Chapter 7
Quicksort

Algorithm Analysis

School of CSEE

Quicksort

- Proposed by C. A. R. Hoare in 1962
- Divide-and-conquer algorithm
- Sorts “in place”
- Very practical (with tuning)

Divide-and-Conquer

- Quicksort an n-element array:

1. *Divide:* Partition the array into two subarrays around a *pivot* x such that elements in lower subarray $\leq x \leq$ elements in upper subarray.

$\leq x$	x	$\geq x$

2. *Conquer:* Recursively sort the two subarrays.
3. *Combine:* Trivial.

Key: Linear- time partitioning subroutine.

Partitioning subroutine

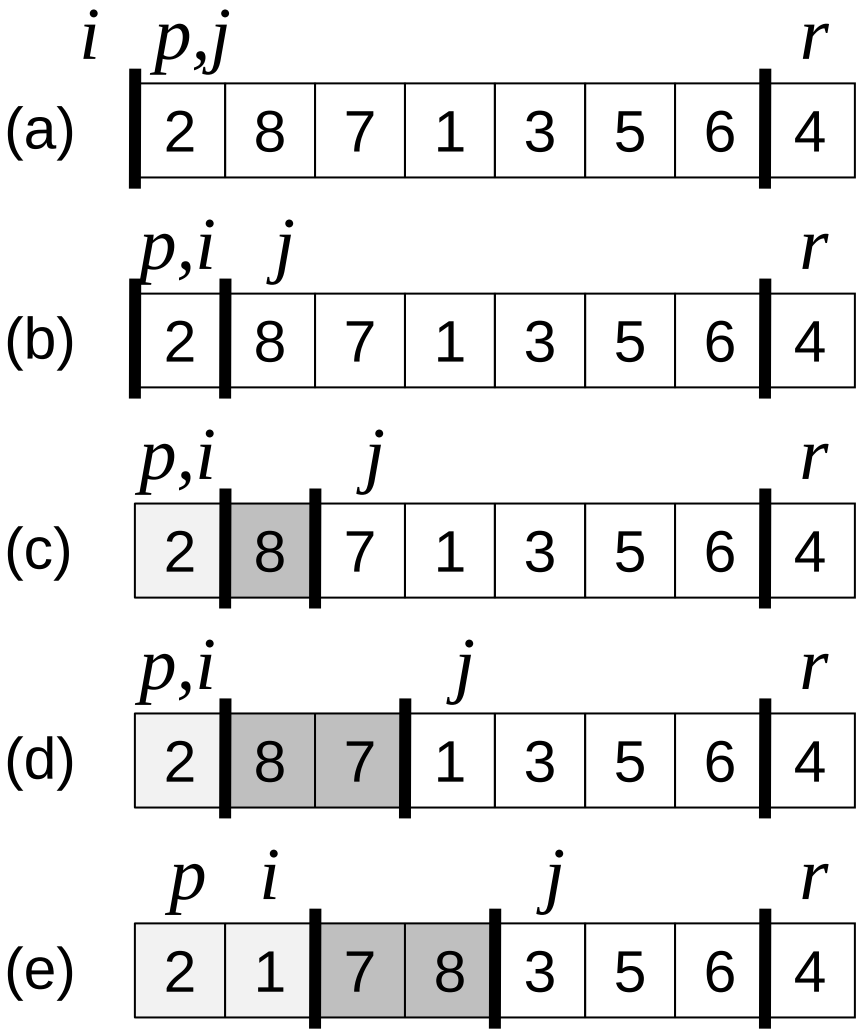
```
PARTITION(A, p, r)
x ← A[r]
i ← p-1
for j ← p  to r-1
        do if A[j] ≤ x
                then i ← i + 1
                        exchange A[i] ↔ A[j]
exchange A[i+1] ↔ A[r]
return i+1
```

Running Time $= O(n)$

Exercise

- Illustrate the operation of "`PARTITION()`" on the array of A={2, 8, 7, 1, 3, 5, 6, 4}.

