6.046 fall 2005 Quiz Review—modified from 6.046 Spring 2005

1. **Recurrences**
   Solve the following recurrences by giving tight Θ-notation bounds. You do not need to justify your answers, but any justification that you provide will help when assigning partial credit.

   (a) $T(n) = T(n/3) + T(n/6) + \Theta(n^{\sqrt{\log n}})$
   (b) $T(n) = T(n/2) + T(\sqrt{n}) + n$
   (c) $T(n) = 3T(n/5) + \lg^2 n$
   (d) $T(n) = 2T(n/3) + n \lg n$
   (e) $T(n) = T(n/5) + \lg^2 n$
   (f) $T(n) = 8T(n/2) + n^3$
   (g) $T(n) = 7T(n/2) + n^3$
   (h) $T(n) = T(n-2) + \lg n$

2. **True or False**
   Circle **T** or **F** for each of the following statements, and briefly explain why. The better your argument, the higher your grade, but be brief. No points will be given even for a correct solution if no justification is presented.

   T F For all asymptotically positive $f(n)$, $f(n) + o(f(n)) = \Theta(f(n))$.

   T F The worst-case running time and expected running time are equal to within constant factors for any randomized algorithm.

   T F The collection $\mathcal{H} = \{h_1, h_2, h_3\}$ of hash functions is universal, where the three hash functions map the universe $\{A, B, C, D\}$ of keys into the range $\{0, 1, 2\}$ according to the following table:

x	$h_1(x)$	$h_2(x)$	$h_3(x)$
A	1	0	2
B	0	1	2
C	0	0	0
D	1	1	0

3. **Short Answers**
   Give *brief*, but complete, answers to the following questions.

   (a) Argue that any comparison based sorting algorithm can be made to be stable, without affecting the running time by more than a constant factor.
   (b) Argue that you cannot have a Priority Queue in the comparison model with both the following properties.

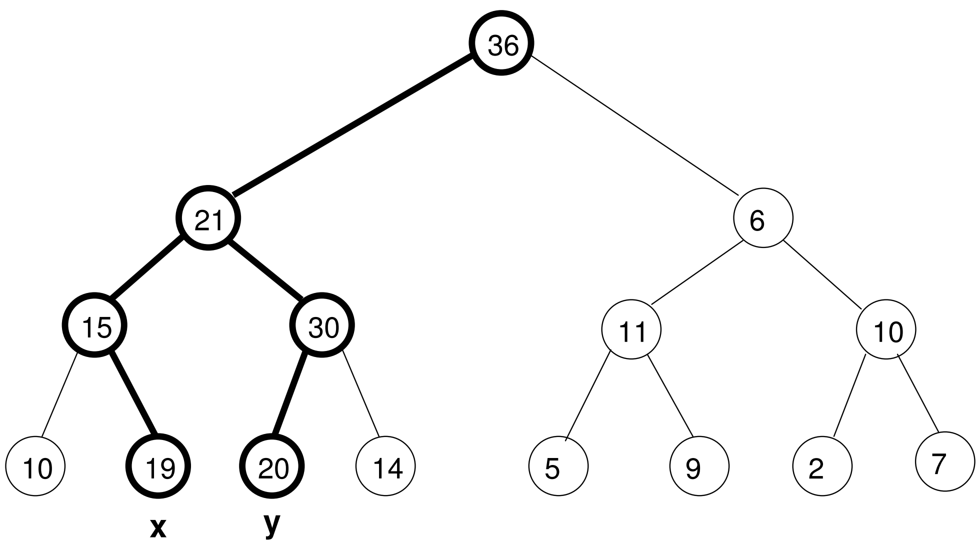
- EXTRACT-MIN runs in $\Theta(1)$ time.
- BUILD-HEAP runs in $\Theta(n)$ time.

(c) Given a max-heap in an array $A[1 \ldots n]$ with $A[1]$ as the maximum key (the heap is a max heap), give pseudo-code to implement the following routine, while maintaining the max heap property.
DECREASE-KEY(i, δ) – Decrease the value of the key currently at $A[i]$ by δ. Assume $\delta \geq 0$.

(d) Given a sorted array A of n *distinct* integers, some of which may be negative, give an algorithm to find an index i such that $1 \leq i \leq n$ and $A[i] = i$ provided such an index exists. If there are many such indices, the algorithm can return any one of them.

4. Suppose you are given a complete binary tree of height h with $n = 2^h$ leaves, where each node and each leaf of this tree has an associated "value" v (an arbitrary real number).
   If x is a leaf, we denote by $A(x)$ the set of ancestors of x (including x as one of its own ancestors). That is, $A(x)$ consists of x, x's parent, grandparent, etc. up to the root of the tree. Similarly, if x and y are distinct leaves we denote by $A(x, y)$ the ancestors of *either* x or y. That is,
   $$A(x, y) = A(x) \cup A(y) .$$

   Define the function $f(x, y)$ to be the sum of the values of the nodes in $A(x, y)$.

   A(x,y) shown in bold

   f(x,y) = 19+15+21+36+20+30 = 141

   Give an algorithm (pseudo-code not necessary) that efficiently finds two leaves x_0 and y_0 such that $f(x_0, y_0)$ is as large as possible. What is the running time of your algorithm?

5. **Sorting small multisets**

For this problem A is an array of length n objects that has at most k distinct keys in it, where $k < \sqrt{n}$. Our goal is to sort this array in time faster than $\Omega(n \log n)$. We will do so in two phases. In the first phase, we will compute a *sorted* array B that contains the k *distinct* keys occuring in A. In the second phase we will sort the array A using the array B to help us.

Note that k might be very small, like a constant, and your running time should depend on k as well as n. The n objects have satellite data in addition to the keys.

Example: Let $A = \left[5, 10^{10}, \pi, \frac{128}{279}, 10^{10}, \pi, 5, 10^{10}, \pi, \frac{128}{279}\right]$. Then $n = 10$ and $k = 4$.

In the first phase we compute $B = \left[\frac{128}{279}, \pi, 5, 10^{10}\right]$.

The output after the second phase should be $\left[\frac{128}{279}, \frac{128}{279}, \pi, \pi, \pi, 5, 5, 10^{10}, 10^{10}, 10^{10}\right]$.

Your goal is to design and analyse efficient algorithms and analyses for the two phases. Remember, the more efficient your solutions, the better your grade!

(a) Design an algorithm for the first phase, that is computing the sorted array B of length k containing the k distinct keys. The value of k is not provided as input to the algorithm.

(b) Analyse your algorithm for part (a).

(c) Design an algorithm for the second phase, that is, sorting the given array A, using the array B that you created in part (a). Note that since the objects have satellite data, it is not sufficient to count the number of elements with a given key and duplicate them. *Hint: Adapt Counting Sort.*

(d) Analyse your algorithm for part (c).