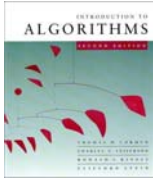


CS 3343 -- Fall 2007




Quicksort

Carola Wenk

Slides courtesy of Charles Leiserson with small changes by Carola Wenk


9/27/07 CS 3343 Analysis of Algorithms 1



Quicksort

- Proposed by C.A.R. Hoare in 1962.
- Divide-and-conquer algorithm.
- Sorts “in place” (like insertion sort, but not like merge sort).
- Very practical (with tuning).

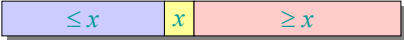
9/27/07 CS 3343 Analysis of Algorithms 2



Divide and conquer

Quicksort an n -element array:


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



- Conquer:** Recursively sort the two subarrays.
- Combine:** Trivial.

Key: Linear-time partitioning subroutine.

9/27/07 CS 3343 Analysis of Algorithms 3



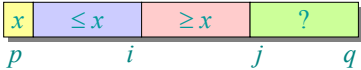
Partitioning subroutine

```


PARTITION( $A, p, q$ ) //  $A[p..q]$ 
 $x \leftarrow A[p]$  // pivot =  $A[p]$ 
 $i \leftarrow p$ 
for  $j \leftarrow p+1$  to  $q$ 
  do if  $A[j] \leq x$ 
    then  $i \leftarrow i+1$ 
        exchange  $A[i] \leftrightarrow A[j]$ 
exchange  $A[p] \leftrightarrow A[i]$ 
return  $i$ 

```

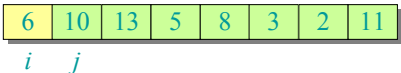
Running time = $O(n)$ for n elements.

Invariant: 


9/27/07 CS 3343 Analysis of Algorithms 4



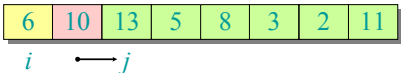
Example of partitioning



9/27/07 CS 3343 Analysis of Algorithms 5



Example of partitioning



9/27/07 CS 3343 Analysis of Algorithms 6

Example of partitioning

Initial array: 6, 10, 13, 5, 8, 3, 2, 11

Pivot: 10

Element 5 is being moved to the position before the pivot.

9/27/07 CS 3343 Analysis of Algorithms 7

Example of partitioning

Array after moving 5: 6, 5, 13, 10, 8, 3, 2, 11

Pivot 10 is in its final position.

9/27/07 CS 3343 Analysis of Algorithms 8

Example of partitioning

Array after moving 5: 6, 5, 13, 10, 8, 3, 2, 11

Pivot 10 is in its final position.

9/27/07 CS 3343 Analysis of Algorithms 9

Example of partitioning

Array after moving 5: 6, 5, 13, 10, 8, 3, 2, 11

Pivot 10 is in its final position.

9/27/07 CS 3343 Analysis of Algorithms 10

Example of partitioning

Array after moving 5: 6, 5, 13, 10, 8, 3, 2, 11

Array after moving 13, 8, 3, 2: 6, 5, 3, 10, 8, 13, 2, 11

Pivot 10 is in its final position.

9/27/07 CS 3343 Analysis of Algorithms 11


Example of partitioning

Array after moving 5: 6, 5, 13, 10, 8, 3, 2, 11

Array after moving 13, 8, 3, 2: 6, 5, 3, 10, 8, 13, 2, 11

Pivot 10 is in its final position.

9/27/07 CS 3343 Analysis of Algorithms 12




Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11

$\longleftarrow i$ $j \longrightarrow$

9/27/07 CS 3343 Analysis of Algorithms 13




Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11

i $\longleftarrow j$

9/27/07 CS 3343 Analysis of Algorithms 14




Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11

i $\longrightarrow j$

9/27/07 CS 3343 Analysis of Algorithms 15




Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11
2	5	3	6	8	13	10	11

i

9/27/07 CS 3343 Analysis of Algorithms 16




Pseudocode for quicksort

```

QUICKSORT( $A, p, r$ )
  if  $p < r$ 
    then  $q \leftarrow$  PARTITION( $A, p, r$ )
         QUICKSORT( $A, p, q-1$ )
         QUICKSORT( $A, q+1, r$ )

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

9/27/07 CS 3343 Analysis of Algorithms 17



Analysis of quicksort

- Assume all input elements are distinct.
- In practice, there are better partitioning algorithms for when duplicate input elements may exist.
- Let $T(n)$ = worst-case running time on an array of n elements.

9/27/07 CS 3343 Analysis of Algorithms 18



Worst-case of quicksort

