Homework 1 Solutions
Fundamental Algorithms, Fall 2004, Professor Yap

Due: Thu Sep 23, in class
SOLUTION PREPARED BY T.A. Vikram Sharma and Chee Yap

INSTRUCTIONS:

- Please read questions carefully. When in doubt, please ask.
- There are links from the homework page to the old home-works from previous classes, including solutions. Feel free to study these.

---

1. (15 Points)
In the first lecture, we described a conventional program for merging two sorted lists.
(a) Please draw a comparison-tree for merging two sorted lists of numbers, $(x < y)$ and $(a < b < c < d)$. Your comparison-tree should be obtained by "unwinding" the algorithm described in class. HINT: you can appeal to symmetry to only draw a portion of the comparison-tree.
(b) What is the height of your comparison tree?
(c) Determine $M(2,4)$ (give as good an upper and lower bound as you can). Do not quote some known result – you need to do an argument from first principles. HINT: make two separate arguments for upper and lower bounds, respectively.

Solution:
(a) Omitted, but most of the students got it right. However, one has to make sure that the tree covered all the 15 possible cases that could occur as input while merging, implying that there should be 15 leaves in the tree.
(b) should be 5.
(c) The exact answer id $M(2,4) = 5$. However, this is non-trivial to prove. You receive full credit if you can show $4 \leq M(2,4) \leq 5$, which is what the following argument shows. Upper bound is 5 from the part (b). Lower bound: assume $a < x < b < y < c$. These 4 comparisons, i.e, the ones between x and a, b and the ones between y and $b.c$ are necessary.
Comments: 1) Some students were not clear about the definition of height of a tree; which is the length of the longest path from the root to a leaf, and not the number of nodes along the path (which is one more than the height). 2) For part (c) we desired a "good" upper and lower bound. For lower bounds one can trivially say that there is at least one comparison, but as shown above, this is not good enough. Better than that is to say that we need at-least n comparisons, where n is the number of elements in the smaller list — such an argument was also proposed by some students. For the complete argument of why $M(2,4) = 5$ you can refer Knuth's Vol.3 on Sorting and Searching Page 198. Gave $3 - 4$ points to those who showed $3 \leq M(2,4) \leq 5$.

2. (20 Points)
We want to consider the "best case time" in comparison trees. If T is a tree, let $B(T)$ denote the length of a *shortest* path from the root to a leaf of T.
(a) Define $M'(m,n)$ to be the length of a **shortest path** in some comparison-tree for merging two lists of sizes m and n. More precisely, $M'(m,n) = \min_T\{B(T)\}$ where T ranges over all comparison-trees for merging two lists of sizes m and n. Determine $M'(m,n)$.
(b) Define $S'(n)$ to be the length of the length of a **shortest path** in some comparison-tree for sorting a list of size n, i.e., $S'(n) = \min_T\{B(T)\}$ where T ranges over all comparison-trees for sorting n elements. Determine $S'(n)$.

HINT: (a) and (b) are quite easy, once you understand what the problem is about.

Solution:
(a)$M'(m,n) = 1$. This question assumed that the two lists are already sorted. Then suppose our algorithm compares the largest element of one list to the smallest of other and if the former is smaller than the latter then our algorithm terminates; otherwise, proceeds to carry the merging, say using the standard algorithm. The comparison tree of this algorithm has a minimum hight of one, which is the least we can obtain. This shows that $M'(m,n) \leq 1$, but we know that we have to make one comparison whatsoever may be the case, thus $M'(m,n) = 1$.
(b) $S'(n) = n-1$. Again, suppose that our algorithm verifies if the input is already sorted. More precisely, suppose the input elements are $x_1, x_2, \ldots, x_n$ and the algorithm begins by checking if $x_1 < x_2$, $x_2 < x_3, \ldots x_{n-1} < x_n$, which are $n-1$ comparisons; if each of these hold then the algorithm terminates, otherwise proceeds to sort the input using some sorting algorithm. The minimum length of the comparison tree corresponding to this algorithm is $n-1$ which happens when we are fortunate enough to have the input of the form $x_1 < x_2 < \ldots < x_n$. This gives us that $S'(n) \leq n-1$. However, note that any sorting algorithm needs to determine the relative order of the input elements for which it has to make these, or an equivalent set (i.e, those which will also lead us to the same conclusion about the relative order of the inputs) of, $n-1$ comparisons. Thus $S'(n) = n-1$.
Comments: In this problem many students did not realize that the problem asked to minimize the functions $M'(m,n)$ and $S'(n)$ over all the possible algorithms for merging two lists and sorting a list, respectively. Once this is realized one can guess, and that is what we did, the best case and the algorithm first verifies the guess and terminates if the input satisfied the guess and otherwise proceeds to solve the respective problem. Three-four points were given for each part if the answer was close but not correct.

