Advanced Data Structures, Practice Exam Questions, Dec 26, 2011
Purpose: Determining which topics are understood less and need to be reviewed in the Dec 28 lecture.
Instructions:
Fill in the excel sheet provided to answer whether you can solve each question in the exam or not within a reasonable time (15-30mins, depending on the question).
0: I can not solve this.
1: I can solve this, but it is difficult.
2: I can solve this, it is not too difficult.
You must submit the excel sheet by Tuesday Dec 27, noon.

Q1) Solve the following recurrences, give tight bounds.

$T(n) = T(n/3) + T(n/6) + \Theta(n^{\sqrt{\log n}})$
$T(n) = 3T(n/5) + \lg^2 n$
$T(n) = T(n-2) + \lg n$

Q2) True or False?
Q2.1.
T F For all asymptotically positive $f(n)$, $f(n) + o(f(n)) = \Theta(f(n))$.

Q2.2.
T F The worst-case running time and expected running time are equal to within constant factors for any randomized algorithm.

Q2.3.
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

Solution: True. A hash family $\mathcal{H}$ that maps a universe of keys U into m slots is *universal* if for each pair of distinct keys $x, y \in U$, the number of hash functions $h \in \mathcal{H}$ for which $h(x) = h(y)$ is exactly $|\mathcal{H}|/m$. In this problem, $|\mathcal{H}| = 3$ and $m = 3$. Therefore, for any pair of the four distinct keys, exactly 1 hash function should make them collide. By consulting the table above, we have:

$$\begin{array}{lll}
h(A) = h(B) & \text{only for } h_3 & \text{mapping into slot 2} \\
h(A) = h(C) & \text{only for } h_2 & \text{mapping into slot 0} \\
h(A) = h(D) & \text{only for } h_1 & \text{mapping into slot 1} \\
h(B) = h(C) & \text{only for } h_1 & \text{mapping into slot 0} \\
h(B) = h(D) & \text{only for } h_2 & \text{mapping into slot 1} \\
h(C) = h(D) & \text{only for } h_3 & \text{mapping into slot 0}
\end{array}$$

Q2.4.

T F Any comparison based sorting algorithm can be made to be stable, without affecting the running time by more than a constant factor.

Q2.5.

T F You can 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.

Q2.6.

T F Every binary search tree on n nodes has height $O(\log n)$.

Q2.7.

T F Computing the median of n elements takes $\Omega(n \log n)$ time for any algorithm working in the comparison-based model.

Q2.8.

T F There exists a data structure to maintain a dynamic set with operations Insert(x,S), Delete(x,S), and Member?(x,S) that has an expected running time of $O(1)$ per operation.

Q2.9.

T F The total amortized cost of a sequence of n operations (i.e., the sum over all operations, of the amortized cost per operation) gives a lower bound on the total actual cost of the sequence.

Q2.10.

T F Let A_1, A_2, and A_3 be three sorted arrays of n real numbers (all distinct). In the comparison model, constructing a balanced binary search tree of the set $A_1 \cup A_2 \cup A_3$ requires $\Omega(n \lg n)$ time.

Q2.11.

T F Let T be a complete binary tree with n nodes. Finding a path from the root of T to a given vertex $v \in T$ using breadth-first search takes $O(\lg n)$ time.

Q2.12.

T F Given an unsorted array $A[1 \, . \, . \, n]$ of n integers, building a max-heap out of the elements of A can be performed asymptotically faster than building a red-black tree out of the elements of A.

Q2.13.

T F Suppose we use a hash function h to hash n distinct keys into an array T of length m. Assuming simple uniform hashing, the expected number of colliding pairs of elements is $\Theta(n^2/m)$.

(Note: Simple uniform hashing means that the probability of element i hashing to slot k is 1/m.)

Q3. Algorithms:

Q3.1.

Given a heap in an array $A[1 \dots n]$ 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 $\delta \geq 0$.

Q3.2.

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.

