CSE 373 Homework 1 Solution

April 26, 1998

1. (Weiss, 2.2) :

   (a) $T_1(N) + T_2(N) = O(f(N))$: True

   (b) $T_1(N) - T_2(N) = o(f(N))$: False

   (c) $T_1(N)/T_2(N) = O(1)$: False

   (d) $T_1(N) = O(T_(N))$: False

2. The table below shows the value of variables *ThisSum and* MaxSum at the end of each iteration:

Sequence		3	-2	4	5	-11	8	3	-4	5	-14	5
ThisSum	0	3	1	5	10	0	8	11	7	12	0	5
MaxSum	0	3	3	5	10	10	10	11	11	12	12	12

   The maximum sum of the sequence is 12.

3. The pseudocode for mergesort is given below:

```
list Merge (list l1, list l2)
 /* Merges two sorted lists into a sorted list and returns it
  * This function destroys l1 and l2  */
{
     if (l1 == NULL)
        RETURN l2;
     if (l2 == NULL)
        RETURN l1;

      /* The first node of the merged list is the smaller
       * of the value of  first node of l1 and l2.
       */
     if (l1.val < l2.val) {
        l1.next = Merge (l1.next, l2);
        RETURN l1;
```

```
    }
    else {
       l2.next = Merge (l1, l2.next);
       RETURN l2;
    }
}

list MergeSort (list l)
/* Sorts l in increasing order using mergesort */
{
    if (l == NULL)
       RETURN l;

     duo d = split (l);

     RETURN ( Merge (MergeSort(d.first), MergeSort(d.second)) );
 }
```

4. Let n be the number of nodes in the set.

   (a) Printing the element in a set requires traversing the linked list which can be done in $O(n)$ time. The running time is the same for both singly linked and doubly linked list representations.

   (b) Union of two sets: If we assume that the two sets don't have any elements in common, we can compute the union set in constant time - just by linking the last node in set A to the first node in B. Finding the last node is easy for a doubly linked list (last node is first.prev). For a singly linked list, the naive approach might take $O(n)$ time to get to the last node. However, we can maintain a pointer to the last node in a list as an additional field in the set data structure, without additional cost in set operations. In that case, for both the representations, union takes $O(1)$ time.

   However, if the two sets may have elements in common, we need to check for each element in set B if it is present in set A. If it is not present, we insert it in the union. Checking the presence of an element in set A may take $O(n_1)$ time. So overall running time is $O(n_1 * n_2)$, where $n_1 = |A|$, and $n_2 = |B|$.

   The above applies to both the list representations.