Quicksort

交通大學 資訊工程系

Quicksort

- Worst case running time $\Theta(n^2)$.
- Expected running time $\Theta(n \lg n)$.
  - The constant factors hidden in the $\Theta(n \lg n)$ notation are quite small.
- Advantage
  - Sorting in place.
- Based on the *divide-and-conquer* paradigm.
  - Divide: Partition $A[p \,..\, r]$ into $A[p \,..\, q-1]$ and $A[q+1 \,..\, r]$, such that $A[p \,..\, q-1]$ is less than or equal to $A[q]$ and $A[q]$ is less than or equal to $A[q+1 \,..\, r]$.

Quicksort

- **Conquer: Sort *A*[*p* . . *q-1*] and *A*[*q+1* . . *r*] by recursive calls to Quicksort.**
- **Combine: No work is needed to combine them.**

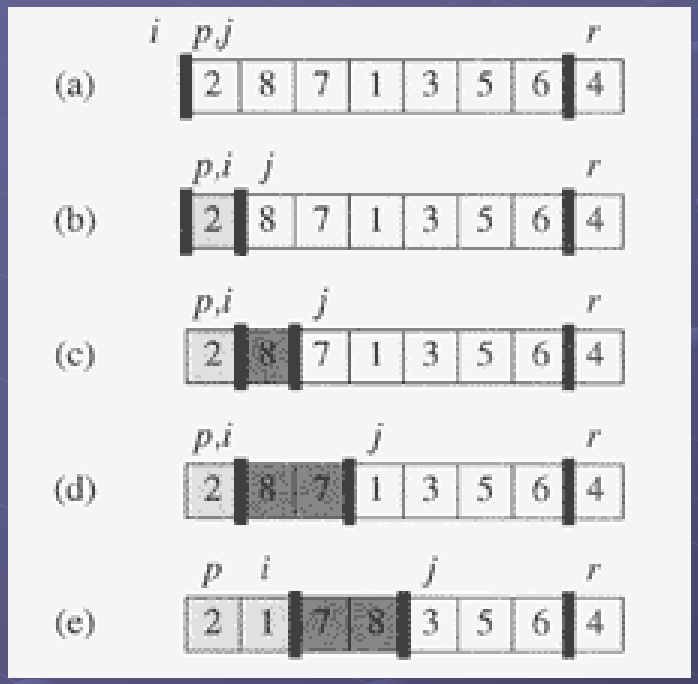
```
QUICKSORT(A, p, r)
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)
```

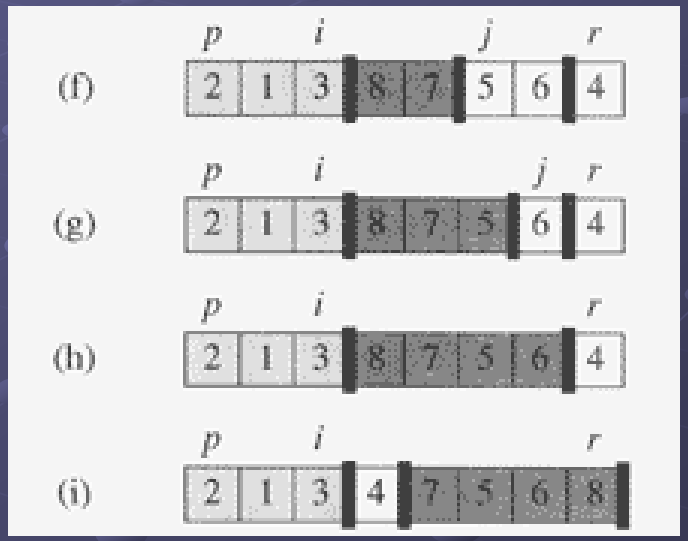
Quicksort

- **Partitioning the array**
  - ***Pivot*** **point: $x = A[r]$.**

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4       if A[j] ≤ x
5            i = i + 1
6            exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

Quicksort

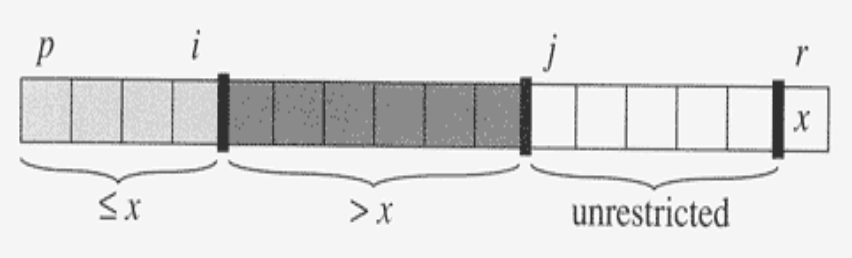

(a)
i p,j r
2 8 7 1 3 5 6 4
(b)
p,i j r
2 8 7 1 3 5 6 4
(c)
p,i j r
2 8 7 1 3 5 6 4
(d)
p,i j r
2 8 7 1 3 5 6 4
(e)
p i j r
2 1 7 8 3 5 6 4


