Written Assignment 1

Haoyu Zhen

March 14, 2022

Problem 1

Firstly, we have

$$T(n) = a^{\log_b n} \times \mathcal{O}(1) + \mathcal{O}(n^d \log^w n) + a \times \mathcal{O}\left(\left(\frac{n}{b}\right)^d \log^w \left(\frac{n}{b}\right)\right) + \cdots + a^m \times \mathcal{O}\left(\left(\frac{n}{b^m}\right)^d \log^w \left(\frac{n}{b^m}\right)\right)$$

where $m = \log_b n$. More compactly,

$$T(n) = \mathcal{O}\big(a^{\log_b n}\big) + \sum_{i=0}^{m} \mathcal{O}\left(\left(\frac{a}{b^d}\right)^i n^d \log^w \left(\frac{n}{b^i}\right)\right) \quad .$$

Case I: If $a = b^d$:

$$T(n) = \mathcal{O}(n^d) + \sum_{i=0}^{m} \mathcal{O}\left(n^d \log^w \left(\frac{n}{b^i}\right)\right) = \sum_{i=0}^{m} \mathcal{O}\big(n^d \log^w n\big) = m \times \mathcal{O}(n^d \log^w n) = \mathcal{O}(n^d \log^{w+1} n).$$

Case II: If $a < b^d$:

$$\begin{aligned}
T(n) &= \mathcal{O}(n^{\log_b a}) + \sum_{i=0}^{m} \mathcal{O}\left(\left(\frac{a}{b^d}\right)^i n^d \log^w n\right) \\
&= \mathcal{O}(n^{\log_b a}) + \frac{1 - \big(a/b^d\big)^m}{1 - a/b^d} \mathcal{O}\big(n^d \log^w n\big). \\
&= \mathcal{O}(n^{\log_b a}) + \mathcal{O}(n^d \log^w n) \\
&= \mathcal{O}(n^d \log^w n)
\end{aligned}$$

Case III: If $a > b^d$: $\forall \varepsilon > 0$

$$\begin{aligned}
T(n) &= \mathcal{O}(n^{\log_b a}) + \sum_{i=0}^{m} \mathcal{O}\left(\left(\frac{a}{b^d}\right)^i n^d \left(\frac{n}{b^i}\right)^\varepsilon\right) \\
&= \mathcal{O}(n^{\log_b a}) + \frac{1 - \big(a/b^{d+\varepsilon}\big)^m}{1 - a/b^{d+\varepsilon}} \mathcal{O}\big(n^{d+\varepsilon}\big). \\
&= \mathcal{O}(n^{\log_b a}) + \big(a/b^{d+\varepsilon}\big)^m \mathcal{O}\big(n^{d+\varepsilon}\big) \\
&= \mathcal{O}(n^{\log_b a + \varepsilon}) \longrightarrow \mathcal{O}\big(n^{\log_b a}\big)
\end{aligned}$$

Finally the generalization of the master theorem holds. Reference: CS315 Richmond University.

Problem 2

Design the function Merge recursivly:

Algorithm 1 Merge sort with one third dividing approach

Input: A sequence $\{a_i\}$ to be sorted where $i \in \{1, 2, \cdots, L\}$.
Output: $\{R_i\}$.

```
if L == 1 then
    R_1 = a_1
    return
end if
if L == 2 then
    R_1 = min(a_1, a_2) and R_2 = max(a_1, a_2)
    return
end if
{b_i} ← {a_1, a_2, ··· a_[n/3]}
{c_i} ← {a_[n/3]+1, a_[n/3]+2, ··· a_L}
{S_i} ← Merge({b_i}, [n/3])
{T_i} ← Merge({c_i}, n − [n/3])
f_1 ← 1, f_2 ← 1 and i ← 1.
while (f_1 ≤ [n/3] ∧ f_2 ≤ n − [n/3]) do
    if f_2 > n − [n/3] or b_{f_1} < c_{f_2} then
        R_i ← b_{f_1}
        f_1 ← f_1 + 1
    else
        R_i ← c_{f_2}
        f_2 ← f_2 + 1
    end if
    i ← i + 1
end while
return
```

The Soundness of ALG.1:
Proof by induction. Trivially, when $n = 1, 2$, satisfied. Suppose that $\text{Merge}(A, n)$ could successfully sort A for any $n \le k$. Then the result R_i of $\text{Merge}(A, k+1)$ satisfies that: $\forall i$, assume $R_i = S_k$, then $R_{i+1} = S_{k+1}$ or $T_k > R_i$ by the "if" condition. Thus R is sorted.

