Lab 6

# Problem 1

Number of comparisons (comparisons-based algorithms) in worst case = height of the decision tree (h)

h >= ceil(log n!) => h >= ceil(log 4!) => h >= ceil(log 120) => h >= ceil(4.584963) => h >= 5

* Number of comparisons in worst case >= 5

# Problem 2

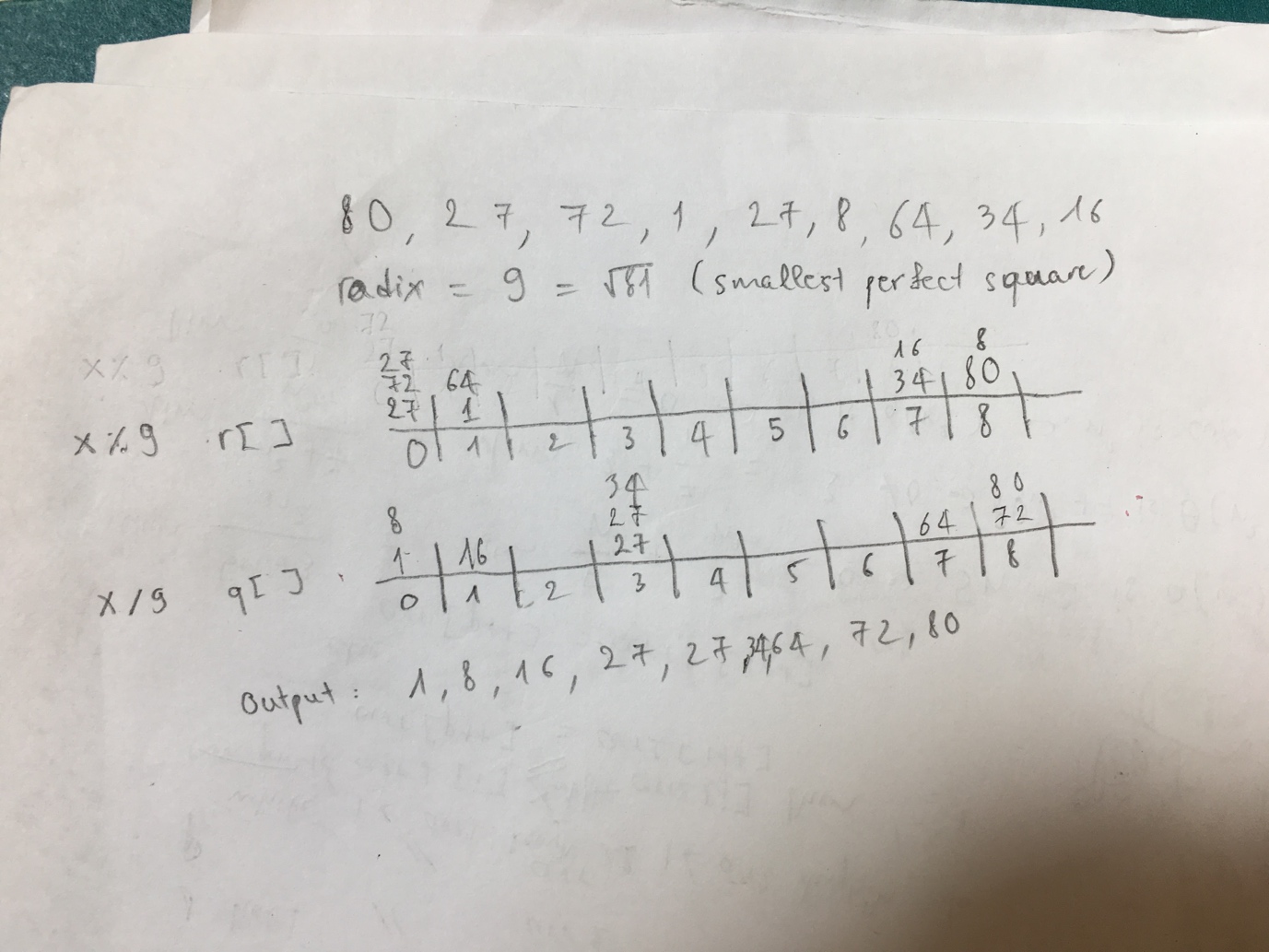
Use MergeSort to sort the array A.

Create new array B with length equal to A.

Use two pointers i, j. i run from left, j run from right. Put A[i] to B[0], put A[j] to B[1], increase i 1, decrease j 1 until i equal to j.

The fastest running time is time of MergeSort (O(n log n)) + time of put elements from A to B (O(n)) => The running time is O(n log n) + O(n) = O(n log n + n) = O(n log n)

# Problem 3



# Problem 4

Algorithm findFirstNoDupElement(A)  
 Input An array A  
 Output First integer that occurs in the array only once  
  
 bucket <- new int[3 \* A.length]  
  
 for i <- 0 to A.length - 1 do  
 bucket[A[i]] <- bucket[A[i]] + 1  
  
 for i <- 0 to A.length - 1 do  
 if bucket[A[i]] = 1  
 return A[i]

We initialize new array with length is 3n and run 2 for loops from 0 to n so the running time is O(3n) + O(n) + O(n) = O(5n) = O(n)