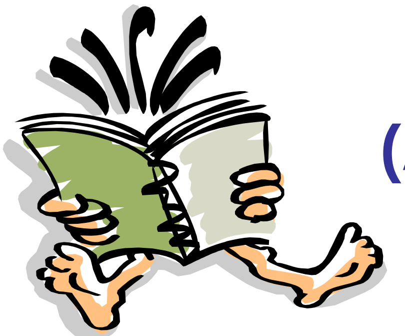
Analysis of Algorithms

Randomizing Quicksort



(Appendix C.2 , Appendix C.3)

(Chapter 5, Chapter 7)

Randomizing Quicksort

- Randomly permute the elements of the input array before sorting
- OR ... modify the PARTITION procedure
  - At each step of the algorithm we exchange element $A[p]$ with an element chosen at random from $A[p...r]$
  - The pivot element $x = A[p]$ is equally likely to be any one of the $r - p + 1$ elements of the subarray

Randomized Algorithms

- No input can elicit worst case behavior
  - Worst case occurs only if we get “unlucky” numbers from the random number generator
- Worst case becomes less likely
  - Randomization can NOT eliminate the worst-case but it can make it less likely!

Randomized PARTITION

Alg.: RANDOMIZED-PARTITION(A, p, r)

$i \leftarrow$ RANDOM(p, r)

exchange $A[p] \leftrightarrow A[i]$

return PARTITION(A, p, r)

Randomized Quicksort

Alg. : RANDOMIZED-QUICKSORT(A, p, r)

if $p < r$

then $q \leftarrow$ RANDOMIZED-PARTITION(A, p, r)

RANDOMIZED-QUICKSORT(A, p, q)

RANDOMIZED-QUICKSORT$(A, q + 1, r)$

Formal Worst-Case Analysis of Quicksort

- $T(n)$ = worst-case running time

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

- Use substitution method to show that the running time of Quicksort is $O(n^2)$
- Guess $T(n) = O(n^2)$
  - Induction goal: $T(n) \le cn^2$
  - Induction hypothesis: $T(k) \le ck^2$ for any $k < n$

Worst-Case Analysis of Quicksort

- Proof of induction goal:

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

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

- The expression $q^2 + (n-q)^2$ achieves a maximum over the range $1 \le q \le n-1$ at one of the endpoints

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

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

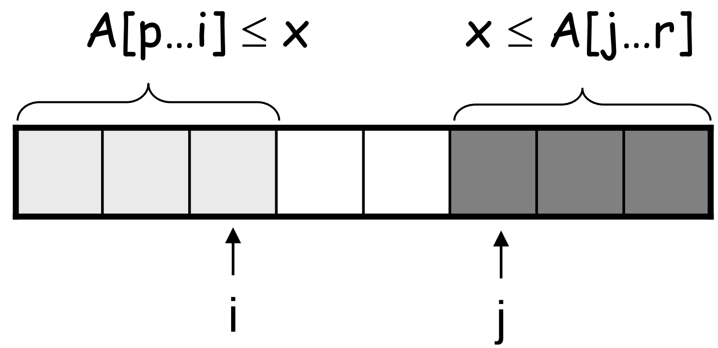
$$\le cn^2$$

Revisit Partitioning

- Hoare's partition
  - Select a pivot element **x** around which to partition
  - Grows two regions

