Lecture 4

Quicksort – Hoare 1962

- divide & conquer
- sorts "in place"
- very practical (with tuning)

insertion sort is "in place"
merge sort is not "in place"

Divide and conquer

1) **Divide**: Partition array into 2 subarrays around pivot x, ∋ elements in lower subarray $\le x \le$ elems in upper subarray.

| $\le x$ | x | $\ge x$ |

2) **Conquer**: recursively sort [redacted] 2 subarrays

3) **Combine**: trivial

Quicksort → recursive partitioning, key is linear-time $\Theta(n)$ partitioning

vs.

Mergesort → recursive merging

Partition (A, p, q)

```
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     ← move i
           exch A[i] ↔ A[j]
  exch A[p] ↔ A[i]
  return i
```

$\underbrace{A[i]}_{\ge x} \leftrightarrow \underbrace{A[j]}_{\le x}$

Invariant

| p | $\le x$ | i | $\ge x$ | j | ? | q |

[redacted]

Time = $\Theta(n)$

Ex: $x \leftarrow 6$ (pivot)

6 10 13 5 8 3 2 11
i j

6 5 13 10 8 3 2 11
i j

6 5 3 10 8 13 2 11
i j

6 5 3 2 8 13 10 11
i j

$\underbrace{2\ 5\ 3}_{\le \text{pivot}}$ $\underbrace{6}_{\text{pivot}}$ $\underbrace{8\ 13\ 10\ 11}_{\ge \text{pivot}}$

Quicksort (A, p, q)

```
Quicksort (A, p, q)
  if p < q
    then r ← Partition (A, p, q)
         Quicksort (A, p, r-1)
         Quicksort (A, r+1, q)
```

zero or single element A as base case

Init. call: Quicksort (A, 1, n)

usually have a separate routine for small arrays down the recursion tree

Analysis - assume all elems distinct.

$T(n)$ = worst-case time

- input sorted or reverse sorted } pick pivot where everything is $\geq$ or $\leq$ $\Rightarrow$ not going to partition well

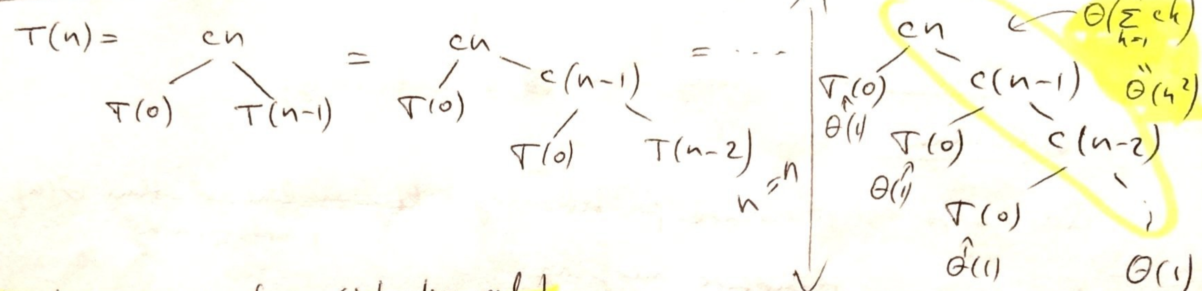
- one side of partition has no elems.

$$T(n) = T(0) + T(n-1) + \Theta(n) = \Theta(1) + T(n-1) + \Theta(n) = T(n-1) + \Theta(n)$$

one side no elems ↑   ↑ other side has $n-1$ elems

$= \Theta(n^2)$ arithmetic series, like insertion sort

Recursion tree $\quad T(n) = T(0) + T(n-1) + cn$



total: $T(n) = \Theta(n) + \Theta(n^2) = \Theta(n^2)$

Best case analysis (intuition only)

If we are really lucky

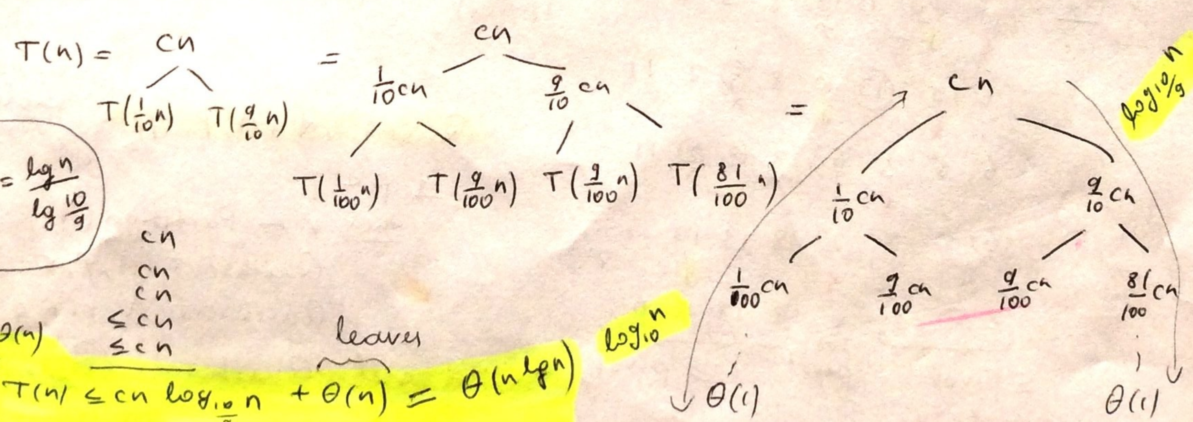
Partition splits the array $n/2 : n/2$

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n \lg n) \quad \text{case 2}$$

