Ch12: Quick Sort

305233, 305234
Algorithm Analysis and Design

Jiraporn Pooksook
Naresuan University

Quick sort

- Worst-case running time is $\Theta(n^2)$ on an input array of n numbers.
- Expected running time is $\Theta(n \lg n)$
- Based on the divide and conquer paradigm.

Quicksort(A, p,r)

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

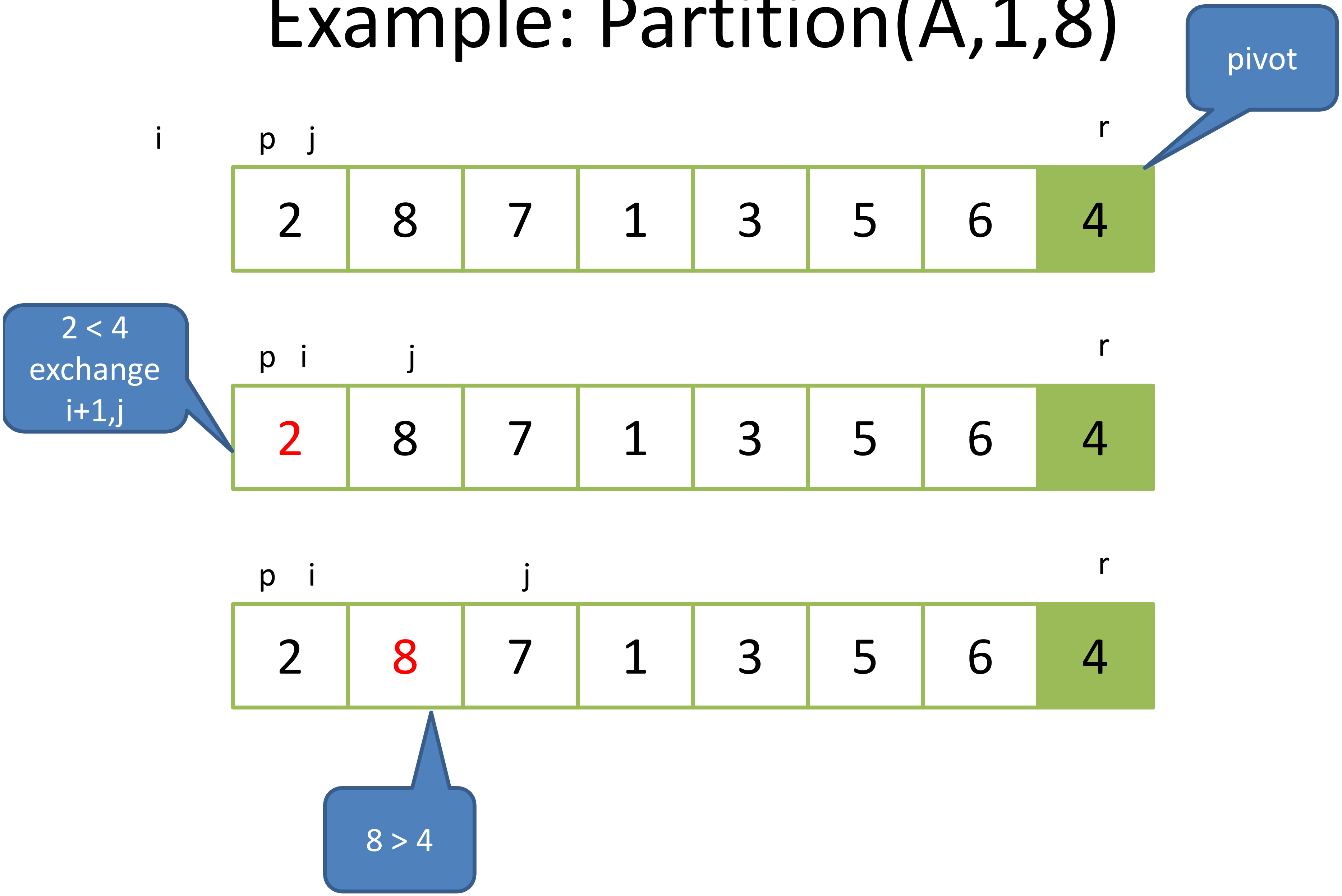
Partition(A,p,r)

- **To partition data is to divide it into two groups**
- One group contains items with a key value higher than the reference value.
- The other group contains items with a key value lower than the reference value.
- A reference value is also called a pivot value