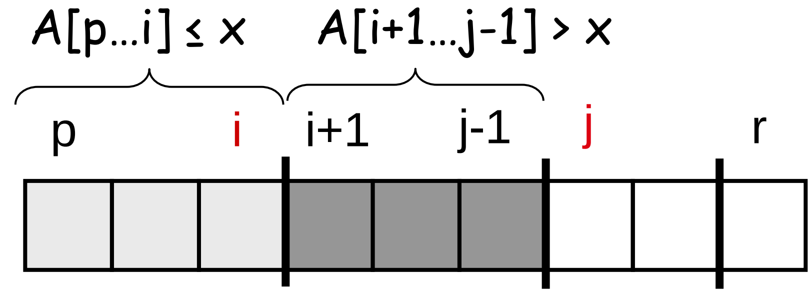
    $A[p \dots i] \leq x$

    $x \leq A[j \dots r]$



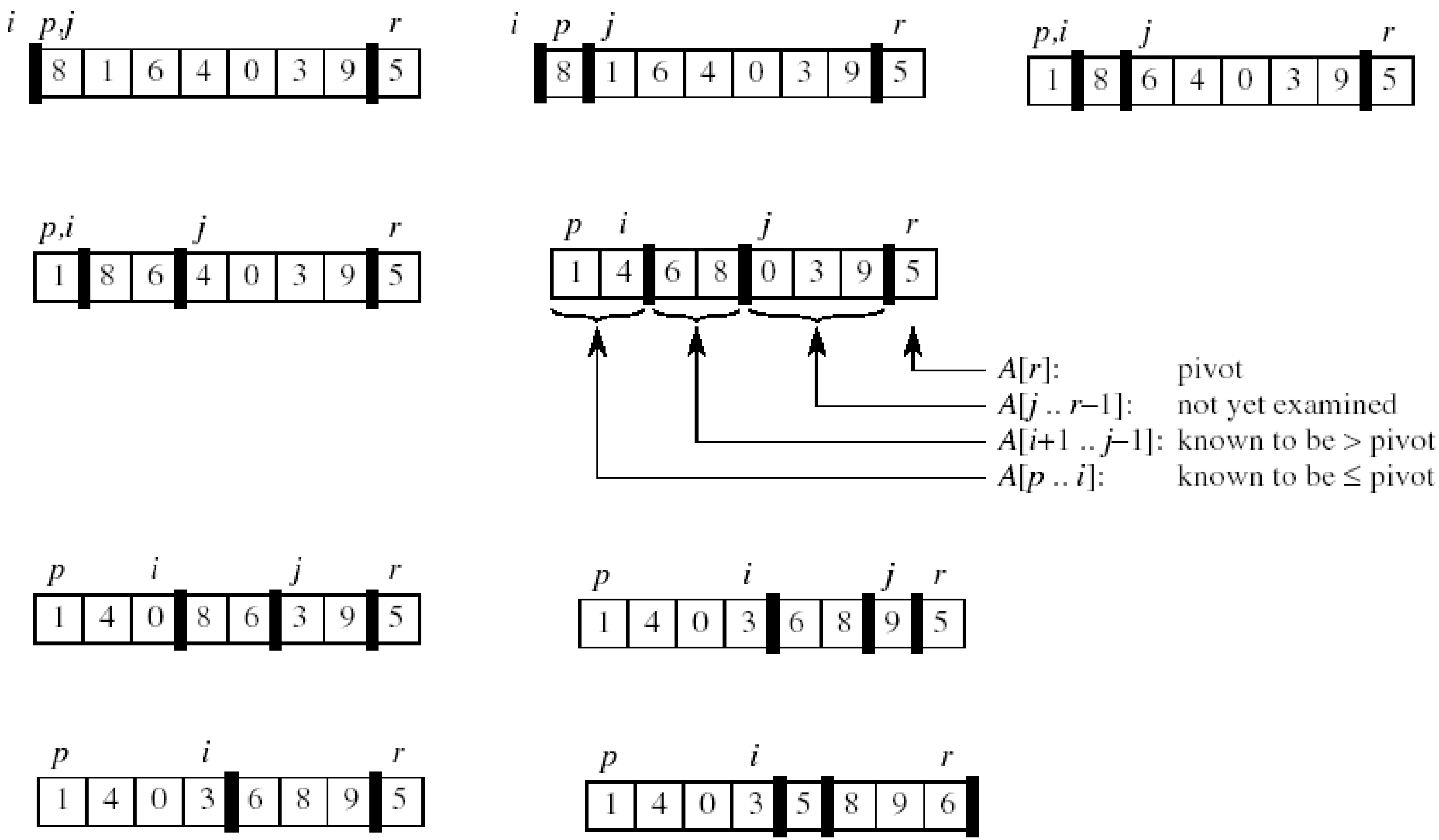
Another Way to PARTITION (Lomuto's partition – page 146)

- Given an array A, partition the array into the following subarrays:



  - A pivot element $x = A[q]$
  - Subarray $A[p..q-1]$ such that each element of $A[p..q-1]$ is smaller than or equal to x (the pivot)
  - Subarray $A[q+1..r]$, such that each element of $A[p..q+1]$ is <u>strictly</u> greater than x (the pivot)
- The pivot element is <u>not included</u> in any of the two subarrays

Example



at the end,
swap pivot

Another Way to PARTITION (cont'd)

