MIDTERM
Fundamental Algorithms, Spring 2008, Professor Yap
March 10, 2008

INSTRUCTIONS:

0. This is a closed book exam, with one 8"x11" (2-sided) cheat sheet.
1. Please answer ALL questions (there is ONE extra credit question)
2. Please write clearly. Use complete sentences. This CAN AFFECT your grade. If necessary, consider printing.
3. Write answers ONLY on the front side of each booklet page. Use the reverse side for scratch and to show your working.
4. Read carefully, think deeply, write sparsely.
5. Begin each question in its own page.

SHORT QUESTIONS

(4 Points Each)
No proofs required for this part (one line of informal justification can be entertained). Just state upper and lower bounds for the following sums or recurrence functions. Ideally, we want Θ-bounds.

- (a) $\sum_{i=1}^{n} i^3(i!)$
- (b) $\sum_{i=1}^{n} \frac{i^3}{2^i}$
- (c) $T(n) = 3T(n/10) + \sqrt{n}/\log n$.
- (d) $T(n) = 10T(n/3) + \sqrt{n}\log n$.
- (e) Order these 7 functions in increasing Θ-order:

$$\frac{n^2}{\lg n}, \quad n^2 \lg\lg n, \quad 2^{\lg n}, \quad 2^n, \quad 4^{\lg n}, \quad n!, \quad n^{\lg n},$$

- (f) What is the minimum number of nodes in an AVL tree of height 4?
- (g) What is the minimum number of nodes in a $(3, 4)$-tree of height 3?
- (h) Begin with the AVL tree in Figure 1, and insert the key 8.8. Show the final AVL tree.

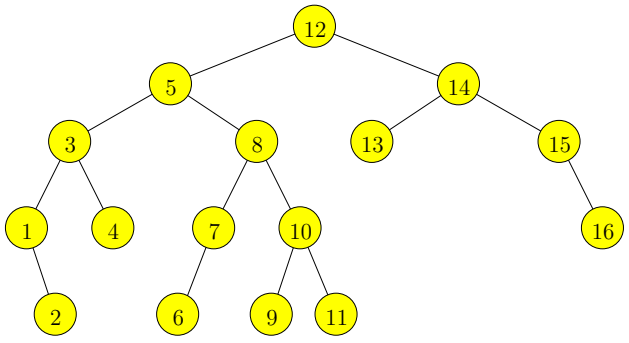

Figure 1: An AVL tree

- (i) Begin with the AVL tree in Figure 1, and delete the key 15. Show the final AVL tree.

SOLUTION: Note that we provide justifications below, but you need not.
(a) $\Theta(n^3(n!))$.
REMARK: Justification is by our summation rule for exponentially large sums. Of course, you did not have to say this. Our instructions said to JUST STATE the bound. Put any derivation on the work sheet if you want, but NOT on the answer part (some students write several pages for one subpart!).
Some students think this is a polynomial sum.
(b) $\Theta(1)$.
REMARK: Justification is by our summation rule for exponentially small sums. Some students say that this is $O(n^3/2^n)$. Why is this wrong?
(c) $\Theta(\sqrt{n}\lg n)$.
REMARK: To justify, you need to see that $\log_{10} 3 < \log_9 3 = 1/2$. Thus the driving functions dominates. You also need to check the regularity condition: if $f(n) = n$. then for some $0 < c < 1$, $cf(n) \geq 10f(n/3)$.
(d) $\Theta(n^\alpha)$ where $\alpha = \log_3(10) > 2$.
REMARK: Again by Master Theorem. You need to see that $\log_3 10 > \log_3 9 > 2$.
(e)

$$2^{\lg n}(= n) \preceq \frac{n^2}{\lg n} \preceq 4^{\lg n}(= n^2) \preceq n^2 \lg\lg n \preceq n^{\lg n}(= 2^{\log^2 n}) \preceq 2^n \prec n!(= 2^{\Theta(n \log n)}).$$