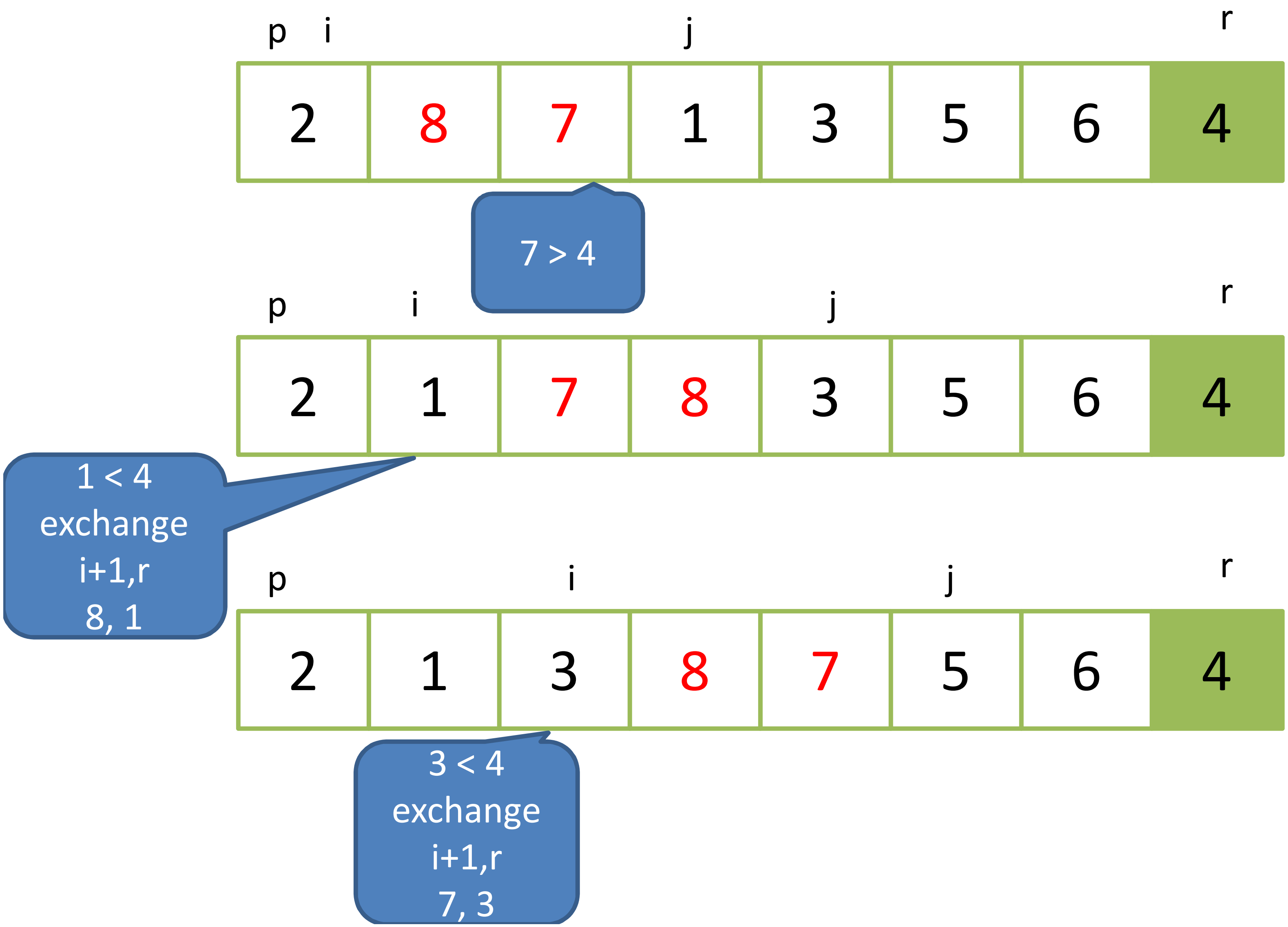
Partition(A, p,r)

```
x = A[r]
i = p-1
for j = p to r-1
        do if A[j] <= x
                then i = i +1
                exchange A[ i ] and A[ j ]
exchange A[i+1] and A[r]
return i+1
```