Alg.: PARTITION(*A, p, r)*

$x \leftarrow A[r]$

$i \leftarrow p - 1$

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

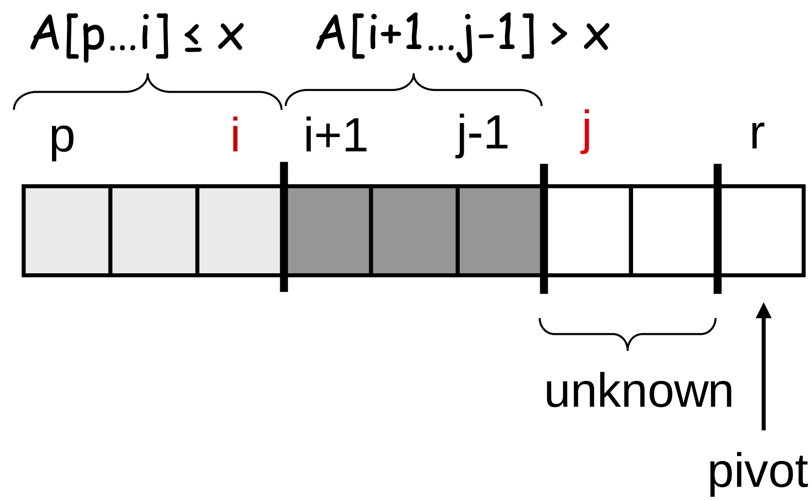
  do if $A[\,j\,] \leq x$

    then $i \leftarrow i + 1$

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

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

return $i + 1$



Chooses the last element of the array as a pivot
Grows a subarray [p..i] of elements ≤ x
Grows a subarray [i+1..j-1] of elements >x
Running Time: $\Theta(n)$, where n=r-p+1

Randomized Quicksort (using Lomuto's partition)

Alg. : RANDOMIZED-QUICKSORT(A, p, r)

if p < r

then $q \leftarrow$ RANDOMIZED-PARTITION(A, p, r)

RANDOMIZED-QUICKSORT(A, p, q - 1)

RANDOMIZED-QUICKSORT(A, q + 1, r)

The pivot is <u>no longer</u> included in any of the subarrays!!

Analysis of Randomized Quicksort

Alg. : RANDOMIZED-QUICKSORT(A, p, r)

The running time of Quicksort is dominated by PARTITION !!

if p < r

then $q \leftarrow$ RANDOMIZED-PARTITION(A, p, r)

RANDOMIZED-QUICKSORT(A, p, q - 1)

RANDOMIZED-QUICKSORT(A, q + 1, r)

PARTITION is called at most n times

(at each call a pivot is selected and never again included in future calls)

