Quicksort

Quicksort

- Unlike mergesort, which divides elements according to their position in array, quicksort divides according to their value.
- A partition is an arrangement of the array's elements so that all the elements to the left of some element *A[s] are less than* or equal to *A[s], and all the elements to the right of A[s] are greater than or equal* to it:

$$\underbrace{A[0]\ldots A[s-1]}_{\text{all are } \leq A[s]} \quad A[s] \quad \underbrace{A[s+1]\ldots A[n-1]}_{\text{all are } \geq A[s]}$$

e.g.

4	8	6	9	5	7	3
4	3	5	6	9	7	8

s= 3

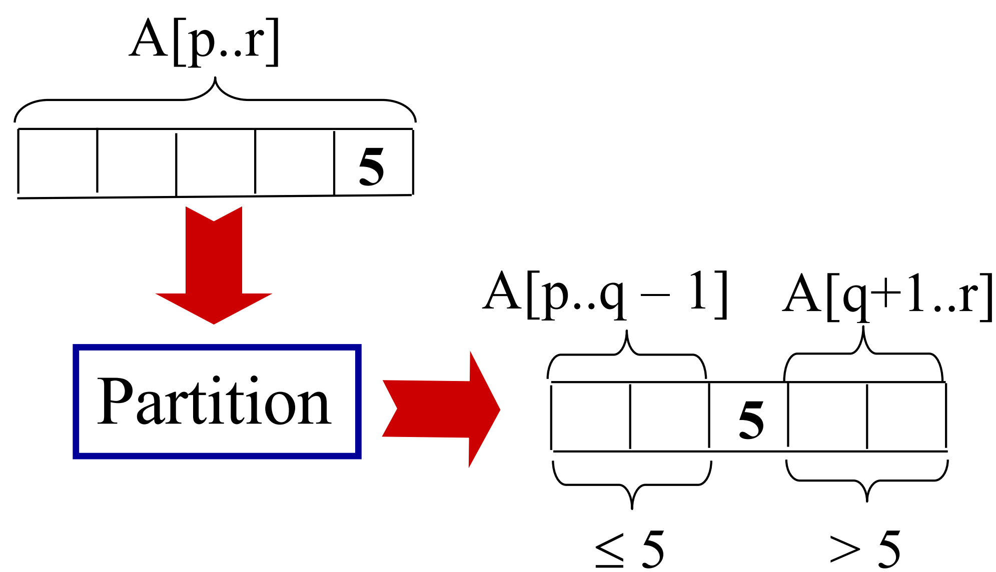
Quicksort (cont.)

- After a partition is achieved, *A[s] will be in its final position in the* sorted array, and we can continue sorting the two subarrays to the left and to the right of *A[s] independently (e.g., by the same method). E.g.,*

4 3 5 6 9 7 8

Final sorted: 3 4 5 6 7 8 9

```
Quicksort(A, p, r)
    if p < r then
        q := Partition(A, p, r);
        Quicksort(A, p, q – 1);
        Quicksort(A, q + 1, r)
    fi
```



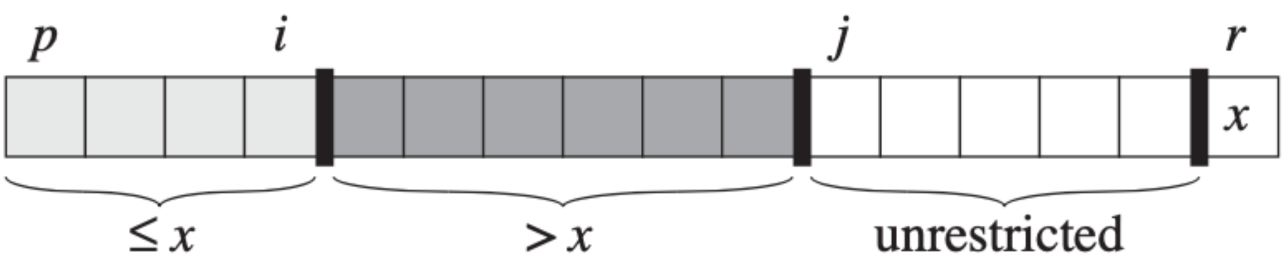
Partitioning

```
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;
            A[i] ↔ A[j]
        fi
    od;
    A[i + 1] ↔ A[r];
    return i + 1
```

With j scan left to right, and put behind i the elements ≤ pivot.

