Algorithms: MTH 524(A)
Department of Mathematics and Statistics
Indian Institute of Technology - Kanpur January 13, 2015

Problem Set 1

Exercises are for extra practice and should not be turned in.

Exercise 1 *2.3-6 from CLRS :* Observe that the while loop of lines 5−7 of the INSERTION-SORT procedure in Section 2.1 (or in Lecture 2) uses a linear search to scan (backward) through the sorted subarray $A[1], \ldots, A[j-1]$. Can we use a binary search instead to improve the overall worst-case running time of insertion sort to $\Theta(n \lg n)$?

Exercise 2 *2.3-7 from CLRS :* Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

Exercise 3 *3.1-4 from CLRS :* Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

Exercise 4 Rank the following functions by increasing order of growth; that is, find an arrangement $g_1, g_2, \ldots, g_{11}$ of the functions satisfying $g_1 = O(g_2), g_2 = O(g_3), \ldots, g_{10} = O(g_{11})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. All the logs are in base 2.

$$\begin{array}{lll} \binom{n}{100}, & 3^n, & n^{100}, \\ 1/n, & 2^{2n}, & 10^{100}n, \\ 3^{\sqrt{n}}, & 1/5, & 4^n, \\ n\log n, & \log(n!). & \end{array}$$

Exercise 5 Prove or disprove each of the following properties related to asymptotic notation. In each of the following assume that f, g, and h are asymptotically non-negative functions.

i. $f(n) = O(g(n))$ and $g(n) = O(f(n))$ implies that $f(n) = \Theta(g(n))$.
ii. $f(n) + g(n) = \Theta(\max(f(n), g(n)))$.
iii. $f(n) = O(g(n))$ implies that $h(f(n)) = O(h(g(n)))$.
iv. $f(n) = O(g(n))$ and $g(n) = O(h(n))$ implies that $f(n) = O(h(n))$.
v. $f(n) + o(f(n)) = \Theta(f(n))$.

Exercise 6 The Fibonacci numbers are define as

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}, \ n \geq 2.$$

i. Show that

$$F_n = \frac{\alpha^n - \hat{\alpha}^n}{\sqrt{5}},$$

where $\alpha = (1 + \sqrt{5})/2$ is the golden ratio and $\hat{\alpha}$ is its conjugate.

ii. Consider the following recursive algorithm for computing the nth Fibonacci number :

FIB(n)
if $n = 0$
  then return 0
else if $n = 1$
  then return 1
return FIB($n - 1$) + **FIB**($n - 2$)

If $T(n)$ is the worst case running time of FIB(n), the show, using substitution method, that $T(n) = \Theta(F_n)$.

iii. Consider the following algorithm for computing nth Fibonacci number :

FIB′(n)
if $n = 0$
  then return 0
else if $n = 1$
  then return 1
sum $\leftarrow 1$
for $k \leftarrow 1$ **to** $n - 2$
  do *sum* $\leftarrow$ *sum*+**FIB′**(k)
return *sum*

Prove the correctness of this algorithms. What is the asymptotic running time $T'(n)$ of FIB′(n)? Is this an improvement over the FIB(n) algorithm?

Exercise 7 Solve for G(n) defined by the recurrence

$$G(0) = c, \quad G(1) = c, \quad G(n) = G(n-1) + G(n-2) + c, \ n \geq 2,$$

where $c > 0$ is a given number.

Exercise 8 If we have two linear polynomials $ax+b$ and $cx+d$, we can multiply them using the four coefficient multiplications

$$\begin{aligned} m_1 &= a \cdot c \\ m_2 &= a \cdot d \\ m_3 &= b \cdot c \\ m_4 &= b \cdot d \end{aligned}$$

to form the polynomial

$$m_1 x^2 + (m_2 + m_3)x + m_4.$$

i. Give a divide-and-conquer algorithm for multiplying two polynomials of degree-bound n based on this formula.

ii. Give and solve a recurrence for the worst-case running time of your algorithm.

iii. Show how to multiply two linear polynomials $ax + b$ and $cx + d$ using only three coefficient multiplications.

iv. Give a divide-and-conquer algorithm for multiplying two polynomials of degree-bound n based on your formula from part (iii).

v. Give and solve a recurrence for the worst-case running time of your algorithm in part (iv).

Exercise 9 For $n > 2$, show that

$$\sum_{k=2}^{n-1} k \lg k \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2.$$

The following problem is due **Friday, February 6** at **11:59 PM**.
Steps you should follow :

1. You should do all your work for this problem set in a directory called GroupID-PS1 where ID stands for your group number. For example, if you are homework group 05, then the directory name will be Group05-PS1; if you are homework group 13, then the directory name will be Group13-PS1. For parts a. and b., the pseudo codes and corresponding time complexities should be included as comments in your program.

2. Upon completion of the work, once you are ready for submission, go the directory that contains the directory GroupID-PS1 and compress it using following command at the xterm/terminal command prompt

```
$ tar cvfz GroupID-PS1.tgz GroupID-PS1
```

which will create a file called GroupID-PS1.tgz .

3. Attach this file in an email with subject GroupID-PS1 and send it to akasha@iitk.ac.in by 11.59 pm on February 6. Late submission will not get any credit.