(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 r
2 1 3 8 7 5 6 4
(i)
p i r
2 1 3 4 7 5 6 8

Quicksort

Loop invariant

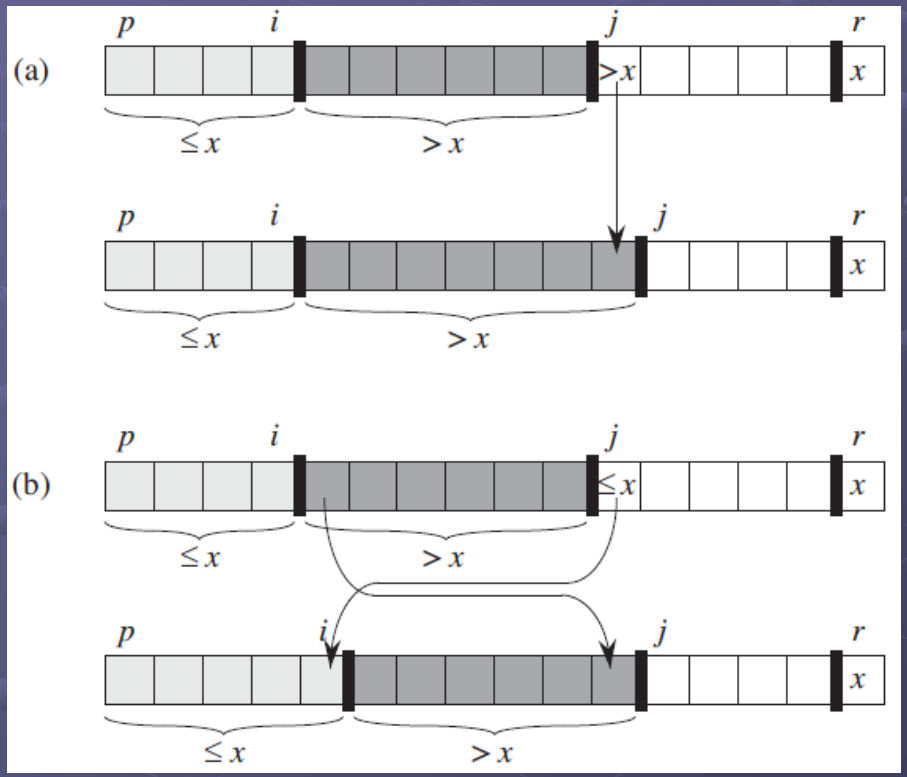
1. if $p \le k \le i$, then $A[k] \le x$.
2. if $i + 1 \le k \le j - 1$, then $A[k] > x$.
3. if $k = r$, then $A[k] = x$.



- Initialization: $i = p - 1$ and $j = p$. No values between p and i, and no values between $i + 1$ and $j - 1$.

Quicksort

- Maintenance



- Termination: At termination, $j = r$.

Quicksort

☞ **Hoare partition**

```
HOARE-PARTITION(A, p, r)
 1  x = A[p]
 2  i = p − 1
 3  j = r + 1
 4  while TRUE
 5       repeat
 6            j = j − 1
 7       until A[j] ≤ x
 8       repeat
 9            i = i + 1
10       until A[i] ≥ x
11       if i < j
12            exchange A[i] with A[j]
13       else return j
```

Performance

- **The running time of quicksort depends on**
  - **Whether the partitioning is balanced or unbalanced.**
  - **Which elements are used for partitioning.**
- **Worst-case partitioning**
  - **One subproblem with n–1 elements and one with 0 elements.**
  - **The partitioning costs $\Theta(n)$ time.**