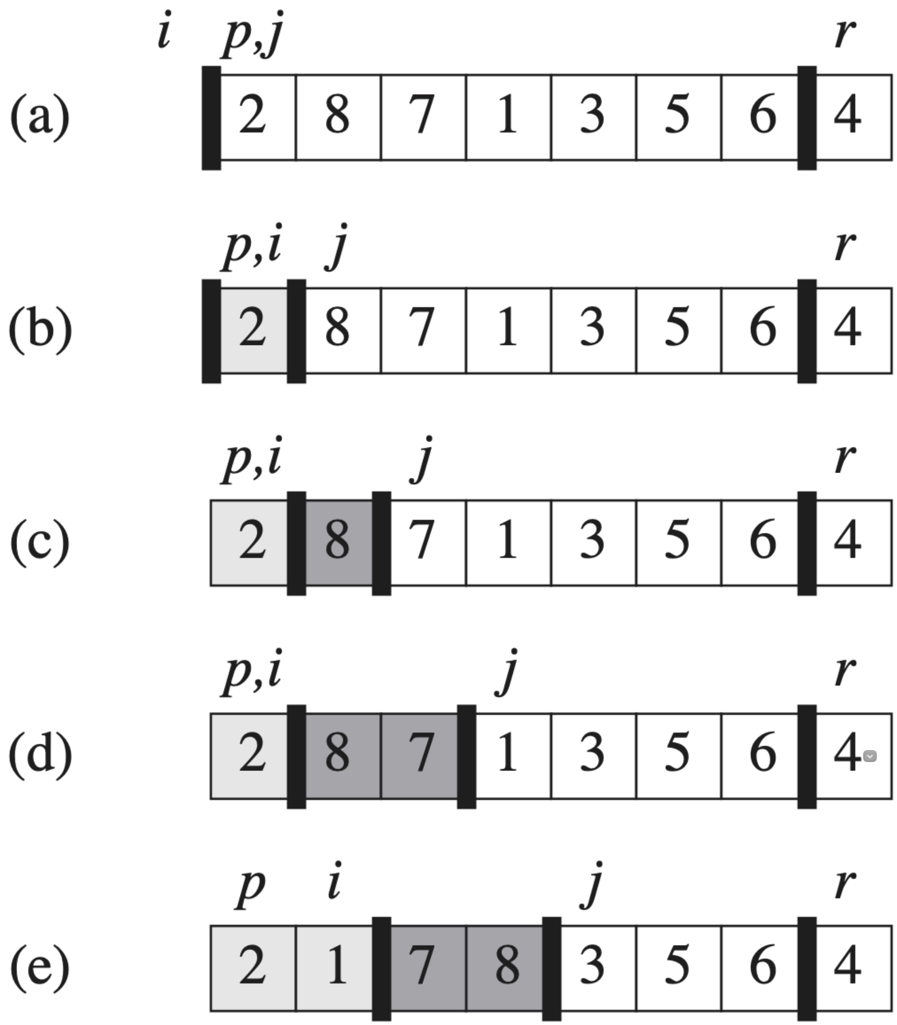
A[0…i]: elements ≤ pivot

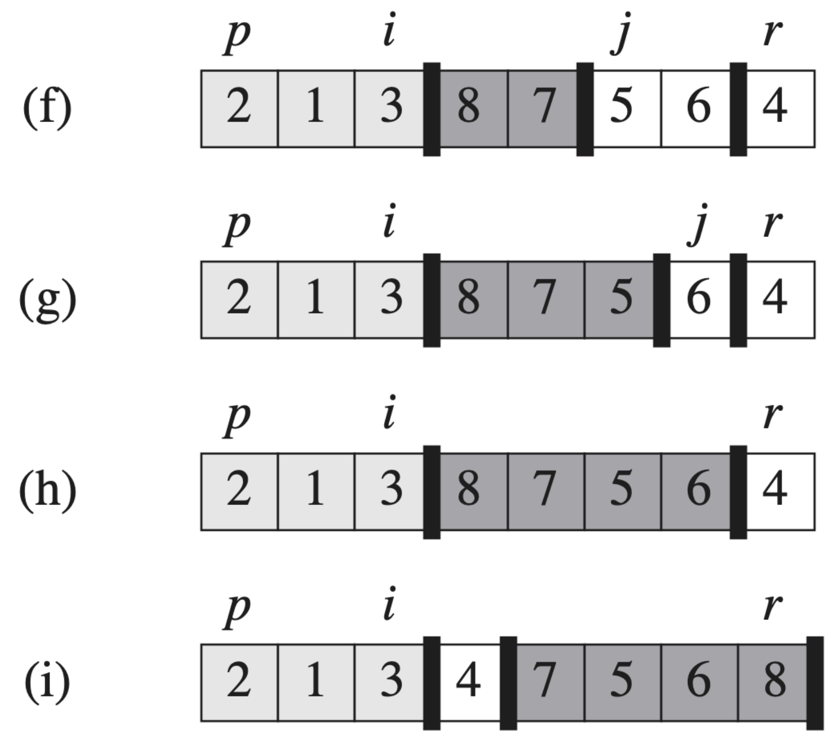
Time: $\Theta(n)$



Example

if A[j] $\leq$ x **then**
i := i + 1;
A[i] $\leftrightarrow$ A[j]
fi





Array entry A[r] is the pivot x. Lightly shaded array elements are all in the first partition with values no greater than x. **Heavily shaded** elements are in the second partition with values greater than x. The unshaded elements have not yet been put in one of the first two partitions, and the final white pivot x.

Time Analysis

Worst-Case: Always get a completely unbalanced partition.

$$
\begin{aligned}
T(n) &= T(n-1) + \Theta(n) \\
&= \sum_{k=1}^{n} \Theta(k) \\
&= \Theta\left(\sum_{k=1}^{n} k\right) \\
&= \Theta\left(n^2\right)
\end{aligned}
$$

When can this happen?

Time Analysis

Worst-Case: Always get a completely unbalanced partition.

$$
\begin{aligned}
T(n) &= T(n-1) + \Theta(n) \\
&= \sum_{k=1}^{n} \Theta(k) \\
&= \Theta\left(\sum_{k=1}^{n} k\right) \\
&= \Theta\left(n^2\right)
\end{aligned}
$$

When a list is sorted, reverse sorted, or all elements are equal!

What is the Best Case scenario?

Time Analysis

Worst-Case: Always get a completely unbalanced partition.

$$
\begin{aligned}
T(n) &= T(n-1) + \Theta(n) \\
&= \sum_{k=1}^{n} \Theta(k) \\
&= \Theta\left(\sum_{k=1}^{n} k \right) \\
&= \Theta\left(n^2\right)
\end{aligned}
$$

When a list is sorted, reverse sorted, or all elements are equal!

Best-Case: Always get a perfectly balanced partition.

$$T(n) \leq 2T(n/2) + \Theta(n) = \Theta(n \lg n)$$

Average-Case Time Analysis

What happens if we get sort-of-balanced partitions, e.g., something like:

$$T(n) \le T(9n/10) + T(n/10) + \Theta(n)?$$

Still get $\Theta(n \lg n)$!!

