) + 10*5 = 7*1 + 50 = 57
   T(1) = 1

   So, T(125) = 5793

2. Since quicksort uses first item in list as pivot
   a. Worst case means the first item won't partition the list in half. List of 15 integers will be 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
   b. Best case means the first item will partition the list in half recursively. List of 15 integers will be 7, 3, 11, 2, 9 , 5, 13, 2, 14, 1, 10, 6, 15, 12, 4
3.
   a. Assume the algorithm split into 1/5 and 4/5. We can simply design the algorithm as follow
      i. Item needs to be find is x and array is numOfBars
      ii. The lower bound is *a* and upper bound *b*
      iii. While $a \leq$ b, do
         1. Find the "midpoint" m = *a* + (4/5)(*b* – *a*)
         2. If numOfBars[m] is equal to x, return m
         3. If numOfBars[m] is greater than x, set *b* = m – 1
         4. If numOfBars[m] is less than x, set *a* = m + 1
         5. If not found, return False

   b. For best case, both algorithm is O(1) when the item is found in the first search. Worst case will take original BS time complexity of $O(\log_2 n)$. There are two cases for the variant BS. If the item is found in the 1/5 part, variant will be faster than original since $O(\log_5 n)$ but if the item is in 4/5 part, variant will be slower than original since $O(\log_{1.25} n)$. Therefore, if there were 1000 items in the array, the original BS search will make 10 comparisons while the variant BS search will make 31comparisons in the worst case.

4. Solve the recurrence relations

   $T(n) = 2T(n-1) + 3$

   $= 2(2T(n-2) + 3) + 3 = 2^2T(n-2) + 2*3 + 3$

   $= 2(2(2T(n-3) + 3) + 3) + 3 = 2^3T(n-3) + 2^2*3 + 2*3 + 3$

   ……

   $= 2^nT(n-n) + \ldots + 2^2*3 + 2*3 + 3$

   $= 2^nT(0) + \ldots + 2^2*3 + 2*3 + 3$

   $= 2^n*3 + \ldots + 2^2*3 + 2*3 + 3$

   $= 3*2^{n+1}$

   $= O(2^n)$

   So, the time complexity is $O(2^n)$