1 Homework 3

Name: Kangdong Yuan

1.1 probelm1

a).I did not work in a group.
b).I did not consult without anyone my group members
c).I did not consult any non-class materials.

1.2 problem2

Master's theorem, we can split the recurrence into $T(n) = aT(\frac{n}{b}) + (n^k * log^p(n))$
a). $T(n) = 11T(\frac{n}{5}) + 13n^{1.3},\ a = 11,\ b = 5,\ k = 1.3,\ p = 0$
$log_5(11) = 1.489 > 1.3 = k$ Then the running time is $T(n) = \Theta(n^{log_5(11)})$

b). $T(n) = 6T(n/2) + n^{2.8},\ a = 6,\ b = 2,\ k = 2.8,\ p = 0$
$log_2(6) = 2.58498 < 2.8 = k$ Then the running time is $T(n) = \Theta(n^{log_2(6)})$

c). $T(n) = 5T(n/3) + log^2(n),\ a = 5,\ b = 3,\ k = 0,\ p = 2$
$log_3(5) = 1.464 > 0 = k$ Then the running time is $n^{log_3(5)} = \Theta(n^{1.464})$

d).$T(n) = T(n-2) + log(n)$, we can unfold it $T(n) = T(n-4) + log(n-2) + log(n) = T(n-6) + log(n-4) + log(n-2) + log(n)$
Finally, if n is a even number $T(n) = T(0) + \sum_{k=1}^{\frac{n}{2}} log(2k)$
if n is a odd number, $T(n) = T(1) + \sum_{k=1}^{\frac{n}{2}} log(2k+1)$

The running time of
$Min(T(1)+\sum_{k=1}^{\frac{n}{2}} log(2k+1), T(0)+\sum_{k=1}^{\frac{n}{2}} log(2k)) \leq T(n) \leq Max(T(1)+\sum_{k=1}^{\frac{n}{2}} log(2k+1), T(0) + \sum_{k=1}^{\frac{n}{2}} log(2k))$
$T(n) = O(n * log(n))$ and $T(n) = \Omega(n * log(n))$, So, for all n, the running time of $T(n)$ is $\Theta(n * log(n))$

1.3 problem3

We assume the n is the power of 2. Find the $A[i] = i$, first we know it is a sorted array, so we can use this feature to find when is the $A[i] = i$.
By using the divide and conquer, first divide this array by 2, and find the mid item. Because there are $n+1$ items in this array (the start index of this

array is 0), the mid item in this array is $A[\frac{n}{2}]$ then check whether $A[\frac{n}{2}] = \frac{n}{2}$, if the conditions are true, we return the true.
If the conditions are not true,
if $A[\frac{n}{2}] > \frac{n}{2}$ we need do the same search in lower interval which from 0 to $\frac{n}{2}$.
But if $A[\frac{n}{2}] < \frac{n}{2}$ we need do the same search in upper interval which from $\frac{n}{2}$ to n.
We do this recurrence search process until we find the $A[i] = i$, or the search interval becomes 1.
The recurrence function is $T(n) = T(\frac{n}{2} + 1)$, then we solve the running time by Master's theorem. $a = 1,\ b = 2,\ k = 0,\ p = 0,\ log_2(1) = 0 = k, p > -1$ so we can know that running time is $O(n^k * log^{p+1}(n)) = O(log(n))$

1.4 problem4

For each array, there are n items in this array (the index of this array start from 1), and there are m different values in this array.
The sorted algorithm for this list is go through all the items in the array, we create m different new array to to store the values for each items. For example, the a new array store only value 5, then we go through all the items in array, we put all items = 5 into this array. Then, after put all items into new arrays, we compare the value in each sub-array by insertion sort. Finally, after sort each sub-array, we combine all sub-array with wanted order.
We analysis the running time of this algorithm, when we go through all the items in array it take O(n), when we sort M sub-array by insertion sort, the average running time is O(M) (if the value in array is uniformly distributed). The total running time is O(n+M).
For small M, the running time is also O(n+m), the lower bound is cannot be the $\Omega(nlog(n))$