Lab01-Algorithm Analysis

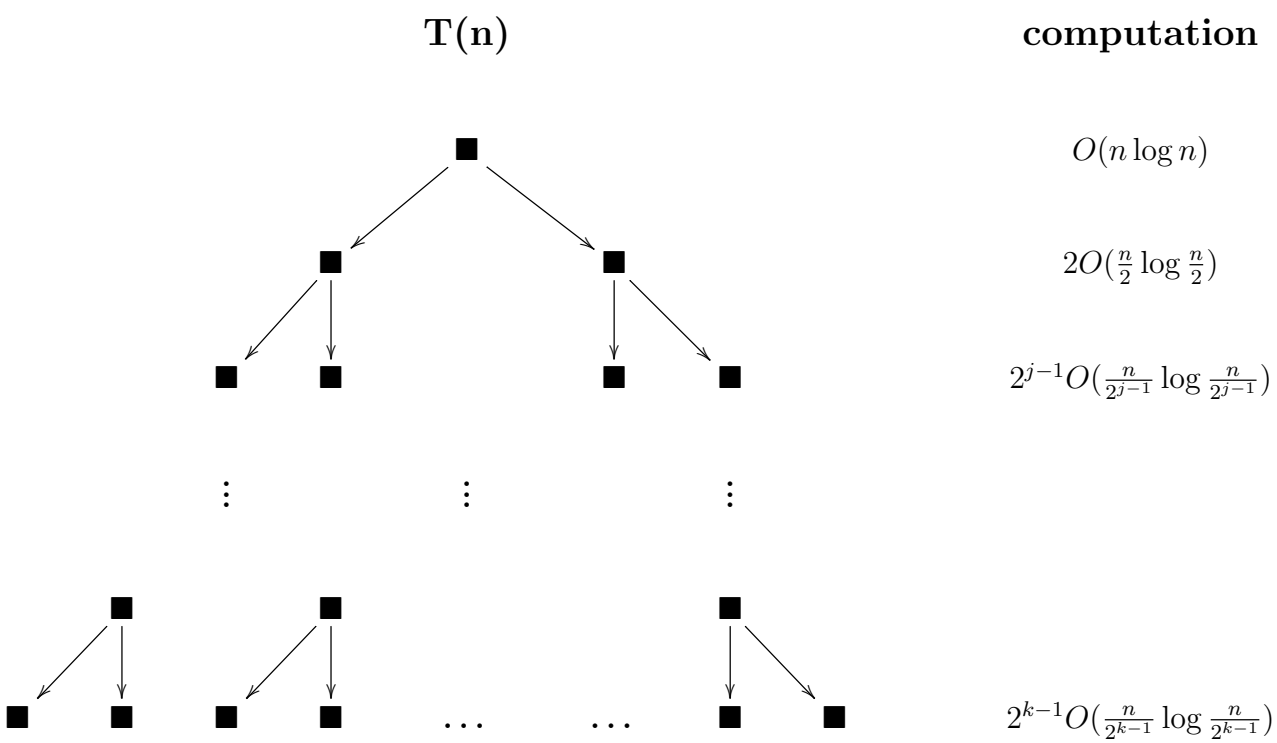
CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2019.

∗ If there is any problem, please contact TA Mingran Peng. Also please use English in homework.

∗ Name: 田雪飞 Student ID: 515030910347 Email: 13487426939@qq.com

1. **Solution.** The following is solution process:

   (a) recurrence tree T(n):



   Complexity of T(n)=sum up all computations at each level.

   (b) **No,because $O(n\log n)$ is not same as $O(n^d)$.**
   As we all know:$k=\log n$

$$T(n)=\sum_{0}^{k-1}2^{j-1}O(\frac{n}{2^{j-1}}\log\frac{n}{2^{j-1}})=\sum_{1}^{k}2^{j}O(\frac{n}{2^{j}}\log\frac{n}{2^{j}})=\sum_{1}^{k}O(n\log\frac{n}{2^{j}}) \tag{1}$$

$$=\sum_{1}^{k}O(n\log\frac{2^k}{2^{j}})=\sum_{1}^{k}O(n\log 2^{j})=O(n\log^2 n) \tag{2}$$

□

2. **Solution.** The following is solution process:

   (a) The answer is in homework2.cpp

   (b) recurrence T(n) by using Master Theorem:

$$T(n)=2T(\frac{n}{2})+O(n^2)=O(n^2) \tag{3}$$

□

3. **Solution.** The following is solution process:

(a) induction:

- n=1,we do not need sort:;
- n=2,obviously,we can use a comparator to sort two elements;
- we assume that we can sort k elements by using sort working:
- n=k+1;we can construct a function:

$$f(x) = \begin{cases} 0 & \text{if } x < a_{k+1} \\ 1 & \text{if } x >= a_{k+1} \end{cases}$$

we can sort k elements to $\{0, 0 \ldots, 1, 1\}$;
$a_{k+1} = 1$,and we can compare a_{k+1} with element in $\{0, 0 \ldots, 1, 1\}$ one-by-one. obviously,we can find a right index for a_{k+1} after k comparisons.
So, a transposition network with n input is a sorting network if and only if it sort the sequence $\{n, n-1 \ldots, 1\}$.

(b) The answer is in homework2.py

□