Indian Institute of Technology Kharagpur CS21203 Algorithms-I, Autumn 2022

Class Test 1

31-August-2022	6.30pm - 7.30pm	Full Marks: 20

Write your answers in the question paper itself. Be brief and precise. Answer all questions.

Roll Number: _____

1. (5 marks) Solve the following recurrence relation:

$$T(n) = 297^2$$
 . $T(\sqrt[297]{n}) + 297$. $(\log n)^2$

Solution

Name: _

Let $n=2^m$. Then $T(2^m)=297^2$. $T(2^{\frac{m}{297}})+297$. m^2 Now let $S(m)=T(2^m)$. Then $S(m)=297^2$. $S(\frac{m}{297})+297$. m^2 Comparing with the general form of recursion T(n)=a $T(\frac{n}{b})+f(n)$, we get $a=297^2$ and b=297. So $n^{\log_b a}=n^{\log_{297}297^2}=n^2$. Hence f(n) and $n^{\log_b a}$ are asymptotically same. So by case 2 of Master Theorem we obtain, $S(m)=\Theta(m^2\log m)$. Changing back from S(m) to T(n) we obtain,

$$T(n) = T(2^m) = S(m) = \Theta(m^2 \log m) = \Theta((\log n)^2 \log \log n)$$

2. (5 marks) Work out the computational complexity of the following piece of code.

```
for ( i=1; i < n; i *= 2 ) {
  for ( j = n; j > 0; j /= 2 ) {
    for ( k = j; k < n; k += 2 ) {
      sum += (i + j * k );
    }
}</pre>
```

Solution:

Running time of the inner, middle, and outer loop is proportional to n, $\log n$, and $\log n$, respectively. Thus the overall Big-Oh complexity is $O(n(\log n)^2)$.

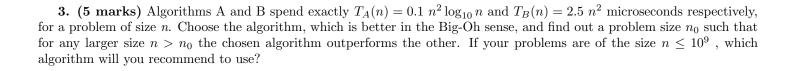
More detailed analysis: Let $n=2^k$. Then the outer loop is executed k times, the middle loop is executed k+1 times, and for each value $j=2^k, 2^{k-1}, \dots, 2, 1$, the inner loop has different execution times as follows:

j	Inner iterations	
2^k	1	
2^{k-1}	$(2^k - 2^{k-1})\frac{1}{2}$	
2^{k-2}	$(2^k - 2^{k-2})\frac{1}{2}$	
	• • •	
2^{1}	$(2^k - 2^1)\frac{1}{2}$	
2^{0}	$(2^k - 2^0)\frac{1}{2}$	

In total, the number of inner/middle steps is:

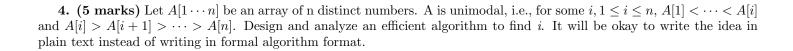
$$1 + k \cdot 2^{k-1} - (1 + 2 + \dots + 2^{k-1}) \frac{1}{2} = 1 + k \cdot 2^{k-1} - (2^k - 1) \frac{1}{2}$$
$$= 1 \cdot 5 + (k-1) \cdot 2^{k-1} \equiv (\log_2 n - 1) \frac{n}{2}$$
$$= O(n \log n)$$

Hence the total complexity is, $O(n(\log n)^2)$.



Solution:

In the Big-Oh sense, the algorithm B is better. It outperforms the algorithm A when $T_B(n) \leq T_A(n)$, that is, when $2.5 \ n^2 \leq 0.1 \ n^2 log_{10} n$. This inequality reduces to $log_{10} n \geq 25$, or $n \geq n_0 = 10^{25}$. If $n \leq 10^9$, the algorithm of choice is A.



Solution:

Probe at the middle of the array, i.e. $A\left[\frac{n}{2}\right]$ and compare it with $A\left[\frac{n}{2}-1\right]$ and $A\left[\frac{n}{2}+1\right]$. Now consider the following cases:

Case-1: $(A[\frac{n}{2}-1] < A[\frac{n}{2}] < A[\frac{n}{2}+1])$: The mode will be between $A[\frac{n}{2}+1]$ and A[n] and we can recurse on the part between $\frac{n}{2}+1$ and n.

Case-2: $(A[\frac{n}{2}-1] > A[\frac{n}{2}] > A[\frac{n}{2}+1])$: The mode will be between A[1] and $A[\frac{n}{2}-1]$ and we can recurse on the part between A[1] and $A[\frac{n}{2}-1]$.

<u>Case-3</u>: (otherwise:) So, here $A[\frac{n}{2}]$ is greater than both $A[\frac{n}{2}-1]$ and $A[\frac{n}{2}+1]$, and hence $A[\frac{n}{2}]$ is the peak element.

In all of the above (exclusive as well as exhaustive cases), we perform at most three comparisons and reduce the problem to a size of at most half of the original problem. Thus, we have an $O(\log n)$ time algorithm.