15210: Parallel and Sequential Data Structures and Algorithms

ParenLab

Zikang Wang (zikangw)

5.1

For a sequence, function subDist S would go along S to find the longest distance starting from its first element.

Function max S is a recursive function, it would compare the value of subDist S and the subsequence of S that has the largest parenthesis distance, then return the larger one of the two.

The base case of this case is the last two elements of S, which would return 0 or 2.

Therefore, the work of this algorithm can be represented as

$$n + (n-1) + (n-2) + \cdots + 2 = \frac{(n+2)(n-1)}{2}$$

That is,

$$W(n) = O(n^2)$$

Since we used recursion and computed the distance in each turn of the recursion,

$$S = O(n^2)$$

One improvement is saving the largest distances starting from each element in a sequence first, then compare, so it can be parallel and S would reduce to n.

But I'm busy and lazy…

5.2

1.

Guess:

$$W(n) = An + Blogn + C$$

So we need to find A, B and C

Base Case: $n = 1$

$$W(1) = A + C = log1 = 0$$

Inductive Case: $n \geq 2$

$$W(n) = 2T(\frac{n}{2}) + logn = 2A(\frac{n}{2}) + 2Blog\frac{n}{2} + 2C + logn$$

$$= An + (2B+1)logn + 2C - 2Blog2$$

To finish the inductive,

$$B = 2B + 1$$

$$C = 2C - 2Blog2$$

$$A + C = 0$$

Therefore,

$$B = -1$$

$$C = -2log2$$

$$A = 1$$

So,

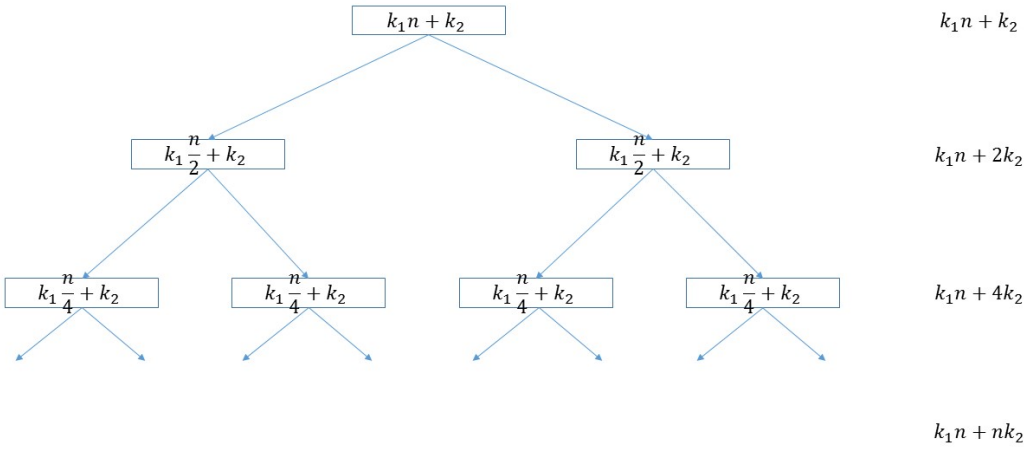
$$W(n) = n - logn - 2log2$$

$$W(n) = \Theta(n - logn - 2log2)$$

2.

If $W_{showt} \in \Theta(n)$, then we can assume $W_{showt} = k_1 n + k_2$, then

$$\mathrm{W(n)} = 2W\left(\frac{n}{2}\right) + k_1 n + k_2$$



Sum all the layers up,

$$sum = k_1 n \log n + (2n - 1)k_2$$

Therefore,

$$W(n) = \Theta(n \log n) + \Theta(n) = \Theta(n \log n)$$

5.3

Use brick method,

Number of Nodes	Work per Node	Total Work
1	n^2	n^2
9	$(\frac{n}{3})^2$	$9 \cdot (\frac{n}{3})^2 = n^2$
……		

Work on two layers are the same $W_i = W_{i+1}$, this is a balanced tree

Define d = depth then on the leaf node, the work is,

$$(\frac{n}{3^{d-1}})^2 = 1$$

Therefore,

$$d = \log_3 n + 1$$
$$W(n) = n^2 \times d = \Theta(n^2 \log n)$$

5.4

Similar to 5.3,

Number of Nodes	Work per Node	Total Work
1	$\log n$	$\log n$

1	$\log\sqrt{n}$	$\frac{1}{2}\log n$
.....		
1	1	

Total work is decreasing, $cW_i < W_{i-1}$, this is a root dominant tree, therefore

$$W(n) = W_{root}(n) = \Theta(\log n)$$

5.5

Use the brick method:

Number of Nodes	Work per Node	Total Work
1	$\log^2 n$	$\log^2 n$
1	$\log^2 \frac{n}{4}$	$\log^2 \frac{n}{4}$
.....		
1	1	

Total work is decreasing, $cW_i < W_{i-1}$, this is a root dominant tree, therefore

$$W(n) = \Theta(\log^2 n)$$

5.6

Number of Nodes	Work per Node	Total Work
1	1	1
2	1	2
4	1	4
.....		
n	1	n

Define d = depth, we have

$$2^{d-1} = n$$

$$\mathrm{d} = \mathrm{logn} + 1$$

Therefore,

$$\mathrm{W(n)} = \frac{1 \cdot (2^d - 1)}{2 - 1} = 2\mathrm{n} - 1 = \mathrm{O(n)}$$

5.7

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(\mathrm{logn})$$

Guess:

$$T(n) \leq k_1 n^x + k_2 logn$$

Base Case: $n = 1$

$$T(1) = k_1 = O(1) \leq O(logn)$$

Inductive Case: $n > 1$

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O\log(n) \le [k_1(\frac{n}{5})^x + k_2 log\frac{1}{5}n] + [k_1(\frac{7n}{10})^x + k_2 log\frac{7}{10}n] + logn$$

$$= [(\frac{1}{5})^x + (\frac{7}{10})^x]k_1 n^x + k_2 logn + (k_2 log\frac{7n}{50} + logn) = k_1 n^x + k_2 logn$$

To make the above equation valid, let:

$$(\frac{1}{5})^x + (\frac{7}{10})^x = 1$$

$$k_2 log\frac{7n}{50} + logn = const$$

Therefore, the parameters should be:

$$k_1 = c,$$
$$k_2 = -1$$