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

Performance

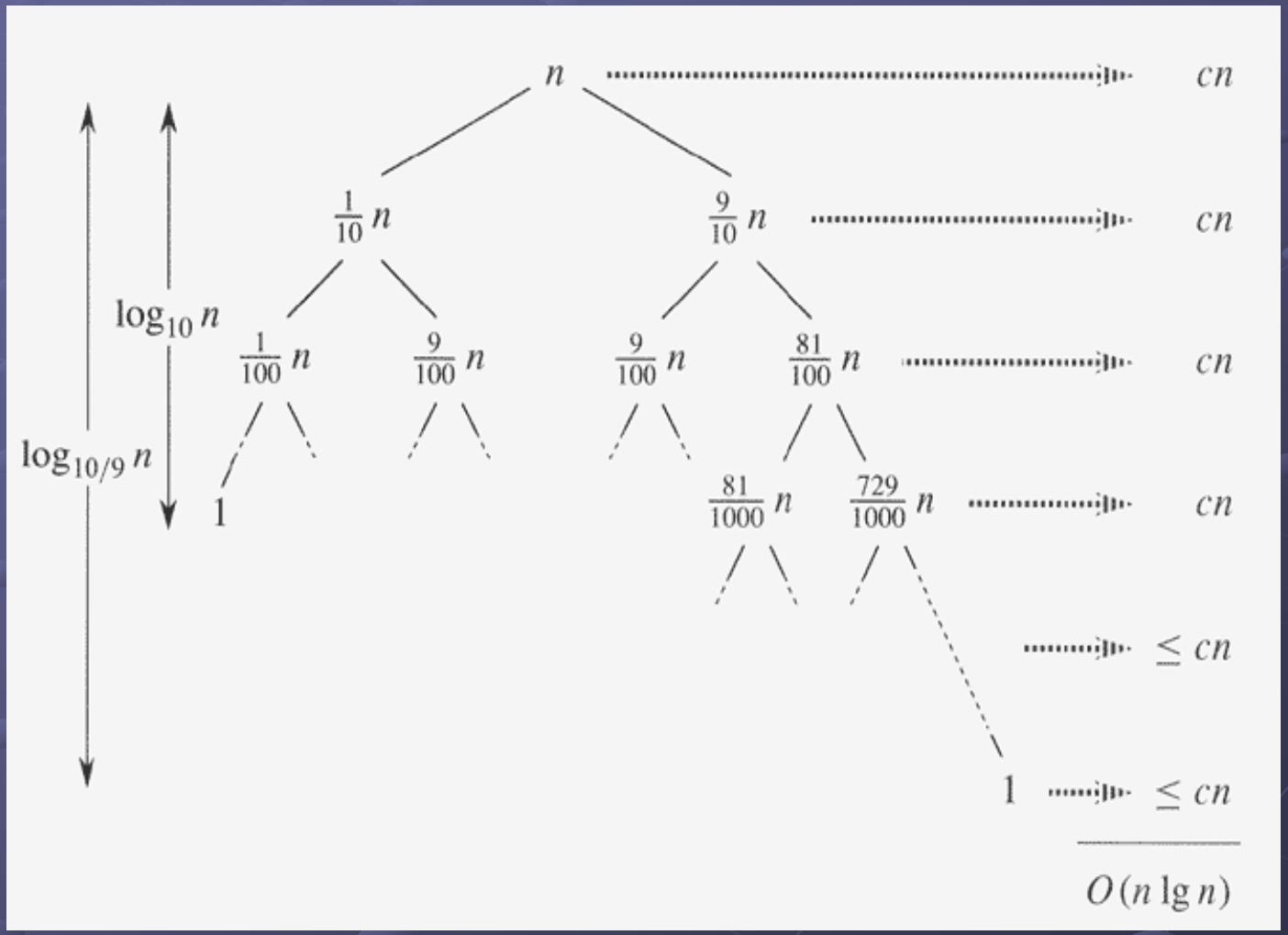
- **Best-case partitioning**
  - **One subproblem is of size $\lfloor n/2 \rfloor$ and one of size $\lfloor n/2 \rfloor$ - 1.**

$$T(n) \le 2T(n/2) + \Theta(n)$$
$$= \Theta(n \lg n) \quad \text{(By case 2 of master theorem)}$$

- **Balanced partitioning**
  - **The average case running time is much closer to the best case than to the worst case.**
  - **For example: 9-to-1 proportional split.**

