1 Problem 1 (10 points: each subproblem is 2 points)

For each statement below state if it is true or false.

1. If $f(n) = \Theta(g(n))$, then $g(n) = \Theta(f(n))$.

   true

2. If$f(n) = O(g(n))$ and $f(n) = \Omega(h(n))$, then $g(n) = \Theta(h(n))$.

   false

3. $2\log^2(n) = O(\log(n) + 1)$

   false

4. $\log(n) + \sqrt{n} = o(n)$

   true

5. $3n^2 + 3n + 5\sqrt{n} = \Theta(n^2)$

   true

2 Problem 2 (20 points; each subproblem is 10 points)

Give asymptotic upper bounds for the following recurrences. You can use Master theorem, when it is applicable. Assume that $T(0) = T(1) = 1$.

1. $T(n) = T(n/2) + n^2 + n$
   Answer:
   $a = 1, b = 2, log_b a = 0 < 2$. So by case 3 of master theorem, $T(n) = \Theta(n^2)$.

2. $T(n) = T(n/2) + T(n/4) + n^2$
   Answer:
   Define $A(n) = 2A(n/2) + n^2$. Clearly, $T(n) \leq S(n)$. By master theorem case 3, $S(n) = \Theta(n^2)$. Therefore, $T(n) = O(n^2)$. Since $T(n) = T(n/2) + T(n/4) + n^2 \geq n^2$, $T(n) = \Omega(n^2)$. Finally, $T(n) = \Theta(n^2)$.

3 Problem 3 (60 points)

Given an array A of size n, design an expected linear time algorithm to sort A. Each element $A[i]$ is either some element from $\{n, n+1, n+2, \cdots, n^{100}\}$ or a real number chosen uniformly at random from $(0, 1]$.

20 points will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem by using the time worse than expected linear time.(3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

The full credit will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem in expected linear time, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

Answer: Assume without loss of generality that the buckets are being sorted in increasing order.

Step 1:

Use one pass to put numbers into 2 buckets:

Bucket B_1 contains all numbers x in A such that
$x \in \{n, n+1, n+2, \cdots, n^{100}\}$.

Bucket B_2 contains all numbers y in A such that $l \in (0, 1]$.

I.e. if the element is less than 1, then put it into B_2. Otherwise, put it into B_1.

Step 2 and 3:

2. Sort bucket B_1 with Radix Sort, using base-n.
   It must use base-n to run in expected linear time, as the expected performance of Radix Sort is dependent on the number of digits. With elements from $n, n+1, n+2, \cdots, n^{100}$, using base-$n$ gives a constant 100 number of digits.
3. Sort bucket B_2 with Bucket Sort.

Step 4:

Merge the two buckets by putting the sorted elements of bucket B_2 before the sorted elements of B_1.

Time Complexity

Step 1 requires a single pass that does a comparison on each element, running in $O(n)$.

Step 2 and 3 employ Radix Sort and Bucket Sort, both of which run in *expected* $O(n)$ time.

Step 4 is a merge which requires no comparisons, simply copying elements in a single pass over each item in $O(n)$.

The steps are sequential, so the entire algorithm will run in expected $O(n)$ time.

Correctness

Buckets B_1 and B_2 will be sorted by the correctness of Radix Sort and Bucket Sort.

Note that $\forall(x, y)$, where $x \in \{n, n+1, n+2, \cdots, n^{100}\}$ and $y \in (0, 1]$, $y < x$. All elements in B_2 should therefore come before all elements of B_1. The elements of each bucket are sorted relative to each other. Thus, the merged buckets are sorted.

4 Problem 4 (60 points)

You are given n intervals $l_i = [a_i, b_i]$ on the real line, where a_i, b_i are real numbers and $a_i \leq b_i$ and $1 \leq i \leq n$. Give an algorithm that computes the measure of this set of intervals, that is, the length of $\cup_{i=1}^{n} l_i$ in $O(n \log n)$ time.

For example, there is a set of intervals $\{[1, 3], [2, 4.5], [6, 9], [7, 8]\}$, the measure of this set of intervals is 6.5.

20 points will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem by using the time worse than $O(n \log n)$.(3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

The full credit will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem in $O(n \log n)$ time, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

Answer:

Let's first sort the intervals by a_i in ascending order. The idea is trying to merge the intervals that have overlaps with other intervals. Starting from $[a_1, b_1]$, check $[a_2, b_2]$: if $a_2 \geq b_1$, do nothing; if $a_2 < b_1$ and $b_2 \leq b_1$, delete interval $[a_2, b_2]$; if $a_2 < b_1$ and $b2 > b1$, change interval $[a_1, b_1]$ to $[a_1, b_2]$. Then continue check $[a_3, b_3]$. After scanning all the intervals, the new set of intervals don't have overlap and just sum all the lengths of the intervals.

For the sorting part, if using merge sort, it takes $O(n \log n)$. For the 'merge' part, just one pass over the set of intervals, which needs $O(n)$ time. Finally, sum all the lengths of the intervals needs another one pass taking $O(n)$ time. So the total time complexity is $O(n \log n)$.

Correctness

1. When two intervals $[a_i, b_i], [a_j, b_j]$ are disjoint, the length of the union of these two intervals is just sum them all.

2. When two intervals $[a_i, b_i], [a_j, b_j]$ have overlap, the algorithm merge the two intervals to a new interval $[a_i, b_j]$ and the length of the union of these two intervals is just the length of $[a_i, b_j]$.

3. When interval $[aj, b_j]$ is included in $[a_i, b_i]$, the the length of the union of two intervals is just the length of $[a_i, b_i]$.

After recursively running the algorithm, all the intervals on the real line don't have overlap and the summation of them is the length of the union.