COT 6405 Introduction to Theory of Algorithms

Topic 8. Quicksort

Quicksort

- Sorts "in place"
  - Only a constant number of elements stored outside the sorted array
- Sorts O(n lg n) in the average case
- Sorts $O(n^2)$ in the worst case
- So why people use it instead of merge sort?
  - Merge sort does not sort "in place"

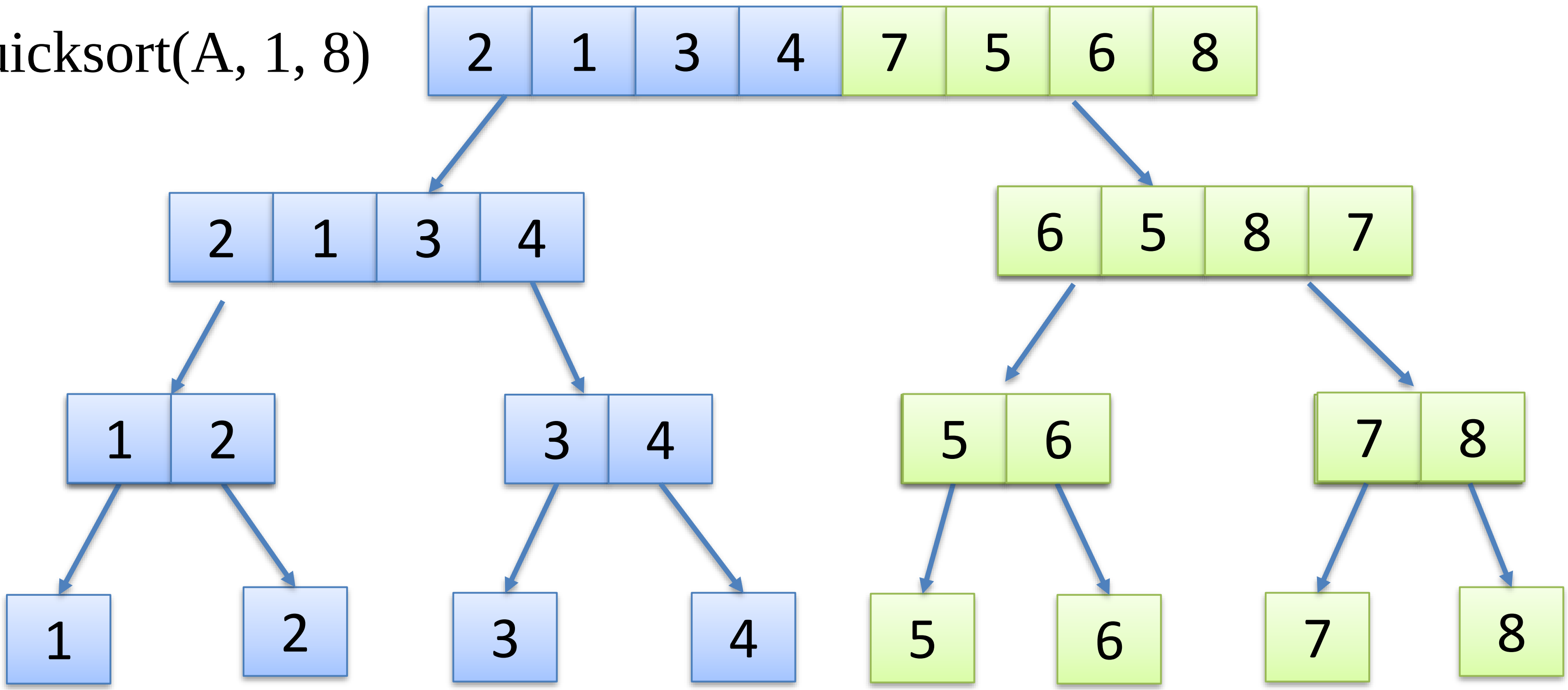
Quicksort: divide and conquer

- **Divide**: Array A[p..r] is partitioned into two non-empty subarrays A[p..q] and A[q+1..r]
  - All elements in A[p..q] are less than all elements in A[q+1..r]
- **Conquer:** The subarrays are recursively sorted by calls to quicksort
- **Combine:** No work is needed to combine the subarrays, because they are sorted in place.