3. (15 Points)

Use the Rote Method to solve the following recurrence:

$$T(n) = n \lg n + 4T(n/2).$$

Be sure to indicate each of the EGVS steps. You can choose your own initial conditions (the "strong form"). NOTE: $\lg n$ means $\log_2 n$.

Solution

$$\begin{array}{rcll} T(n) & = & 4T(n/2) + n \lg n & \\ & = & 4(4T(n/4) + n/2 \lg(n/2)) + n \lg n & \text{(expanding one step)} \\ & = & 4^2 T(n/4) + n \lg n + 2n \lg(n/2) & \\ & = & 4^2(4T(n/8) + n/4 \lg(n/4)) + n \lg n + 2n \lg(n/2) & \text{(expanding one more step)} \\ & = & 4^3 T(n/2^3) + n \lg n + 2n \lg(n/2) + 4n \lg(n/4) & \\ & = & 4^{i+1} T(n/2^{i+1}) + \sum_{j=0}^{i} 2^j n \lg(n/2^j), & \end{array}$$

this is our guess for the $i^t h$ step. We verify it by expanding the recurrence once more:

$$\begin{array}{rl} = & 4^{i+1}(4T(n/2^{i+2}) + \sum_{j=0}^{i} 2^j n \lg(n/2^j) + n/2^{i+1} \lg(n/2^{i+1}) \\ = & 4^{i+2} T(n/2^{i+2}) + \sum_{j=0}^{i+1} 2^j n \lg(n/2^j). \end{array}$$

Hence we have verified the guess. We now use properties of geometric progression to solve the recurrence. Also, for stopping criterion we choose $i = \lfloor \lg n \rfloor$; thus $\frac{1}{2} \le \frac{n}{2^{i+1}} < 1$. We now choose us the boundary condition to be $T(n) = 0$, for $n < 1$. Using this we get:

$$\begin{array}{rcl} T(n) & = & \sum_{j=0}^{\lfloor \lg n \rfloor} 2^j n \lg(n/2^j) \\ & = & n \left[\sum_{j=0}^{\lfloor \lg n \rfloor} 2^j \lg n \right] - n \left[\sum_{j=0}^{\lfloor \lg n \rfloor} 2^j j \right] \\ & = & n \left[\lg n 2^{\lfloor \lg n \rfloor + 1} \right] - n \left[\lg n 2^{\lfloor \lg n \rfloor + 1} + 2^{\lfloor \lg n \rfloor + 1} + 2 \right] \\ & = & n 2^{\lfloor \lg n \rfloor + 1} + 2n \\ & = & \Theta(n^2). \end{array}$$

In the above argument we used the following summation:

$$\sum_{j=0}^{k} j x^j = \frac{x((k+1)(x-1)x^{k+1} - (x^{k+1} - 1))}{(x-1)^2}$$

with x replaced by 2.
Comments: In this problem many students were successful in carrying out the expansion and the guess. However, to verify the guess one has to expand the recursive term once more in the guess, i.e, certain term of the form $T(g(n))$ which appears on the right hand side of the guess. Once that is done the result should have the same form as the guess except with the index , in this case i, replaced by $i + 1$. For this problem one also needed some knowledge of geometric series and those derived from it, since the exact sum gives the cancellation of $n^2 \lg n$ term leading the

summation, otherwise if one were to simply proceed by $\mathcal{O}$ notation the answer will seem to be $\Theta(n^2 \lg n)$, which we now know to be wrong. One would get most points if they were to carry successfully the EGV steps, for the stopping criterion one point was due, and I took off a point or two for being unable to get the $\Theta(n^2)$ form. In case the guess was wrong then roughly $4-5$ points could be lost, since then the rest of the argument fails, the same applies for the case of expansion.

4. (20 Points)
Consider the real recurrences:

$$T_0(n) = n + 2T_0(n/2)$$

and

$$T_1(n) = n + 2T_1(\lfloor n/2 \rfloor + 2).$$

The initial conditions are given by $T_0(n) = T_1(n) = 0$ for $n \leq 4$.
(a) Use real induction to show that $T_0(n) \leq T_1(n)$. HINT: Use the fact that $T_1(n)$ is non-decreasing in n.
(b) Use real induction to show that for all $0 \leq k < k_0$, $T_0(n+k) = T_0(n) + O(k_0 \lg n)$.
(c) Show that $T_1(n) = O(T_0(n))$.