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

Performance

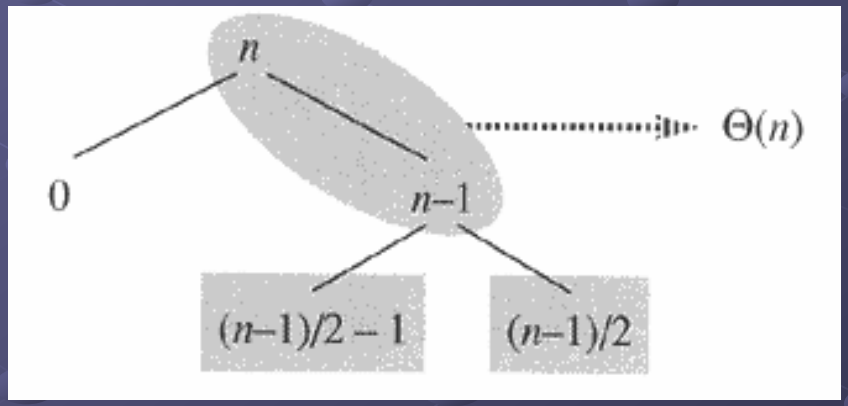


Performance

- Every level of the tree has cost cn, until a boundary condition is reached at depth $log_{10}n = \Theta(lg\ n)$, and the levels have cost at most cn.
- The recursion terminates at depth $log_{10/9}n = \Theta(lg\ n)$.
- The total cost is $\Theta(n\ lg\ n)$.
- Even a 99-to-1 split yields an $\Theta(n\ lg\ n)$ Running time.
  - Any split of constant proportionality yields a recursion tree of depth $\Theta(lg\ n)$.
  - The cost at each level is $O(n)$.

Performance

- **Intuition for the average case**
  - **The behavior of quicksort is determined by the relative ordering of the values in the input array.**
  - **In the average case, PARTITION produces a mix of "good" and "bad" splits.**
  - **Suppose for the sake of intuition, the good and bad splits alternate levels in the tree.**

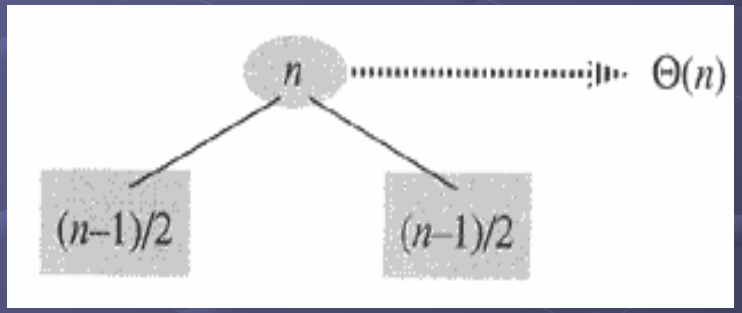


Performance

- The combination of the bad split followed by the good split produces three subarrys of size 0, $(n-1)/2-1$, and $(n-1)/2$ at a combined partitioning cost of

  $$\Theta(n) + \Theta(n-1) = \Theta(n).$$

- The above running time is like the running time for good splits alone.



A Randomized Version of Quicksort

- **Assumption**
  - In exploring the average-case behavior of quicksort.
  - All permutations of the input numbers are equally likely.
- **Random sampling**
  - Instead of always using $A[r]$ as the pivot, we will use a randomly chosen element from the subarray $A[p \ldots r]$.
  - Because the pivot element is randomly chosen, we expect the split of the input array to be reasonably well balanced on average.

A Randomized Version of Quicksort

```
RANDOMIZED-PARTITION(A, p, r)
1  i = RANDOM(p, r)
2  exchange A[r] with A[i]
3  return PARTITION(A, p, r)
```

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if p < r
2      q = RANDOMIZED-PARTITION(A, p, r)
3      RANDOMIZED-QUICKSORT(A, p, q − 1)
4      RANDOMIZED-QUICKSORT(A, q + 1, r)
```

Analysis of Quicksort

- **Worst-Case analysis – Using substitution method**

$$T(n) = \max_{0 \le q \le n-1} (T(q) + T(n-q-1)) + \Theta(n)$$

Guess $T(n) \le cn^2$ for some constant c.

$$T(n) \le \max_{0 \le q \le n-1} (cq^2 + c(n-q-1)^2) + \Theta(n)$$

$$= c \max_{0 \le q \le n-1} (q^2 + (n-q-1)^2) + \Theta(n)$$

$$\le cn^2 - 2c(n-1) + \Theta(n)$$

$$\le cn^2$$

Analysis of Quicksort

- **Expected running time**
  - **RANDOMIZED-PARTITION puts any constant fraction of the elements on one side of the partition.**
    - **The recursion tree has depth $\Theta(lg\ n)$, and $O(n)$ work is performed at each level.**
    - **Even new levels with the most unbalanced split between these levels.**
  - **The running time of QUICKSORT is dominated by the time spent in the PARTITION procedure.**

Analysis of Quicksort

- **Lemma 7.1**
  - **Let X be the number of comparisons performed in line 4 of PARTITION over the entire execution of QUICKSORT on an *n*-element array. Then the running time of QUICKSORT is O(*n* + X).**
  - **Proof:**

$$X_{ij} = \mathrm{I}\{z_i \text{ is compared to } z_j\}$$

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

Analysis of Quicksort

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

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

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

Analysis of Quicksort

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

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

Analysis of Quicksort

$$
\begin{aligned}
\mathrm{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\
&< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \\
&= \sum_{i=1}^{n-1} O(\lg n) \\
&= O(n \lg n) \ .
\end{aligned}
$$