PARTITION

Alg.: 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] ↔ A[j]
exchange A[i + 1] ↔ A[r]
return i + 1
```

O(1) - constant

# of comparisons: X_k between the pivot and the other elements

O(1) - constant

Amount of work at call k: $c + X_k$

Average-Case Analysis of Quicksort

- Let X = **total number of comparisons performed in <u>all calls</u> to PARTITION:** $X = \sum_k X_k$

- The total work done over the **entire** execution of Quicksort is

$$O(nc+X)=O(n+X)$$

- Need to estimate E(X)

Review of Probabilities

- **Definitions**

  - random experiment: an experiment whose result is not certain in advance (e.g., throwing a die)

  - outcome: the result of a random experiment

  - sample space: the set of all possible outcomes (e.g., $\{1,2,3,4,5,6\}$)

  - event: a subset of the sample space (e.g., obtain an odd number in the experiment of throwing a die = $\{1,3,5\}$)

Review of Probabilities

- **Probability of an event** (discrete case)

  - The likelihood that an event will occur if the underlying random experiment is performed

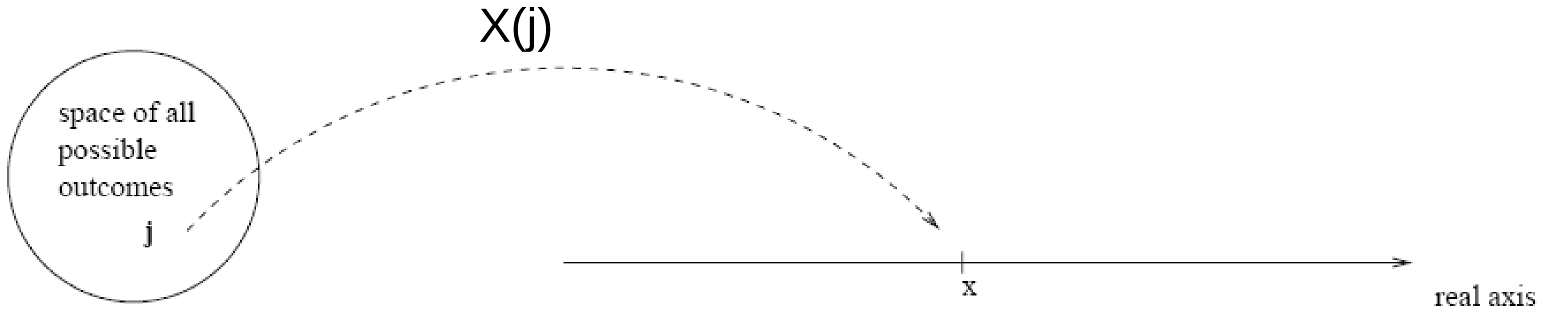
$$P(event) = \frac{number\ of\ \ favorable\ outcomes}{total\ \ number\ of\ \ possible\ outcomes}$$

Example: $P(obtain\ an\ odd\ \ number) = 3/6 = 1/2$

Random Variables

*Def.: **(Discrete) random variable X**: a function from a sample space S to the real numbers.*

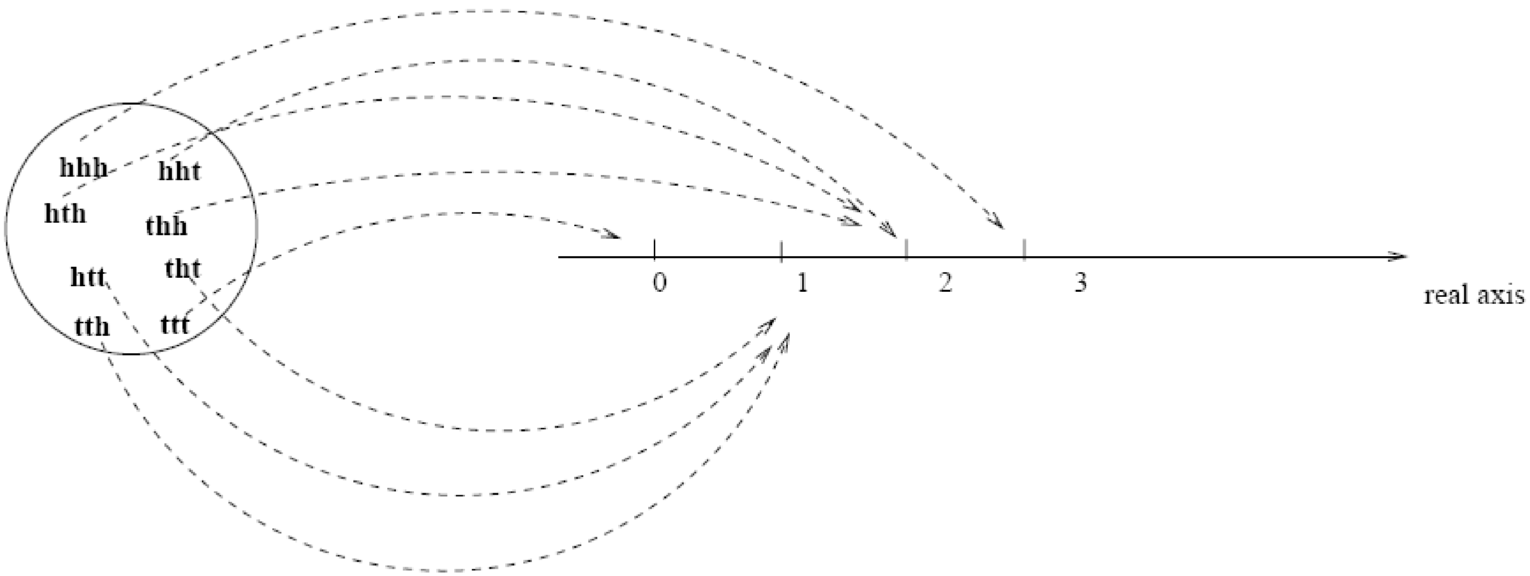
– It associates a real number with each possible outcome of an experiment.



Random Variables

E.g.: Toss a coin three times

define X = “numbers of heads”



Computing Probabilities Using Random Variables

- Example: consider the experiment of throwing a pair of dice

Define the r.v. X="sum of dice"

$X = x$ corresponds to the event $A_x = \{s \in S / X(s) = x\}$

(e.g., $X = 5$ corresponds to $A_5 = \{(1,4),(4,1),(2,3),(3,2)\}$

$$P(X = x) = P(A_x) = \sum_{s:X(s)=x} P(s)$$

$(P(X = 5) = P((1,4)) + P((4,1)) + P((2,3)) + P((2,3)) = 4/36 = 1/9)$

Expectation

