Introduction to Algorithms: 6.006
Massachusetts Institute of Technology — February 1, 2011
Professors Erik Demaine, Piotr Indyk, and Manolis Kellis — Problem Set 1

Problem Set 1

This problem set is divided into two parts: Part A problems are theory questions, and Part B problems are programming tasks.

Both Part A and Part B questions are due **Monday, February 14** at **11:59PM**.
Solutions should be turned in through the course website. Your solution to Part A should be in PDF format using LaTeX or scanned handwritten solutions. Your solution to Part B should be a valid Python file which runs from the command line. A template for writing up solutions in LaTeX is available on the course website. Remember, your goal is to communicate. Full credit will be given only to a correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Part A:

Problem 1-1. [18 points] **Asymptotic Growth**

For each group of four functions below, rank the functions by increasing order of growth; that is, find an arrangement g_1, g_2, g_3, g_4 of the functions satisfying $g_1 = O(g_2)$, $g_2 = O(g_3)$, $g_3 = O(g_4)$. (For example, the correct ordering of n, n^2, n^3, n^4 is n, n^2, n^3, n^4.)

(a) [6 points] Group 1:

$$f_1 = \sqrt{n} \qquad f_2 = \binom{n}{2} \qquad f_3 = 2^{2^{100000}} \qquad f_4 = \log n$$

(b) [6 points] Group 2:

$$f_1(n) = 1/n \qquad f_2(n) = \log\log\log n \qquad f_3(n) = n/\log n \qquad f_4(n) = n^{0.99}$$

(c) [6 points] Group 3:

$$f_1(n) = 2^n \qquad f_2(n) = n \cdot 2^{n/2} \qquad f_3(n) = \log n^n \qquad f_4(n) = \sum_{i=0}^{n} i$$

Problem 1-2. [6 points] **Unimodal Maximum**

Define an array $A[0\,..\,n-1]$ of numbers to be ***unimodal*** if there exists a maximum element $A[k]$ such that

$$A[i] > A[i-1] \text{ for all } i \text{ such that } 0 < i \le k, \text{ and}$$
$$A[i] < A[i-1] \text{ for all } i \text{ such that } k < i < n$$

For example, $[1, 4, 8, 9, 7, 6, 2]$ is unimodal, while $[1, 3, 5, 4, 1, 2]$ is not.

Assume that all the numbers in A are distinct.

Devise an efficient algorithm to find the maximum element in a unimodal array A. Prove that your algorithm runs in $O(\log n)$ time.

Problem 1-3. [6 points] **(Re)writing History**

For next year's State of the Onion address, President Banach Bahama needs help crafting his speech so that it's not too similar to the one he gave this year. He's been advised that MIT 6.006 students have been taught about document distance and has found this to be a reasonable metric to use in crafting his new speech. And thus he has turned to you to help advise him in writing his new speech.

Recall that a document distance of $\theta = 0$ means the speeches are identical and that $\theta = 90°$ means they don't even share a word. Starting with a copy of this year's speech, Banach wants to know whether each of the following strategies will improve his speech (i.e., increase θ) or not help at all (i.e., keep θ the same). He also requires an explanation for each of your answers.

(a) [2 points] Strategy 1: re-arrange the order of the words while somehow keeping it coherent (or not).

(b) [2 points] Strategy 2: pad the speech with extra words, none of which appeared in his speech for this year.

(c) [2 points] Strategy 3: remove some number of words from his speech for this year.

Problem 1-4. [20 points] **Augmented Binary Search Trees**

Classes have started and you are still searching for an "Introduction to Algorithms" textbook. As you search for textbooks, you decide to use an AVL tree to keep track of the prices you come across. Besides the standard operations of an AVL tree (e.g., insert(k) and search(k)), you also want to the tree to support num_ textbooks_ in_ range(a, b), which finds how many textbooks fall within the price range a to b inclusive. For example, if the set of textbook prices were [6, 9, 16, 3, 5], then num_ textbooks_ in_ range(5, 11) = 3, since the prices 5, 6, and 9 are in the specified range of [5, 11].

(a) [6 points] Describe an implementation of the num_ textbooks_ in_ range(a, b) operation on a regular (non-augmented) AVL tree. What is its running time?

If we augment the AVL tree so that each node maintains the size of the subtree rooted at that node (in addition to a price and pointers to parent and children), we can speed up the num_ textbooks_ in_ range operation to take $O(\log n)$ time.

(b) [14 points] Describe your implementation of the num_ textbooks_ in_ range(a, b) operation. Argue that it takes $O(\log n)$ time.

Part B:

Problem 1-5. [50 points] **Peak Finding**

Consider an array $A[0\,..\,n-1]$ of n integers. Define a ***peak*** of A to be an index i with $0 \le i < n$ such that $A[i-1] \le A[i]$ and $A[i] \ge A[i+1]$, where we imagine $A[-1] = A[n] = -\infty$. In other words, a peak x is greater than or equal to its neighbors in A, where we treat the first and last elements as having only one neighbor. Note that A might have multiple peaks.

For example, if $A = [10, 6, 4, 3, 12, 19, 18]$, then A has two peaks, 10 and 19.

Note that the absolute maximum of A is always a peak, but it requires $\Omega(n)$ time to compute.

(a) [20 points] Write `quick_find_1d_peak(A)` to compute a peak of an array $A[0\,..\,n-1]$ of integers in $O(\log n)$ time, using the algorithm described in lecture.

Now consider an $n \times n$ matrix B of integers. Define the ***neighborhood*** of element $B[i][j]$ to consist of $B[i+1][j]$, $B[i-1][j]$, $B[i][j+1]$, and $B[i][j-1]$. To properly handle the boundary, we imagine $B[-1][j] = B[i][-1] = B[i][n] = B[n][j] = -\infty$. Define an element $B[i][j]$ to be a ***peak*** of B if it is greater than or equal to all of its neighbours. Note that the maximum element of B is a peak, but that requires $\Omega(n^2)$ time to compute.

(b) [30 points] Write `quick_find_2d_peak(B)` to compute a peak of $n \times n$ array B of integers in $O(n)$ time, using the algorithm described in lecture.

For Python coding help, we have provided code skeletons for you to fill in, including a few helpful auxiliary routines, and an implementation of the $O(n \log n)$ algorithm for 2D peak finding described in lecture (`medium_find_2d_peak`).