Exercise




```
PARTITION(A, p, r)
    x ← A[r]
    i ← p-1
    for j ← p  to r-1
        do if A[j] ≤ x
            then i ← i + 1
                exchange A[i] ↔ A[j]
    exchange A[i+1] ↔ A[r]
    return i+1
```

Exercise

(f)

p		i		j			r
2	1	3	8	7	5	6	4

(g)

p		i				j	r
2	1	3	8	7	5	6	4

(h)

p		i				j	r
2	1	3	8	7	5	6	4

(i)

p		i					r
2	1	3	4	7	5	6	8

```
PARTITION(A, p, r)
   x ← A[r]
   i ← p-1
   for j ← p  to r-1
        do if A[j] ≤ x
             then i ← i + 1
                  exchange A[i] ↔ A[j]
   exchange A[i+1] ↔ A[r]
   return i+1
```

Pseudocode for quicksort

- QUICKSORT (A, p, r)

  if $p < r$

  then $q \leftarrow$ PARTITION(A, p, r)

  QUICKSORT $(A, p, q\text{-}1)$

  QUICKSORT $(A, q+1, r)$

Initial call: QUICKSORT $(A, 1, n)$

Correctness

- Loop invariant :
  - All entries in $A[p..i]$ are ≤ *pivot*.
  - All entries in $A[i+1 .. j-1]$ are > *pivot*.
  - $A[r]$ = *pivot*

- Initialization : True, because $A[p..i]$, $A[i+1..j-1]$ are empty.

  (Prior to the first iteration of the loop, $i=p-1$, and $j=p$.)

Correctness

- Maintenance : While the loop is running, if $A[j] \leq$ *pivot*, then $A[j]$ and $A[i+1]$ are swapped and then i and j are incremented. If $A[j] >$ *pivot*, then increment only j.
- Termination : When the loop terminates, $j = r$, so elements in A are partitioned into one of the three cases.

- Running time : $\Theta(n)$

Worst-case of quicksort

- When the partitioning routine produces one subproblem with n-1 elements and one with 0 element.
  - input is sorted in ascending or descending order

Partitioning cost

$$
\begin{aligned}
T(n) &= T(n-1) + T(0) + \Theta(n) \\
&= T(n-1) + \Theta(1) + \Theta(n) \\
&= T(n-1) + \Theta(n) \\
&= \Theta(n^2)
\end{aligned}
$$

Best-case of quicksort

- When the partitioning routine produces two subproblems, each of size no more than $n/2$.

Partitioning cost

$$
\begin{aligned}
T(n) &\le 2T(\frac{n}{2}) + \Theta(n) \\
&= O(n \lg n)
\end{aligned}
$$

This is the Case 2 of the master theorem.

Average-case of Quicksort

- Average-case running time of quicksort is much closer to best case than to the worst case.
- Book discusses two solutions for average-case analysis:
  - Randomize the input array – Intuitive analysis
  - Randomized version of quicksort (*Pick a random pivot element*) – formal analysis

