Homework #1 Solutions

A.

Most of you got the basic idea right on this problem. A few things to watch out for:

1. n, the problem size, should be an integer, use the floor function where appropriate to insure this.

2. 1 micro-second is 10^{-9} seconds (not 10^{-6})

Here is a sample entry in the table...

	1s	1m	1h	1d	1m	1y	1c
n!	9	11	12	13	15	16	17

B.

a) (3,4) (1,5) (2,5) (3,5) (4,5)

b) The reverse-sorted array <n, n-1,...,2,1> has (n-1)+(n-2)+...+2+1 = n(n+1)/2 inversions.

c) There is a direct relationship between the running time of insertion sort and the number of inversions in the input array. Consider the insertion sort procedure on p.3 of [CLR]. Each element A[j] 2 <= j <= length(A) is inserted into the already sorted list A[1..j-1]. The while loop of line 5 is executed once for every A[i], i < j where A[i] > A[j], but this is the definition of an inversion. Furthermore, since A[1..j-1] is sorted the body of the while loop, lines 6-7, exactly once for each inversion.

d) Use merge_sort(A,p,r) of p. 13 [CLR]. Modify the procedure merge(A,p,q,r) by adding a global counter and the line:

counter += q - left_index+1

at the point where an element from the right sub-list is about to be added to the sorted list.

C. (in decreasing order and w/o explanation) a)

$2^{2^{\wedge}(n+1)}$

$2^{2^{\wedge}n}$

$(n+1)!$

$n!$

e^{n}

$n2^{n}$

2^{n}

$(3/2)^{n}$

$(\lg n)^{\lg n}$ == $n^{\lg \lg n}$

$(\lg n)!$

n^{3}

n^{2} == $\lg (n!)$

$(\sqrt{2})^{\lg n}$

$2^{\sqrt{(2 \lg n)}}$

$\lg^{2} n$

$\ln n$

$\sqrt{\lg n}$

$\ln \ln n$

$2^{\lg^{*} n}$

$\lg^{*}(\lg n)$ == $\lg^{*} n$

$\lg(\lg^{*} n)$

$n^{(1/\lg n)}$ == 1

b) $f(n)(1+\sin(n))$ w/ $f(n) = \Omega(2^{2^{\wedge}(n+1)})$, $2^{2^{\wedge}(n+2)}$ for example.

D.

I. Sort all x_i's and re-number so that $x_1 <= x_2 <= \ldots <= x_n$

II.

```
max = -∞; old_x = -∞; maximal = 0;
for (i=n; i>=1; i--)
      count = 0;
      if (x_i != old_x && maximal != 0)
            output (maximal);
            maximal = 0;
      else
            old_x = x_i;
            if (y_i > max)
```

```
                        maximal = i;
    if (maximal != 0)
        output (maximal);
```

E.

a) $\theta(n \lg n)$

For $n=3^k$ shown in [CLR]. Next show...

1. T(n) is non-decreasing by induction (as in handout)
   So, we know: $T(3^{floor(\lg_3 n)}) <= T(n) <= T(3^{ceil(\log_3 n)})$

2. Now prove: $T(3^{floor(\lg_3 n)}) = O(n \lg n)$
   $T(3^{ceil(\log_3 n)}) = \Omega(n \lg n)$

b) $\theta(n)$ by case 1 of the Master Theorem

c) $\theta(n)$

Prove by induction:

1. $T(n) <= c(n+\sqrt{n})$
2. $T(n) >= (n+\sqrt{n})$

WARNING: Many of you tried to use an induction argument that went like this:

Assume T(n) = cn for n' < n then...

$T(n) = \max \{T(k) + T(n-k) + \sqrt{k} \mid 1 <= k <= n/2\}$
$T(n) = \max \{ck + c(n-k) + \sqrt{k} \mid 1 <= k <= n/2\}$
$T(n) = \max \{cn + \sqrt{k} \mid 1 <= k <= n/2\}$
$T(n) = cn + \sqrt{(n/2)} = \theta(n)$

This argument does NOT work. When using induction you must prove the induction hypothesis exactly. In this case T(n) = cn.

d) $\theta(n^{\log_3 5})$ by case 1 of the Master Theorem

e) $\theta(n^{\log_3 5})$ by case 1 of the Master Theorem

F.

Start with the lsb. Step through A[], examining bit k and separating the numbers into 2 auxiliary arrays depending on weather bit k is a 1 or 0. If, at the end of this separation, the number of elements in the 0 array is less than or equal to the number of elements in the 1 array then we throw out the 1 array and move on to the next bit and repeat. Otherwise if the number of elements in the 0 array is greater than the number of elements in the 1 array we do the same thing except this time we throw out the 0 array.

$T(n) = T(n/2) + n = \theta(n)$

G.

a) Consider the case when ALL the chips are bad. Then any subset of the bad chips can conspire to pretend that they are good by calling each other good and everyone else bad.

b) Do floor(n/2) pairwise tests. Keep only one chip from each pair that returns GG. If there is an unpaired chip keep it only if the number of GG pairs is even.

c) Repeatedly apply b) to find one good chip. The do n-1 pairwise tests to find all other good chips.

$T(n) = 0$ if $n = 1$
$T(n) = \max\{T(k) + \text{floor}(n/2) \mid 1 <= k <= n/2\}$

I graded pretty loosely on this one.

H.

24-2.1

Assume that you want to get MST T from Kruskals Algorithm. Sort the edges in such a way that the edges in T come before any other edges of equal weight. Prove by contradiction that Kruskals Algorithm will actually produce T.

24-2.4

Use bucket sort to sort in linear time. This does not improve the overall running time of $O(V + E\lg^* E)$.