COMMON mistakes:
Students did not recognize $2^{\lg n} = n$, or $4^{\lg n} = n^2$. You should know a standard transformation here: $a^{\log_b c} = c^{\log_b a}$ for all a, b, c.
Some think that $2^n \preceq n^{\log n}$. To see that this is wrong, take log of both sides, and you will get $n \leq C + \log^2 n$ for some C. This is clearly wrong.
Some think $n^2 \lg\lg n \preceq n$. I see how you come to this conclusion – you think $\lg\lg n$ grows so slowly that it overwhelms that fact that n^2 grows faster than n...but you cannot argue this way. The correct way is to order a "product of powers" in lexicographic order – we consider the most important power which is different in the 2 products. Then the decision is determined by the power of this term only!
(f) There are 12 nodes. Use the formula that $\mu(h+1) = \mu(h-1) + \mu(h)$, where $\mu(-1) = 0$, $\mu(0) = 1$.
(g) There are 26 nodes. Use the fact that the root must have 2 children and the remaining internal nodes have 3 children each. STANDARD MISTAKE: you assume the root must have 3 children.
(h) First insert 8.8 into the tree, then do `rotate`$^2(8)$. This is basically the same as the insertion of 9.5, described in Figure 14 in Lecture III.
(i) First delete 15 from the tree, then do `rotate`(5). This is basically the same as the deletion of 13, described in Figure 16 in Lecture III.

REGULAR QUESTIONS

1. (4 Points Each)
   If true, prove it. If false, give counter example. For this problem, assume that $f(x) > 0$ for all x.
   - (a) T/F: If f is polynomial-type, so is f^a for any real $a > 0$.
   - (b) T/F: If f is polynomial-type, so is $\lg f$.
   - (c) T/F: If f is exponential-type, then f^a is exponential-type for any real $a > 0$.
   - (d) T/F: If f is a increasing exponential-type, so is 2^f.

SOLUTION:

- (a) True: If f is polynomial-type, so is f^a for any real $a > 0$.
  Pf: Let $f(n) \leq Cf(n/2)$ (ev.) for some $C > 0$. Then $f(n)^a \leq C^a f(n/2)^a$.
- (b) False: If f is polynomial-type, so is $\lg f$.
  Counter example: let $f(n) = (n-1)/n = 1 - (1/n)$. Then $f(n) \leq 2f(n/2)$ but $\lg f(n) < 0$ means that $\lg f(n) \neq \mathcal{O}(f(n/2))$.
  REMARK: Some students give $1/n$ as a counter-example. Why is this not polynomial-type?
- (c) True: If f is exponential-type, then f^c is exponential-type for any real $c > 0$.
  Pf: For any $K > 1$, if $f(n-1) \geq Kf(n-1)$ implies $f(n)^c \geq K^c f(n-1)^c$ and $K^c > 1$. For any $k < 1$, if $f(n-1) \leq kf(n-1)$ implies $f(n)^c \leq k^c f(n-1)^c$ and $k^c < 1$.
  REMARK: do not forget that you need to show this for exponentially large as well as exponentially small functions.
- (d) True: If f is increasing exponentially, so is 2^f.
  Pf: Suppose $f(n-1) \geq Kf(n-1)$ for some $K > 1$. Note that $f(n) \geq c$ (ev.) for any $c > 0$. Hence $Kf(n) \geq (K-1)c + f(n)$ (ev.). Thus

$$\begin{array}{rcll} 2^{f(n)} & \geq & 2^{Kf(n-1)} & \\ & \geq & 2^{(K-1)c+f(n-1)} & \text{(ev.)} \\ & \geq & K' 2^{f(n-1)} & \end{array}$$

  where $K' := 2^{(K-1)c} > 1$.
- (e) True: If f is a shrinking exponential-type, so is 2^{-f}.
  Pf: Suppose $f(n-1) \leq kf(n-1)$ for some $0 < k < 1$. Note that $kf(n) \leq -k + f(n)$ (ev.) since $f(n) \to 0$. Thus

$$\begin{array}{rcll} 2^{f(n)} & \leq & 2^{kf(n-1)} & \\ & \leq & 2^{-k+f(n-1)} & \text{(ev.)} \\ & = & 2^{-k} 2^{f(n-1)} & \\ 2^{-f(n)} & \geq & 2^k 2^{-f(n-1)} & \end{array}$$

  where $2^k > 1$.

2. (2+8 Points)
The following statement is a fact: *a planar graph on n vertices has at most $3n-6$ edges.* Let us restate it as follows:

(G is a planar graph and has n vertices) $\Rightarrow$ (G has $\leq 3n-6$ edges).

(i) State the contra-positive of this statement.
(ii) The complete graph on 5 vertices, denoted by K_5 is shown in Figure 2. Using the contra-positive statement in part (i), prove that K_5 is not planar.

SOLUTION:

(i) Constra-positive:

(G has $> 3n-6$ edges) $\Rightarrow$ (G is not a planar graph or does not have n vertices).

(ii) Let $n = 5$. Then K_5 has $> 3n - 6 = 9$ edges. By the contra-positive of (i), we conclude that K_5 is not a planar graph or does not have 5 vertices. Since it does have 5 vertices, it must not be planar.