A. For a given positive real number a, let the cube with side length a, denoted by C, be

$$C = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \ : \ 0 \le x_i \le a, \ i = 1, 2, 3\}$$

and for a given positive integer L, the family of cubes with side length a/L be

$$\{C_\ell \ : \ \ell = (\ell_1, \ell_2, \ell_3) \text{ with } \ell_i \in \{0, \dots, L-1\}\}$$

and the family of spheres of radius $2a/L$ be

$$\{S_\ell \ : \ \ell = (\ell_1, \ell_2, \ell_3) \text{ with } \ell_i \in \{0, \dots, L-1\}\},$$

with

$$C_\ell = \left\{(x_1, x_2, x_3) \in \mathbb{R}^3 \ : \ \left(\frac{\ell_i}{L}\right) a \le x_i \le \left(\frac{\ell_i + 1}{L}\right) a, \ i = 1, 2, 3\right\}$$

and

$$S_\ell = \left\{(x_1, x_2, x_3) \in \mathbb{R}^3 \ : \ \sum_{i=1}^{3} \left(x_i - \frac{2\ell_i + 1}{2L} a\right)^2 = \left(\frac{2a}{L}\right)^2\right\}.$$

a. Write an algorithm **PARTITION(r, L)** that, for given input $r \in C$, and L, returns all ℓ such that $r \in C_\ell$. Analyze your algorithm for the worst-case running time.
Let $R = \{r^1, r^2, \dots, r^K\}$ be a given set containing K points in C. Let the symbol R_ℓ denote the set of points in R that is contained in C_ℓ, that is,

$$R_\ell = R \cap C_\ell.$$

Write **PARTITION-ALL(R, L)** that uses the **PARTITION** algorithm to obtain the set R_ℓ for all $\ell = (\ell_i, \ell_2, \ell_3)$ with $\ell_i \in \{0, \dots, L-1\}$.

b. Let $P = \{p^1, p^2, \dots, p^N\}$ be a given set of N points in $\mathbb{R}^3$ that lie on the boundary of C_ℓ, $Q = \{q^1, q^2, \dots, q^M\}$ be a given set of M points in $\mathbb{R}^3$ that lie on S_ℓ. Let $\mathcal{A}$ be a matrix whose (i, j)-th element $\mathcal{A}_{ij}$ is given by

$$\mathcal{A}_{ij} = \frac{\sin |q^i - p^j|}{|q^i - p^j|}.$$

Write an algorithm **BUILD-MATRIX(L, ℓ, P, Q)** to compute the square matrix $\mathcal{A}^T \mathcal{A}$, where $\mathcal{B}^T$ denotes the transpose of $\mathcal{B}$, for given inputs P and Q. Analyze your algorithm for the worst-case running time.

c. Implement the algorithms **PARTITION-ALL** and **BUILD-MATRIX**.

The function prototype for **PARTITION-ALL** should read

```
bool partition-all(int L, char* inputFileName, char* outputFileName);
```

where the `TRUE` value of the output bool indicate successful execution of the function; the input string `inputFileName` is the name of the file in which the data for set R is saved – one point per line. For example, a file containing the data for set $R = \{(1,0,0.5),(0.3,0.4,0.1),(0,1,0.8)\}$ should read

```
1.0 0.0 0.5
0.3 0.4 0.1
0.0 1.0 0.8
```

The fourth input parameter `outputFileName` is the name of the output file in which the data for R_ℓ is to be saved for all L^3 number of ℓ's. In the output file, the data for each R_ℓ should begin with the corresponding ℓ values, followed by the number of points in R_ℓ and then the point data for R_ℓ. For example, if $L = 2$, and $R_{0,0,0} = \{(0.1,0.3,0.4),(0.1,0.2,0.1),(0.2,0.3,0.3)\}$, $R_{0,0,1} = \{(0.1,0,0.55)\}$, $R_{0,1,0} = \{\}$, $R_{0,1,1} = \{\}$, $R_{1,0,0} = \{(0.6,0.3,0.4),(0.75,0.25,0.01)\}$, etc., then the output files reads

```
0 0 0
3
0.1 0.3 0.4
0.1 0.2 0.1
0.0 1.0 0.8
0 0 1
1
0.1 0.01 0.55
0 1 0
0
0 1 1
0
1 0 0
2
0.6 0.3 0.4
0.75 0.25 0.01
- - -
- - -
```

The function prototype for **BUILD-MATRIX** should read

```
bool matrix(int L, int l[3], char* fileForP, char* fileForQ,
char* fileForMatrix);
```

where the `TRUE` value of the output bool indicate successful execution of the function; the input strings `fileForP` and `fileForQ` are the names of the file containing the data for sets P and Q respectively in the "one point per line" format. The fifth input parameter `fileForMatrix` is the name of the output file in which the data for $\mathcal{A}^T\mathcal{A}$ is to be saved. Each line of this output file should have three numbers in the form

```
i j value
```

where the `value` corresponds to the (i, j)-th entry of the matrix, i.e., $(\mathcal{A}^T\mathcal{A})_{ij} = \text{value}$. *Your implementations should perform all the necessary checks to ensure validity of the input data.*