- Expected value (expectation, mean) of a discrete random variable X is:

$$E[X] = \Sigma_x \, x \Pr\{X = x\}$$

  – "Average" over all possible values of random variable X

Examples

Example: X = face of one fair dice

E[X] = 1·1/6 + 2·1/6 + 3·1/6 + 4·1/6 + 5·1/6 + 6·1/6 = 3.5

Example: X="sum of dice"

Events												
Sum	1	2	3	4	5	6	7	8	9	10	11	12
Probability	0/36	1/36	2/36	3/36	4/36	5/35	6/36	5/36	4/360	3/36	2/36	1/36

$$E(X) = 1P(X=1) + 2P(X=2) + \ldots + 12P(X=12) = (0 + 2 + \ldots + 12)/36 = 7$$

Indicator Random Variables

- Given a sample space S and an event *A*, we define the ***indicator random variable*** I{A} associated with A:
  - $I\{A\} = \begin{cases} 1 & \text{if A occurs} \\ 0 & \text{if A does not occur} \end{cases}$
- The expected value of an indicator random variable $X_A = I\{A\}$ is:

$$E[X_A] = Pr\{A\}$$

- Proof:

$$E[X_A] = E[I\{A\}] = 1 * Pr\{A\} + 0 * Pr\{\bar{A}\} = Pr\{A\}$$

Average-Case Analysis of Quicksort

- Let X = **total number of comparisons performed in all calls to PARTITION:** $X = \sum_k X_k$

- The total work done over the **entire** execution of Quicksort is

  O(n+X)

- Need to estimate E(X)

Notation

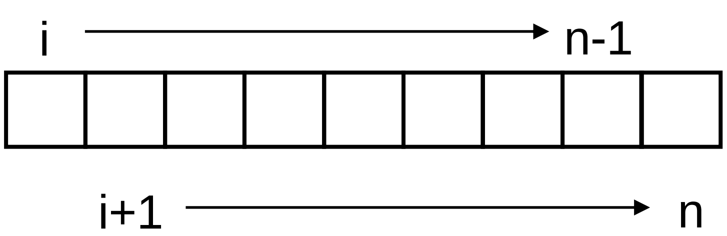
z_2	z_9	z_8	z_3	z_5	z_4	z_1	z_6	z_{10}	z_7
2	9	8	3	5	4	1	6	10	7

- Rename the elements of A as $z_1, z_2, \ldots, z_n$, with z_i being the *i*-th smallest element
- Define the set $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$ the set of elements between z_i and z_j, inclusive

Total Number of Comparisons in PARTITION

- Define $X_{ij} = I\{z_i \text{ is compared to } z_j\}$
- Total number of comparisons X performed by the algorithm:

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



Expected Number of Total Comparisons in PARTITION

- Compute the **expected value of X:**

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

by linearity
of expectation

indicator
random variable

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

the expectation of X_{ij} is equal
to the probability of the event