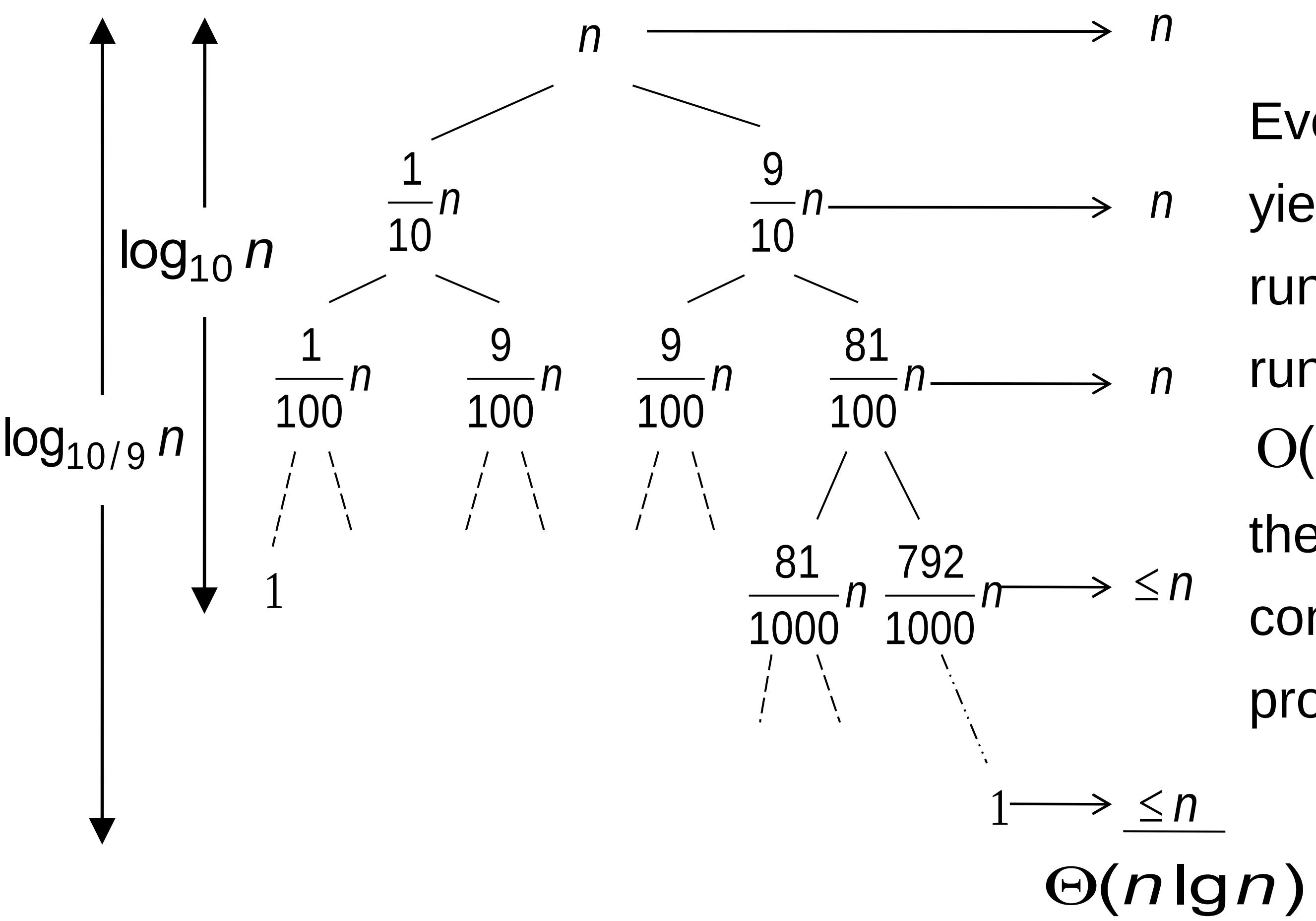
Intuition : Balanced(?) partitioning

- First, a more intuitive explanation/example:
  - Suppose that *partition*() always produces a 9-to-1 split. This looks quite unbalanced!
  - The recurrence is thus:

    $T(n) \leq T(9n/10) + T(n/10) + cn$

  - *How deep will the recursion go?* (draw it)

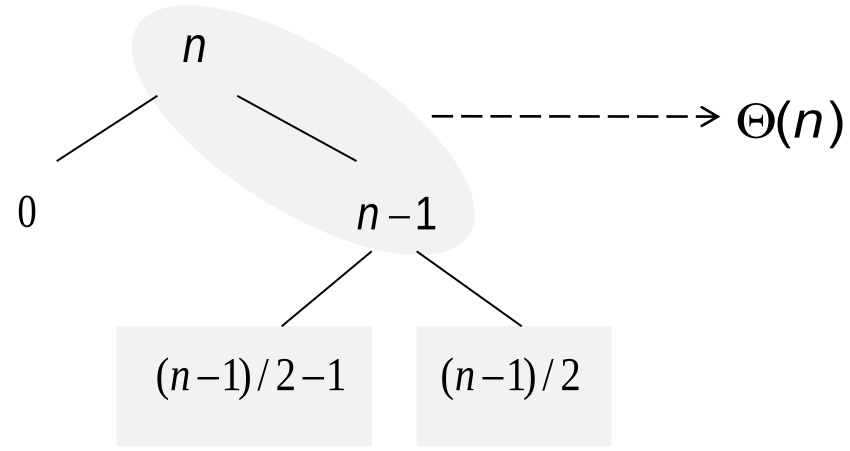
9-to-1 proportional split of quicksort



Even a 99-to-1 split yields an $O(n \lg n)$ running time. The running time is $O(n \lg n)$ whenever the split has constant proportionality.

