Homework 2 Solutions
Fundamental Algorithms, Spring 2008, Professor Yap

Due: Wed Feb 20, in class.
HOMEWORK with SOLUTION, prepared by Instructor and T.A.s

INSTRUCTIONS:

- Please read questions carefully. When in doubt, please ask. You may also post general questions to the homework discussion forum in class website. Also, bring your questions to recitation on Monday.
- There are links from the homework page to the old homeworks from previous classes, including solutions. Feel free to study these.

---

1. (4 Points)
Exercise A.3 in Lecture 1. (De Morgan's Law applied to quantifiers)
Consider the following sentence:

$$(\forall x \in \mathbb{Z})(\exists y \in \mathbb{R})(\exists z \in \mathbb{R}) \left[(x > 0) \Rightarrow ((y < x < y^{-1}) \wedge (z < x < z^2) \wedge (y < z))\right] \qquad (1)$$

Note that the range of variable x is $\mathbb{Z}$, not $\mathbb{R}$. This is called a **universal sentence** because the leading quantifier is the universal quantifier ($\forall$). Similarly, we have **existential sentence**.
(i) Negate the sentence (1), and then apply De Morgan's law to rewrite the result as an existential sentence.
(ii) Give a counter example to (1).
(iii) By changing the clause "$(x > 0)$", make the sentence true. Indicate why it would be true.

SOLUTION:
(i)

$$(\exists x \in \mathbb{Z})(\forall y \in \mathbb{R})(\forall z \in \mathbb{R}) \left[(x > 0) \wedge (\neg(y < x < y^{-1}) \vee \neg(z < x < z^2) \vee (z \leq y))\right]$$

(ii) A counter example is $x = 1$. If $z < x$, then we must have $z < 1$ and so $z^2 < z$. Now, z must be positive, as otherwise, it would not satisfy the condition $y < z$. Hence $0 < z < 1$. Thus $(z < x < z^2)$ cannot hold.
(iii) We change $(x > 0)$ to $(x > 1)$. This removes the counter example. Nor we can always choose z so that $(z < x < z^2)$. Now if we choose a positive y sufficiently small, we can also satisfy the remaining clauses.

2. (10 Points)
Exercise 7.6, Lecture 1. (Redoing question from hw1, with new assumptions)
Provide either a counter-example when false or a proof when true. The base b of logarithms is arbitrary but fixed, and $b > 1$. Unlike in hw1, we now assume the functions f, g are unbounded and ≥ 0 eventually.
(a) $f = \mathcal{O}(g)$ implies $g = \mathcal{O}(f)$.
(b) $\max\{f, g\} = \Theta(f + g)$.
(c) If $g > 1$ and $f = \mathcal{O}(g)$ then $\ln f = \mathcal{O}(\ln g)$. HINT: careful!
(d) $f = \mathcal{O}(g)$ implies $f \circ \log = \mathcal{O}(g \circ \log)$. Assume that $g \circ \log$ and $f \circ \log$ are complexity functions.
(e) $f = \mathcal{O}(g)$ implies $2^f = \mathcal{O}(2^g)$.
(f) $f = o(g)$ implies $2^f = \mathcal{O}(2^g)$.
(g) $f = \mathcal{O}(f^2)$.
(h) $f(n) = \Theta(f(n/2))$.

SOLUTION: Many false statements are now true:

(a) False, as before.

(b) True. $\max(f,g) \le f+g$ (ev), since both functions are ≥ 0 (ev). But $f+g \le 2\max(f,g)$ (ev).

(c) True. Since $f = \mathcal{O}(g)$, we have $f \le Cg$ (ev) for some $C > 1$. So $\lg f \le \lg C + \lg g$. As g is grows unbounded, we have $g \ge C$ (ev) and so $\lg f \le \lg C + \lg g = 2\lg g$ (ev).

