Lec 04 快排及随机化算法

Quick Sort, by Tony Hoare in 1962

- divide and conquer
- sorts "in place" (rearranges elements where they are)
  节省内存
- very practical (with tuning)

△ divide and conquer

1. divide: (★)

   to partition array into 2 subarrays, around an element called pivot x, such that elements in the lower subarray are less than or equal to x, and elements in the upper subarray are greater than or equal to x.

$\le x$	x	$\ge x$

2. conquer:

   to recursively sort the 2 subarrays

3. combine:

   trivial

△ key: linear-time ($\Theta(n)$) partitioning subroutine

```
partition (A, p, q)   // A[p...q]
    x ← A[p]      // pivot = A[p]
    i ← p
    for j ← p+1 to q
        do if A[j] ≤ x
            then i ← i+1
                exchange A[i] ↔ A[j]
    exchange A[p] ↔ A[i]
    return i
```

	$\le x$	≥ 0	?	
p	i	j		q

"移动界限的位置"

Example: 6 10 13 5 8 3 2 11 $x \leftarrow 6$ (pivot)

```
6   10  13  5   8   3   2   11
i   j
. . .   . . .   . . .
6   10  13  5   8   3   2   11
i   i+1     j
6   5   13  10  8   3   2   11
    i           j
```

6 5 13 10 8 3 2 11
$\quad$ i $\qquad\qquad\quad$ j

6 5 3 10 8 13 2 11
$\qquad$ i $\qquad\qquad\quad$ j

6 5 3 2 8 13 10 11 (loop terminates)
$\qquad\quad$ i $\qquad\qquad\quad$ j

6 5 3 2 8 13 10 11 (to put the pivot element in the middle between the two subarrays)
p $\qquad\quad$ i

$\underbrace{2\ \ 5\ \ 3}_{\leq \text{pivot}}\ \underbrace{\boxed{6}}_{\text{pivot}}\ \underbrace{8\ \ 13\ \ 10\ \ 11}_{\geq \text{pivot}}$

QuickSort (A, p, q)
$\quad$ if $p < q$
$\quad\quad$ then $r \leftarrow$ partition (A, p, q)
$\quad\quad\quad$ QuickSort $(A, p, r-1)$
$\quad\quad\quad$ QuickSort $(A, r+1, q)$

initial call: QuickSort $(A, 1, n)$

- analysis

worst case:

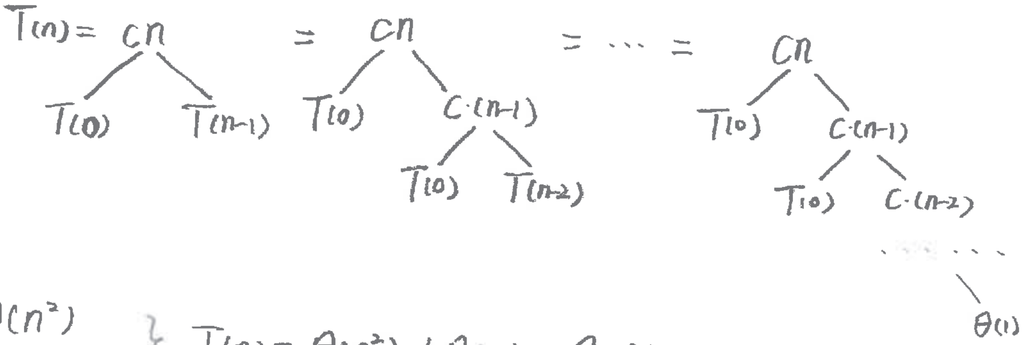
if you always pick the pivot, and everything is greater than or everything is less than this pivot, you are not going to partition the array very well.

$\Leftrightarrow$ if it is already sorted or reverse sorted

in those cases, one side of each partition has no elements

$$
\begin{aligned}
T(n) &= T(0) + T(n-1) + \theta(n) \\
&= \theta(1) + T(n-1) + \theta(n) \\
&= T(n-1) + \theta(n) \\
&= \theta(n^2) \quad \text{(arithmetic series)}
\end{aligned}
$$

recursion tree for $T(n) = T(0) + T(n-1) + cn$



height $= n$

$$\left.\begin{aligned} &\theta\left(\sum_{k=1}^{n} c\cdot k\right) = \theta(n^2) \\ &n \cdot T(0) = n \cdot \theta(1) = \theta(n) \end{aligned}\right\} \quad T(n) = \theta(n^2) + \theta(n) = \theta(n^2)$$

best case (intuition only):

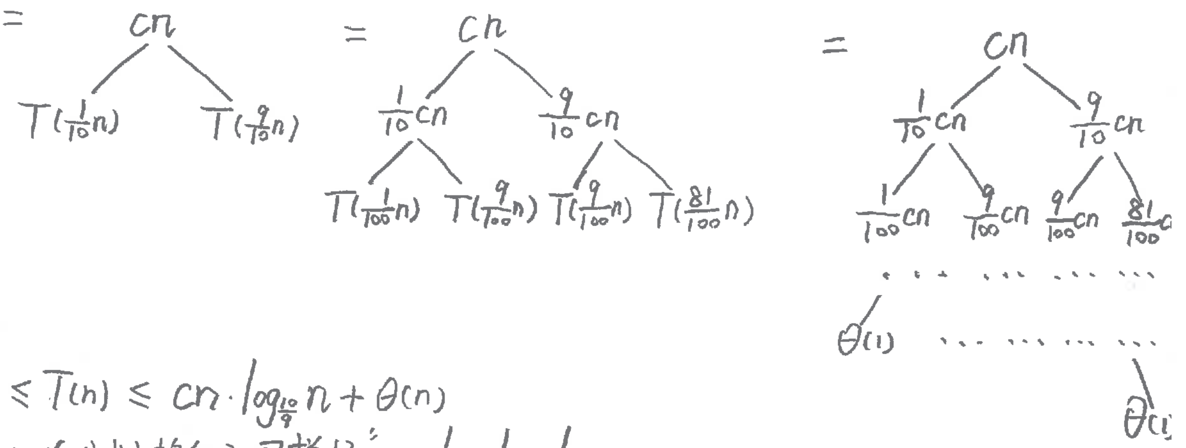
if we are really lucky, partition splits the array $n/2 : n/2$

