طراحی الگوریتم‌ها – مرتب‌سازی سریع

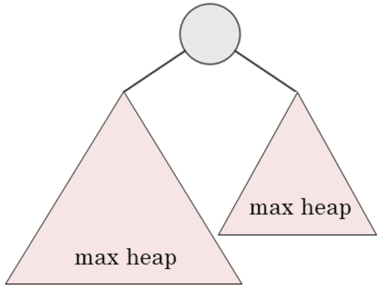
Introduction to Algorithm

مهدی جوانمردی

۱۴۰۱

مرور جلسه قبل

Max-Heapify



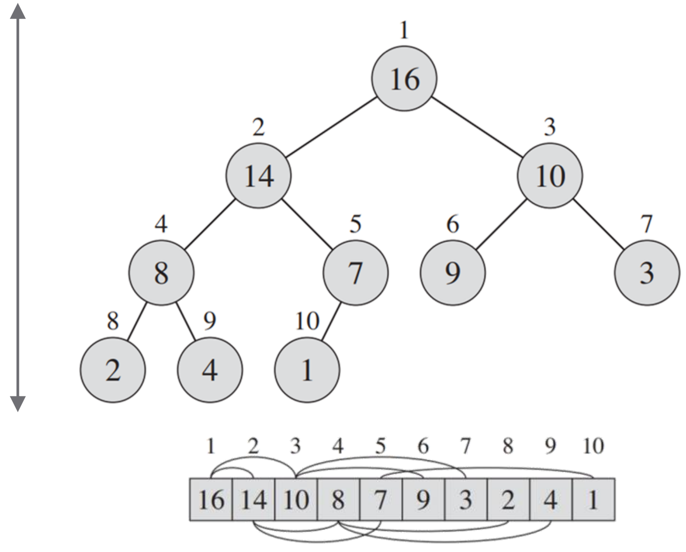
$\boxed{T(n) = O(\lg n)}$

```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i]
    largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
    largest = r
if largest ≠ i
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
```

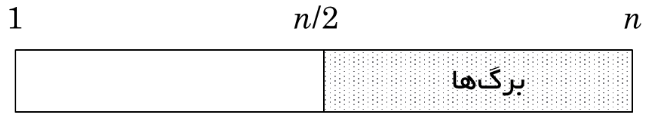
هرم یا Heap

height

$\Theta(\lg n)$



Build-heap & Heapsort




```
BUILD-MAX-HEAP(A)
A.heap-size = A.length
for i = ⌊A.length/2⌋ downto 1
    MAX-HEAPIFY(A, i)
```

$O(n)$

```
HEAPSORT(A)
BUILD-MAX-HEAP(A)                          O(n)
for i = A.length downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1              n times
    MAX-HEAPIFY(A, 1)                      O(lg n)
```

$O(n \lg n)$

عملیات‌های روی Heap

Maintain/Restore the max-heap property — $O(\lg n)$ $O(h)$
MAX-HEAPIFY

Create a max-heap from an unordered array — $O(n)$
BUILD-MAX-HEAP

Sort an array in place — $O(n \lg n)$
HEAPSORT

Priority queues

فصل هفتم: مرتب‌سازی سریع | Quicksort

- معرفی Quicksort
- بازدهی Quicksort
- نسخه تصادفی Quicksort
- تحلیل Quicksort

II Sorting and Order Statistics

مقایسه الگوریتم‌های مرتب‌سازی

only a constant number of elements of the input are ever stored outside the array.

- الگوریتم Insertion sort

مرتبه زمانی: worst-case running time $\Theta(n^2)$; expected running time $\Theta(n^2)$	$\boxed{\text{sorts } \textit{in place}}$	fast in-place sorting algorithm for small input

- الگوریتم Merge sort

مرتبه زمانی: running time $\Theta(n \lg n)$	~~sorts *in place*~~ MERGE procedure	

- الگوریتم Heap sort

مرتبه زمانی: worst-case running time $O(n \lg n)$	$\boxed{\text{sorts } \textit{in place}}$	important data structure, called a heap; priority queue

- الگوریتم Quick sort

مرتبه زمانی: worst-case running time $\Theta(n^2)$; expected running time $\Theta(n \lg n)$; outperforms heapsort in practice	$\boxed{\text{sorts } \textit{in place}}$	quicksort has tight code; popular algorithm for sorting large input arrays

مقایسه الگوریتم‌های مرتب‌سازی

$\boxed{\text{we can beat this lower bound of } \Omega(n \lg n)}$

if we can gather information about the sorted order of the input

- الگوریتم Counting sort

مرتبه زمانی

worst-case running time $\Theta(k + n)$

expected running time $\Theta(k + n)$

the input numbers are in the set $\{0, 1, \ldots, k\}$

- الگوریتم Radix sort

مرتبه زمانی

worst-case running time $\Theta(d(n + k))$

expected running time $\Theta(d(n + k))$

there are n integers to sort

integer has d digits

digit can take on up to k possible values

- الگوریتم Bucket sort

مرتبه زمانی

worst-case running time $\Theta(n^2)$

average-case running time $\Theta(n)$

requires knowledge of the probabilistic distribution of numbers in the input array

real numbers uniformly distributed in the half-open interval $[0, 1)$

مرتب‌سازی سریع یا Quicksort

quicksort has tight code.
popular algorithm for sorting large input arrays

sorts ***in place***

worst-case running time $\Theta(n^2)$
expected running time $\Theta(n \lg n)$
outperforms heapsort in practice

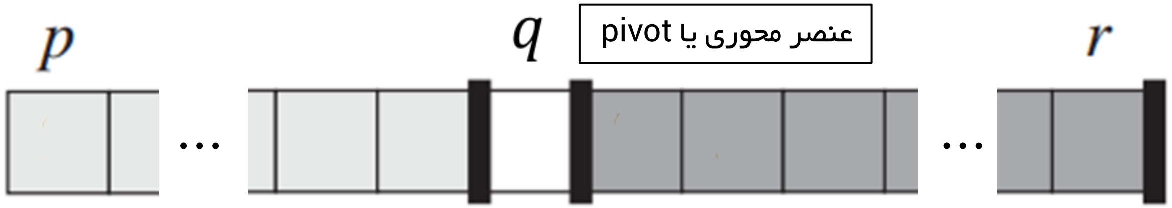
مرتبه زمانی

- Quicksort با وجود بدترین زمان اجرا $O(n^2)$ معمولا بهترین انتخاب مرتب‌سازی
- به دلیل expected خوب و ثابت‌های خیلی کوچک در تقریب O و درجا بودن عملیات
- به دلیل درجا بودن عملیات و جابجایی کم مناسب برای حافظه‌های مجازی Virtual Memory
- مشابه Mergesort از روش تقسیم و حل استفاده میکند

نحوه کار Quicksort

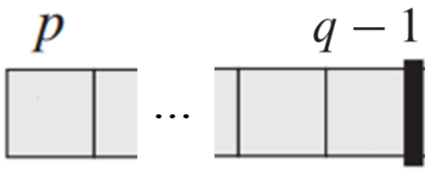
Partition (rearrange) the array $A[p\,..\,r]$

تقسیم



$$A[p\,..\,q-1] \;\le\; A[q] \;\le\; A[q+1\,..\,r]$$

حل

Quicksort(

)