"z_i is compared to z_j"

Comparisons in PARTITION : Observation 1

- Each pair of elements is compared **at most once** during the entire execution of the algorithm
  - Elements are compared only to the pivot point!
  - Pivot point is excluded from future calls to PARTITION

Comparisons in PARTITION: Observation 2

- Only the pivot is compared with elements in both partitions!

z_2	z_9	z_8	z_3	z_5	z_4	z_1	z_6	z_{10}	z_7
2	9	8	3	5	4	1	6	10	7

$Z_{1,6} = \{1, 2, 3, 4, 5, 6\}$ $\{7\}$ $Z_{8,9} = \{8, 9, 10\}$

pivot

Elements between different partitions are <u>never</u> compared!

Comparisons in PARTITION

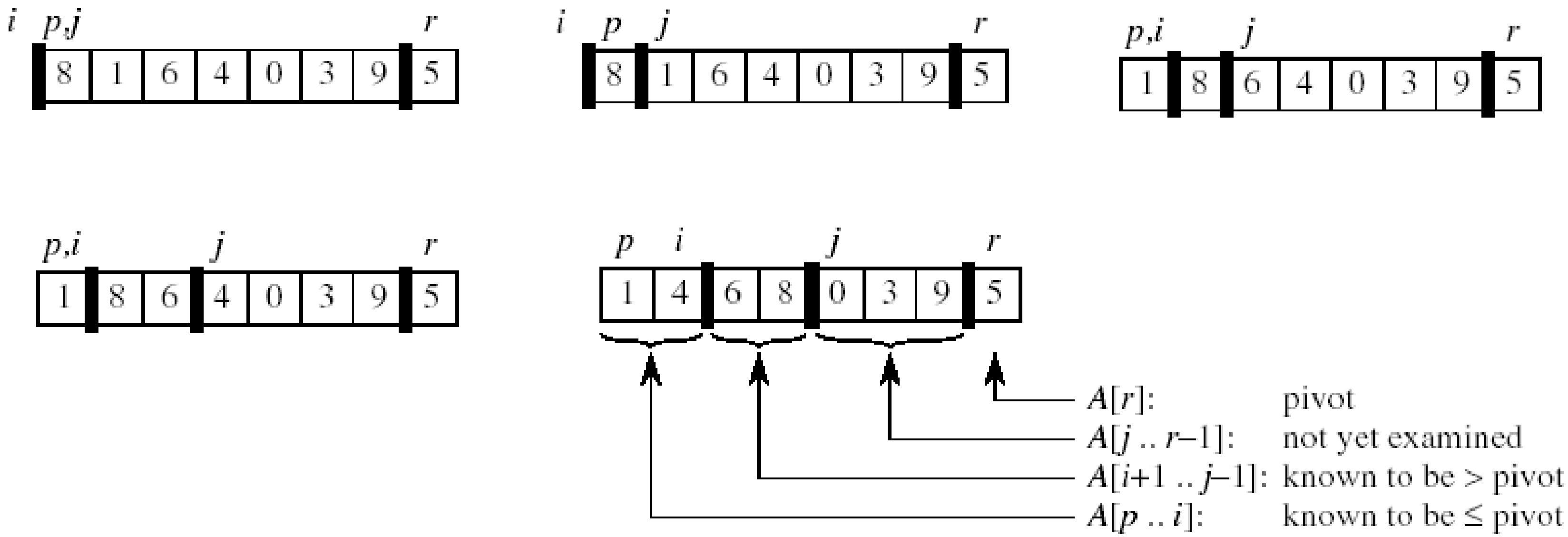
z_2	z_9	z_8	z_3	z_5	z_4	z_1	z_6	z_{10}	z_7
2	9	8	3	5	4	1	6	10	7

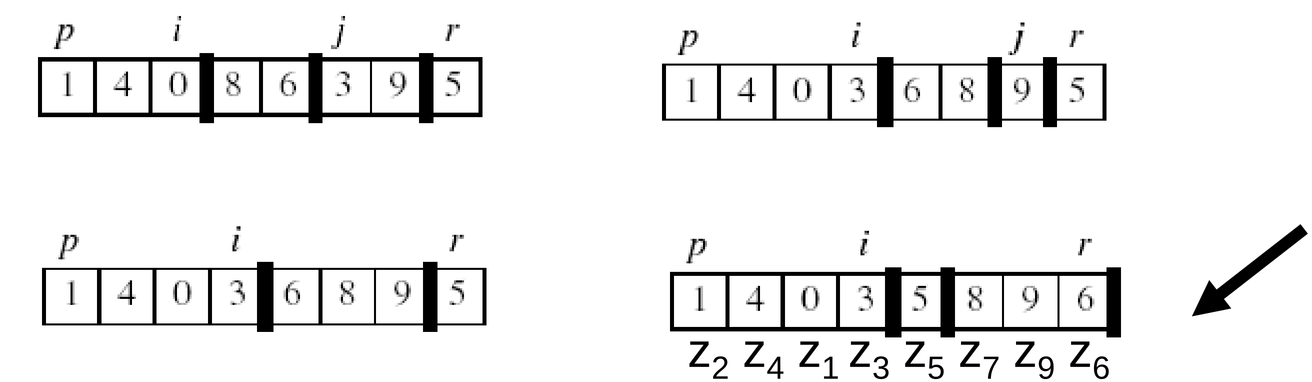
$Z_{1,6}$= {1, 2, 3, 4, 5, 6}   {7} $Z_{8,9}$ = {8, 9, 10}

$$\Pr\{z_i \textit{ is compared to } z_j\}?$$

- Case 1: pivot chosen such as: $z_i < x < z_j$
  - z_i and z_j will never be compared
- Case 2: z_i or z_j is the pivot
  - z_i and z_j will be compared
  - only if one of them is chosen as pivot before any other element in range z_i to z_j

See why ☺





z2 will never be compared with z6 since z5 (which belongs to [z_2, z_6]) was chosen as a pivot first !

Probability of comparing z_i with z_j

Pr{ z_i is compared to z_j } =

Pr{ z_i is the first pivot chosen from Z_{ij} }

+

Pr{ z_j is the first pivot chosen from Z_{ij} }

$= 1/(j - i + 1) + 1/(j - i + 1) = 2/(j - i + 1)$

- There are $j - i + 1$ elements between z_i and z_j
  - Pivot is chosen randomly and independently
  - The probability that any particular element is the first one chosen is $1/(j - i + 1)$

Number of Comparisons in PARTITION

Expected number of comparisons in PARTITION:

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr\{z_i \textit{ is compared to } z_j\}$$

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

(set k=j-i) (harmonic series)

$$= O(n \lg n)$$

$\Rightarrow$ Expected running time of Quicksort using RANDOMIZED-PARTITION is $O(n\lg n)$