Intuition for the average case

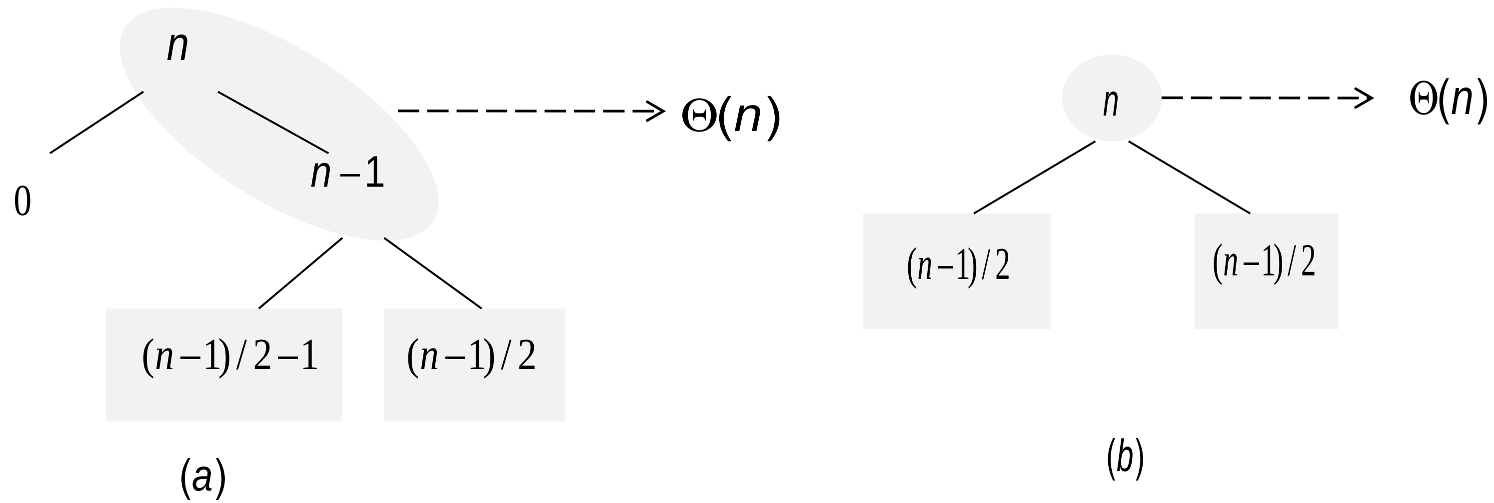
- Intuitively, a real-life run of quicksort will produce a mix of "bad" and "good" splits
  - Randomly distributed among the recursion tree
  - Pretend for intuition that they alternate between best-case [(*n*-1)/2 : (*n*-1)/2] and worst-case [*n*-1 : 0]



Intuition for the average case

– *What happens if we bad-split root node, then good-split the resulting size (n-1) node?*

- We end up with three subarrays, size 0, (*n*-1)/2 -1, and (*n*-1)/2
- Combined cost of splits = **Θ**(*n*) + **Θ**(*n* -1) = **Θ**(*n*)
- No worse than if we had good-split alone!



Randomization

- If input array *A* is almost or already sorted, choosing the last element as a pivot yields a poor performance.
- Instead, choose a random element as a pivot!
- Randomization of quicksort stops any specific type of array from causing worst-case behavior.

A randomized version of quicksort

```
Randomized-Partition(A,p,r)
    i = random(p,r)
    exchange A[r] with A[i]
    return Partition(A,p,r)
```

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

Analyzing Quicksort: Average Case

- For simplicity, assume:
  - All inputs are distinct (no repeats)
  - **`Randomized-partition()`** procedure
    - partition around a random element from the subarray.
    - all splits (0:n-1, 1:n-2, 2:n-3, … , n-1:0) are equally likely to happen.
- *What is the probability of a particular split happening?*
- Answer: 1/n

Analyzing Quicksort: Average Case

- So partition generates splits
  (0:n-1, 1:n-2, 2:n-3, … , n-2:1, n-1:0)
  each with probability 1/n
- If $T(n)$ is the expected running time,

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

- *What is each term under the summation for?*
- *What is the $\Theta(n)$ term for?*

Analyzing Quicksort: Average Case

- So…

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

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

Analyzing Quicksort: Average Case

- We can solve this recurrence using the substitution method.
  - Guess the answer
  - Assume that the inductive hypothesis holds
  - Substitute it in for some value < n
  - Prove that it follows for n

Analyzing Quicksort: Average Case

- We can solve this recurrence using the dreaded substitution method.
  - Guess the answer
    - $T(n) = O(n \lg n)$
  - Assume that the inductive hypothesis holds
    - $T(n) \leq an \lg n + b$ for some constants a and b
  - Substitute it in for some value < n
    - The value k in the recurrence
  - Prove that it follows for n
    - Grind through it…

Analyzing Quicksort: Average Case

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

The recurrence to be solved

$$\le \frac{2}{n}\sum_{k=0}^{n-1} (ak\lg k + b) + \Theta(n)$$

Plug in inductive hypothesis

$$\le \frac{2}{n}\left[b + \sum_{k=1}^{n-1} (ak\lg k + b)\right] + \Theta(n)$$

Expand out the k=0 case

$$= \frac{2}{n}\sum_{k=1}^{n-1} (ak\lg k + b) + \frac{2b}{n} + \Theta(n)$$

2b/n is just a constant, so fold it into Θ(n)

$$= \frac{2}{n}\sum_{k=1}^{n-1} (ak\lg k + b) + \Theta(n)$$

Note: leaving the same recurrence as the book

Analyzing Quicksort: Average Case

$$T(n) = \frac{2}{n}\sum_{k=1}^{n-1}\left(ak\lg k + b\right) + \Theta(n)$$

The recurrence to be solved

$$= \frac{2}{n}\sum_{k=1}^{n-1} ak\lg k + \frac{2}{n}\sum_{k=1}^{n-1} b + \Theta(n)$$