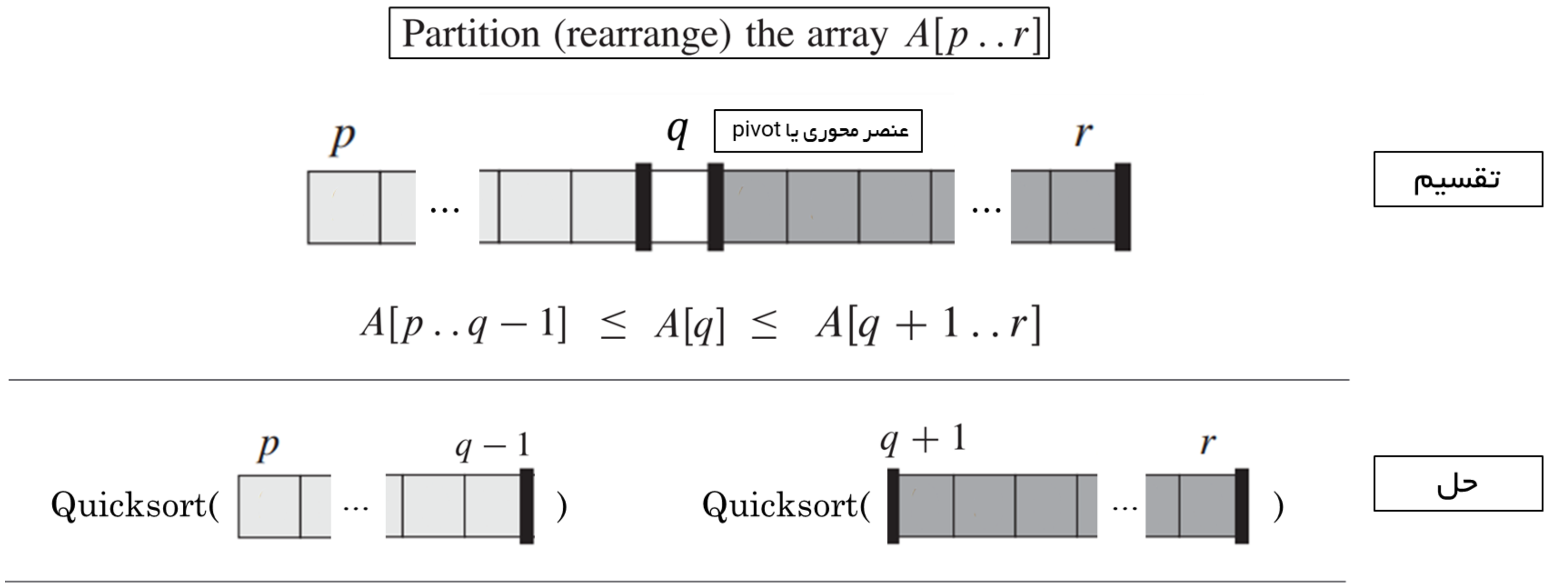
Quicksort(

)

ترکیب

$A[p\,..\,r]$ is now sorted

شبه کد Quicksort بصورت بازگشتی



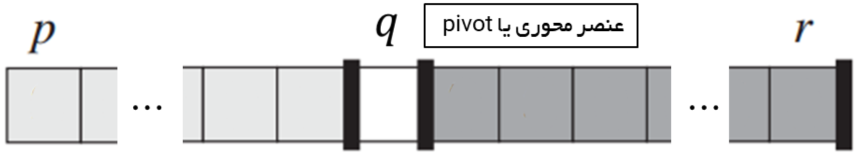
QUICKSORT(A, p, r)

```
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q - 1)
    QUICKSORT(A, q + 1, r)
```

اولین فراخوانی تابع:

$\boxed{\text{QUICKSORT}(A, 1, A.length).}$

نحوه کار Partition

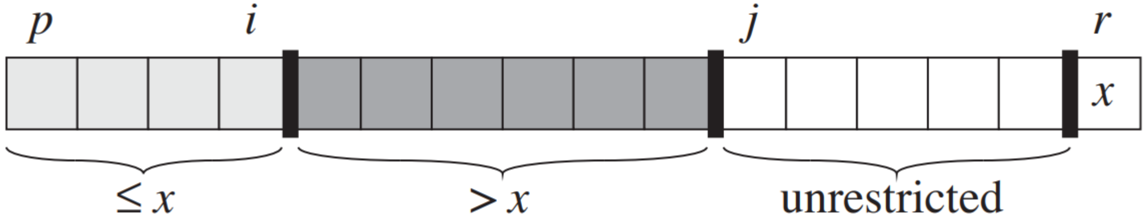


$$A[p \mathinner{.\,.} q-1] \le A[q] \le A[q+1 \mathinner{.\,.} r]$$

- انتخاب آخرین عنصر به عنوان pivot
- تغییر چیدمان عناصر آرایه با رعایت قوانین چهار بخش
- قرارداد pivot در جایگاه مناسب

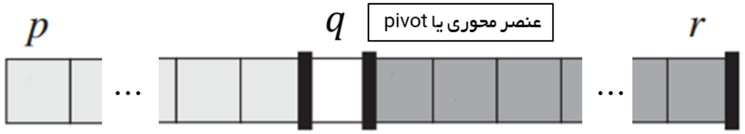
PARTITION(A, p, r)

```
x = A[r]
i = p - 1
for j = p to r - 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



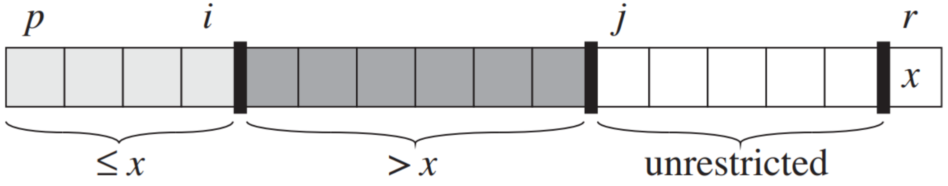
نمونه Partition

هدف نهایی

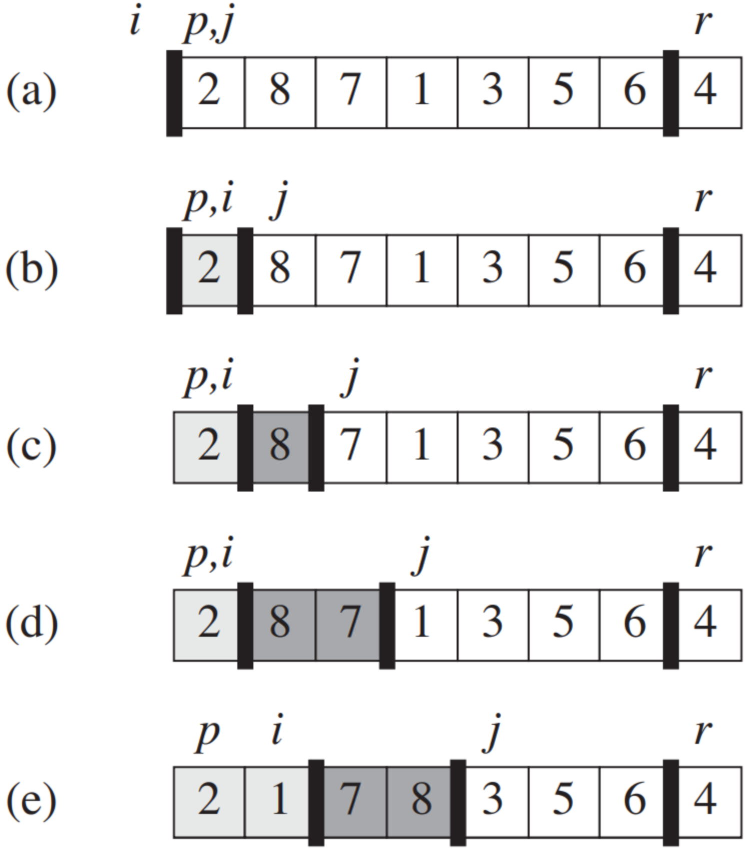


$$A[p \,..\, q-1] \;\le\; A[q] \;\le\; A[q+1 \,..\, r]$$

شرایط میانی

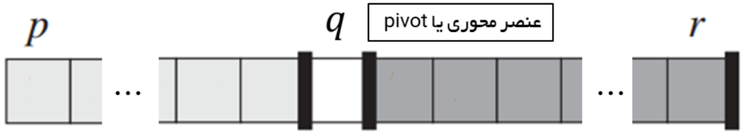