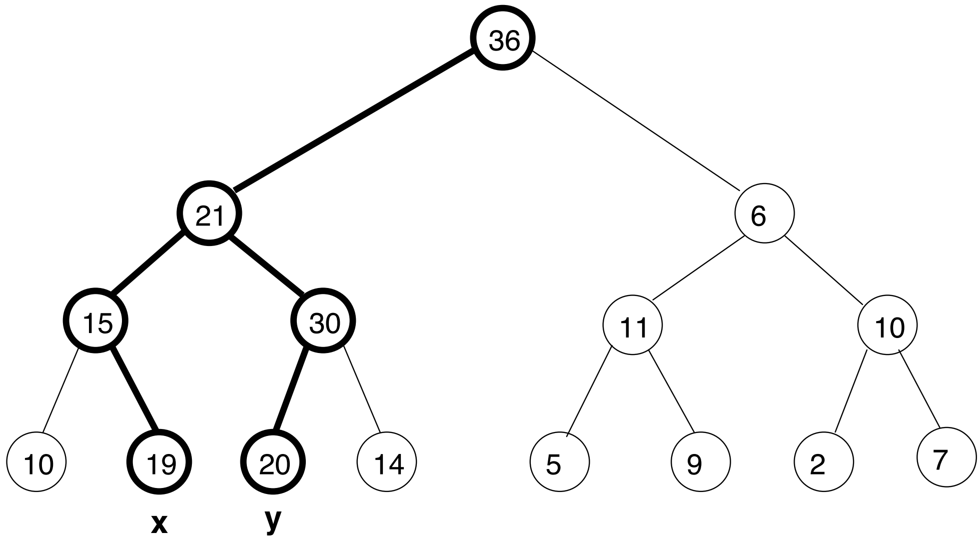
Q3.3.

Problem -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,

$$A(x, y) = A(x) \cup A(y) .$$



A(x,y) shown in bold

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

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

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?

Q3.4.

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).

Q3.5.

By applying his research in warm fission, Professor Uriah's company is now manufacturing and selling the Queueinator™, a priority-queue hardware device which can be connected to an ordinary computer and which effectively supports the priority-queue operations INSERT and EXTRACT-MIN in $O(1)$ time per operation. The professor's company has a customer, however, who actually needs a "double-ended" priority queue that supports not only the operations INSERT and EXTRACT-MIN, but also EXTRACT-MAX. Redesigning the Queueinator™ hardware to support the extra operation will take the professor's company a year of development. Help the professor by designing an efficient double-ended priority queue using software and one or more Queueinator™ devices.

Q3.6.

A matrix $M[1\,..\,n, 1\,..\,n]$ contains entries drawn from $\mathbb{R} \cup \{\infty\}$. Each row contains at most 10 finite values, some of which may be negative. The goal of the problem is to transform M so that every entry is nonnegative by using only ***pivot*** operations:

PIVOT(M, i, x)
for $j \leftarrow 1$ **to** n
  do $M[i, j] \leftarrow M[i, j] + x$
    $M[j, i] \leftarrow M[j, i] - x$

Give an efficient algorithm to determine whether there exists a sequence of pivot operations with various values for i and x such that, at the end of the sequence, $M[i, j] \geq 0$ for all $i, j = 1, 2, \ldots, n$.

Q4. Fill in the blanks.

Q4.1.

Consider a modification to QUICKSORT, such that each time PARTITION is called, the median of the partitioned array is found (using the SELECT algorithm) and used as a pivot.

The worst-case running time of this algorithm is:

Q4.2.

If a data structure supports an operation `foo` such that a sequence of n `foo`'s takes $\Theta(n \log n)$ time to perform in the worst case, then the amortized time of a `foo` operation is $\Theta\left(\boxed{}\right)$, while the actual time of a single `foo` operation could be as high as $\Theta\left(\boxed{}\right)$.

Q4.3.

Fill in the complexities of the following algorithms in O() or **Θ**() notation.

