CS 430 Introduction to Algorithms – Fall 2017
Recitation Class Note 3
Xiaolang Wang

A Little More on Merge Sort and Quicksort

1. Time complexity of Merge
    - Merge: given 2 sorted subarrays with size $\frac{n}{2}$, get a sorted array with size n.
    - Only compare the first element in the remaining of each subarray.

2. Please consider some special inputs for quicksort, what's the time complexity?
    - Sorted array
    - Reversed sorted array
    - An array of all same numbers

3. Average Case Study of Quicksort
    - Average = Expectation
    - $T_{avg} = \sum T(Case_i) \times \Pr[Case_i]$
    - $T_{avg} \neq T(average\ of\ the\ input)$, unless T is a linear function
    - For example, the running time of quicksort, $T_{avg} \neq T\left(\frac{1}{4}n\right) + T\left(\frac{3}{4}n\right) + \Theta(n)$
    - Real average case (Professor Reingold's Notes)
        - ❖ Given array of size n, a partition yields one subarray of size k, and the other subarray has a size (?)

          $$T(n) = \Theta(n) + \sum_{k=0}^{n-1} (T(k) + T(n-1-k) \times \Pr[k\ elements\ less\ than\ the\ pivot]$$

          $$= \Theta(n) + \sum_{k=0}^{n-1} \frac{1}{n} (T(k) + T(n-1-k)$$

          $$= \Theta(n) + \frac{2}{n} \times \sum_{k=0}^{n-1} T(k)$$

          $$= an + b + \frac{2}{n} \times \sum_{i=0}^{n-1} T(i) \ldots\ldots\ldots\ldots replace\ k\ with\ i\ from\ here\ on$$
        - ❖ Times n on both sides

          $$n \times T(n) = an^2 + bn + 2\sum_{i=0}^{n-1} T(i) \ldots\ldots\ldots\ldots (1)$$
        - ❖ Equation (1) holds for $n-1$

          $$(n-1) \times T(n-1) = a(n-1)^2 + b(n-1) + 2\sum_{i=0}^{n-2} T(i) \ldots\ldots\ldots\ldots (2)$$
        - ❖ (1) – (2):

$nT(n) - (n-1)T(n-1) = a(2n-1) + b + 2T(n-1)$
$\Rightarrow nT(n) - (n+1)T(n-1) = a(2n-1) + b$

- Divide by $n(n+1)$:
$\frac{T(n)}{n+1} - \frac{T(n-1)}{n} = \frac{(2n-1)a+b}{n(n+1)} = \frac{3a-b}{n+1} + \frac{b-a}{n} = \frac{2a}{n} - (3a-b)\left(\frac{1}{n} - \frac{1}{n+1}\right)$
- Given first several values $T(0) = t_0, T(1) = t_1 \ldots T(n_0) = t_{n_0}, n > n_0$; sum up both sides from $n_0 + 1$ to n
$\sum_{i=n_0+1}^{n} (\frac{T(i)}{i+1} - \frac{T(i-1)}{i}) = \sum_{i=n_0+1}^{n} (\frac{2a}{i} - (3a-b)\left(\frac{1}{i} - \frac{1}{i+1}\right))$
- $LHS = \frac{T(n)}{n+1} - \frac{T(n_0)}{n_0+1}$
- $RHS = 2a\left(H_n - H_{n_0}\right) - (3a-b)\left(\frac{1}{n_0+1} - \frac{1}{n+1}\right)$
- Combining both sides we can get:
$T(n) = 2anH_n + 2aH_n + 3a - b + (n+1)(\frac{T(n_0)}{n_0+1} - 2H_{n_0} - \frac{3a-b}{n_0+1})$
  - Here H_n is the n^th Harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} \ldots \frac{1}{n} = \ln(n+1)$, when n is large.
- We have $H_n = \ln n + O(1)$, thus we have:
$T(n) = 2anH_n + O(n) = 2a\, n \ln n + O(n) = (2a \ln 2) n \lg n + O(n)$
$\approx (1.386a) n \lg n + O(n)$
- Average is about 38.6% slower than the best case, and still $\Theta(n \lg n)$
  - Using the same assumption for Partition() has a time complexity $\Theta(n) = an + b$, we can get the best case $T(n) = 2\left(\frac{T}{2}\right) + an + b = (an+b)\lg n =$
$an \lg n + \Theta(\lg n)$.