```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



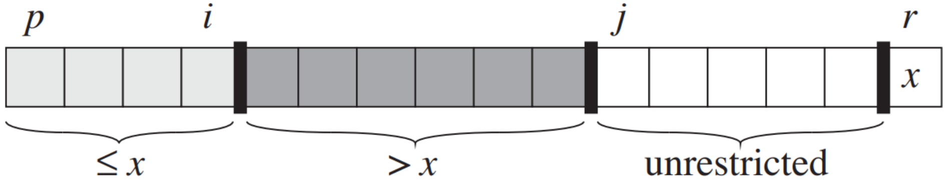
نمونه Partition

هدف نهایی



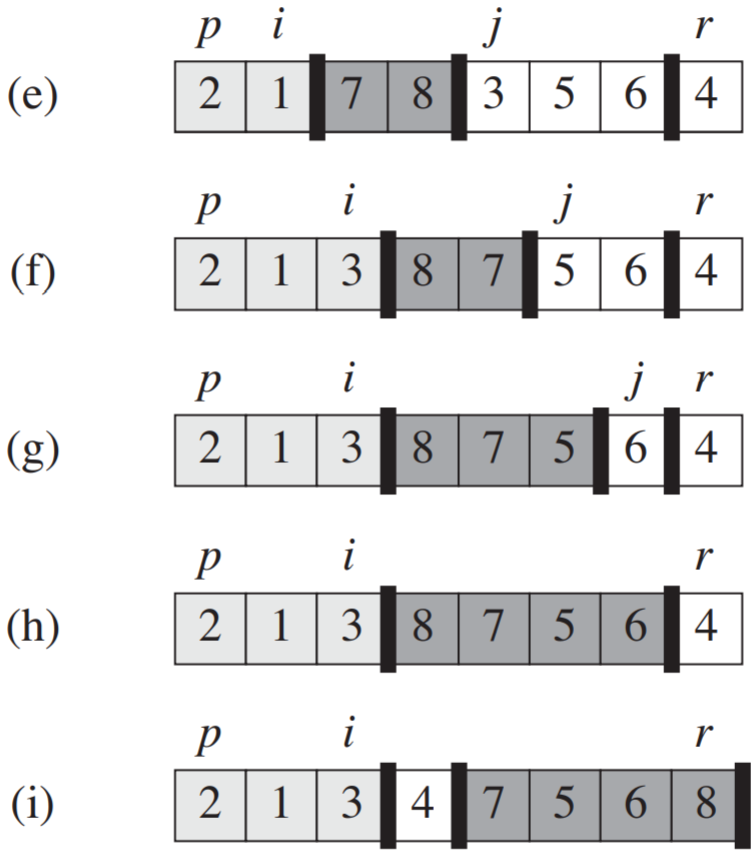
$$A[p\,..\,q-1] \le A[q] \le A[q+1\,..\,r]$$

شرایط میانی



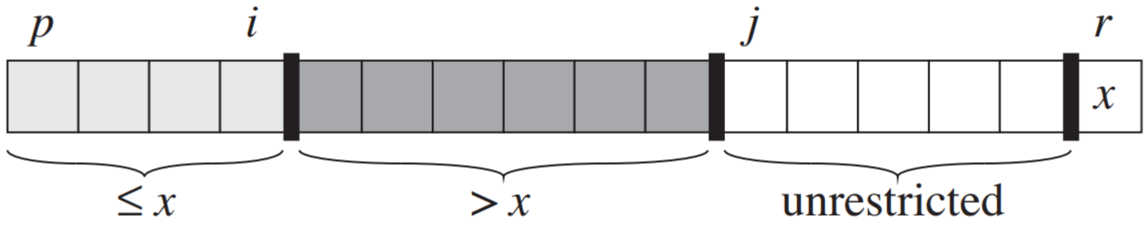
PARTITION(A, p, r)

```
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



اثبات شبه کد Partition


```
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
```



مستقل از حلقه

At the beginning of each iteration of the loop of lines 3–6, for any array index k,

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

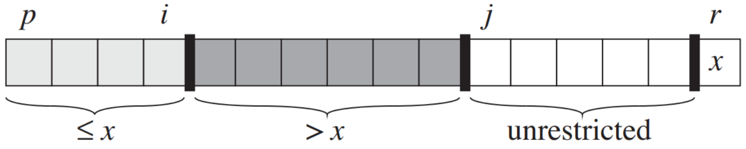
اثبات شبه کد Partition

```
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
```

مستقل از حلقه

At the beginning of each iteration of the loop of lines 3–6, for any array index k,

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 $j = p$

اثبات شبه کد Partition

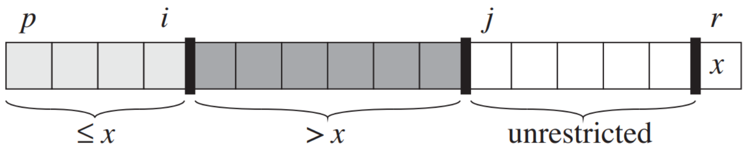
PARTITION(A, p, r)