Alternative proof (based on limits, which I generally avoid): As f is increasing, eventually $f > 1$, so $\ln(f) > 0$ and moreover, it too is an increasing function of n. Looking at the limit as n goes to infinity of $\ln(f)/\ln(g) = (1/f)/(1/g) = g/f = C$, we see that $ln(f) = \mathcal{O}(\ln(g))$.

(d) True, as before.

(e) False, as before.

(f) True. as before.

(g) True. We have $1 \le f$ (ev) and hence $f \le f^2$ (ev). Thus $f = \mathcal{O}(f^2)$.

(h) False, as before.

3. (12 Points)
Exercise 0.2, Lecture 2. Let $T(n) = aT(n/b) + n$, where $a > 0$ and $b > 1$. How sensitive is this recurrence to the initial conditions? More precisely, if $T_1(n)$ and $T_2(n)$ are two solutions corresponding to two initial conditions, what is the strongest relation you can infer between T_1 and T_2?

SOLUTION: We can be slightly more general, and let the driving function be $d(n)$; our problem assumes $d(n) = n$. We can use rote expansion

$$\begin{aligned} T(n) &= aT(n/b) + d(n) \\ &= a^2T(n/b^2) + ad(n/b) + d(n) \quad (\mathit{Expand}) \\ &= \cdots \\ &= a^iT(n/b^i) + \textstyle\sum_{j=0}^{i-1} a^j d(n/b^j) \end{aligned}$$

after expanding i times. Choose $i = \log_b n - C$ for some $C > 0$ where where both $T_1(n/b^i)$ and $T_2(n/b^i)$ are $\Theta(1)$. In the special case where $d(n) = n$, we see that

$$\begin{aligned} T(n) &= a^iT(n/b^i) + n\sum_{j=0}^{i-1}(a/b)^j \\ &= \Theta(a^i) + \Theta\begin{cases} n(a/b)^i & \text{if } a > b \\ ni & \text{if } a = b \\ n & \text{if } a < b \end{cases} \\ &= \Theta\begin{cases} n^{\log_b a} & \text{if } a > b \\ n\log_b n & \text{if } a = b \\ n & \text{if } a < b. \end{cases} \end{aligned}$$

This shows that the initial conditions have no effects on the final solution!

4. (10 Points)
Exercise 2.8, Lecture 2. (Google problem) Recall the problem of computing a minimum cover for a set $W = \{w_1, \ldots, w_k\}$ of k key words in a file. For each key word w_i, we are given a sorted list $P(w_i)$ of the positions where w_i occurs in the file.
(a) Solve the minimum cover problem for $k = 2$ in linear time.

(b) Suppose $P(w_i) = (s_i, t_i)$ for $i = 1, \ldots, k$, i.e., each keyword has just two positions. Give an $O(k \log k)$ algorithm to find the minimum cover J for $w_1, \ldots, w_k$.

▷ ____

SOLUTION (a) Let $P(w_1)$ and $P(w_2)$ be merged into the list $A[1..n]$. The minimum cover has the form $J = [A[i], A[i+1]]$ where (i) $A[i] - A[i+1]$ has minimum absolute value, and (ii) $A[i] \in P(w_1)$ iff $A[i+1] \in P(w_2)$.

(b) Let $P(w_i) = (s_i, t_i)$ for $i = 1, \ldots, k$. Wlog, assume $s_1 < s_2 < \cdots < s_k$. Let $S = \{s_1, \ldots, s_k\}$ and and $T = \{t_1, \ldots, t_k\}$. We first sort the set $S \cup T$ in $O(k \log k)$ time. Let L be the sorted list whose elements are $S \cup T$, sorted in increasing order (so the first element of L is s_1). Let the set of minimal covers be $C_1, \ldots, C_m$ where $C_i = [c_i, d_i]$. We may assume so that $c_1 < c_2 < \cdots < c_m$, and hence (by minimality) also $d_1 < d_2 < \cdots < d_m$. We will first show (BASE CASE) how to compute C_1, and from C_i, we show (INDUCTIVE CASE) how to compute C_{i+1} or else determine that $i = m$. The computation of $C_1, \ldots, C_m$ will take $O(k)$, hence linear time.

