Algorithm Design and Analysis (Fall 2023)

Assignment 1

Deadline: Nov 1, 2023

Yuqing Tu 522030910152

1. (25 points) Prove the following generalization of the master theorem. Given constants $a \geq 1, b > 1, d \geq 0$, and $w \geq 0$, if $T(n) = 1$ for $n < b$ and $T(n) = aT(n/b) + n^d \log^w n$, we have

$$T(n) = \begin{cases} O(n^d \log^w n) & \text{if } a < b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \\ O(n^d \log^{w+1} n) & \text{if } a = b^d \end{cases}.$$

Proof:

The running time of solving all size-1 problem is

$$a^{\log_b n} O(1) = O(n^{\log_b a})$$

The total running time for combinig is

$$\begin{aligned} &c(n^d)(\log^w n) + ac(\frac{n}{b})^d(\log^w(\frac{n}{b})) + \ldots + a^k c(\frac{n}{b^k})^d(\log^w(\frac{n}{b^k})) + \ldots + a^{\log_b n} c(\log^w(\frac{n}{b^{\log_b n}})) \\ &< cn^d \log^w n(1 + (\frac{a}{b^d}) + \ldots + (\frac{a}{b^d})^k + \ldots + (\frac{a}{b^d})^{\log_b n}) \end{aligned}$$

(a) $a < b^d$:

$$T(n) = O(n^d \log^w n)(1 + (\frac{a}{b^d}) + \ldots + (\frac{a}{b^d})^k + \ldots + (\frac{a}{b^d})^{\log_b n})$$

Since $a < b^d$, then $\frac{a}{b^d} < 1$

$$T(n) = O(n^d \log^w n)$$

(b) $a > b^d$:

$\because a > b^d$

$\therefore \log_b a > d$

$\therefore \exists \varepsilon > 0,\ s.t.\ n^d \log^w n \leq n^{\log_b a - \varepsilon}$

$\therefore n^d \log^w n = O(n^{\log_b a - \varepsilon})$

$\therefore$ According to https://blog.csdn.net/qq_41739364/article/details/101224786

$$T(n) = O(n^{\log_b a})$$

(c) $a = b^d$:

$$T(n) = O(n^d \log^w n)(1 + (\frac{a}{b^d}) + \ldots + (\frac{a}{b^d})^k + \ldots + (\frac{a}{b^d})^{\log_b n})$$

Since $a = b^d$, then $\frac{a}{b^d} = 1$

$$T(n) = O(n^d \log^{w+1} n)$$

2. (25 points) Recall the median-of-the-medians algorithm we learned in the lecture. It groups the numbers by 5. What happens if we group them by $3, 7, 9, \ldots$? Please analyze those different choices and discuss which one is the best.

Solution:

If we group them by k, we should find $\frac{n}{k}$ medians and find the median v of them. Then we can draw a rectangle with length $\frac{n}{k}$ and width k. In this rectangle, there is a smaller rectangle with length $\frac{n}{2k}$ and width $\frac{k+1}{2}$, which contains all numbers greater or less than v. So at least $\frac{k+1}{4k}n$ numbers are definitely greater or less than v. Each recursion can eliminate $\frac{n}{3}$ numbers for $k = 3$, can eliminate $\frac{2n}{7}$ numbers for $k = 7$, can eliminate $\frac{5n}{18}$ numbers for $k = 9$.

When k increases, the fewer numbers we can eliminate in each recursion and it leads to more recursive calls, but we spend less time to find the median-of-the-medians.

To sum up, we should choose a suitable k according to n, so that the total time between recursive calls and finding the median-of-the-medians is minimized.

3. (25 points) Let X and Y be two sets of integers. Write $X \succ Y$ if $x \geq y$ for all $x \in X$ and $y \in Y$. Given a set of m integers, design an $O(m \log(m/n))$ time algorithm that partition these m integers to k groups $X_1, \ldots, X_k$ such that $X_i \succ X_j$ for any $i > j$ and $|X_1|, \ldots, |X_k| \leq n$. Notice that k is not specified as an input; you can decide the number of the groups in the partition, as long as the partition satisfies the given conditions. You need to show that your algorithm runs in $O(m \log(m/n))$ time.

Remark: We have not formally define the asymptotic notation for multi-variable functions in the class. For f and g be functions that maps $\mathbb{R}^k_{>0}$ to $\mathbb{R}_{>0}$, we say $f(\mathbf{x}) = O(g(\mathbf{x}))$ if there exist constants $M, C > 0$ such that $f(\mathbf{x}) \leq C \cdot g(\mathbf{x})$ for all $\mathbf{x}$ with $x_i \geq M$ for some i. The most rigorously running time should be written as $O(m \cdot \max\{\log(m/n), 1\})$, although it is commonly just written as $O(m \log(m/n))$ for this kind of scenarios.

Solution:

Put these m integers in array A, and execute the function $Partition(0, m-1, n)$.