An example of Quicksort

2	8	7	1	3	5	6	4

Quicksort(A, 1, 8)



Quicksort Code

```
Quicksort(A, p, r)
{
    if (p < r)
    {
        q = Partition(A, p, r);
        Quicksort(A, p, q-1);
        Quicksort(A, q+1, r);
    }
}  // what is the initial call?
```

Partition

- Clearly, all the actions take place in the **`partition()`** function
  - Rearranges the subarray “in place”
  - End result:
    - Two subarrays
    - All values in 1st subarray < all values in 2nd
  - Returns the index of the “pivot” element separating the two subarrays
- How do we implement this function?

Partition array $A[p..r]$

```
PARTITION(A, p, r)
    x ← A[r]        // select the pivot
    i ← p − 1
    for j ← p to r − 1
        if A[ j ] ≤ x
            i ← i + 1
            exchange A[i ] ↔ A[ j ]
    // move the pivot between the two subarraies
    exchange A[i + 1] ↔ A[r]
    // return the pivot
    return i + 1
```

What is the running time of `partition()`?

An example of Partition

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

PARTITION(A, p, r)

$x \leftarrow A[r]$ **// select the pivot**

$i \leftarrow p - 1$

for $j \leftarrow p$ **to** $r - 1$

  if $A[\,j\,] \le x$

    $i \leftarrow i + 1$

    exchange $A[i\,] \leftrightarrow A[\,j\,]$

exchange $A[i + 1] \leftrightarrow A[r]$

return $i + 1$

An example of Partition (cont’d)

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

```
PARTITION(A, p, r)
        x ← A[r]          // select the pivot
        i ← p − 1
        for j ← p to r − 1
                if A[ j ] ≤ x
                        i ← i + 1
                        exchange A[i ] ↔ A[ j ]
        exchange A[i + 1] ↔ A[r]
        return i + 1
```

An example of Partition (cont'd)

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

PARTITION(A, p, r)

$x \leftarrow A[r]$ **// select the pivot**
$i \leftarrow p - 1$
for $j \leftarrow p$ **to** $r - 1$
    if $A[j] \leq x$
        $i \leftarrow i + 1$
        exchange $A[i] \leftrightarrow A[j]$
exchange $A[i + 1] \leftrightarrow A[r]$
return $i + 1$

An example of Partition (cont'd)

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

```
PARTITION(A, p, r)
    x ← A[r]            // select the pivot
    i ← p − 1
    for j ← p to r − 1
        if A[ j ] ≤ x
            i ← i + 1
            exchange A[i ] ↔ A[ j ]
    exchange A[i + 1] ↔ A[r]
    return i + 1
```

An example of Partition (cont'd)

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

```
PARTITION(A, p, r)
    x ← A[r]          // select the pivot
    i ← p − 1
    for j ← p to r − 1
        if A[ j ] ≤ x
            i ← i + 1
            exchange A[i ] ↔ A[ j ]
    exchange A[i + 1] ↔ A[r]
    return i + 1
```

An example of Partition (cont'd)

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

```
PARTITION(A, p, r)
    x ← A[r]        // select the pivot
    i ← p − 1
    for j ← p to r − 1
        if A[ j ] ≤ x
            i ← i + 1
            exchange A[i ] ↔ A[ j ]
    exchange A[i + 1] ↔ A[r]
    return i + 1
```

An example of Partition (cont'd)

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

PARTITION*(A, p, r)*
    $x \leftarrow A[r]$ **// select the pivot**
    $i \leftarrow p - 1$
    for $j \leftarrow p$ to $r - 1$
        if $A[j] \leq x$
            $i \leftarrow i + 1$
            exchange $A[i] \leftrightarrow A[j]$
    exchange $A[i + 1] \leftrightarrow A[r]$
    return $i + 1$

An example of Partition (cont'd)

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

```
PARTITION(A, p, r)
        x ← A[r]        // select the pivot
        i ← p − 1
        for j ← p to r − 1
                if A[ j ] ≤ x
                        i ← i + 1
                        exchange A[i ] ↔ A[ j ]
        exchange A[i + 1] ↔ A[r]
        return i + 1
```