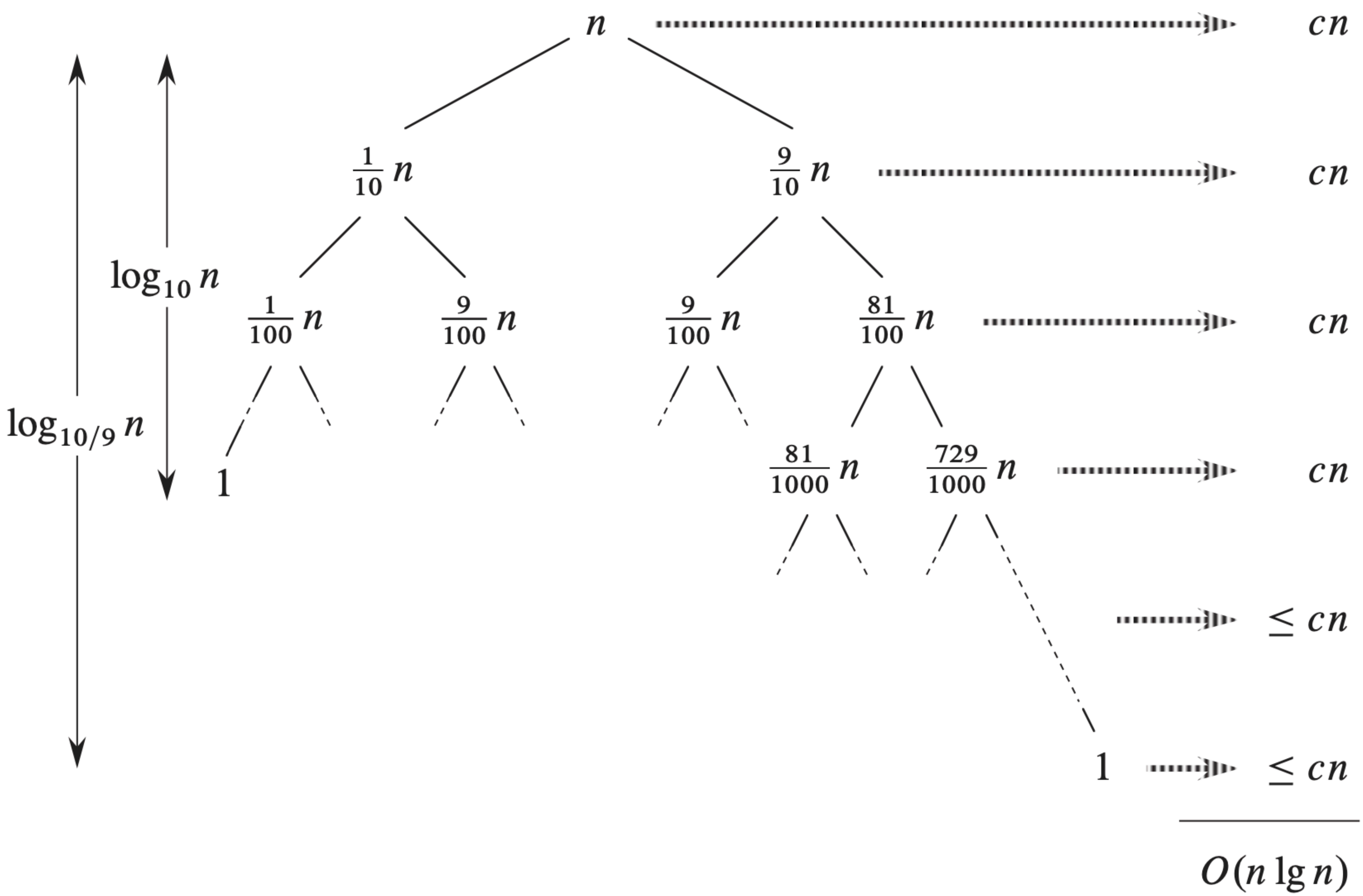
Intuition: Can divide n by $c > 1$ only $\Theta(\lg n)$ times before getting 1.

$$\left.\begin{array}{c} n \\ \downarrow \\ n/c \\ \downarrow \\ n/c^2 \\ \downarrow \\ \vdots \\ \downarrow \\ 1 \end{array}\right\} \text{roughly, } \log_c n \text{ steps}$$

Intuition holds even if c is very close to 1, e.g., 100/99.