Solution:
(a)

$$\begin{array}{rcll} T_0(n) & = & n + 2T_0(n/2) & \text{(recurrence for } T_0(n)) \\ & \leq & n + 2T_1(n/2) & \text{(by induction hypothesis)} \\ & \leq & n + 2T_1(\lfloor n/2 \rfloor + 2) & \text{(since } T_1(n) \text{ is non-decreasing in } n) \\ & = & T_1(n) & \text{(recurrence for } T_1(n)) \end{array}$$

(b) Assume that inductively, there is a $C > 0$ such that for all $0 \leq k$, $T_0(n+k) \leq T_0(n) + Ck \lg n$. Then,

$$\begin{array}{rcll} T_0(n+k) & = & (n+k) + 2T_0(n/2 + k/2) & \text{(by recurrence for } T_0) \\ & \leq & (n+k) + 2T_0(n/2) + Ck\lg(n/2) & \text{(by induction)} \\ & = & (n+k) + 2T_0(n/2) + Ck\lg n - Ck & \\ & \leq & n + 2T_0(n/2) + Ck\lg n & \text{(for } C \geq 1) \\ & = & T_0(n) + Ck\lg n & \text{(recurrence for } T_0) \end{array}$$

COMMENT: Note that this proof has shown the bound for any k, not just those less than some fixed k_0. But in the next part (c), we only need this result for $k \leq k_0 = 2$.

(c) Assume that inductively, $T_1(n) \leq C'T_0(n)$ for some $n \geq n_0$.

$$
\begin{aligned}
T_1(n) &= n + 2T_1(\lfloor n/2 \rfloor + 2) && \text{(by recurrence for } T_1\text{)} \\
&\leq n + 2C'T_0(\lfloor n/2 \rfloor + 2) && \text{(by induction hypothesis)} \\
&\leq n + 2C'T_0(n/2 + 2) && \text{(since } T_0 \text{ is increasing)} \\
&\leq n + 2C'(T_0(n/2) + 2C\lg(n/2)) && \text{(by recurrence for } T_0\text{)} \\
&= 2C'T_0(n/2) + n + 4CC'\lg(n/2) \\
&= 2C'T_0(n/2) + n + 4C'\lg(n/2) && \text{(since we can choose } C = 1 \text{ from (b) above)} \\
&\leq C'(2T_0(n/2) + n) && \text{(if } C'n \geq n + 4C'\lg(n/2)\text{)} \\
&= C'T_0(n).
\end{aligned}
$$

We let $C' = 2$ then we want $n \geq 8\lg(n/2)$ which holds for all values of $n \geq 32$.
Comments: A general remark regarding operations with $\mathcal{O}$ in respect to the inductive step: One has to make sure that the constant hidden by $\mathcal{O}$ should be the same at the end of the induction, as is done in the above argument, because if it were increasing as the argument proceeds then we cannot bound the recurrence by $\mathcal{O}$. This was the reason why the hint was wrong, but I elucidate in more detail. Suppose we were trying to show that for some constant $k \geq 0, T_1(n) = T_0(n+k)$, as suggested by the hint. Then we have,

$$
\begin{aligned}
T_1(n) &= 2T_1(\lfloor n/2 \rfloor + 2) + n && \text{(from recurrence for } T_1\text{)} \\
&= 2T_0(\lfloor n/2 \rfloor + 2 + k) + n && \text{(by induction hypothesis)} \\
&= 2\left[T_0(\lfloor n/2 \rfloor) + C(k+2)\lg(n/2)\right] + n && \text{(using part (b))} \\
&= 2T_0(\lfloor n/2 \rfloor) + 2C(k+2)\lg(n/2) + n \\
&= 2T_0(\lfloor n/2 \rfloor) + n + 2C(k+2)\lg(n/2) \\
&\leq 2T_0(n/2) + n + 2C(k+2)\lg(n/2) && \text{(since } T_0 \text{ is increasing)} \\
&= T_0(n) + 2C(k+2)\lg(n/2) \\
&= T_0(n) + Ck\lg(n) && \text{(if } Ck\lg n = 2C(k+2)\lg(n/2)\text{).}
\end{aligned}
$$

Canceling out C on both sides we want $k\lg n = 2(k+2)\lg(n/2)$, for all n, but this does not hold since the right hand side is increasing 2 times faster than the left. This subtle point will be eluded from you if you were to carry a straight forward argument using $\mathcal{O}$. The distribution of points was 7, 7, 5, for (a), (b), (c) respectively. One point was kept for those who could successfully get the δ involved in the real induction. This was assuming that the argument avoided the pitfalls mentioned above, in which case one or two points could be lost.