An example of Partition (cont'd)

p $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ r

2	1	3	4	7	5	6	8

$x=4$

j

i

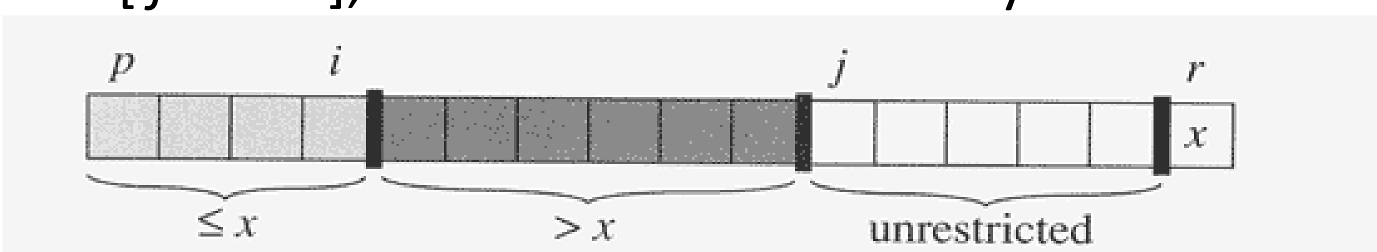
PARTITION*(A, p, r)*

```
PARTITION(A, p, r)
        x ← A[r]          // select the pivot
        i ← p − 1
        for j ← p to r − 1
                if A[ j ] ≤ x
                        i ← i + 1
                        exchange A[i ] ↔ A[ j ]
        exchange A[i + 1] ↔ A[r]
        return i + 1
```

Partitioning

- PARTITION first selects the **pivot** (How?)
  – the last element $A[r]$ in the subarray $A[p\ .\ .\ r]$
- The array is partitioned into <u>four regions</u>
  – some of which may be empty
- **<u>Loop invariant</u>:**

1. All entries in $A[p\ .\ .\ i]$ are ≤ pivot.
2. All entries in $A[i + 1\ .\ .\ j - 1]$ are > pivot.
3. $A[r]$ = pivot.
4. It's not needed as part of the loop invariant, but the fourth region is $A[j\ .\ .\ r-1]$, whose entries have not yet been examined.



Analyzing Quicksort

- Worst-case performance
  - The worst-case behavior for quicksort occurs when the partitioning routine produces with n-1 elements and one with 0 elements
- The recurrence is
  - T(n) = T(n-1) + T(0) + $\Theta(n)$

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

Exercise

- For T(n) = T(n-1) + $\Theta(n)$, use substitution method to show that the $\mathrm{T(n)} = \Theta(n^2)$.

Exercise (cont'd)

- T(n) = T(n-1) + $\Theta(n)$
- Basis: n = 1, T(1) = Θ (1)
  Inductive step: suppose T(k) $\leq$ ck^2 for all k < n, then

T(n) $\leq$ c$(n-1)^2$ + $c'n$

$=$ cn^2- 2cn + c +$c'n$

= cn^2 - (2c- c')n + c

$\leq$ cn^2 - (2c- c')n + cn (n > 1)

$\leq$ cn^2 when -(2c-c')n + cn ≥ 0 - > n_0= 1 and c'$\geq$ c

Thus, T(n) = O(n^2)

Exercise (cont'd)

- T(n) = T(n-1) + $\Theta(n)$
- Basis: n = 1, T(1) = Θ (1)
  Inductive step: suppose T(k) $\geq$ ck^2 for all k < n, then

  T(n) $\geq$ c$(n-1)^2$ + $c'n$

  $\geq$ cn^2- 2cn + c +$c'n$

  $\geq$ cn^2 - 2cn +$c'n$

  $\geq$ cn^2 if -2cn + $c'n \geq 0$ (c $\leq$ c'/2)

A question

- Will any particular input elicit the worst case?
  - Yes, the array is already sorted in the reverse order
  - Or it is already sorted

15	14	11	9	6	5	3	1

1	3	5	6	9	11	14	15

PARTITION(A, p, r)

$x \leftarrow A[r]$ // select the pivot
$i \leftarrow p - 1$
for $j \leftarrow p$ **to** $r - 1$
  if $A[j] \leq x$
    $i \leftarrow i + 1$
    exchange $A[i] \leftrightarrow A[j]$
exchange $A[i + 1] \leftrightarrow A[r]$
return $i + 1$

```
Quicksort(A, p, r)
{
    if (p < r)
    {
        q = Partition(A, p, r);
        Quicksort(A, p, q-1);
        Quicksort(A, q+1, r);
    }
}
```

Best-case performance

- The best-case behavior occurs when Partition() produces two sub-problems of equal size, the total size of two sub-problems is n-1.
- The recurrence for the running time is
  - $T(n) = 2T(n-1/2) + \Theta(n)$
  - By case 2 of the master theorem, $T(n) = \Theta(n\lg n)$

Performance of quicksort

- The running time of quicksort depends on the partitioning of the subarrays:
  - If the subarrays are balanced, then quicksort can run as fast as mergesort.
  - If they are unbalanced, then quicksort can run as slowly as insertion sort.