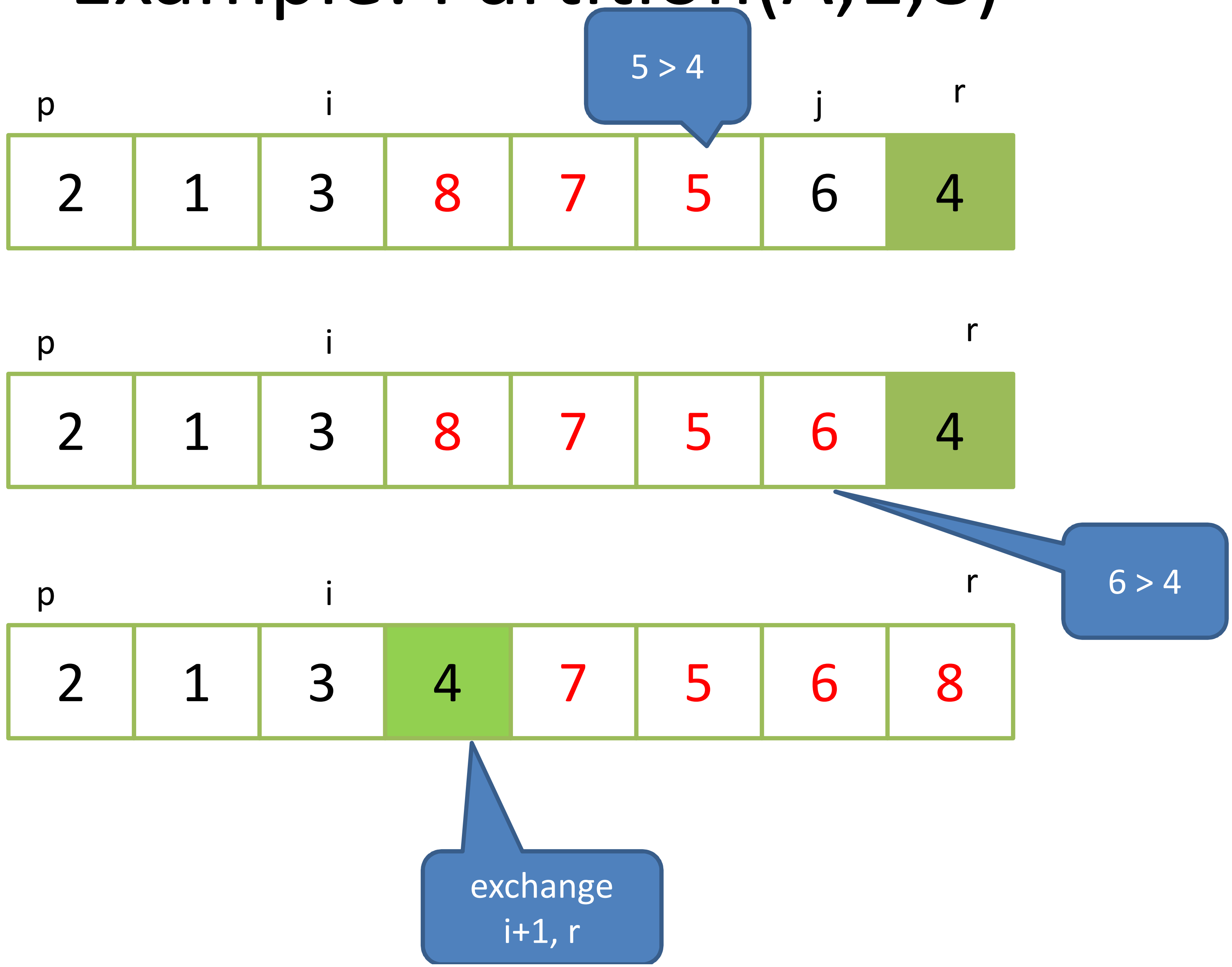
Example: Partition(A,1,8)