Each cover $C = [c, d]$ is represented by a pair of pointers to the numbers corresponding to c and d in the sorted list L. We call t_i the **partner** of s_i; and if $c \in S$, we can speak of the partner of c. For each $c \in L$, let the **successor** of c be the next larger element in the list L. If c is equal to s_i (for some i) then the **partner** of c is t_i in the list L; if $c \in T$, then the partner of c is undefined.

First we will show how to find $C_1 = [c_1, d_1]$. It is easy to see that no d_i's can be smaller that s_k. Moreover, we see that $[s_1, s_k]$ is a cover, not necessary minimal. Hence $d_1 = s_k$. To find c_1, we initialize the variable c to s_1: $c \leftarrow s_1$. Then we enter a while-loop:

```
$c \leftarrow s_1$; $d_1 \leftarrow s_k$
  ◁ *LOOP INVARIANT:* $[c, d_1]$ *is a cover*
while ($c \in S$ and the partner of $c$ lies in $[c, d_1]$)
    $c \leftarrow$ successor of $c$.
$c_1 \leftarrow c$; ▷ *ASSERTION:* $[c_1, d_1]$ *is a minimal cover*
```

When we exit from the while-loop, there are two cases: either (1) $c \in T$ or (2) $c \in S$ but the partner of c is larger than d_1. In either case, we can set c_1 to this value of c. To see that $[c_1, d_1]$ is minimal, we only have to note that if c_1 is replaced by its successor, it would no longer be a cover (by virtue of (1) or (2)). This concludes our construction of C_1.

Inductively, suppose we are given $C_i = [c_i, d_i]$. If $c_i \in T$, then we see that C_i is the last minimal cover (i.e., $i = m$). If $c_i \in S$, then by minimality of C_i, the partner of c_i must be larger than d_i. We let d_{i+1} be the partner of c_i. It is easy to see that $[succ(c_i), d_{i+1}]$ is a cover, This proves that d_{i+1} is indeed correctly computed. It remains to compute c_{i+1}. We use the same while-loop as before. Namely, we initialize c to $succ(c_i)$, and as long as $c \in S$ and the partner of c lies in $[c, d_{i+1}]$, we replace c by its successor. When we exit from this loop, we set c_{i+1} to c. It is easy to see that $[c_{i+1}, d_{i+1}]$ is a minimal cover. This concludes our algorithm.

Since the work amounts to following pointers to successors or partners, and these pointers point from smaller to larger, the complexity of these pointer jumping is $O(k)$.

____ ◁

5. (0 Points)
Exercise 3.1, Lecture 2.

6. (18+5 Points)
Exercise 6.1, Lecture II. (a) Verify that the examples in (55), (56), and (57) in Lecture II are polynomial-type or exponential-type, as claimed.
(b) Is the summation $\sum_{i=1}^{n} i^{\lg i}$ an exponential type or polynomial type? Give bounds for the summation.

SOLUTION: We will show one from each section.

(a) $f(i) = i \log i$. We will show this is of polynomial-type. Let us compare $f(i)$ and $f(i/2)$ by looking at the following for some $C > 0$ (we will determine the actual C later).

$$i \log i \le C(i/2)\log i/2$$

Moving the constant $1/2$ into the constant C and dividing both sides by i and $\log i/2$, we obtain the relationship

$$\log i / \log(i/2) \le C$$

In other words, we are asking: does the ratio on the left hand side have a finite limit as we let i goto infinity? Using L'Hospitals rule, we take limits of the top and bottom to see that this limit is in fact finite. Moreover, this ratio approaches the limit from above, monotonically. Therefore, for any N, we let $C = \log N/\log(N/2)$ and the result follows.