```

QUICKSORT(A, p, r)
  if p < r
    then q ← PARTITION(A, p, r)
         QUICKSORT(A, p, q-1)
         QUICKSORT(A, q+1, r)

```

- Input sorted or reverse sorted.
- Partition around min or max element.
- One side of partition always has no elements.

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



Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + cn$$



Worst-case recursion tree

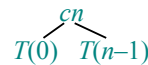
$$T(n) = T(0) + T(n-1) + cn$$

$T(n)$



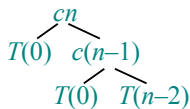
Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + cn$$



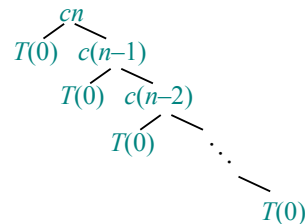
Worst-case recursion tree


$$T(n) = T(0) + T(n-1) + cn$$



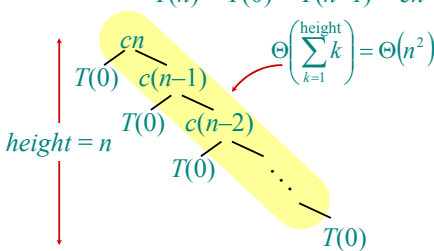
Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + cn$$





Worst-case recursion tree


$$T(n) = T(0) + T(n-1) + cn$$


$$\Theta\left(\sum_{k=1}^{\text{height}} k\right) = \Theta(n^2)$$

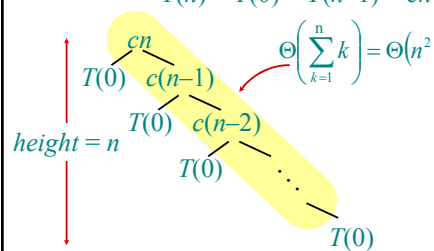
height = n

$T(0)$

9/27/07 CS 3343 Analysis of Algorithms 25



Worst-case recursion tree


$$T(n) = T(0) + T(n-1) + cn$$


$$\Theta\left(\sum_{k=1}^n k\right) = \Theta(n^2)$$

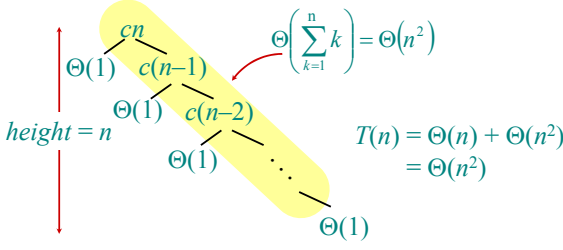
height = n

$T(0)$

9/27/07 CS 3343 Analysis of Algorithms 26



Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + cn$$



$$\Theta\left(\sum_{k=1}^n k\right) = \Theta(n^2)$$

height = n

$\Theta(1)$

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

9/27/07 CS 3343 Analysis of Algorithms 27



Best-case analysis

(For intuition only!)

If we're lucky, PARTITION splits the array evenly:

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


$$= \Theta(n \log n) \quad (\text{same as merge sort})$$

What if the split is always $\frac{1}{10} : \frac{9}{10}$?

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

What is the solution to this recurrence?


9/27/07 CS 3343 Analysis of Algorithms 28



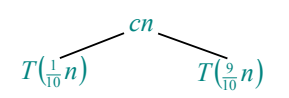
Analysis of "almost-best" case

$$T(n)$$

9/27/07 CS 3343 Analysis of Algorithms 29



Analysis of "almost-best" case



$$T\left(\frac{1}{10}n\right) \quad T\left(\frac{9}{10}n\right)$$

9/27/07 CS 3343 Analysis of Algorithms 30

Analysis of “almost-best” case

cn
 $\frac{1}{10}cn$ $\frac{9}{10}cn$
 $T(\frac{1}{100}n)T(\frac{9}{100}n)$ $T(\frac{9}{100}n)T(\frac{81}{100}n)$

9/27/07 CS 3343 Analysis of Algorithms 31

Analysis of “almost-best” case

cn
 $\frac{1}{10}cn$ $\frac{9}{10}cn$
 $\frac{1}{100}cn$ $\frac{9}{100}cn$ $\frac{9}{100}cn$ $\frac{81}{100}cn$
 $\Theta(1)$ $O(n)$ leaves $\Theta(1)$

9/27/07 CS 3343 Analysis of Algorithms 32

Analysis of “almost-best” case

cn
 $\frac{1}{10}cn$ $\frac{9}{10}cn$
 $\frac{1}{100}cn$ $\frac{9}{100}cn$ $\frac{9}{100}cn$ $\frac{81}{100}cn$
 $\Theta(1)$ $O(n)$ leaves $\Theta(1)$

$\Theta(n \log n)$
 $cn \log_{10} n \leq T(n) \leq cn \log_{10} n + O(n)$

9/27/07 CS 3343 Analysis of Algorithms 33

Quicksort Runtimes

- Best case runtime $T_{\text{best}}(n) \in O(n \log n)$
- Worst case runtime $T_{\text{worst}}(n) \in O(n^2)$
- Worse than mergesort? Why is it called quicksort then?
- Its average runtime $T_{\text{avg}}(n) \in O(n \log n)$
- Better even, the expected runtime of **randomized quicksort** is $O(n \log n)$

9/27/07 CS 3343 Analysis of Algorithms 34

Average Runtime

The **average runtime** $T_{\text{avg}}(n)$ for Quicksort is the average runtime over **all possible inputs** of length n .

- What kind of inputs are there?
- How many inputs are there?

9/27/07 CS 3343 Analysis of Algorithms 35

Average Runtime

- What kind of inputs are there?
 - Do $[1, 2, \dots, n]$ and $[5, 6, \dots, n+5]$ cause different runtimes of Quicksort?
 - No. Therefore only consider all permutations of $[1, 2, \dots, n]$.