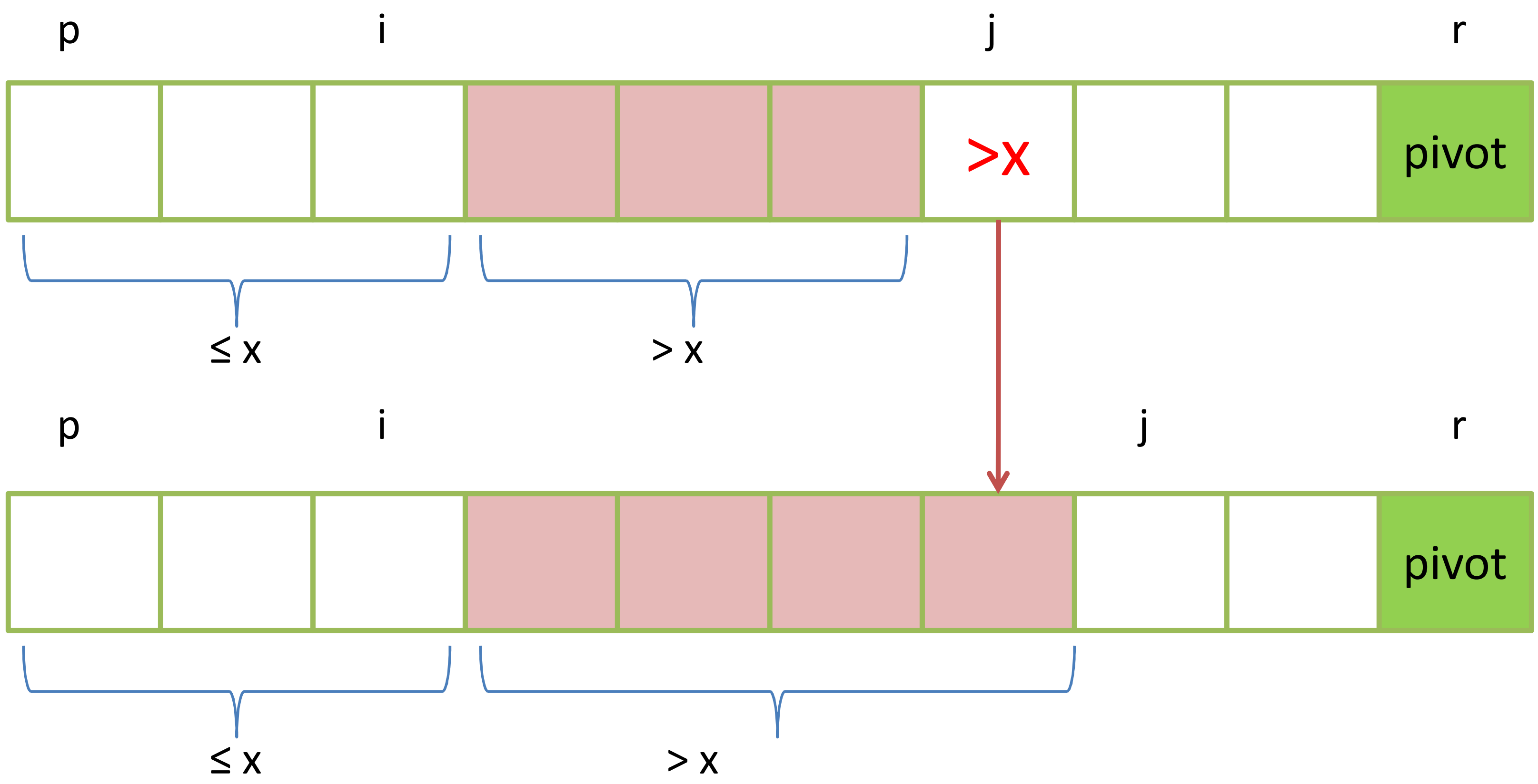
Example: Partition(A,1,8)