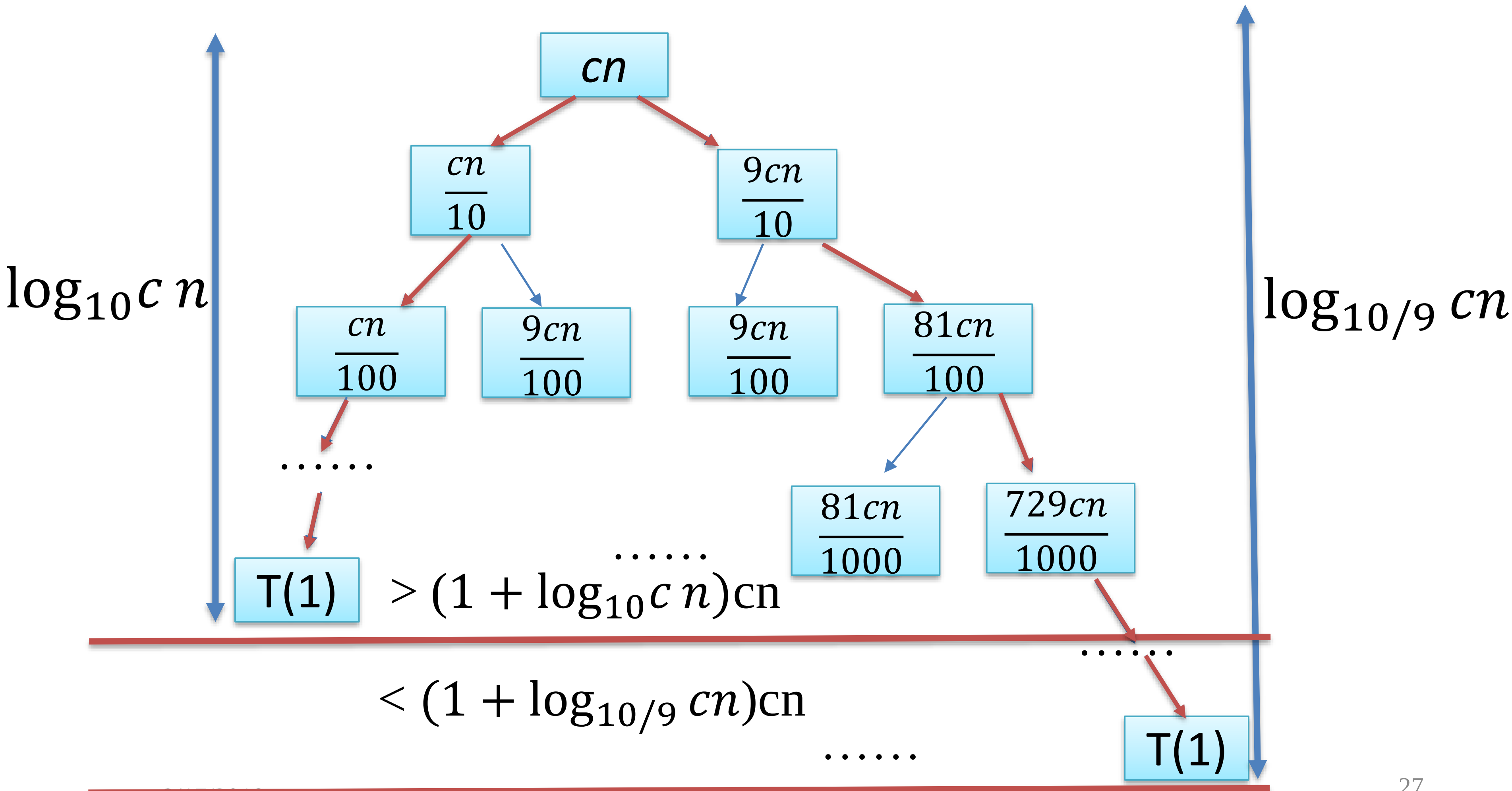
Analyzing Quicksort: Average Case

- Assuming random input → average-case running time is much closer to $O(n \lg n)$ than $O(n^2)$
- First, a more intuitive explanation/example:
  - Suppose that partition() always produces a 9-to-1 split. This looks quite unbalanced!
  - The recurrence is thus:

    $T(n) = T(9n/10) + T(n/10) + cn$
  - How deep will the recursion go? (draw it)

Average Case (cont'd)

- T(n) = T(9n/10) + T(n/10) + cn



Average Case (cont'd)

- For shortest path for the root to the leaf
  - The subproblem size for a node at depth *i* is $(\frac{1}{10})^i$cn
  - The subproblem size hits T(1), when $(\frac{1}{10})^i$cn= 1, or $i = \log_{10} cn$
  - Thus, the length of the shortest path is $\log_{10} cn$

Average Case (cont'd)

- For longest path for the root to the leaf
  - The subproblem size for a node at depth *i* is $(\frac{9}{10})^i$cn
  - The subproblem size hits T(1), when$(\frac{9}{10})^i$cn= 1, or $i = \log_{10/9} cn$
  - Thus, the length of the longest path is $\log_{10/9} cn$

Average Case (cont'd)

- Notice that every level of the tree has a cost of cn , until the recursion reaches a boundary condition at depth $\log_{10} cn = \Theta(lgn)$
- Then, the levels have cost at most cn
- The recursion terminates at depth $\log_{10/9} n = \Theta(lgn)$

Average Case (cont'd)

- The total cost of quicksort T(n)

  T(n) $> (1 + \log_{10} cn)$cn -> Ω (nlgn)

  T(n) $< (1 + \log_{10/9} cn)$cn -> O(nlgn)

  T(n) = $\Theta(\mathrm{nlgn})$

Analyzing Quicksort: 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)
  - What happens if we bad-split root node, then good-split the resulting size (n-1) node?
    - We end up with 3 subarrays, size 0, (n-1)/2-1, (n-1)/2
    - Combined cost of splits = n + n -1 = 2n -1 = $\Theta(n)$
    - No worse than if we had good-split the root node!
      - Good-split: $T(n) = 2T(n-1/2) + \Theta(n)$
      - Mix-split: $T(n) = T(0) + T(n-1/2) + T((n-1)/2-1)+ \Theta(n)$