Algorithm 1 $Partition(begin, end, n)$

1: **if** $m \leq n$ **then**
2: **return** $A[begin, end]$
3: **else**
4: Use the median-of-the-medians algorithm to find the $\frac{begin+end}{2}$-th smallest integer x^* in $A[begin, end]$
5: $Partition(begin, \frac{begin+end}{2}, n)$
6: $Partition(\frac{begin+end}{2} + 1, end, n)$
7: **end if**

Prove the algorithm runs in $O(m \log \frac{m}{n})$ time:

$$T(m) = 2T(\frac{m}{2}) + O(m)$$

Prove by induction that $T(m) \leq Am \log \frac{m}{n}$ for some constant A

Since the median-of-the-medians algorithm runs in $O(n)$ times,

Base Step: $T(m) = 1$ for $m \leq n$

Inductive Step: Suppose $T(i) \leq Ai \log \frac{i}{n}$ for $i > n$ and $i < m$ for $B - A \log 2 \leq 0$

$$\begin{aligned} T(m) &\leq 2A\frac{m}{2} \log \frac{m}{2n} + Bm \\ &= Am \log \frac{m}{2n} + Bm \\ &= Am(\log \frac{m}{n} - \log 2) + Bm \\ &= Am \log \frac{m}{n} + (B - A \log 2)m \\ &\leq Am \log \frac{m}{n} \end{aligned}$$

So $T(n) = O(m \log \frac{m}{n})$.

4. (25 points) Given an array $A[1, \ldots, n]$ of integers sorted in ascending order, design an algorithm to **decide** if there exists an index i such that $A[i] = i$ for each of the following scenarios. Your algorithm only needs to decide the existence of i; you do notGiven an array $A[1, \ldots, n]$ of integers sorted in ascending order, design an a need to find it if it exists.

   (a) The n integers are positive and distinct.

   (b) The n integers are distinct.

   (c) The n integers are positive.

   (d) The n integers are positive and are less than or equal to n.

   (e) No further information is known for the n integers.

   Prove the correctness of your algorithms. For each part, try to design the algorithm with running time as low as possible.

Solution:

(a) **Algorithm 2** Decide the existence of i

```
1: if A[1] = 1 then
2:     return True
3: else
4:     return False
5: end if
```

Prove by contradiction.

$\Longrightarrow$:

Assume there is no i such that $A[i] = i$ when $A[1] = 1$. Obviously there is a contradiction

$\Longleftarrow$:

Assume $A[1] \neq 1$ when there is an i such that $A[i] = i$.

$\because$ the n integers are positive and $A[1] \neq 1$

$\therefore A[1] > 1$

$\because A[1, \ldots, n]$ is ascending and the n integers are distinct

$\therefore i - 1 \leq A[i] - A[1]$

But there is an i such that $A[i] = i$, then $i - 1 > A[i] - A[1]$.

There is a contradiction, so the algorithm is correct.

(b) **Algorithm 3** $deciding(begin, end)$

1: **if** $A[begin] > begin$ or $A[end] < end$ **then**
2: **return** $False$
3: **end if**
4: **if** $A[begin] = begin$ or $A[end] = end$ **then**
5: **return** $True$
6: **else**
7: **return** $deciding(begin + 1, \frac{begin+end}{2}) \bigcup deciding(\frac{begin+end}{2} + 1, end - 1)$
8: **end if**

To prove this algorithm's correctness, we just need to prove if $A[begin] > begin$ or $A[end] < end$, then there is no i for $begin < i < end$.
$\because A[begin] > begin$ and $A[begin + k] \geq A[begin] + k$ for $k = 1, 2, \cdots, end - begin$
$\therefore A[begin + 1] > begin + 1$
. . . . . .
$\therefore A[end] > end$
$\therefore$ There is no i for $begin < i < end$.
Similarly, we can prove if $A[end] < end$, then there is no i for $begin < i < end$. So the algorithm is correct.

(c) **Algorithm 4** $deciding(begin, end)$

1: **if** $begin == end$ and $A[begin] \neq begin$ **then**
2: **return** $False$
3: **end if**
4: **if** $A[begin] > begin$ and $A[end] < end$ **then**
5: **return** $True$
6: **else**
7: **return** $deciding(begin + 1, \frac{begin+end}{2}) \bigcup deciding(\frac{begin+end}{2} + 1, end - 1)$
8: **end if**

To prove this algorithm's correctness, we just need to prove if $A[begin] > begin$ and $A[end] < end$, then there is an i for $begin < i < end$.
Prove by contradiction.
Assume $A[begin] > begin$ and $A[end] < end$ but there is no i for $begin < i < end$.
Assume $A[begin] - begin = k$, so $A[begin + k] > A[begin] = begin + k$
. . . . . .
$\therefore A[end] > end$
There is a contradiction, so the algorithm is correct.

(d) It is always true.
Prove by contradiction.
Assume $0 < A[i] \leq n$ and there is no i for $A[i] = i$.
$\therefore A[1] > 1$
$\because A[2] \geq A[i]$ and $A[2] \neq 2$
$\therefore A[2] > 2$
.....
$\therefore A[n] > n$
There is a contradiction, so the algorithm is correct.

(e) For every i, $i = 1, \ldots, n$:
If there is an i that satisfies $A[i] = i$, then there exists an index i such that $A[i] = i$.
The correctness is obvious.

5. How long does it take you to finish the assignment (including thinking and discussion)? Give a score (1,2,3,4,5) to the difficulty. Do you have any collaborators? Please write down their names here.

It took me about ten hours to finish the assignment. I spent about four hours thinking and discussing, and the rest of the time getting familiar with the use of Latex and search for relevant syntax. I'll give a 4 to the difficulty. My collaborator is Junqi You and his student ID is 522030910204.