Distribute the summation

$$= \frac{2a}{n}\sum_{k=1}^{n-1} k\lg k + \frac{2b}{n}(n-1) + \Theta(n)$$

Evaluate the summation:
b+b+…+b = b (n-1)

$$\le \frac{2a}{n}\boxed{\sum_{k=1}^{n-1} k\lg k} + 2b + \Theta(n)$$

Since n-1<n, 2b(n-1)/n < 2b

This summation gets its own set of slides later

Analyzing Quicksort: Average Case

$$T(n) \le \frac{2a}{n}\sum_{k=1}^{n-1} k \lg k + 2b + \Theta(n)$$

The recurrence to be solved

$$\le \frac{2a}{n}\left(\frac{1}{2}n^2 \lg n - \frac{1}{8}n^2\right) + 2b + \Theta(n)$$

We'll prove this later

$$= an \lg n - \frac{a}{4}n + 2b + \Theta(n)$$

Distribute the (2a/n) term

$$= an \lg n + b + \left(\Theta(n) + b - \frac{a}{4}n\right)$$

Remember, our goal is to get $T(n) \le an \lg n + b$

$$\le an \lg n + b$$

Pick a large enough that an/4 dominates $\Theta(n)+b$

Analyzing Quicksort: Average Case

- So $T(n) \le an \lg n + b$ for certain a and b
  - Thus the induction holds
  - Thus $T(n) = O(n \lg n)$
  - Thus quicksort runs in $O(n \lg n)$ time on average (phew!)
- Oh yeah, the summation…

The Key Summation

$$\sum_{k=1}^{n-1} k \lg k = \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg k + \sum_{k=\lceil n/2 \rceil}^{n-1} k \lg k$$

Split the summation for a tighter bound

$$\leq \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg k + \sum_{k=\lceil n/2 \rceil}^{n-1} k \lg n$$

The* lg *k* *in the second term is bounded by* lg *n

$$= \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

Move the* lg *n* *outside the summation

The Key Summation

$$\sum_{k=1}^{n-1} k \lg k \le \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

The summation bound so far

$$\le \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg(n/2) + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

The lg k in the first term is bounded by lg n/2

$$= \sum_{k=1}^{\lceil n/2 \rceil - 1} k(\lg n - 1) + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

lg *n*/2 = lg *n* - 1

$$= (\lg n - 1) \sum_{k=1}^{\lceil n/2 \rceil - 1} k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

Move (lg n - 1) outside the summation

The Key Summation

$$\sum_{k=1}^{n-1} k \lg k \le (\lg n - 1) \sum_{k=1}^{\lceil n/2 \rceil - 1} k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

The summation bound so far

$$= \lg n \sum_{k=1}^{\lceil n/2 \rceil - 1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

Distribute the (lg n - 1)

$$= \lg n \sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k$$

The summations overlap in range; combine them

$$= \lg n \left(\frac{(n-1)(n)}{2} \right) - \sum_{k=1}^{\lceil n/2 \rceil - 1} k$$

The Guassian series

The Key Summation

$$\sum_{k=1}^{n-1} k \lg k \le \left(\frac{(n-1)(n)}{2}\right)\lg n - \sum_{k=1}^{\lceil n/2 \rceil - 1} k$$

The summation bound so far

$$\le \frac{1}{2}[n(n-1)]\lg n - \sum_{k=1}^{n/2-1} k$$

Rearrange first term, place upper bound on second

$$\le \frac{1}{2}[n(n-1)]\lg n - \frac{1}{2}\left(\frac{n}{2}\right)\left(\frac{n}{2}-1\right)$$

X Guassian series

$$\le \frac{1}{2}\left(n^2 \lg n - n \lg n\right) - \frac{1}{8}n^2 + \frac{n}{4}$$

Multiply it all out

The Key Summation

$$\sum_{k=1}^{n-1} k \lg k \le \frac{1}{2}\left(n^2 \lg n - n \lg n\right) - \frac{1}{8}n^2 + \frac{n}{4}$$

$$\le \frac{1}{2}n^2 \lg n - \frac{1}{8}n^2 \text{ when } n \ge 2$$

Done !!