Maoting Ren
mren

2-1. Practice Solving Recurrences. Please do problem 13 in the draft course textbook (chapter 2).

(a). $p = \log_b a = \log_3 6 < \alpha = 2$, so $T(n) = \Theta(n^2)$
(b). $p = \log_b a = \log_4 5 > \alpha = 1$, so $T(n) = \Theta(n^{\log_4 5}) \approx \Theta(n^{1.16})$
(c). $p = \log_b a = \log_2 3 > \alpha = 3/2$, so $T(n) = \Theta(n^{\log_2 3}) \approx \Theta(n^{1.58})$
(d). $p = \log_b a = \log_2 4 = \alpha = 2$, so $T(n) = \Theta(n^2 \log n)$
(e). $1/6^p + 1/2^p + 1/3^p = 1$, so $p = 1$, and $p = \alpha = 1$. so $T(n) = \Theta(n \log^3 n)$
(f). we can simplify the recurrence to: $T(n) = 4T(n/3) + 5T(n/2) + 2T(2n/3) + n^{62} \log^{37} n$
$4/3^p + 5/2^p + 2/(3/2)^p = 1$.
assume $f(p) = 4/3^p + 5/2^p + 2/(3/2)^p$, as p increase the result will decrease. As we can see $f(62) < 1$, so we know $p < 62 = \alpha$, hence $T(n) = \Theta(n^{62} \log^{37} n)$
(g) $T(n) = \log \log n$.
because $m = \log_2 n$, so $n = 2^m$, $T(2^m) = T(2^{m/2}) + 1$
assume $G(m) = T(2^m)$, so we can know $G(m) = G(m/2) + 1 = \Theta(\log m) = T(2^m)$, instead m to n, we will get $T(n) = \log \log n$.

2-2. Sorting Fractions. Please do problem 48 (chapter 3).

solution
we can convert this problem back to sort integers, and this means we should multiply an integer to convert the fractions to integers. But we should make sure that even the smallest difference between two fractions should be big or equal than 1 after converted to integers. Because a_i and b_i are in the range $[1, n^c]$, so $1/(n^c-1) - 1/n^c$ is the smallest difference. Therefore, we can multiply $(n^c-1)*n^c$ to convert all fractions to integers, then use radix sort to sort them.

2-3. In-Place Merge Sort. Please do problem 44 (chapter 3).

solution
Each time we sort half of the array, and the rest half as scratch space.
The specific approach is:
1. sort the second half array, and using the first half array as scratch space.
2. sort the first quarter array, and using the second quarter array as scratch space.
3. merge the first quarter and the second half array by the second quarter as scratch space. This makes the unsorted elements move to the first quarter.

Then sort the first half unsorted array, and use the rest as scratch space. Then merge the newly sorted element into sorted array.

And by doing this recursively, we can done the in-place merge sort.
Time complexity is O(n log n). Because $T(n) = T(n/2) + T(n/4) + O(n)$
so $T(n) = O(n \log n)$.

2-4. Counting Distant Pairs. Please do problem 51(k) (chapter 3). For simplicity, you only need to worry about the special case where a = b = n/2 to obtain full credit (although you should feel welcome to try the problem in its full generality if you would like a challenge).

solution

We are going to calculate how many pairs satisfy this condition: $|x_i - x_j| \geq a$ and $|y_i - y_j| \geq b$.
It maybe a little tricky to compute it directly, but we can divide all pairs into four parts:

1. $|x_i - x_j| \geq a$ && $|y_i - y_j| \geq b$
2. $|x_i - x_j| \geq a$ && $|y_i - y_j| < b$
3. $|x_i - x_j| < a$ && $|y_i - y_j| \geq b$
4. $|x_i - x_j| < a$ && $|y_i - y_j| < b$

As we all know that there are N*(N-1)/2 pairs in total, so if we can compute the sum of part **2, 3, 4**, we will get the answer of this problem.
To make this problem more simple, we can separately compute the sum of pairs in
① $|x_i - x_j| < a$
② $|y_i - y_j| < b$
③ $|x_i - x_j| < a$ && $|y_i - y_j| < b$
The sum of pairs in part **2, 3, 4** is the sum of pairs (① + ② – ③).

First, we compute ③. Sort the points by x coordinate first. Then process the points in this order, and keep a window that all points in this window satisfy $|x_i - x_j| < a$. Use the points which in the window to construct an **order statistic tree**. By using this data structure, when insert a new point, we can know the number of points satisfy $|y_i - y| < b$ in (log n) time cost, y_i is the newly add point and y is the point in tree. Each point will be inserted into the tree once, and removed once. Each time add one point into the tree, we can compute how many pairs newly add. By this way we can get the sum of all pairs satisfy $|x_i - x_j| < a$ && $|y_i - y_j| < b$.

Then the ① and ② will be pretty simple to compute. When add point into the window, we can computer how many pairs satisfy $|x_i - x_j| < a$ newly add, by this way we can get the number of pairs $|x_i - x_j| < a$. We can use the some way to compute ② – pairs satisfy $|y_i - y_j| < b$.

By doing the above things, we have get the number of pairs of **2**, **3**, **4**, then to get the number of pair of **1**, we can subtract N*(N-1) by this three zone. Done!