Alternative Average-Case Analysis of Quicksort

- See Problem 7-2, page 16

- Focus on the expected running time of each individual recursive call to Quicksort, rather than on the number of comparisons performed.

- Use Hoare partition in our analysis.

Alternative Average-Case Analysis of Quicksort

- *Idea*: consider all possible splits and find their probability (assume all inputs are distinct)

- Given *L[p..r]* and *x=L[random(p,r)]* (pivot point), define

$$rank(x) = \text{number of elements} <= x$$

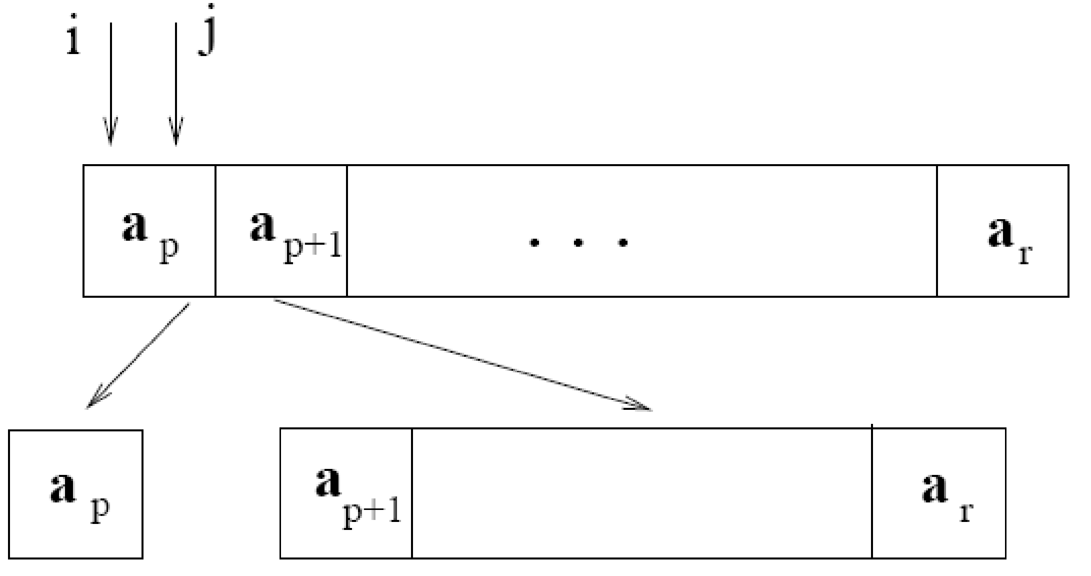
(if $L = [2, 1, 5, 4, 12]$ and $x = 5$, then $rank(5) = 4$)

- $P(rank(x) = i) = 1/n,\ i = 1, 2, \ldots, n\ (n = r - p + 1)$

(i.e., any element has the same probability to be chosen as pivot)

Alternative Average-Case Analysis of Quicksort

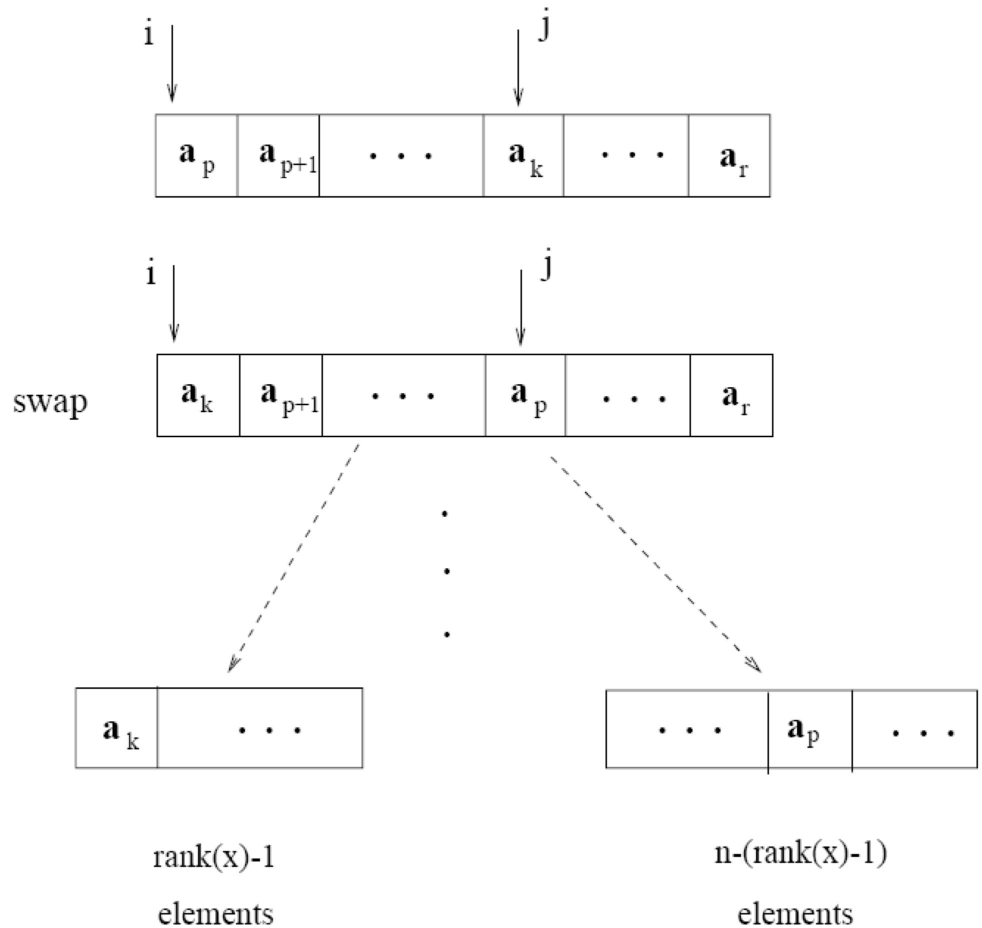
Case1: If $rank(x) = 1$, $x = a_p$



This is a $1: n-1$ split, $P(rank(x) = 1) = 1/n$

Alternative Average-Case Analysis of Quicksort

Case2: If $rank(x) \geq 2$, $x = a_p$



Alternative Average-Case Analysis of Quicksort

rank(x)	split	probability
2	**1 : n-1**	**P(rank(x)=2) = 1/n**
3	**2 : n-2**	**P(rank(x)=3) = 1/n**
.	.	.
.	.	.
.	.	.

- So, what are the possible splits and their probability ?

$$P(1: n-1) = 2/n,\ P(i: n-i) = 1/n\ (i = 2, 3, ..)$$

Alternative Average-Case Analysis of Quicksort

- What is the average time ? $(E(Q(n)))$

$$Q(n) = Q(q) + Q(n-q) + \Theta(n) \text{ or}$$

$$E(Q(n)) = E(Q(q) + Q(n-q)) + E(\Theta(n)) \text{ or}$$

$$E(Q(n)) = E(Q(q) + Q(n-q)) + \Theta(n)$$

- 1:n-1 splits have 2/n probability
- all other splits have 1/n probability

Alternative Average-Case Analysis of Quicksort

$$E(Q(n)) = 2/n(Q(1) + Q(n-1)) + 1/n \sum_{q=2}^{n-1} (Q(q) + Q(n-q)) + \Theta(n) =$$

$$1/n(Q(1) + Q(n-1)) + 1/n \sum_{q=1}^{n-1} (Q(q) + Q(n-q)) + \Theta(n)$$

- From worst-case analysis:

$$1/n(Q(1) + Q(n-1)) = 1/n(\Theta(1) + O(n^2)) = O(n)$$
(can be absorbed in the $\Theta(n)$)

Alternative Average-Case Analysis of Quicksort

$$E(Q(n)) = 1/n \sum_{q=1}^{n-1} (Q(q) + Q(n-q)) + \Theta(n) =$$