```
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
```

مستقل از حلقه

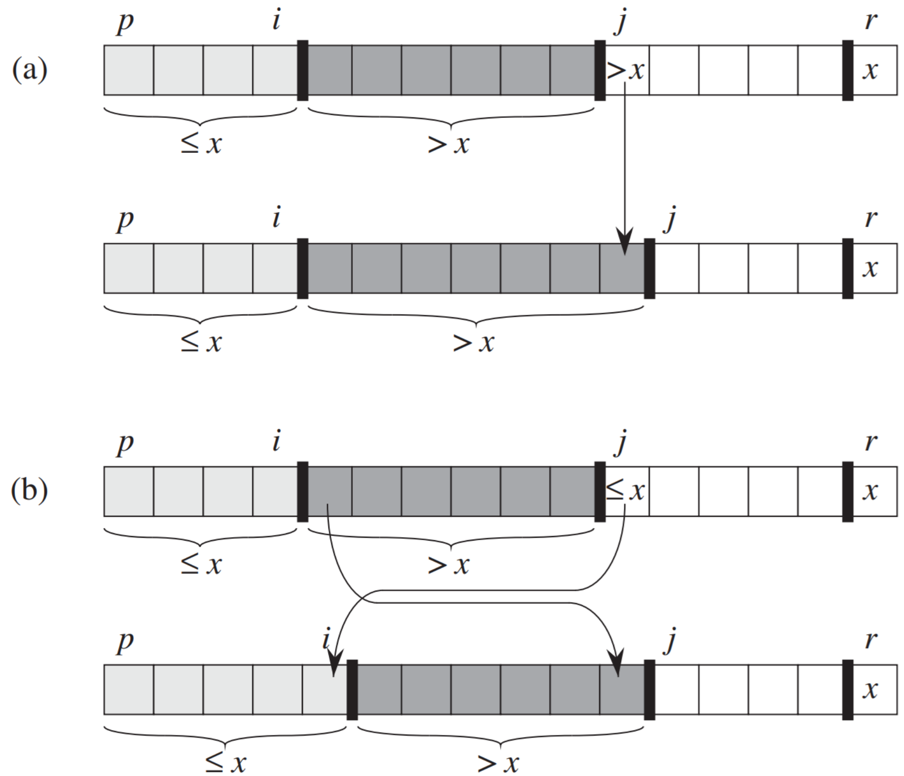
At the beginning of each iteration of the loop of lines 3–6, for any array index k,

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 $A[k] = x$.



$\boxed{A[j] > x}$

$\boxed{A[j] \le x}$



Maintenance: $\overset{\checkmark}{\boxed{j-1}} \longrightarrow \overset{?}{\boxed{j}}$

اثبات شبه کد Partition

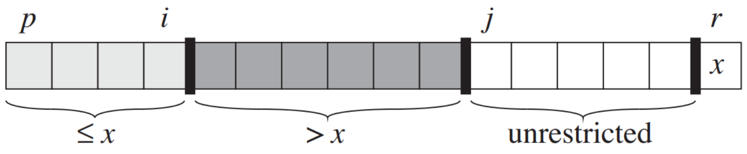
PARTITION(A, p, r)

1 $x = A[r]$
2 $i = p - 1$
3 **for** $j = p$ **to** $r - 1$
4   **if** $A[j] \leq x$
5         $i = i + 1$
6   exchange $A[i]$ with $A[j]$
7 exchange $A[i+1]$ with $A[r]$
8 **return** $i + 1$

مستقل از حلقه

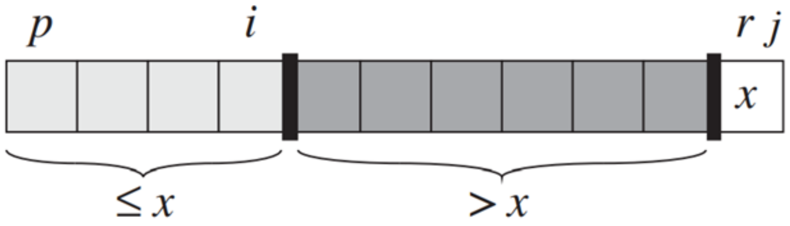
At the beginning of each iteration of the loop of lines 3–6, for any array index k,

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

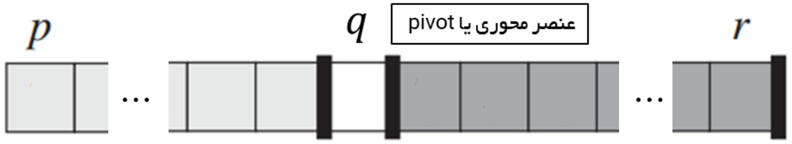


Termination: $j = r$

three sets:



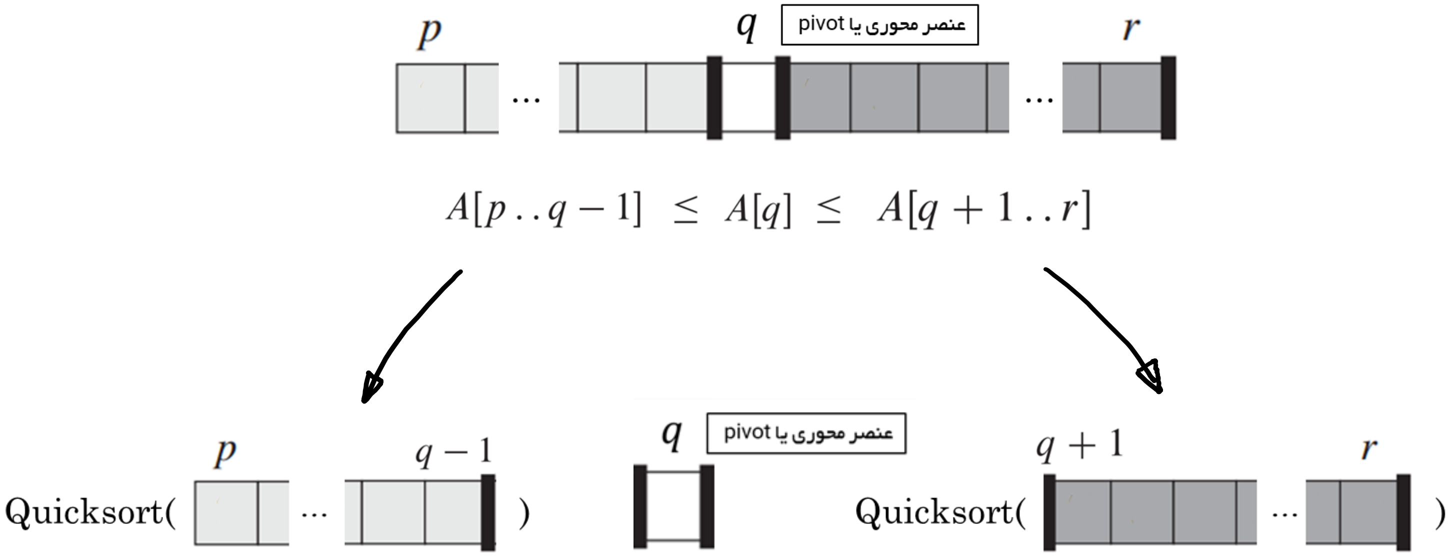
$\boxed{7 \quad \text{exchange } A[i+1] \text{ with } A[r]}$



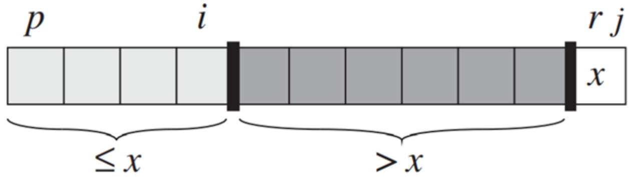
$A[p \,..\, q-1] \leq A[q] \leq A[q+1 \,..\, r]$

تحلیل زمان اجرای Quicksort

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

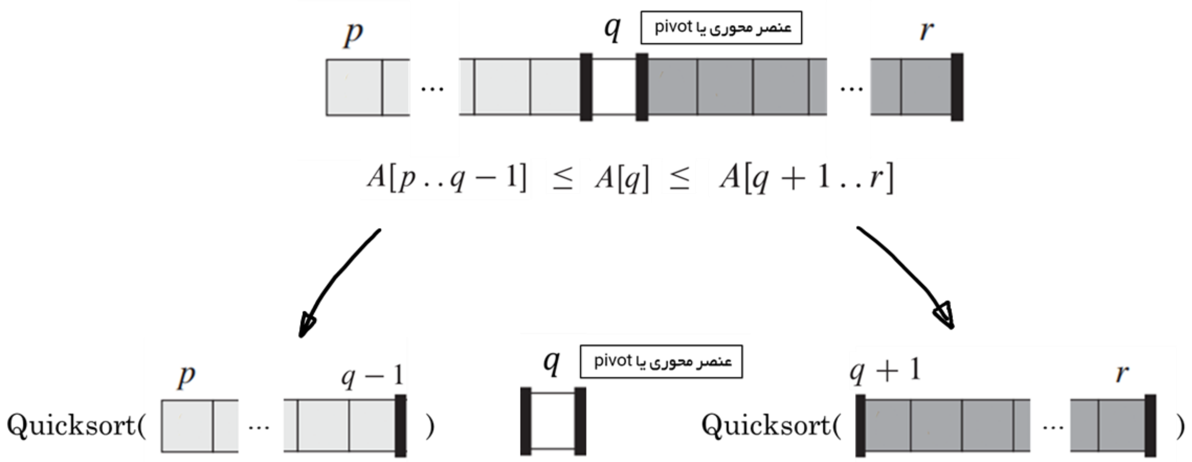


- چقدر متقارتن تقسیم شده؟
- چه حالتی بهترست؟
- چه پارامتری تعیین کننده است؟



تحلیل شهودی بدترین زمان اجرای Quicksort