        $\quad\quad <= 2T(n-1/2) + \Theta(n)$ -> good split complexity

Partition cost in Elliptical Shading

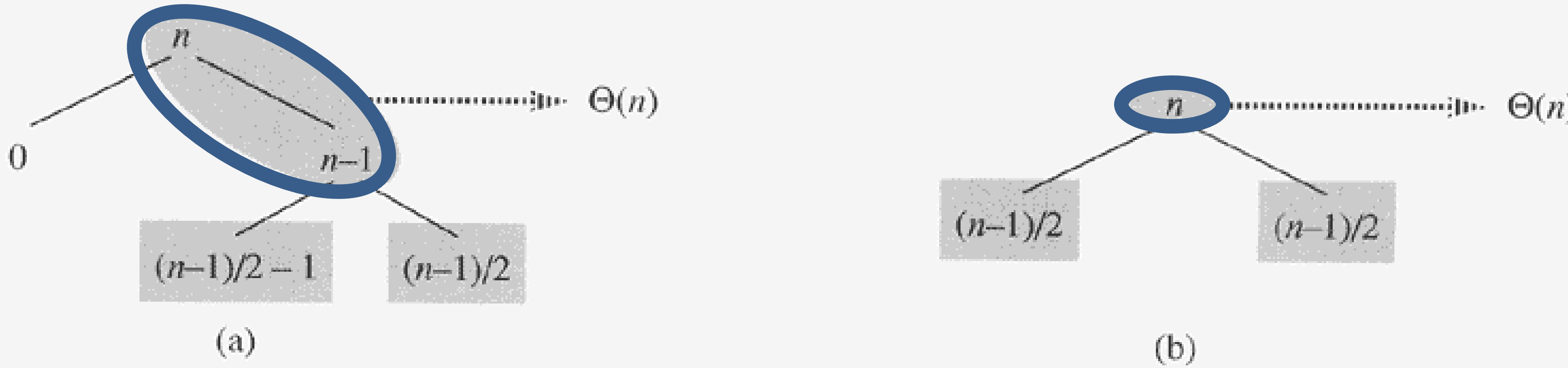


Figure 7.5 **(a)** Two levels of a recursion tree for quicksort. The partitioning at the root costs n and produces a "bad" split: two subarrays of sizes 0 and $n-1$. The partitioning of the subarray of size $n-1$ costs $n-1$ and produces a "good" split: subarrays of size $(n-1)/2 - 1$ and $(n-1)/2$. **(b)** A single level of a recursion tree that is very well balanced. In both parts, the partitioning cost for the subproblems shown with elliptical shading is $\Theta(n)$. Yet the subproblems remaining to be solved in (a), shown with square shading, are no larger than the corresponding subproblems remaining to be solved in (b).

Analyzing Quicksort: Average case

- Intuitively, the O(n) cost of a bad split (or 2 or 3 bad splits) can be absorbed into the O(n) cost of each good split
- Thus running time of alternating bad and good splits is still O(n lg n), with slightly higher constants
- How can we be more rigorous?

Analyzing Quicksort: Average case

- For simplicity, assume:
  - All inputs distinct (no repeats)
- Partition around a random element
  - all splits (0:n-1, 1:n-2, 2:n-3, … , n-1:0) are equally likely
  - In general, a split can be represented by (k : n-1-k)
- 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
- T(n) is the expected running time, T(n) = ?

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

- What is each term under the summation for?
- What is the Θ(n) term for?

Average case

- T(n) $= \frac{1}{n}\sum_{k=0}^{k=n-1} T(k) + T(n-1-k) + \Theta(n)$
- We can rewrite the above equation as
- T(n) $= \frac{2}{n}\sum_{k=0}^{k=n-1} T(k) + \Theta(n)$

Why?

- T(n) $=\frac{1}{n}$(T(0)+T(n-1)+T(1)+T(n-2)+…+ T(n-1)+T(0))

Average case (cont'd)

- 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
    - What's the inductive hypothesis?
    - $T(k) \leq a\, k \lg k + b$ for some constants $a>0$ and $b>0$ *and* $k<n$

Average case (cont'd)

- The recurrence to be solved
  - T(n) = $\frac{2}{n}\sum_{k=0}^{k=n-1} T(k) + \Theta(n)$
- What next?
  - Plug in the inductive hypothesis
  - $T(n) \leq \frac{2}{n}\sum_{k=0}^{k=n-1}(aklgk + b) + \Theta(n)$

Average case (cont'd)

- The recurrence to be solved
  - $T(n) \leq \frac{2}{n} \sum_{k=0}^{k=n-1} (aklgk + b) + \Theta(n)$
- What next?
  - Expand out the k=0 case
  - For simplicity, when *n* = 0, we define
    - $anlgn = \lim_{n \to 0} anlgn = 0$
  - T(n) $\leq \frac{2}{n} \left[b + \sum_{k=1}^{k=n-1} (aklgk + b) \right] + \Theta(n)$

Average case (cont'd)

- The recurrence to be solved
  - T(n) $\leq \frac{2}{n}\left[b+\sum_{k=1}^{k=n-1}(aklgk+b)\right]+\Theta(n)$

    $= \frac{2}{n}\left[\sum_{k=1}^{k=n-1}(aklgk+b)\right]+\frac{2b}{n}+\Theta(n)$
- 2b/n is just a constant, so fold it into Θ(n)
  - T(n) $\leq \frac{2}{n}\sum_{k=1}^{k=n-1}(aklgk+b)+\Theta(n)$

Average case (cont'd)

- The recurrence to be solved
  - T(n) $\leq \frac{2}{n}\sum_{k=1}^{k=n-1}(aklgk + b) + \Theta(n)$
- What next?
  - Distribute the summation
  - T(n) = $\frac{2}{n}\sum_{k=1}^{k=n-1} aklgk + \frac{2}{n}\sum_{k=1}^{k=n-1} b + \Theta(n)$

Average case (cont'd)

