Lab02-Divide and Conquer

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2021.

∗ If there is any problem, please contact TA Haolin Zhou.

∗ Name:Yanjie Ze Student ID:519021910706 Email: zeyanjie@sjtu.edu.cn

1. *Recurrence examples.* Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible.

   (a) $T(n) = 4T(n/3) + n \log n$

   (b) $T(n) = 4T(n/2) + n^2\sqrt{n}$

   (c) $T(n) = T(n-1) + n$

   (d) $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log n$

Solution. (a) Because $n < nlogn < n^2$, and we denote:

$$T_1(n) = 4T(n/3) + n$$

$$T_2(n) = 4T(n/3) + n^2$$

Based on Master Theorem, we have:

$$T_1(n) = O(n^{log_3 4})$$

$$T_2(n) = O(n^2)$$

Therefore:

$$T(n) = O(n^2), T(n) = \Omega(n^{log_3 4})$$

(b)

$$T(n) = 4T(n/2) + n^2\sqrt{n} = 4T(n/2) + n^{\frac{5}{2}}$$

Based on Master Theorem, $a = 4, b = 2, d = \frac{5}{2}$,we have:

$$T(n) = O(n^{\frac{5}{2}})$$

(c)

$$T(n) = \sum_{i=1}^{n} i + T(0) = \frac{n^2+n}{2} + T(0)$$

Thus:

$$T(n) = \Omega(n^2)$$

(d) Assume $n = 2^{2^k}$, then $k = log_2(log_2 n)$.

$$T(2^{2^k}) = 2T(2^{2^{k-1}}) + 2^k log2$$

$$\frac{T(2^{2^k})}{2^k} = \frac{T(2^{2^{k-1}})}{2^{k-1}} + log2$$

Thus we can sum them together and get:

$$\frac{T(2^{2^k})}{2^k} = (k-1)log2 + T(2)$$

Multiply both sides by 2^k and then replace k by n:

$$T(n) = log2 \cdot log_2 n \cdot log_2(log_2 n) + T(2) \cdot log_2 n$$

Therefore:

$$T(n) = \Theta(logn \cdot log(logn))$$

□

2. *Divide-and-conquer.* Given an integer array $A[1..n]$ and two integers $lower \le upper$, design an algorithm using **divide-and-conquer** method to count the number of ranges (i, j) $(1 \le i \le j \le n)$ satisfying

$$lower \le \sum_{k=i}^{j} A[k] \le upper.$$

Example:

Given $A = [1, -1, 2]$, $lower = 1$, $upper = 2$, return 4.

The resulting four ranges are $(1, 1)$, $(3, 3)$, $(2, 3)$ and $(1, 3)$.

(a) Complete the implementation in the provided C/C++ source code (The source code *Code-Range.cpp* is attached on the course webpage).

(b) Write a recurrence for the running time of the algorithm and solve it by recurrence tree (You can modify the figure sources *Fig-RecurrenceTree.vsdx* or *Fig-RecurrenceTree.pptx* to illustrate your derivation).

(c) Can we use the Master Theorem to solve the recurrence above? Please explain your answer.

Solution. (a)

$$T(n) = 2T(\frac{n}{2}) + nlogn$$

□

3. *Transposition Sorting Network.* A comparison network is a **transposition network** if each comparator connects adjacent lines, as in the network in Fig. 1.

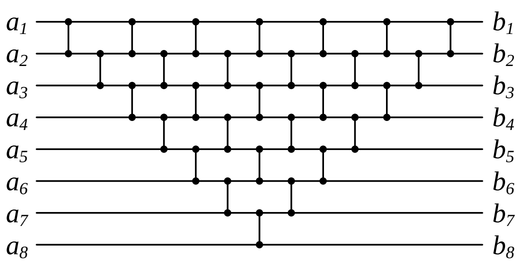

Figure 1: A Transposition Network Example

(a) Prove that a transposition network with n inputs is a sorting network if and only if it sorts the sequence $\langle n, n-1, \cdots, 1\rangle$. (Hint: Use an induction argument analogous to the *Domain Conversion Lemma*.)

(b) (Optional Sub-question with Bonus) Given any $n \in \mathbb{N}$, write a program using Tkinter in Python to draw a figure similar to Fig. 1 with n input wires.