```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



دارای صفر عنصر

دارای $n-1$ عنصر

$$\begin{aligned} T(n) &= T(n-1) + T(0) + \Theta(n) \\ &= T(n-1) + \Theta(n) . \end{aligned}$$

$\longrightarrow$

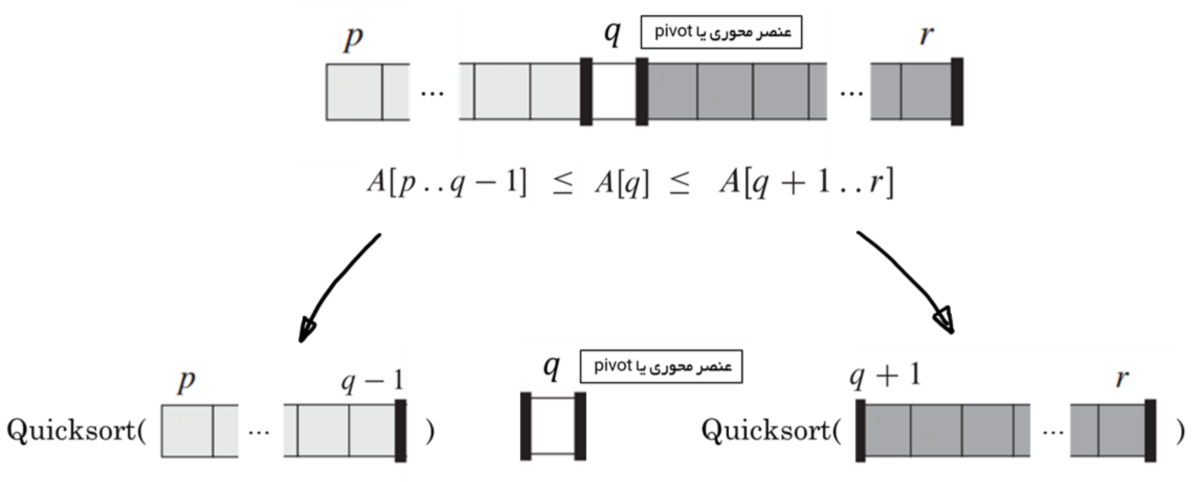
$$\begin{aligned} \sum_{k=1}^{n} k &= \frac{1}{2} n(n+1) \\ &= \Theta(n^2) . \end{aligned}$$

$\longrightarrow$ $\Theta(n^2)$

زمان اجرای quicksort برای آرایه مرتب؟ $\longleftarrow$ برای مرتب‌سازی درجی؟

تحلیل بهترین زمان اجرای Quicksort

```
PARTITION(A, p, r)
x = A[r]
i = p - 1
for j = p to r - 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



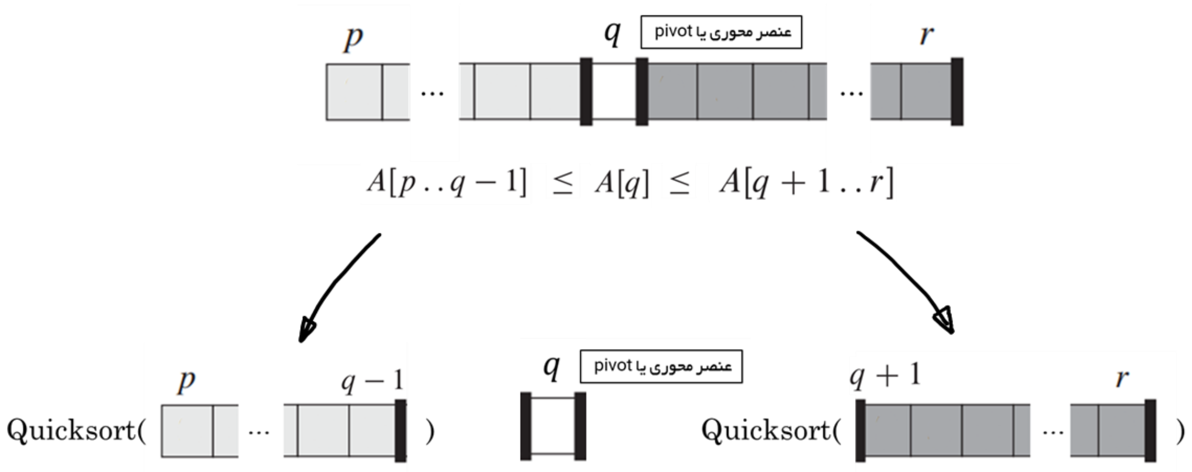
دارای $\lfloor n/2 \rfloor$ عنصر

دارای $\lceil n/2 \rceil - 1$ عنصر

$$T(n) = 2T(n/2) + \Theta(n) ,$$ → حالت دوم قضیه اصلی → $$T(n) = \Theta(n \lg n)$$

تحلیل متوسط زمان اجرای Quicksort

```
PARTITION(A, p, r)
x = A[r]
i = p - 1
for j = p to r - 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



دارای $n/10$ عنصر

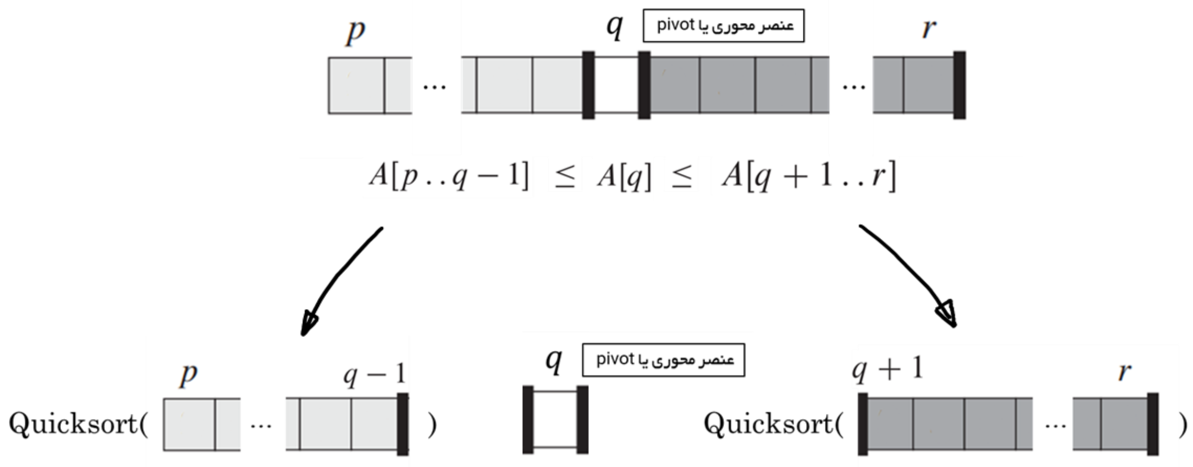
دارای $9n/10$ عنصر

$$T(n) = T(9n/10) + T(n/10) + cn$$ → حدس با درخت بازگشتی

تحلیل متوسط زمان اجرای Quicksort

متوسط زمان اجرای Quicksort به بهترین زمان اجرای آن بسیار نزدیک است

```
PARTITION(A, p, r)
    x = A[r]
    i = p − 1
    for j = p to r − 1
        if A[j] ≤ x
            i = i + 1
            exchange A[i] with A[j]
    exchange A[i + 1] with A[r]
    return i + 1
```