$$T(n) = 2T(\tfrac{n}{2}) + \theta(n)$$
$$= \theta(n\log_2 n)$$

suppose split is always $1/10 : 9/10$.

$$T(n) = T(\tfrac{1}{10}n) + T(\tfrac{9}{10}n) + \underbrace{\theta(n)}_{\leq cn}$$

recursion tree: $T(n) =$



$$cn\log_{10} n + \theta(n) \leq T(n) \leq cn\cdot\log_{\frac{10}{9}} n + \theta(n)$$

"1:9 的分划和 1:1 的分划趋向于同样好" lucky!

suppose we alternate lucky, unlucky, lucky, ...

$L(n) = 2U(\tfrac{n}{2}) + \theta(n)$, lucky step

$U(n) = L(n-1) + \theta(n)$, unlucky step

then $L(n) = 2[L(\tfrac{n}{2}-1) + \theta(\tfrac{n}{2})] + \theta(n)$

$$= 2L(\tfrac{n}{2}-1) + \theta(n)$$
$$= \theta(n\lg n)$$ lucky!

how can we ensure that we are usually lucky?

to randomly choose the pivot, randomized-QuickSort,

then (1) the running time is independent of the input ordering

(2) it makes no assumptions about the input distribution

(3) there is no specific input that can elicit the worst-case behavior "引出,探出,诱出"

(4) the worst-case is determined only by a random-number generator

Analysis

$T(n)$ = random variable for the running time assuming that the random numbers are independent for $k = 0, 1, \cdots, n-1$, let $x_k = \begin{cases} 1, & \text{if partition generates a } k{:}n-k-1 \text{ split} \\ 0, & \text{otherwise} \end{cases}$

$$E(x_k) = 0 \cdot P(x_k = 0) + 1 \cdot P(x_k = 1)$$
$$= P(x_k = 1)$$
$$= \frac{1}{n}$$

$$T(n) = \begin{cases} T(0) + T(n-1) + \theta(n), & \text{if } 0{:}n-1 \text{ split} \\ T(1) + T(n-2) + \theta(n), & \text{if } 1{:}n-2 \text{ split} \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \\ T(n-1) + T(0) + \theta(n), & \text{if } n-1{:}0 \text{ split} \end{cases}$$
$$= \sum_{k=0}^{n-1} x_k \cdot [T(k) + T(n-k-1) + \theta(n)]$$

$$E(T(n)) = E\left(\sum_{k=0}^{n-1} x_k \cdot [T(k) + T(n-k-1) + \theta(n)] \right)$$

期望的线性叠加性：期望的和等于和的期望
vice versa

$$= \sum_{k=0}^{n-1} E\left(x_k \cdot [T(k) + T(n-k-1) + \theta(n)] \right)$$
$$= \sum_{k=0}^{n-1} E(x_k) \cdot E(T(k) + T(n-k-1) + \theta(n))$$
$$= \frac{1}{n} \sum_{k=0}^{n-1} E(T(k) + T(n-k-1) + \theta(n))$$
$$= \underbrace{\frac{1}{n} \sum_{k=0}^{n-1} E(T(k))}_{} + \underbrace{\frac{1}{n} \sum_{k=0}^{n-1} E(T(n-k-1))}_{} + \frac{1}{n} \underbrace{\sum_{k=0}^{n-1} \theta(n)}_{\theta(n^2)}$$

"identical"

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

"to absorb $k = 0, 1$ terms into $\theta(n)$ for technical convenience"

$$= \frac{2}{n} \sum_{k=2}^{n-1} E(T(k)) + \theta(n)$$

prove: $E(T(n)) \leq a \cdot n \cdot \lg n$ for const $a > 0$

proof: choose a big enough so that $a \cdot n \cdot \lg n > E(T(n))$ for small n

use fact: $\sum_{k=2}^{n-1} k \lg k \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$

substitution: $E(T(n)) \leq \frac{2}{n} \sum_{k=2}^{n-1} (a \cdot k \cdot \lg k) + \theta(n)$

$$\leq \frac{2a}{n} \cdot \left(\frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) + \theta(n)$$
$$= an \cdot \lg n - \frac{a}{4} n + \theta(n)$$
$$= \underbrace{a \cdot n \lg n}_{\text{desired}} - \underbrace{\left(\frac{a}{4} n - \theta(n) \right)}_{\text{residual}}$$
$$\leq a \cdot n \cdot \lg n, \text{ if } a \text{ is big enough so that } \text{"} \tfrac{an}{4} > \theta(n) \text{"}$$