Lab02-Divide and Conquer

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

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

* Name:Xin Xu Student ID:519021910726 Email: xuxin20010203@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. We can figure out the time complexity bu the tool of Master Theorem.

   (a) Since $\log n < n^a$ for any real number $a > 0$, by the Master Theorem, $a = 4, b = 3, disalittlegreaterthan1$. So, $d < \log_3 4$, and $T(n) = O(n^{\log_3 4})$.

   (b) By the Master Theorem, $a = 4, b = 2, d = 5/2$. So, $d > \log_b a = 2$. Thus, $T(n) = O(n^{\frac{5}{2}})$.

   (c) Because of recurrence, $T(n) = T(n-1) + n = T(n-2) + (n-1) + n = T(n-3) + (n-2) + (n-1) + n ...... = T(1) + 2 + 3 + ... + (n-2) + (n-1) + n = O(n^2)$.

   (d) Without generity, we assume that $n = 2^{2^k}$. So, $\log_2 n = 2^k, k = \log_2(\log_2 n)$.
   Because of recurrence, $T(n) = 2T(n^{\frac{1}{2}}) + \log n = 2(2T(n^{\frac{1}{2^2}}) + \log\frac{1}{2}) + \log n = 2^2T(n^{\frac{1}{2^2}}) + 2\log n = ... = 2^kT(2) + k\log n = A\log n + B\log n * \log(\log n) = O(\log n \log(\log n))$.

   □

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

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

   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) The .cpp file is in the homework folder.

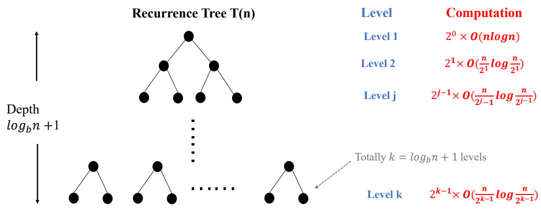

Figure 1: The Recurrence Tree

(b) We assume that the input is $O(n)$ for there are n integers in the array to operate the entry.
Additionally, the time complexity of merge_count is $T(n)$, which satisfies $T(n) = 2T(n/2) + O(n \log n)$.
Considering that $n = 2^k$. With the help of the recurrence Tree, we can figure out that $T(n) = \sum_{i=0}^{k-1} 2^i \times O(\frac{n}{2^i} \log \frac{n}{2^i}) = \sum_{i=0}^{k-1} O(2^i \times \frac{n}{2^i} \log \frac{n}{2^i}) = O(n) \sum_{i=0}^{k-1} \log \frac{n}{2^i} = O(n) \sum_{i=0}^{k-1} (\log n - i \log 2) = O(n \times (\log n)^2)$.

(c) No. If we use the Master Theorem to solve the recurrence above, the output is that $a = 2, b = 2$, and $\log n$ is smaller than n^k , for any $k > 0$, but is greater than 1. Since we know $O(n) < O(n \log n)$, so we can conclude that $n \log n = n^d, d > 1$. And the time complexity $T(n) = O(n \log n)$. This conclusion is contrast from the answer we get above. In fact, we just konw $n \log n$ is just a little greater than n, and we don't know the exact amount it is.In other words, it's like a boundary. And at this special point, we can't figure out the size between $log_b a$ and d. So, we can't use the Master Theorem.

□

3. *Transposition Sorting Network.* A comparison network is a **transposition network** if each comparator connects adjacent lines, as in the network in Fig. 2.

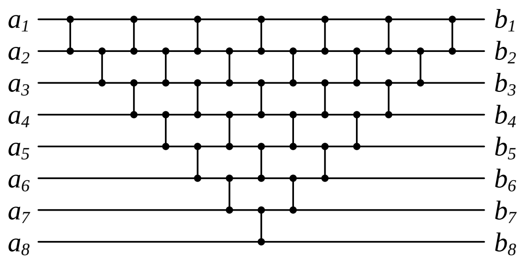

Figure 2: A Transposition Network Example

(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. 2 with n input wires.

Solution. I just finish the first question.

(a) **Proof.** $\Rightarrow$: If a transposition network with n puts is a sorting network, it's obvious that it can sort the sequence $<n, n-1, ..., 1>$ because of the definition of sorting network.
$\Leftarrow$: To improve it, we could use the method of induction.

basic step. For any inputs $<a_1, a_2>$ to a comparator, there is a monotonically increasing function f which maps $<1, 2>$ to $<a_1, a_2>:$ $f(2)$=max(a_1, a_2), $f(1)$=min(a_1, a_2). According to Domain Conversion Lemma, since the transposition network can sort $<1, 2>$, it can also sort $<a_1, a_2>$ just with the replace of elements.
hypothesis. For any depth $d < k, k \geqslant 1$, the network can sort sequence $<a_n, a_{n-1}, ..., a_1>$.
induction. For a comparator with the depth $d = k$ and inputs of $<a_i, a_j>$, it's also true because of our **basic step.** So, we can claim the statement: if a transposition network can sort the sequence $<n, n-1, ..., 1>$, it's a sorting network. $\square$

(b) I'm sorry I can't finish it on time.

$\square$