دارای $n/10$ عنصر

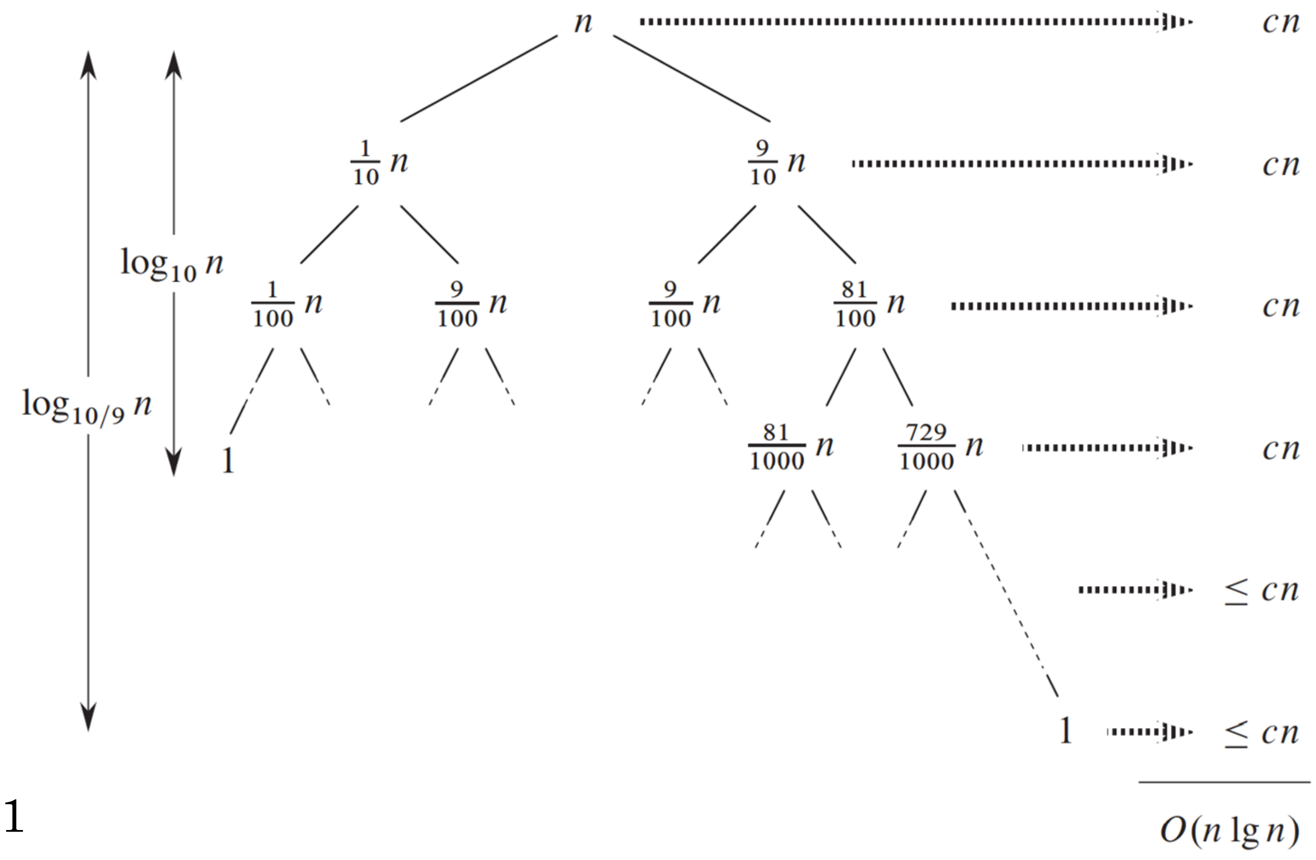
دارای $9n/10$ عنصر

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

→ حدس با درخت بازگشتی

تحلیل متوسط زمان اجرای Quicksort

$$\boxed{T(n) = T(9n/10) + T(n/10) + cn}$$ → حدس با درخت بازگشتی

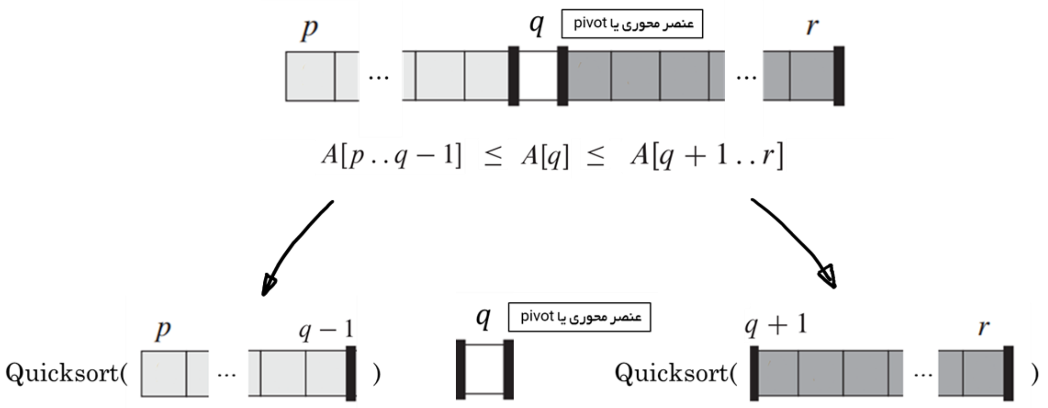


برای تقسیم ۹۹ به ۱ چطور؟؟

برای تقسیم ثابت همواره $T(n) = \Theta(n \lg n)$

تحلیل شهودی متوسط زمان اجرای Quicksort

```
PARTITION(A, p, r)
  x = A[r]
  i = p - 1
  for j = p to r - 1
      if A[j] ≤ x
          i = i + 1
          exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```



برای محاسبه دقیق زمان متوسط اجرای Quicksort نیاز به فرضیاتی نسبت به چیدمان ورودی داریم

صرفا چیدمان اهمیت دارد و نه خود اعداد ورودی

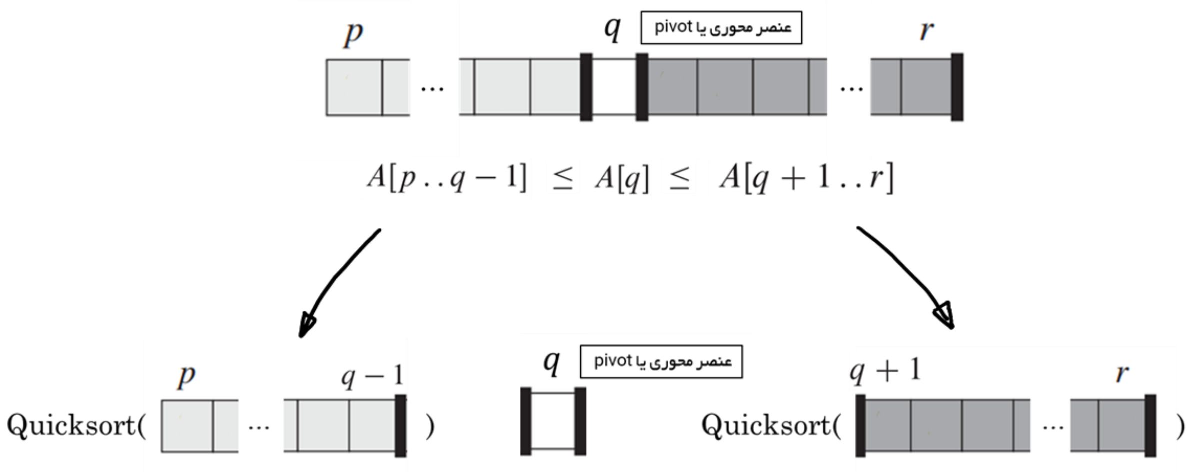
$\downarrow$

فرض: تمام جایگشت‌های اعداد ورودی با احتمال یکسان رخ میدهند