(**Remember:** Different base logs are related by a constant.)

Observation through Recursion Tree



Randomized Version

Want to make running time independent of input ordering.

```
Randomized-Partition(A, p, r)
    i := Random(p, r);
    A[r] ↔ A[i];
    Partition(A, p, r)
```

```
Randomized-Quicksort(A, p, r)
    if p < r then
        q := Randomized-Partition(A, p, r);
        Randomized-Quicksort(A, p, q – 1);
        Randomized-Quicksort(A, q + 1, r)
    fi
```

Average Case Analysis of Randomized Quicksort

- Let RV X = number of comparisons over all calls to Partition.
- Suffices to compute E[X]. **Why?**
- Note that each pair of elements is compared at most once. Why?

Average Case Analysis of Randomized Quicksort

- Let RV X = number of comparisons over all calls to Partition.

- Suffices to compute E[X]. **Why?**

- Note that each pair of elements is compared at most once. Why?

  Elements are compared only to the pivot element and, after a particular call of PARTITION finishes, the pivot element used in that call is never again compared to any other elements.

Average Case Analysis of Randomized Quicksort

- Let RV X = number of comparisons over all calls to Partition.

- Suffices to compute $E[X]$. **Why?**

Notation:

- Let $z_1, z_2, \ldots, z_n$ denote the list items (in sorted order).
- Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$.

Let RV $X_{ij} = \begin{cases} 1 & \text{if } z_i \text{ is compared with } z_j \\ 0 & \text{otherwise} \end{cases}$ This is an example of an **indicator random variable**.

Thus, $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$. Considering all pairs

Analysis (Continued)

We have:

$$E[X] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right]$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P[z_i \text{ is compared to } z_j]$$

Note:

$E[X_{ij}] = 0 \cdot P[X_{ij}=0] + 1 \cdot P[X_{ij}=1]$

$= P[X_{ij}=1]$

This is a nice property of indicator RVs.

So, all we need to do is to compute $P[z_i$ is compared to $z_j]$.

Analysis (Continued)

Note: z_i and z_j are compared iff the first element to be chosen as a pivot from Z_{ij} is either z_i or z_j.

Proof? Hint: compared only with pivot.

Once a pivot x is chosen (randomly and independently), all numbers are compared with it → two partitions with x in the middle.

A number from one partition is never compared with a number from the other partition.

Being the smallest and the largest, z_i and z_j must be in two separate partitions right after the first call of Partition().

Hence, they were compared only if one of them was the first pivot.

Analysis (Continued)

Note: z_i and z_j are compared iff the first element to be chosen as a pivot from Z_{ij} is either z_i or z_j.

Hence,

$$\begin{aligned}
\mathrm{P}[z_i \text{ is compared to } z_j] &= \mathrm{P}[z_i \text{ or } z_j \text{ is first pivot from } Z_{ij}] \\
&= \mathrm{P}[z_i \text{ is first pivot from } Z_{ij}] \\
&\quad + \mathrm{P}[z_j \text{ is first pivot from } Z_{ij}] \\
&= \frac{1}{j-i+1} + \frac{1}{j-i+1} \\
&= \frac{2}{j-i+1}
\end{aligned}$$

Analysis (Continued)

Therefore,

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

$$= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1}$$

$$< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k}$$

$$= \sum_{i=1}^{n-1} O(\lg n) \longleftarrow \text{Think from Integration}$$

$$= \boxed{O(n \lg n).}$$

Analysis (Continued)

The entire analysis can be made much easier if you consider where your expected partition is each time.

Analysis

What is the complexity of Randomized Quicksort in the worst case?