- How many inputs are there?
 - There are $n!$ different permutations of $[1, 2, \dots, n]$

9/27/07 CS 3343 Analysis of Algorithms 36



Average Runtime

- Therefore, $T_{\text{avg}}(n)$ has to average the runtimes over all $n!$ different input permutations
 - Disadvantage of considering average runtime:
 - There are still worst-case inputs that will have a $O(n^2)$ runtime
 - Are all inputs really equally likely? That depends on the application
- ⇒ **Better:** Use randomized quicksort

9/27/07

CS 3343 Analysis of Algorithms

37



Randomized quicksort

IDEA: Partition around a *random* element.

- Running time is independent of the input order.
- No assumptions need to be made about the input distribution.
- No specific input elicits the worst-case behavior.
- The worst case is determined only by the output of a random-number generator.

9/27/07

CS 3343 Analysis of Algorithms

38



Randomized quicksort analysis

- $T(n)$ = random variable for the running time of randomized quicksort on an input of size n
- $E(T(n))$ = expected value of $T(n)$, the “average runtime” of randomized quicksort

$$T(n) = \begin{cases} T(0) + T(n-1) + dn & \text{if } 0 : n-1 \text{ split,} \\ T(1) + T(n-2) + dn & \text{if } 1 : n-2 \text{ split,} \\ \vdots & \\ T(n-1) + T(0) + dn & \text{if } n-1 : 0 \text{ split,} \end{cases}$$

9/27/07

CS 3343 Analysis of Algorithms

39



Randomized quicksort analysis

Assume that each split is equally likely, with $1/n$ probability.

⇒ The expected runtime (the “average runtime”) is

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

9/27/07

CS 3343 Analysis of Algorithms

40



Randomized quicksort analysis

Assume that each split is equally likely, with $1/n$ probability.

⇒ The expected runtime (the “average runtime”) is

$$\begin{aligned} E(T(n)) &= \frac{1}{n} \sum_{k=0}^{n-1} (E(T(k)) + E(T(n-k-1)) + dn) \\ &= \frac{2}{n} \sum_{k=0}^{n-1} (E(T(k)) + dn) \end{aligned}$$

9/27/07

CS 3343 Analysis of Algorithms

41



Hairy recurrence

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

(Assume base cases $E(T(0))=E(T(1))=0$.)

Prove: $E[T(n)] \leq cn \log n$ for constant $c > 0$.

- Base case: Choose c large enough so that $cn \log n$ dominates $E[T(n)]$ for sufficiently small $n \geq n_0 = 2$.

Use fact: $\sum_{k=2}^{n-1} k \log k \leq \frac{1}{2} n^2 \log n - \frac{1}{8} n^2$ (exercise).

9/27/07

CS 3343 Analysis of Algorithms

42



Inductive step

$$E[T(n)] \leq \frac{2}{n} \sum_{k=2}^{n-1} ck \log k + dn$$

Substitute inductive hypothesis.



Inductive step

$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=2}^{n-1} ck \log k + dn \\ &\leq \frac{2c}{n} \left(\frac{1}{2} n^2 \log n - \frac{1}{8} n^2 \right) + dn \end{aligned}$$

Use fact.



Inductive step

$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=2}^{n-1} ck \log k + dn \\ &\leq \frac{2c}{n} \left(\frac{1}{2} n^2 \log n - \frac{1}{8} n^2 \right) + dn \\ &= cn \log n - \left(\frac{cn}{4} - dn \right) \end{aligned}$$

Express as **desired** – **residual**.



Inductive step

$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=2}^{n-1} ck \log k + dn \\ &= \frac{2c}{n} \left(\frac{1}{2} n^2 \log n - \frac{1}{8} n^2 \right) + dn \\ &= cn \log n - \left(\frac{cn}{4} - dn \right) \\ &\leq cn \log n, \end{aligned}$$

if $c \geq 4d$.



Quicksort in practice

- Quicksort is a great general-purpose sorting algorithm.
- Quicksort is typically over twice as fast as merge sort.
- Quicksort can benefit substantially from **code tuning**.
- Quicksort behaves well even with caching and virtual memory.



Quicksort runtimes

- Best case: $n \log n$
- Worst case: n^2
- Expected runtime for randomized quicksort: $n \log n$