(b) $f(i) = 2^i$. We will show this is of exponential type. To do this we must show $2^i \ge C * (2^{i-1})$. However, to observe that $2^i = 2 * 2^{i-1}$ and the result follows.

(c) $f(i) = i^{\lg i}$. Let first observe, that this function is not of polynomial type. As above, we look at the limit of the ratio $f(i)/f(i/2)$. Using basic properties of the logarithm, we can write this ratio as

$$i^{\lg(i)}/i^{(\lg(i/2))} * 2^{\lg i/2} = 1/2 * i^2$$

Clearly this limit is unbounded, so f cannot be of polynomial-type.

We now claim that $f(i)$ is not exponential type either! Suppose there is some $C_0 > 1$ such that

$$f(i) \ge C_0 f(i-1) \quad \text{(ev.)}. \tag{2}$$

We first note two useful bounds: From the fact that $H_n = \ln n + \gamma + 1/(2n) + O(n^{-2})$ (see notes on the Harmonic function H_n in §15, Lect.II). we conclude that

$$\ln n + \gamma + (1/n) \ge H_n \ge \ln n + \gamma \quad \text{(ev.)}. \tag{3}$$

Hence,

$$\begin{aligned} \ln n &\le H_n - \gamma \\ &= (1/n) + H_{n-1} - \gamma \\ &\le (1/n) + \ln(n-1) + (1/n) \\ &= \ln(n-1) + (2/n), \end{aligned}$$

eventually. Dividing this by $\ln 2$, we get:

$$\lg n \le \lg(n-1) + (2K/n) \quad \text{(ev.)} \tag{4}$$

where $K = 1/\ln 2$. The following bound is from Appendix A (Lect.II): for $|x| < 1$, $x > \ln(1+x) > x/2 \quad$ (ev.). Again, dividing by $\ln 2$, we get

$$Kx > \lg(1+x) > Kx/2 \quad \text{(ev.)}. \tag{5}$$

We see that

$$\begin{aligned} f(i) &= \left[(i-1)(1+\tfrac{1}{i-1})\right]^{\lg i} \\ &\le (i-1)^{\lg(i-1)+(2K/i)}(1+\tfrac{1}{i-1})^{\lg i} && (\text{ by } (4)) \\ &= f(i-1)\cdot(i-1)^{2K/i}\cdot 2^{\lg(1+\frac{1}{i-1})\lg i} \\ &\le f(i-1)\cdot 2^{2K\lg(i-1)/i}\cdot 2^{\frac{\lg i}{i-1}} && (\text{ by } (5)) \\ &= f(i-1)\cdot C_2(i)\cdot C_3(i) \end{aligned}$$

where $C_2(i) := 2^{2K\lg(i-1)/i}$ and $C_3(i) := 2^{\frac{\lg i}{i-1}}$. But notice that $\lg C_2(i) = 2K\lg(i-1)/i$ and $\lg C_3(i) = \lg i/(i-1)$ both goes to 0 as $i \to \infty$. Hence $C_2(i) \le$

7. (10 Points)
Use the Rote Method to solve the following $T(n) = 11T(n/3) + n^2$ You must indicate clearly the EGVS steps.

SOLUTION: As above in Problem 3 above, we have

$$\begin{aligned} T(n) &= aT(n/b) + d(n) \\ &= a^i T(n/b^i) + \sum_{j=0}^{i-1} a^j d(n/b^j). \end{aligned}$$

In Problem 3, we only do the guessing but did not do the verification. But you must to do this in your solution.
Specializing to our particular recurrence,

$$\begin{aligned} T(n) &= 11^i T(n/3^i) + \sum_{j=0}^{i-1} 11^j (n/3^j)^2 \\ &= 11^i T(n/3^i) + n^2 \sum_{j=0}^{i-1} (11/9)^j \\ &= 11^i T(n/3^i) + n^2 (11/9)^i \end{aligned}$$