Exercise 7.2-6 about 80 percent of the time PARTITION produces a split that is more balanced than 9 to 1

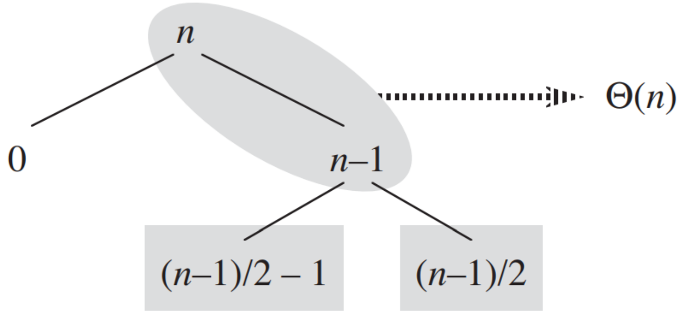
تحلیل شهودی متوسط زمان اجرای Quicksort

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

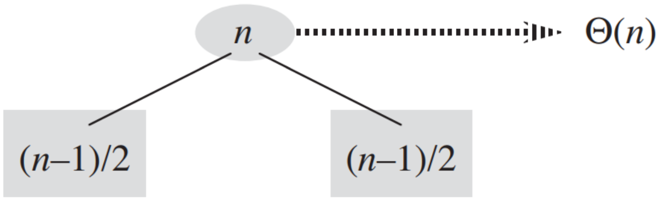


بار اول: دارای $n-1$ عنصر — دارای صفر عنصر

بار دوم: دارای $\lceil n/2 \rceil - 1$ عنصر — دارای $\lfloor n/2 \rfloor$ عنصر

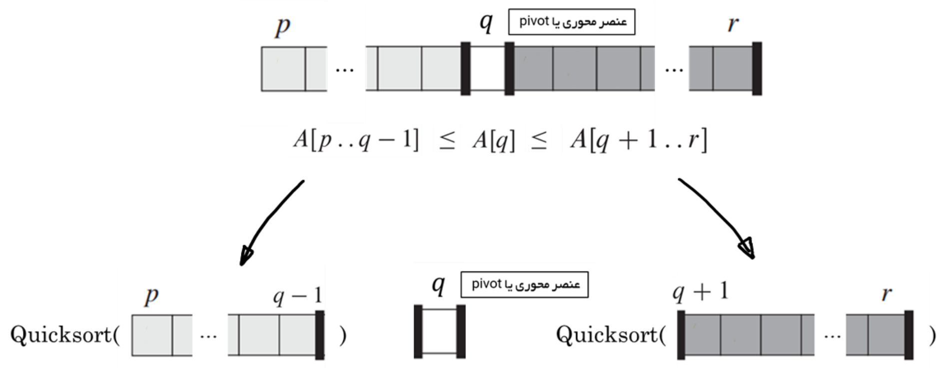


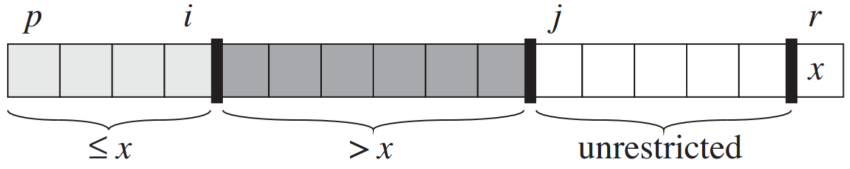
(a)



(b)

مدل تصادفی Quicksort





~~فرض: تمام جایگشت‌های اعداد ورودی با احتمال یکسان رخ میدهند~~

↓

random sampling

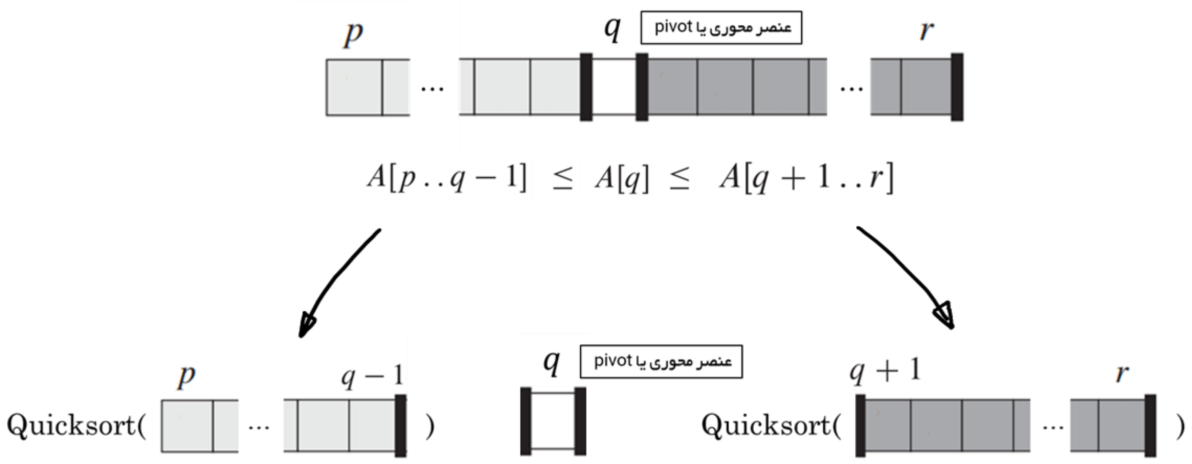
Pivot selection: $A[r] \longrightarrow$ Randomly choose from $A[p\,..\,r]$

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

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

تحلیل ریاضی بدتری زمان اجرای Quicksort