- The recurrence to be solved
  - T(n) = $\frac{2}{n}\sum_{k=1}^{k=n-1} aklgk + \frac{2}{n}\sum_{k=1}^{k=n-1} b + \Theta(n)$
- What next?
  - Evaluate the summation
  - T(n) = $\frac{2}{n}\sum_{k=1}^{k=n-1} aklgk + \frac{2}{n}\sum_{k=1}^{k=n-1} b + \Theta(n)$

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

Average case (cont'd)

- The recurrence to be solved
  - T(n) = $\frac{2a}{n}\sum_{k=1}^{k=n-1} klgk + \frac{2b}{n}(n-1) + \Theta(n)$
- What next?
  - Since n-1<n, 2b(n-1)/n < 2b
  - T(n) $\leq \frac{2a}{n}\sum_{k=1}^{k=n-1} klgk + 2b + \Theta(n)$

Average case (cont'd)

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

It can be proved that

$$\boxed{\sum_{k=1}^{k=n-1} k \lg k \le \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2}$$

This summation gets its own set of slides later

Average case (cont'd)

- The recurrence to be solved
  - T(n) $\leq \frac{2a}{n}\sum_{k=1}^{k=n-1} klgk + 2b + \Theta(n)$
- What next?
  - Substitute $\sum_{k=1}^{k=n-1} klgk \ \leq \ \frac{1}{2}n^2lgn$ - $\frac{1}{8}n^2$
  - T(n) $\leq \frac{2a}{n}$ ($\frac{1}{2}n^2$lg$n$ - $\frac{1}{8}n^2$) $+$ 2b + $\Theta(n)$

Average case (cont'd)

- The recurrence to be solved
  - T(n) $\leq \frac{2a}{n}\left(\frac{1}{2}n^2 \lg n - \frac{1}{8}n^2\right) + 2b + \Theta(n)$
- What next?
  - Distribute the (2a/n) term
  - T(n) $\leq an\lg n - \frac{an}{4} + 2b + \Theta(n)$

Average case (cont'd)

- The recurrence to be solved
  - $T(n) \le an\lg n - \frac{an}{4} + 2b + \Theta(n)$
- What is our goal?
  - Our goal is to get $T(n) \le an\lg n + b$
  - We rewrite $T(n)$ as

$$\boxed{T(n) \le an\lg n + b + [\Theta(n) + b - \frac{an}{4}]}$$

Average case (cont'd)

- T(n) $\leq$ *an*lg*n* + *b* + $[\Theta(n)+b - \frac{an}{4}]$
- What next?
  - $\Theta(n)+b - \frac{an}{4} \leq 0$
    - $\Theta(n)+b - \frac{an}{4} \leq \Theta(n) + bn - \frac{an}{4} \leq$ c'*n* +$bn - \frac{an}{4} \leq$ 0, when *a* $\geq$ 4(*c*' + *b*).
    - Thus, pick *a* large enough that *an*/4 dominates Θ(*n*)+*b*
  - Then, T(n) $\leq$ *an*lg*n* + *b*

Average case summary

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

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

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

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

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

(when n -> ∞, $\frac{2b}{n}$ - > 0)

Average case summary(cont'd)

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

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

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

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

Average case summary (cont'd)

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

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

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

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

$\leq an\lg n + b$

Pick *a* large enough that *an*/4 dominates Θ(n)+b

Average case (cont'd)

- So T(n) ≤ *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
- Now let's prove the summation

$$\boxed{\sum_{k=1}^{k=n-1} k \lg k \le \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2}$$

Tightly Bounding

- Prove $\sum_{k=1}^{k=n-1} klgk \leq \frac{1}{2}n^2 \lg n - \frac{1}{8}n^2$
- Split the summation for a tighter bound

$$\sum_{k=1}^{k=n-1} klgk = \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} klgk + \sum_{k=\lceil\frac{n}{2}\rceil}^{k=n-1} klgk$$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \ = \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} klgk + \sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} klgk$
- The lgk in the second term is bounded by lg n

$$\sum_{k=1}^{k=n-1} klgk \ \leq \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} klgk + \sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} klgn$$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \ \leq \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} klgk$ +$\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} klgn$
- Move the lgn outside the summation

$\sum_{k=1}^{k=n-1} klgk \ \leq \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} klgk$ + lgn$\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} k$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \;\leq \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} klgk$ + lgn$\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} k$
- What next?