Complexity Analysis of ALG.1:
Thus we have

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \mathcal{O}(n).$$

$\exists B > 0$, $\exists C = \left[\frac{1}{3}\log 3 + \frac{2}{3}\log\left(\frac{3}{2}\right)\right]^{-1} B$:

$$\begin{aligned} T(n) &\le C\frac{n}{3}\log\left(\frac{n}{3}\right) + C\frac{2n}{3}\log\left(\frac{2n}{3}\right) + Bn \\ &\le Cn\log(n) - \left(\frac{1}{3}\log 3 + \frac{2}{3}\log\left(\frac{3}{2}\right)\right)Cn + Bn \\ &\le Cn\log n \end{aligned}$$

which means that $T(n) = \mathcal{O}(n \log n)$.

Problem 3

The solution of (b).

Algorithm 2 Find the second largest integer

Input: A squence $\{A_i\}$ where $i \in \left\{1, 2, \cdots, 2^k\right\}$
Output: R, the second largest integer in $\{A_i\}$

$m \leftarrow 1$, $S \leftarrow \{A_i\}$
while $m \le k$ **do**
  Partition S into subsets $\{S_i^m\}$ with size 2.
  For each S_i^m, find the max integer A_i^m. # Comparation 1
  $S \leftarrow \{A_i^m\}$, $m \leftarrow m + 1$
end while
Finally we have the maximum $M \triangleq A_1^k$ in $\{A_i\}$.
$T \leftarrow \bigcup S_i^m - \{M\}$ where $M \in S_i^m$
Use a single step in Bubble Sort to find the maximum M' in T. # Comparation 2
return M'

The Soundness of ALG.2: the second largest integer M' must be compared with M in "while". If not, $\exists\ x$ such that $M > x > M'$ which leads to a contradiction.

Times to compare: $|T| = k$ and $|\{S_i^m\}| = 2^{k-m}$. Thus

$$\text{NumOfComp} = |T| + \sum_{m=1}^{k} |\{S_i^m\}| = 2^k + k - 1 < n + \log n.$$

Problem 4

(a)

Algorithm 3 count the number of pairs when $d = 1$

Input: A, B
  Sum $\leftarrow 0$
  Sort A: $a_1 < a_2 < \cdots < a_k$
  Sort B: $b_1 < b_2 < \cdots < b_l$
  $f_1 \leftarrow 1$, $f_2 \leftarrow 1$ and $f \leftarrow 0$
  while $f_1 \le k$ and $f_2 \le l$ **do**
    if $f_2 > l$ or $a_{f_1} < b_{f_2}$ **then**
      $f_1 \leftarrow f_1 + 1$
      Sum $\leftarrow$ Sum $+ f$
    else
      $f_2 \leftarrow f_2 + 1$
      $f \leftarrow f + 1$
    end if
  end while
  return Sum

The Soundness of ALG.3: Trivial. We just sort A and B. And sum $= \sum_i \left(\sum_j \mathbf{1}(b_j < a_i)\right)$

Complexity Analysis:

$$T(n) = \mathcal{O}(a \log a) + \mathcal{O}(b \log b) + \mathcal{O}(n) = \mathcal{O}(n \log n).$$

(b) Note: we only sort A, B for the first time.

