2019-1 Computer Algorithms Homework #2

Dae-Ki Kang

March 17, 2019

(Deadline : April 2)

1. Watching assignments : watch the following lecture(s).

   (a) (Required) Asymptotic Complexity

      i. 2011 Recitation 1. Asymptotic Complexity, Peak Finding
      `https://www.youtube.com/watch?v=P7frcB_-g4w`

      ii. 2005 Lecture 2: Asymptotic Notation; Recurrences; Substitution, Master Method
      `https://www.youtube.com/watch?v=whjt_N9uYFI`

   (b) (Required) Sorting algorithms

      i. 2011 Lecture 3. Insertion Sort, Merge Sort
      `https://www.youtube.com/watch?v=Kg4bqzAqRBM`

      ii. 2011 Recitation 3. Document Distance, Insertion and Merge Sort
      `https://www.youtube.com/watch?v=4iXLnF3hExw`

      iii. 2011 Lecture 4. Heaps and Heap Sort
      `https://www.youtube.com/watch?v=B7hVxCmfPtM`

      iv. 2011 Lecture 7. Counting Sort, Radix Sort, Lower Bounds for Sorting
      `https://www.youtube.com/watch?v=Nz1KZXbghj8`

      v. 2011 Recitation 7. Comparison Sort, Counting and Radix Sort
      `https://www.youtube.com/watch?v=9bkvws_vqLU`

      vi. 2005 Lecture 5: Linear-time Sorting: Lower Bounds, Counting Sort, Radix Sort
      `https://www.youtube.com/watch?v=0VqawRl3Xzs`

      vii. 2011 Lecture 5. Binary Search Trees, BST Sort
      `https://www.youtube.com/watch?v=9Jry5-82I68`

      viii. 2011 Recitation 5. Recursion Trees, Binary Search Trees
      `https://www.youtube.com/watch?v=r5pXu1PAUkI`

      ix. 2005 Lecture 4: Quicksort, Randomized Algorithms
      `https://www.youtube.com/watch?v=vK_q-C-kXhs`

      x. 2015 Recitation 4. Randomized Select and Randomized Quicksort
      `https://www.youtube.com/watch?v=QPk8MUtq5yA`

   (c) (Required) Divide and conquer

      i. 2015 Lecture 6. Randomization: Matrix Multiply, Quicksort
      `https://www.youtube.com/watch?v=cNB2lADK3_s`

      ii. 2015 Recitation 1. Matrix Multiplication and the Master Theorem
      `https://www.youtube.com/watch?v=09vU-wVwW3U`

      iii. 2011 Lecture 11. Integer Arithmetic, Karatsuba Multiplication
      `https://www.youtube.com/watch?v=eCaXlAaN2uE`

      iv. 2005 Lecture 3: Divide-and-Conquer: Strassen, Fibonacci, Polynomial Multiplication
      `https://www.youtube.com/watch?v=-EQTVuAhSFY`

   (d) Recommended lectures

i. 2015 Lecture 2. Divide & Conquer: Convex Hull, Median Finding
`https://www.youtube.com/watch?v=EzeYI7p9MjU`

ii. 2005 Lecture 6: Order Statistics, Median
`https://www.youtube.com/watch?v=mR_RUjsJnV8`

iii. 2015 Lecture 3. Divide & Conquer: FFT
`https://www.youtube.com/watch?v=iTMn0Kt18tg`

iv. 2011 Lecture 6. AVL Trees, AVL Sort
`https://www.youtube.com/watch?v=FNeL18KsWPc`

v. 2011 Recitation 6. AVL Trees
`https://www.youtube.com/watch?v=IWzYoXKaRIc`

2. Using your favorite computer programming language (but C/C++, Java, Python, C# recommended), write a program that calculate the following:

   Suppose you are given an integer $0 \leq n \leq 10,000$ which is not divisible by 2 or 5. Then, there exists a number which is some multiple of n and is represented as a sequence of 1's in decimal notation.

   Your task is to implement an algorithm that finds how many digits are in the smallest such multiple of n.

   Note that, no matter what number is entered, your program has to return the result and terminate itself in **one second**.

   (a) Input
   Integers (one integer per line)

   (b) Output
   Each output line gives the smallest integer $x > 0$ such that p $= \sum_{i=0}^{x-1} 1 \times 10^i$, where a is the corresponding input integer, $p = a \times b$, and b is an integer greater than zero.

   (c) Sample Input
   3
   7
   9901
   9981
   9967
   9949

   (d) Sample Output
   3
   6
   12
   9972
   9966
   9948

3. For this problem, you can either write a program or calculate by hand.

   Tony is not a morning person and is late for work with 0.5 probability. Andy is a generous supervisor, so tries not to care for Tony's being late. But Andy scolds Tony if Tony is late for work three times in a row. Calculate the probability that Tony is not being scolded at all if he has attended for twenty days.

   Ian is also not a morning person and is late for work with $\frac{2}{3}$ probability. Calculate the probability that Ian is not being scolded by Andy at all if he has attended for twenty days.

4. Describe an alternative Euclid's algorithm for calculating $GCD(a, b)$ using the following properties:

$$GCD(a,b) = \begin{cases} 2GCD\left(\frac{a}{2}, \frac{b}{2}\right) & \text{if } a, b \text{ are even} \\ GCD\left(a, \frac{b}{2}\right) & \text{if } a \text{ is odd and } b \text{ is even} \\ GCD\left(\frac{a-b}{2}, b\right) & \text{if } a, b \text{ are odd} \end{cases}$$

Analyze the efficiency of the proposed algorithm.

5. In discussing the matrix multiplication algorithm, I have claimed that the following block-wise property: if X and Y are $n \times n$ matrices, and

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

where A, B, C, D, E, F, G and H are $\frac{n}{2} \times \frac{n}{2}$ submatrices, then the product XY can be expressed in terms of these blocks:

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

Prove this property.

6. A binary tree is *full* if all of its vertices have either zero or two children. Let B_n denote the number of full binary trees with n vertices.
   (a) Derive a recurrence relation for B_n.
   (b) Prove that B_n is $2^{\Omega(n)}$

7. You are given an infinite array $A[\cdot]$ in which the first n cells contain integers in sorted order and the rest of the cells are filled with ∞. You are *not* given the value of n. Describe an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in $O(\log n)$ time.

8. Given a sorted array of distinct integers $A[1..n]$, you want to find out whether there is an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(\log n)$.

9. Consider the task of searching a sorted array $A[1..n]$ for a given element x: a task we usually perform by binary search in time $O(\log n)$. Show that any algorithm that accesses the array only via comparisons (i.e. by asking questions of the form "is $A[i] \leq z$?") must take $\Omega(\log n)$ steps.