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

  $\sum_{k=1}^{k=n-1} klgk \;\leq \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} klg\,\frac{n}{2}$ + lgn$\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} k$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \ \leq \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} klg\,\frac{n}{2}$ + lgn$\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} k$
- What next?

  lg n/2 = lg n - 1

  $\sum_{k=1}^{k=n-1} klgk \ \leq \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} k(lgn-1)$ + lgn$\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} k$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \ \le \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} k(lgn-1)$ + lgn$\sum_{k=\lceil\frac{n}{2}\rceil}^{k=n-1} k$
- What next?
  – Move (lg n - 1) outside the summation

$$\sum_{k=1}^{k=n-1} klgk \ \le (lgn-1)\sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} k + \text{lgn}\sum_{k=\lceil\frac{n}{2}\rceil}^{k=n-1} k$$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \ \leq (lgn - 1) \ \sum_{k=1}^{k=\lceil\frac{n}{2}\rceil-1} k$+ lgn$\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} k$
- What next?
  – Distribute the (lg n - 1)

$$\sum_{k=1}^{k=n-1} klgk \ \leq \ lgn \sum_{k=1}^{k=\left\lceil\frac{n}{2}\right\rceil-1} k - \sum_{k=1}^{k=\left\lceil\frac{n}{2}\right\rceil-1} k + \text{lgn} \sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} k$$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \;\leq\; lgn \sum_{k=1}^{k=\left\lceil\frac{n}{2}\right\rceil-1} k - \sum_{k=1}^{k=\left\lceil\frac{n}{2}\right\rceil-1} k$ + lgn $\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{k=n-1} k$
- What next?
  - The summations overlap in range; combine them

$$\sum_{k=1}^{k=n-1} klgk \;\leq\; lgn \sum_{k=1}^{k=n-1} k - \sum_{k=1}^{k=\left\lceil\frac{n}{2}\right\rceil-1} k$$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \ \leq \ lgn \sum_{k=1}^{k=n-1} k - \sum_{k=1}^{k=\left\lceil \frac{n}{2} \right\rceil - 1} k$
- What next?
  - The Gaussian series

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

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \ \leq$lgn $(\frac{(n-1)n}{2}) - \sum_{k=1}^{k=\left\lceil\frac{n}{2}\right\rceil-1} k$
- What next?
  - Rearrange first term, place upper bound on second

$$\sum_{k=1}^{k=n-1} klgk \ \leq \frac{1}{2}[n(n-1)]lgn - \sum_{k=1}^{k=n/2-1} k$$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \ \le \frac{1}{2}[n(n-1)]lgn - \sum_{k=1}^{k=n/2-1} k$
- What next?
  - The Gaussian series
  - $\sum_{k=1}^{k=n-1} klgk \ \le \frac{1}{2}[n(n-1)]lgn - \frac{1}{2}\left(\frac{n}{2}\right)\left(\frac{n}{2}-1\right)$

Tightly Bounding (cont'd)

- $\sum_{k=1}^{k=n-1} klgk \le \frac{1}{2}[n(n-1)]lgn - \frac{1}{2}\left(\frac{n}{2}\right)\left(\frac{n}{2}-1\right)$
- What next?
  - Multiply it all out
  - $\sum_{k=1}^{k=n-1} klgk \le \frac{1}{2}(n^2lgn - nlgn)$ - $\frac{1}{8}\ n^2$ + $\frac{n}{4}$

$$= \frac{1}{2}n^2lgn - \frac{1}{8}\ n^2 - \frac{1}{2}nlgn + \frac{n}{4}$$

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

Done !!!!

Tightly Bounding Summary

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

$$\leq \sum_{k=1}^{k=\lceil n/2\rceil-1} klg(\frac{n}{2}) + \sum_{k=\lceil n/2\rceil}^{k=n-1} klgn$$

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

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

Tightly Bounding Summary (cont'd)

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

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

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

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

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

Tightly Bounding Summary(cont'd)

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

$$= \frac{1}{2}[n(n-1)]lgn - \frac{1}{2}\left(\frac{n}{2}\right)\left(\frac{n}{2} - 1\right)$$

$$= \frac{1}{2}(n^2lgn - nlgn) - \frac{1}{8}\ n^2 + \frac{n}{4}$$

$$= \frac{1}{2}n^2lgn - \frac{1}{8}\ n^2 - \frac{1}{2}nlgn + \frac{n}{4}$$

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

Done !!!!