Suppose split is always $\frac{1}{10} : \frac{9}{10}$ (intuition)

$$T(n) = T\left(\frac{1}{10}n\right) + T\left(\frac{9}{10}n\right) + \Theta(n)$$

Recursion tree



$$\log_{\frac{10}{9}} n = \frac{\lg n}{\lg \frac{10}{9}}$$

cn
cn
cn
$\leq$ cn
$\leq$ cn

$cn \log_{10} n + \Theta(n) \leq T(n) \leq cn \log_{\frac{10}{9}} n + \underbrace{\Theta(n)}_{\text{leaves}} = \Theta(n \lg n)$ $\log_{10} n$

Lecture 4

Suppose we alternate: lucky, unlucky, lucky ...

$L(n) = 2U(\frac{n}{2}) + \Theta(n)$ lucky
$U(n) = L(n-1) + \Theta(n)$ unlucky

} system of recurrences

Then $L(n) = 2\left[L(\frac{n}{2}-1) + \Theta(\frac{n}{2})\right] + \Theta(n) = 2L(\frac{n}{2}-1) + \Theta(n)$

$= \Theta(n \lg n)$ lucky

Ranomly choose pivot, or randomly rearrange elements (permute) to avoid worst case

Randomised Quicksort (pivot on rand. element)

- running time is independent of input ordering
  - ↳ may get unlucky due to random # generator
    - ↳ rerun or run a batch
- no assumptions about input distr.
- no specific input [redacted] elicit worst case behavior
- worst case determined only by random number generator
  - ↳ can bound

$T(n)$ = r.v. for running time
assuming rand #'s independent

For $k = 0, 1, \ldots, n-1$, let

$$X_k = \begin{cases} 1 & \text{if partition generates } \overset{\le x}{k} : \overset{\ge x}{n-k-1} \text{ split} \\ 0 & \text{otherwise} \end{cases}$$

n r.v.'s $X_0 \ldots X_{n-1}$ } indicator r.v.'s

$$E[X_k] = 0 \cdot \Pr\{X_k = 0\} + 1 \Pr\{X_k = 1\} = 1 \Pr\{X_k = 1\} = \frac{1}{n}$$

$$T(n) = \begin{cases} T(0) + T(n-1) + \Theta(n) & \text{if } 0 : n-1 \text{ split} \rightarrow \text{worst case (ordered)} \\ T(1) + T(n-2) + \Theta(n) & \text{if } 1 : n-2 \text{ split} \\ \vdots & \\ T(n-1) + T(0) + \Theta(n) & \text{if } n-1 : 0 \text{ split} \rightarrow \text{worst case (reverse-ordered)} \end{cases}$$

My comment → r.v. recurrence

$$= \sum_{k=0}^{n-1} X_k \left[T(k) + T(n-k-1) + \Theta(n)\right]$$

My comment
cannot just take the sum of all splits; [redacted] the value of $T(n)$ [redacted] r.v. is generated each time for one split [redacted] → need X_k

$$E[T(n)] = E[\Sigma \ldots]$$

$$= \sum_{k=0}^{n-1} E\left[X_k \left[T(k) + T(k^{n-}_{-1}) + \Theta(n)\right]\right]$$ linearity of exp.

$$= \sum_{k=0}^{n-1} E[X_k]\, E[T(k) + T(n-k-1) + \Theta(n)]$$ independence

My comment: X_k tells nothing about the value of $T(k)$ [redacted] $T(n-k-1)$ r.v. and vice versa $\Rightarrow$ independent

my note: [redacted]

$$= \frac{1}{n} \sum_{k=0}^{n-1} E\left[T(k) + T(n-k-1) + \Theta(n)\right]$$ linearity

eq.

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

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

Absorb $k=0,1$ into $\Theta(n)$ for tech. convenience

recurrence: for expectation

$$\boxed{E[T(n)] = \frac{2}{n} \sum_{k=2}^{n-1} E[T(k)] + \Theta(n)}$$

Prove: $E[T(n)] \leq a n \lg n$, for const $a > 0$

choose a big enough so that $a n \lg n \geq E[T(n)]$ for small n

why excluded $n = 0,1$

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

My comment

$$E[T(n)] \leq \frac{2}{n} \sum_{k=2}^{n-1} a k \lg k + \Theta(n)$$ by IH (with big enough a for small n)

choose a bigger

$$\leq \frac{2a}{n} \left(\frac{1}{2} n^2 \lg n - \frac{1}{8} n^2\right) + \Theta(n)$$ by fact (above)

$$= \underbrace{a n \lg n}_{\text{desired}} - \underbrace{\left(\frac{an}{4} - \Theta(n)\right)}_{\text{residual}}$$

My comment: a for the inductive step to be true

$\leq a n \lg n$ if a is big enough s.t. $\frac{an}{4} > \Theta(n)$

$a > \frac{4\Theta(n)}{n}$

$$\boxed{E[T(n)] \leq a n \lg n}$$ ✓

in practice randomized quicksort is faster than mergesort