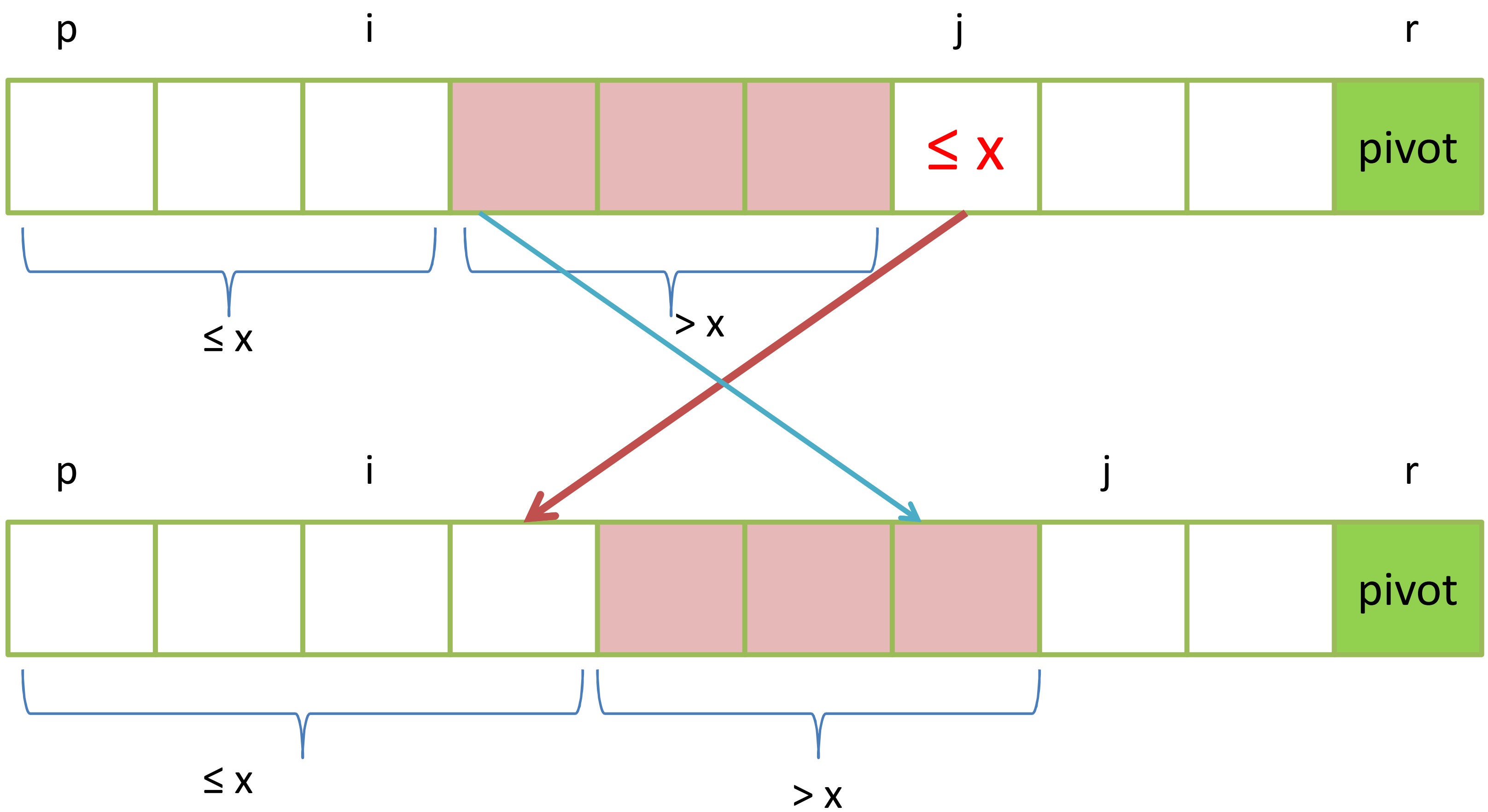
Example: Partition(A,1,8)



Analyzing Partition(A,p,r)



Analyzing Partition(A,p,r)



Loop invariants with Partition(A,p, r)

theoretically

loop invariant =
before running each loop for any array index k,
the following conditions hold:
1. $p \le k \le i$, 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:
Before running loop 0 , i = p-1 and j=p
Condition 1 : there is no value between p and i,
Condition 2: there is no value between i+1 and j-1
Condition 3: line 1 x = A[k]
Hence all 3 conditions hold. (True!!)

Maintenance:
When A[j]<=x, i increases and A[i], A[j] are swapped. Then j increases. Condition 1 satisfies.
When A[j] >= x, j increases. Then condition 2 satisfies.
Condition 3 satisfies from the 1st line.
Hence all 3 conditions hold. (True!!)

Termination:
At termination j = r, the array has partitioned into 3 sets following above conditions. (True!!)

The running time of Quicksort

- Worst-case partitioning
  - Partition with n-1 elements and 0 elements.