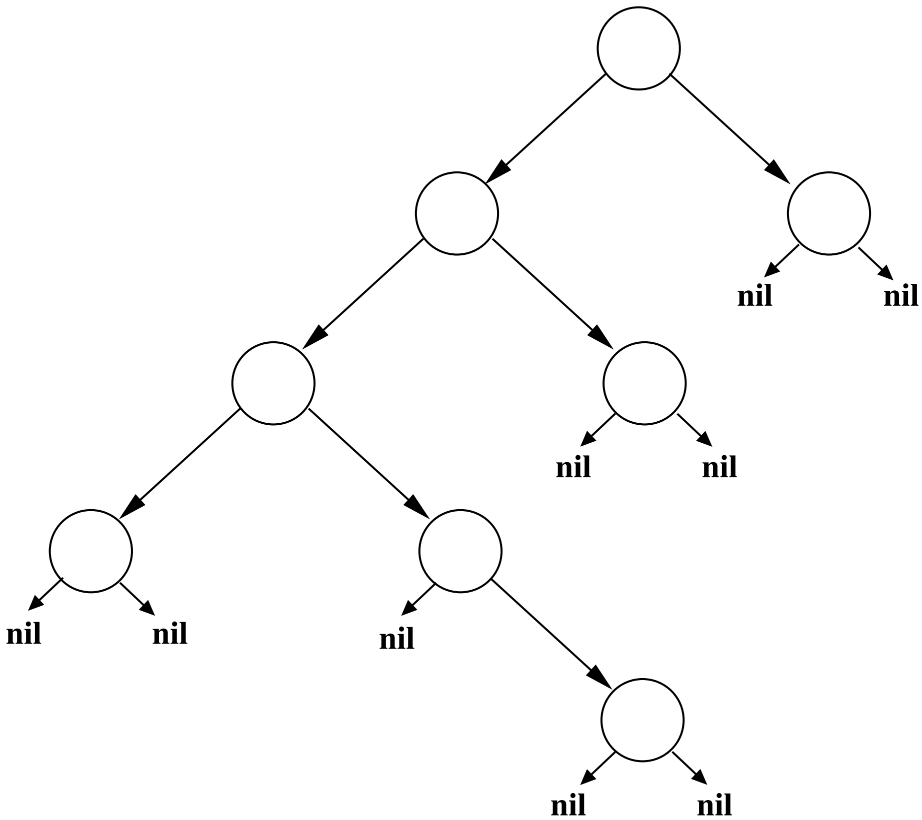
Insertion Sort

Heap Sort

Build-Heap

Q5.a

Assign the keys $2, 3, 5, 7, 11, 13, 17, 19$ to the nodes of the binary search tree below so that they satisfy the binary-search-tree property.



Q5b.

The binary search tree can be transformed into a red-black tree by performing a single rotation. Draw the red-black tree that results, labeling each node with "red" or "black." Include the keys from part (a).

Q6.

An array $A[0\,..\,k-1]$ of bits (each array element is 0 or 1) stores a binary number $x = \sum_{i=0}^{k-1} A[i] \cdot 2^i$.

To add 1 (modulo 2^k) to x, we use the following procedure:

INCREMENT(A, k)

$i \leftarrow 0$
while $i < k$ and $A[i] = 1$
    do $A[i] \leftarrow 0$
        $i \leftarrow i + 1$
if $i < k$
    then $A[i] \leftarrow 1$

Given a number x, define the potential $\Phi(x)$ of x to be the number of 1's in the binary representation of x. For example, $\Phi(19) = 3$, because $19 = 10011_2$. Use a potential-function argument to prove that the amortized cost of an increment is $O(1)$, where the initial value in the counter is $x = 0$.

Q7.
These are learning style questions, which may help us understand why some material is learned better than the others. Answer these as **a** or **b** in the excel worksheet.

Q7.1.
When I think about what I did yesterday, I am most likely to get
(a) a picture.
(b) words.

Q7.2.
I prefer to get new information in
(a) pictures, diagrams, graphs, or maps.
(b) written directions or verbal information.

Q7.3.
In a book with lots of pictures and charts, I am likely to
(a) look over the pictures and charts carefully.
(b) focus on the written text.

Q7.4.
I like teachers
(a) who put a lot of diagrams on the board.
(b) who spend a lot of time explaining.

Q7.5.
I remember best
(a) what I see.
(b) what I hear.

Q7.6.
When I get directions to a new place, I prefer
(a) a map.
(b) written instructions.

Q7.7.
When I see a diagram or sketch in class, I am most likely to remember
(a) the picture.
(b) what the instructor said about it.

Q7.8.
When someone is showing me data, I prefer
(a) charts or graphs.
(b) text summarizing the results.

Q7.9.
When I meet people at a party, I am more likely to remember
(a) what they looked like.
(b) what they said about themselves.

Q7.10.
For entertainment, I would rather
(a) watch television.
(b) read a book.

Q7.11.
I tend to picture places I have been
(a) easily and fairly accurately.
(b) with difficulty and without much detail.

Q7.12.
How do you learn a difficult material best?

Q7.13.
How could you have learned this course better?

From Felder:

How can visual learners help themselves?

If you are a visual learner, try to find diagrams, sketches, schematics, photographs, flow charts, or any other visual representation of course material that is predominantly verbal. Ask your instructor, consult reference books, and see if any videotapes or CD-ROM displays of the course material are available. Prepare a concept map by listing key points, enclosing them in boxes or circles, and drawing lines with arrows between concepts to show connections. Color-code your notes with a highlighter so that everything relating to one topic is the same color.

How can verbal learners help themselves?

Write summaries or outlines of course material in your own words. Working in groups can be particularly effective: you gain understanding of material by hearing classmates' explanations and you learn even more when you do the explaining.