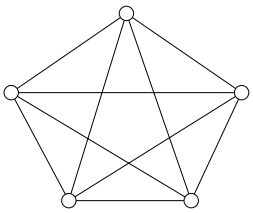

Figure 2: K_5, the complete graph on 5 vertices

3. (10 Points)
Let $size(u)$ denote the number of nodes in the tree rooted at u. Say that node u is **size-balanced** if it is a leaf or else

$$1/2 \leq size(u.\texttt{left})/size(u.\texttt{right}) \leq 2.$$

Use shell programming to give an algorithm $BALANCE(u)$ that returns TRUE iff every node below u is balanced. DO NOT assume any additional fields in the nodes.

Please use the informal style used in our lecture notes (use indentation to indicate the logical structure of your program, and do not declare variables, etc.)

SOLUTION: A common mistake is to assume that you already have the value $size(u)$ for each node u. We will simply use the "return shell" called $rPOST(u)$ in the lecture notes. We assume that $rPOST(u)$ returns the size of the binary tree rooted at u if the tree is size-balanced; otherwise, $rPOST(u) = -1$. The required procedure $BALANCE(u)$ can be easily constructed from $rPOST(u)$. Here is $rPOST$ again:

```
rPOST(u)
    BASE(u)
    L ← rPOST(u.left)
    R ← rPOST(u.right)
    VISIT(u, L, R)
```

We must program the BASE and VISIT macros:

```
BASE(u):
    if (u = nil) return(−1)
    if (u.left = nil and u.right = nil) return(1)
    if (u.left = nil or u.right = nil)
RETURN(−1)
```

The macro VISIT macro is

```
BASE(u, L, R):
    if (1/2 ≤ L/R ≤ 2) return(1 + L + R)
    else return(−1)
```

4. (EXTRA CREDIT ONLY)
Suppose $X_1, \ldots, X_n$ are n sorted lists, each with k elements. We want to the set $X = \bigcup_{i=1}^{n} X_i$. Here is a sketch of an algorithm: at each phase, we merge pairs of lists. The phases stop when there is only one list left. In a general phase with m lists of size ℓ each, the merging produce $m/2$ lists each of size 2ℓ (assume m is even). The merging in this phase take time $O(m\ell)$.

(a) Set up the recurrence for the complexity $T(n,k)$ of this algorithm.

(b) Show that $T(n,k) = \mathcal{O}(nk \log n)$. HINT: domain transformation can be used but it is not necessary.

(c) Using the Information Theoretic Bound, prove that $T(n,k) = \Omega(nk \log n)$. (WARNING: This might be hard for many students.)

SOLUTION: (a) The recurrence is $T(n,k) = T(n/2, 2k) + nk$ when n is a power of 2. By our general practice, we will extend this recurrence for arbitrary real values of n.
(b) Let $N = \lg n$ and $K = \lg k$. Initially we assume n is a power of 2. We can transform $T(n,k)$ to $t(N,K)$ as follows:

$$\begin{aligned} t(N,K) &:= T(2^N, 2^K) \\ &= t(N-1, K+1) + 2^{N+K} \\ &= t(N-2, K+2) + 2 \cdot 2^{N+K} \\ &= \cdots \\ &= t(0, N+K) + N \cdot 2^{N+K} \end{aligned}$$

Now, $t(0, N+K) = T(1, 2^{N+K}) = 0$, clearly. Hence the solution to $T(n,k) = \lg n \cdot nk$. When n is not a power of 2, we stop when we reach $t(\{N\}, N+K)$ where $0 \le \{N\} < 1$, and $\{N\}$ is the fractional part of N.
ALTERNATIVE SOLUTION (Millstone): we maintain a minimum priority queue which normally holds exactly n keys, one key from each of the n lists. To sort, we just `deleteMin`() from the queue, note the list where the deleted item comes from, and `insert` the next item from that list. There are nk `deleteMin`'s and `insert`'s, and each takes time $O(\lg n)$. This proves our upper bound.
(c) Let $C(n,k)$ denote the number of combinatorially distinct possible outcomes when we merge n lists each of size k. This number is the multinomial

$$\begin{aligned} C(n,k) &= \binom{nk}{\underbrace{k, k, \ldots, k}_{n}} \\ &= \frac{(nk)!}{(k!)^n} \end{aligned}$$

Using Stirling's approximation, it is easy to see that $\log C(n,k) = \Omega(nk \log n)$