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

  - The running time is $\Theta(n^2)$
- Best-case partitioning
  - Partition with the floor of n/2 elements

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

  - The running time is $\Theta(n \lg n)$

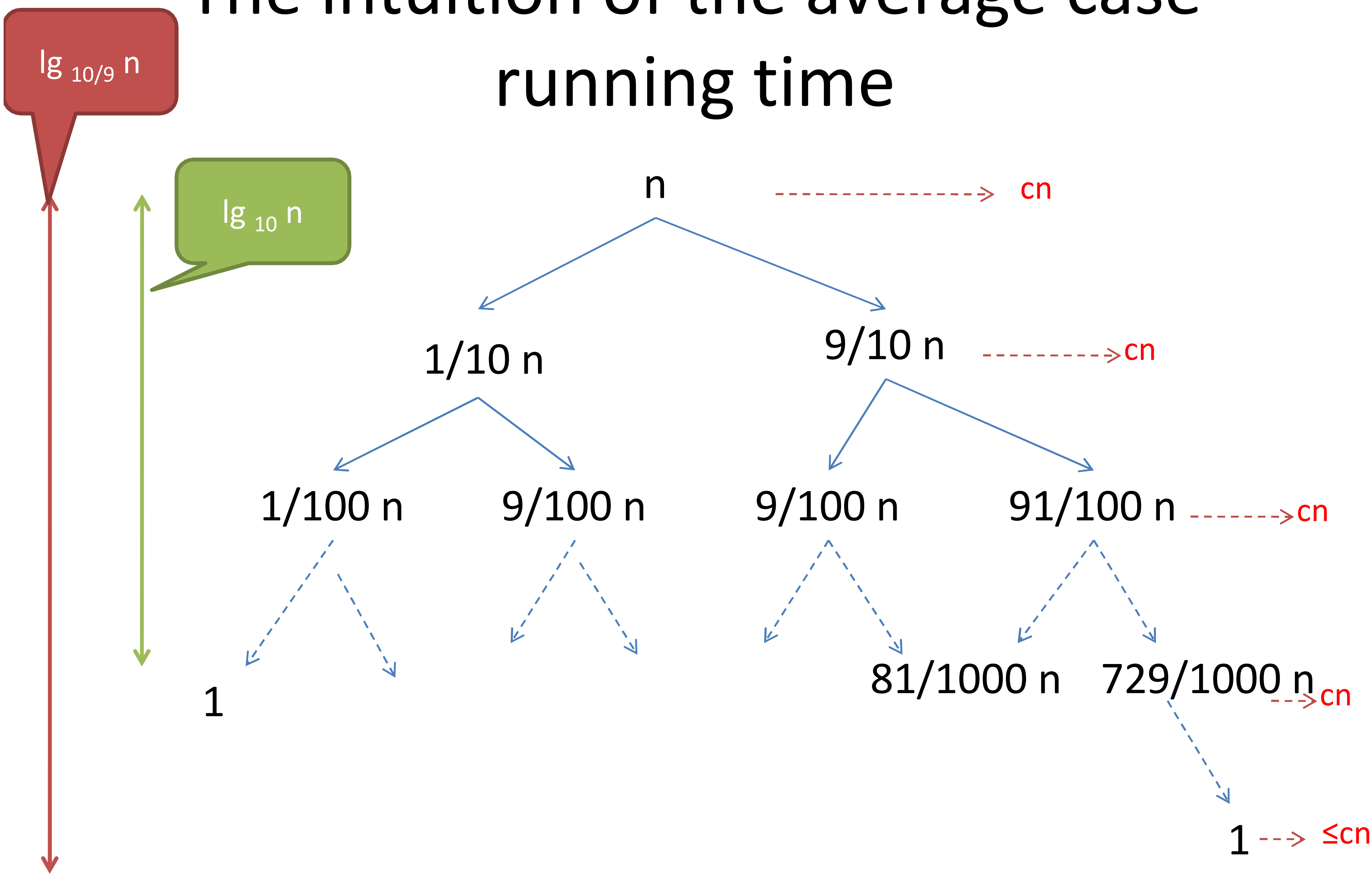
The running time of Quicksort

- Balanced partitioning
- Average running time is closer to the best-case running time.
- For example, partition 9-to-1 proportional split.

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

- The running time is $\Theta(n \lg n)$

The intuition of the average case running time



$O(n \lg n)$

Randomized version of Quicksort

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

Wrapping-up Sorting algorithms

Algorithm	Time	Note
Insertion sort	$O(n^2)$	In-place memory Notoriously slow
Merge sort	O(n lg n)	Linear extra memory Fast(good for large input)
Heap sort	O(n lg n)	In-place memory Fast (worst case is O(n lg n)
Quick sort	O(n lg n)	In-place memory Fastest (optimal for large input but worst case can be $O(n^2)$)

Practice: Quicksort

16	14	51	2	15	19	17	13