$$1/n[(Q(1) + Q(n-1)) + (Q(2) + Q(n-2)) + \ldots + (Q(n-1) + Q(1))] + \Theta(n) =$$

$$1/n[2Q(1) + 2Q(2) + \ldots + 2Q(n-1)] + \Theta(n) = 2/n \sum_{k=1}^{n-1} Q(k) + \Theta(n)$$

recurrence for average case: Q(n) = $2/n \sum_{k=1}^{n-1} Q(k) + \Theta(n)$

use substitution to show: $E(Q(n)) \leq c_1 n lg n + c_2$;

Problem

- Consider the problem of determining whether an arbitrary sequence $\{x_1, x_2, ..., x_n\}$ of *n* numbers contains repeated occurrences of some number. Show that this can be done in $\Theta(n\lg n)$ time.
  - Sort the numbers
    - $\Theta(n\lg n)$
  - Scan the sorted sequence from left to right, checking whether two successive elements are the same
    - $\Theta(n)$
  - Total
    - $\Theta(n\lg n)+\Theta(n)=\Theta(n\lg n)$

Ex 2.3-6 (page 37)

- Can we use Binary Search to improve InsertionSort (i.e., find the correct location to insert A[j]?)

Alg.: INSERTION-SORT(*A*)

for j ← 2 **to** n

do key ← *A*[j]

Insert *A*[j] into the sorted sequence *A*[1 . . j -1]

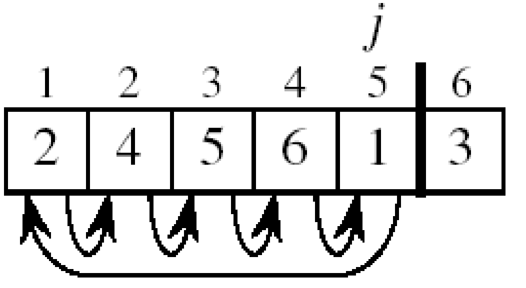
i ← j - 1

while i > 0 and *A*[i] > key

do *A*[i + 1] ← *A*[i]

i ← i - 1

A[i + 1] ← key



Ex 2.3-6 (page 37)

- Can we use binary search to improve InsertionSort (i.e., find the correct location to insert A[j]?)

  – This idea can reduce the number of comparisons from O(n) to O(lgn)
  – Number of shifts stays the same, i.e., O(n)
  – Overall, time stays the same ...
  – Worthwhile idea when comparisons are expensive (e.g., compare strings)

Problem

- Analyze the complexity of the following function:

```
F(i)
  if i=0
    then return 1
  return (2*F(i-1))
```

Recurrence: $T(n)=T(n-1)+c$

Use iteration to solve it .... $T(n)=\Theta(n)$

Problem

- What is the running time of Quicksort when all the elements are the same?

  - Using Hoare partition → best case
    - Split in half every time
    - $T(n)=2T(n/2)+n$ → $T(n)=\Theta(n\lg n)$

  - Using Lomuto's partition → worst case
    - 1:n-1 splits every time
    - $T(n)=\Theta(n^2)$