

Tutorial 02: January 24

1. Expected Runtime Analysis

Give the best-case and expected running time for the following function. You can assume that the Shuffle operation requires $\mathcal{O}(n)$ time and the array A contains no duplicates

Note: the *Shuffle()* function produces each permutation equally likely.

```

MonkeySort(A):
  // Input: Array A of size n
  // Output: None (A is sorted in-place)
  shuffle(A)
  if A is sorted then
    return A
  else do
    MonkeySort(A)

```

2. Average Runtime Analysis

Suppose A is an array containing n distinct elements. In addition, assume each element is in between 1 and n , inclusive. Analyze this pseudo-code to determine a tight bound on the average number of question mark (?) that are printed, rather than a runtime. You may assume n is divisible by 2.

```

mystery(A, n)
  count = 1
  for i = 1 to n-1
    if A[i] is divisible by A[0]
      count++
  for i = 1 to count
    print("?")

```

3. Max-Heap Operations

Insert 27 and 9 into the following heap, and then perform a delete-max operation on the resulting heap.