Algorithm 4 count the number of pairs when $d = 2$

Input: $A, B \in \mathbb{R}^2$
Output: Sum
  if $|A| = 0$ or $|B| = 0$ **then**
    return 0
  end if
  Sum $\leftarrow 0$
  if A, B are not sorted **then**
    Sort A and B for each dimension (x and y) # $\mathcal{O}(n \log n)$
  end if
  Find the median m of $A \cup B$ for x-axis # $\mathcal{O}(n)$
  Patition A into A_1 and A where $A_1 \cup A_2 = A$ and $\forall a \in A_1, a' \in A_2$, $a_x \le m < a'_x$
  Similarly, partition B into B_1 and B_2
  # Do recursively and reduce the dimensionality
  Sum1 $\leftarrow$ Count(A_1, B_1, dimension $= 2$) # $S(n/2)$
  Sum2 $\leftarrow$ Count(A_2, B_2, dimension $= 2$) # $S(n/2)$
  Sum3 $\leftarrow$ Count(A_2, B_1, dimension $= 1$, axis $= y$) # $\mathcal{O}(n)$ (We have already sorted A_y and B_y.)
  return Sum1 + Sum2 + Sum3

The Soundness of ALG.4: same as **The Soundness** of ALG.5.

Complexity Analysis: Thus we have $T(n) = S(n) + \mathcal{O}(n \log n)$ where

$$S(n) = S\left(\frac{n}{2}\right) + \mathcal{O}(n) \Longrightarrow S(n) = \mathcal{O}(n \log n).$$

Finally, $T(n) = \mathcal{O}(n \log n)$.

(c)

Algorithm 5 count the number of pairs when dimension $= d$

Input: $A, B \in \mathbb{R}^2$
Output: Sum
  if $|A| = 0$ or $|B| = 0$ **then**
    return 0
  end if
  Sum $\leftarrow 0$
  if A, B are not sorted **then**
    Sort A and B for each dimension # $\mathcal{O}(dn \log n)$
  end if
  Find the median m of $A \cup B$ for the first axis # $\mathcal{O}(n)$
  Patition A into A_1 and A where $A_1 \cup A_2 = A$ and $\forall a \in A_1, a' \in A_2$, $a_x \le m < a'_x$
  Similarly, partition B into B_1 and B_2
  # Do recursively and reduce the dimensionality
  Sum1 $\leftarrow$ Count(A_1, B_1, dimension $= d$, axis $= \{1, 2, \cdots, d\}$) # $S(n/2, d)$
  Sum2 $\leftarrow$ Count(A_2, B_2, dimension $= d$, axis $= \{1, 2, \cdots, d\}$) # $S(n/2, d)$
  Sum3 $\leftarrow$ Count(A_2, B_1, dimension $= d - 1$, axis $= \{2, 3, \cdots, d\}$) # $S(n, d-1)$ (We have already sorted A and B.)
  return Sum1 + Sum2 + Sum3

The Soundness of ALG.5: Obviously, $\forall a \in A_1$, $b \in B_2$, (a, b) is not the pair we want. Then $\forall a \in A_2$, $b \in B_1$, $a_1 > b_1$. Then we only need to count A_2, B_1 for dimension $d-1$.

Complexity Analysis:
Now we have $T(n,d) = \mathcal{O}(dn\log(n)) + S(n,d)$ where

$$S(n,d) = 2S\Big(\frac{n}{2}, d\Big) + S(n, d-1) + \mathcal{O}(n).$$

Proposition 1.

$$S(n,d) = \mathcal{O}(n\log^{d-1} n).$$

Proof. When $d = 2$, satisfied. Now we suppose that when $d \le k$, $S(n,d) = \mathcal{O}(n\log^{d-1} n)$.
Thus we have

$$S(n,k+1) = 2S\Big(\frac{n}{2}, k+1\Big) + \mathcal{O}(n\log^{k} n).$$

By the general master Thm. in problem 1: $S(n,k+1) = \mathcal{O}(n\log^{k+1} n)$. □

Ultimately,

$$T(n) = \mathcal{O}\Big(n\log^{d-1} n + nd\log(n)\Big).$$

Problem 5

About a day. Difficulty 3. No collaborators.