```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



دارای $\color{red}{n-q-1}$ عنصر

دارای $\color{red}{q}$ عنصر

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

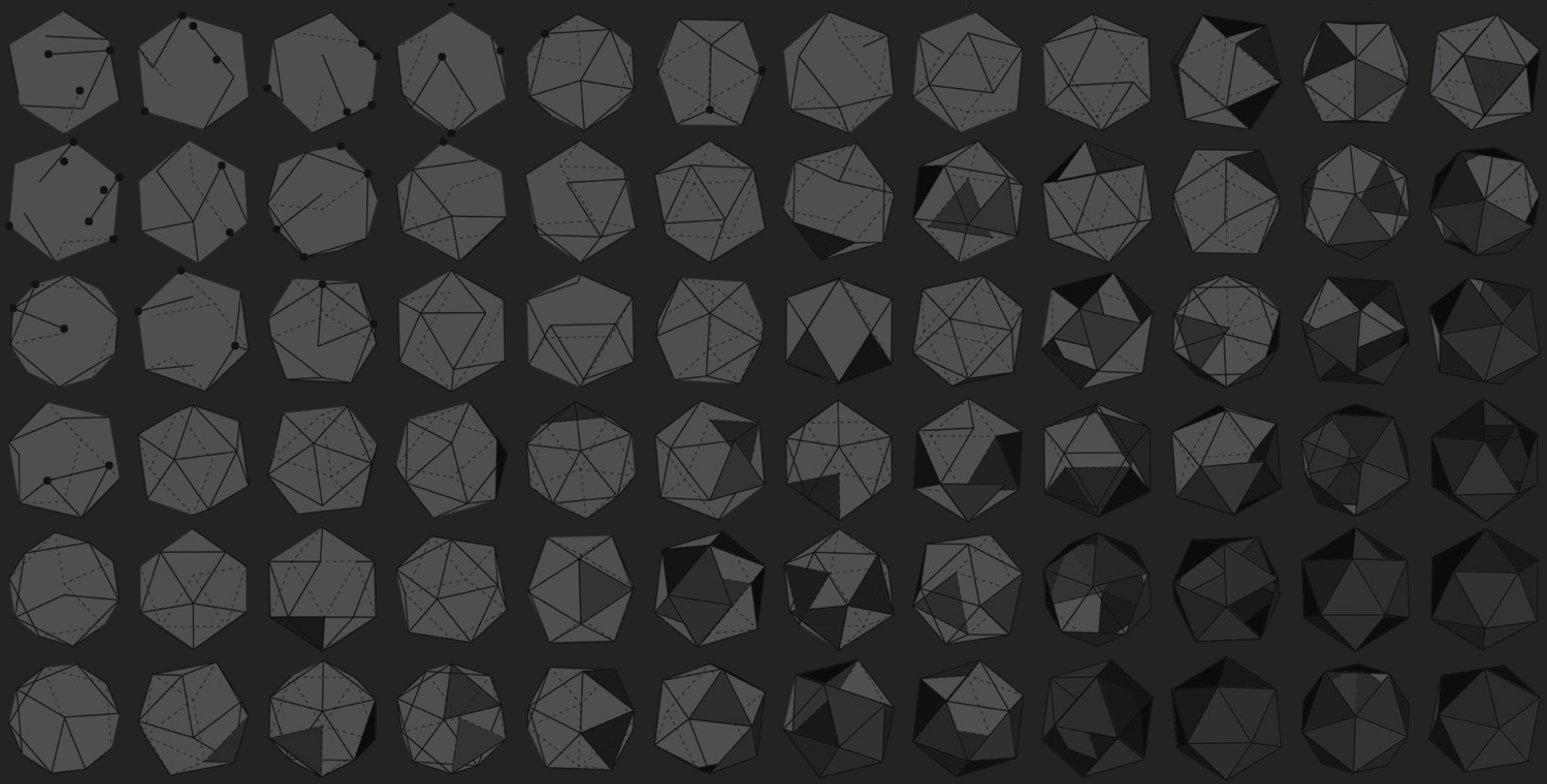
$$\boxed{O(n^2)}$$

$$\boxed{T(n) \leq cn^2}$$

$$\begin{aligned} T(n) &\leq \max_{0 \leq q \leq n-1} (cq^2 + c(n-q-1)^2) + \Theta(n) \\ &= c \cdot \max_{0 \leq q \leq n-1} (q^2 + (n-q-1)^2) + \Theta(n) . \end{aligned}$$

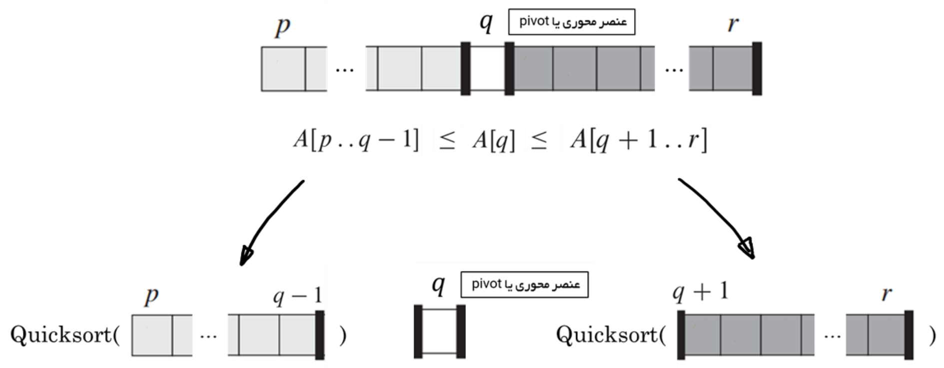
$$\begin{aligned} T(n) &\leq cn^2 - c(2n-1) + \Theta(n) \\ &\leq cn^2 , \end{aligned}$$

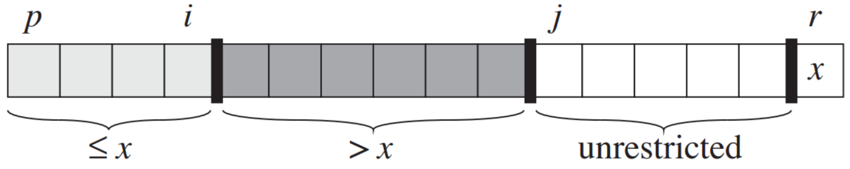
$$\max_{0 \leq q \leq n-1} (q^2 + (n-q-1)^2) \leq (n-1)^2$$



مدل تصادفی مرتب‌سازی سریع

مدل تصادفی Quicksort





~~فرض: تمام جایگشت‌های اعداد ورودی با احتمال یکسان رخ میدهند~~

↓

random sampling

Pivot selection: $A[r] \longrightarrow$ Randomly choose from $A[p..r]$

RANDOMIZED-PARTITION(A, p, r)

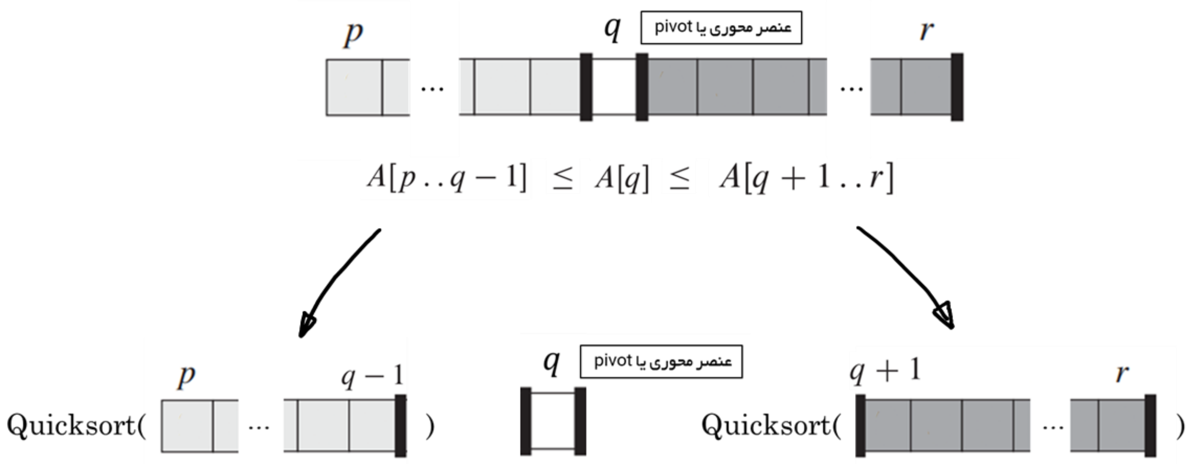
```
i = RANDOM(p, r)
exchange A[r] with A[i]
return PARTITION(A, p, r)
```

RANDOMIZED-QUICKSORT(A, p, r)

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

تحلیل ریاضی بدتری زمان اجرای Quicksort

```
Partition(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



دارای $n-q-1$ عنصر

دارای q عنصر

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

$$\boxed{O(n^2)}$$

$$\boxed{T(n) \le cn^2}$$

$$\begin{aligned} T(n) &\le \max_{0\le q\le n-1}(cq^2 + c(n-q-1)^2) + \Theta(n) \\ &= c \cdot \max_{0\le q\le n-1}(q^2 + (n-q-1)^2) + \Theta(n) . \end{aligned}$$

$$\begin{aligned} T(n) &\le cn^2 - c(2n-1) + \Theta(n) \\ &\le cn^2 , \end{aligned}$$

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