Now we stop by choosing $i = \log_3 n - C$ for some $C > 0$, we get

$$\begin{aligned} T(n) &= \Theta(n^{\log_3 11}) + n^2 (11/9)^{\log_3 n} \\ &= \Theta(n^{\log_3 11}). \end{aligned}$$

8. (20 Points)
Use Real Induction to give good upper and lower bounds on the recurrence $T(n) = T(n/2) + T(n/3) + T(n/4) + n$. HINT: try different values of α, β to get bounds of the form $K' n^\alpha \leq T(n) \leq K n^\beta$ (ev.)

SOLUTION: We want to determine the α such that $T(n) = \Theta(n^\alpha)$ (a general theorem in Lecture 10.3 guarantees that such an α exists). But you can get most of the credits for such questions by giving good upper and lower bounds on α.
For instance, we know that $\alpha \geq 1$ because $T(n) \geq n$. How good is this? We will see that this is surprisingly good.
Let us guess some upper bound, like $\alpha \leq 2$. This amounts to proving that $T(n) \leq Kn^2$ (ev.). So we just try it:

$$\begin{aligned} T(n) &= T(n/2) + T(n/3) + T(n/4) + n \\ &\leq K(n/2)^2 + K(n/3)^2 + K(n/4)^2 + n \\ &= Kn^2 \left[\frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{Kn} \right] \\ &= Kn^2 \end{aligned}$$

provided

$$\left.\begin{array}{l} \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{Kn} \leq 1 \\ \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \frac{1}{Kn} \leq 1 \end{array}\right\}$$

Since $(1/4) + (1/9) + (1/16) < 3/4$, the inequality (6) will clearly hold for $Kn \geq 4$. Hence, we have proven $T(n) \leq Kn^2$ by real induction. You can now try to see if $\alpha \leq 1.5$ (but for this, you probably need a pocket calculator to estimate the value of

$$\frac{1}{2^{1.5}} + \frac{1}{3^{1.5}} + \frac{1}{4^{1.5}}.$$

More generally, let

$$h(x) = \frac{1}{2^x} + \frac{1}{3^x} + \frac{1}{4^x}.$$

As x goes from $-\infty$ to ∞, $h(x)$ changes monotonically from ∞ to 0. Hence there is a unique α such that $h(x) = 1$. Moreover, our theorem in Lecture 10.3 assures us that $T(n) = \Theta(n^\alpha)$.
This is an aside for those of you who want to explore this more: Suppose we want to numerically approximate the $\alpha = 1.???$. For a simple way to do this, use a user-friendly, powerful software like `MATLAB`. For instance, consider the following two lines of `MATLAB` code:

```
>>    h = @(x) 2.^ (-x) + 3.^ (-x) + 4.^ (-x);
>>    for x = 0.9 : 0.1 : 1.2, display([x, h(x)]), end
```

It will produce the values for first of the following four tables:

x	$h(x)$	x	$h(x)$	x	$h(x)$	x	$h(x)$
0.9000	1.1951	1.0700	1.0119	1.0810	1.0011	1.0820	1.0001
1.0000	1.0833	<u>1.0800</u>	1.0021	<u>1.0820</u>	1.0001	<u>1.0821</u>	1.0000
<u>1.1000</u>	0.9828	1.0900	0.9924	1.0830	0.9992	1.0822	0.9999
1.2000	0.8923	1.1000	0.9828	1.0840	0.9982	1.0823	0.9998

NOTE: you should be able to repeat this experiment with `MATLAB` using the public terminals of our department. By changing the stepsize and limits of the for-loop, we can get more correct digits with each iteration. Each of the tables able is based on a new for-loop that yields one extra digit in the decimal expansion of α. Thus, $\alpha \approx 1.0821$. Suggestion: how can you continue this experiment to determine the first 100 digits of α?