Quick Sort

TA 李佳樺

slides: https://reurl.cc/7014gN
records: https://reurl.cc/r973mO

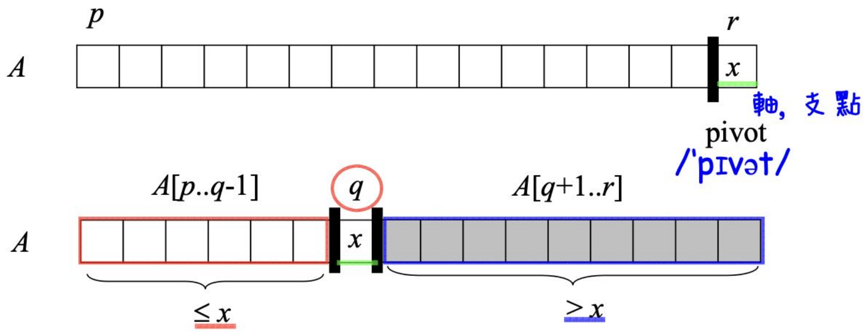
What is Quick Sort?

- Quick Sort is a divide and conquer algorithm, which relies on a partition operation:
    - to partition an array an element called a pivot is selected.
- All elements smaller than the pivot are moved before it, and all greater elements are moved after it.
- The lesser and greater sublists are then recursively sorted

Quick Sort

QuickSort(A[p…r])

Divide: use pivot partition A[p…r] into A[p…q-1] and A[q+1…r]



Conquer: recursively sort A[p...q-1] and A[q+1...r]

Quick Sort

QuickSort(A, p, r)

```
if p < r then
    q := Partition(A, p, r)       /* divide */
    QuickSort(A, p, q - 1)        /* conquer */
    QuickSort(A, q + 1, r)        /* conquer */
```

Partition

Partition(A, p, r)

```
i := p
for j := p to r - 1
    if A[j] <= A[r] then
        exchange(A[i], A[j])
        i := i + 1
exchange(A[i], A[r])
return i
```

Partition

take A[r] as pivot!

7	5	4	8	9	5

Partition(A, p, r)

```
i := p
for j := p to r - 1
    if A[j] <= A[r] then
        exchange(A[i], A[j])
        i := i + 1
exchange(A[i], A[r])
return i
```

Partition

take A[r] as pivot!

Partition(A, p, r)

```
i := p

for j := p to r - 1

    if A[j] <= A[r] then

        exchange(A[i], A[j])

        i := i + 1

exchange(A[i], A[r])

return i
```

Partition

take A[r] as pivot!

Partition(A, p, r)

```
i := p

for j := p to r - 1

    if A[j] <= A[r] then

        exchange(A[i], A[j])

        i := i + 1

exchange(A[i], A[r])

return i
```

Partition

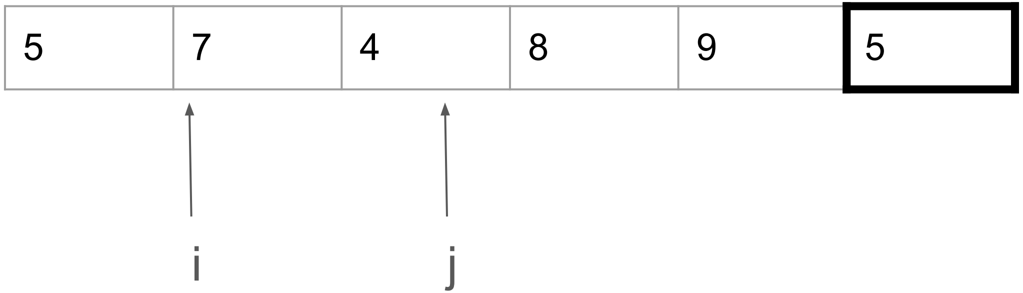
take A[r] as pivot!

Partition(A, p, r)

```
i := p
for j := p to r - 1
    if A[j] <= A[r] then
        exchange(A[i], A[j])
        i := i + 1
exchange(A[i], A[r])
return i
```

Partition

take A[r] as pivot!

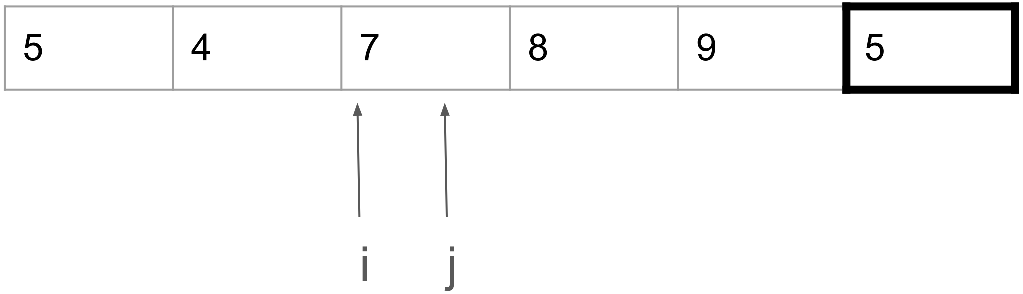


Partition(A, p, r)

```
i := p

for j := p to r - 1

    if A[j] <= A[r] then

        exchange(A[i], A[j])

        i := i + 1

exchange(A[i], A[r])

return i
```

Partition

take A[r] as pivot!

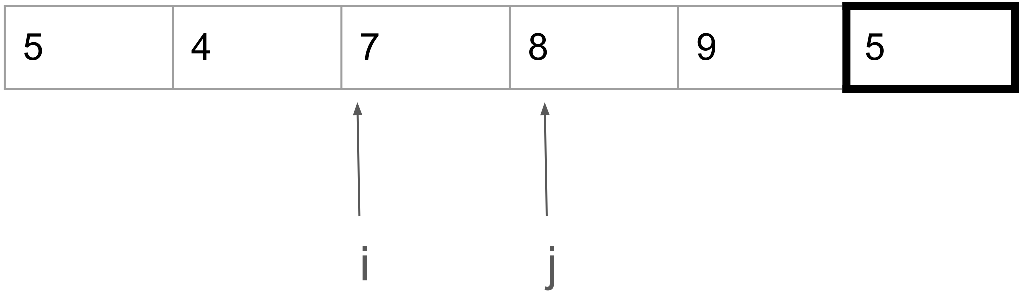


Partition(A, p, r)

```
i := p
for j := p to r - 1
    if A[j] <= A[r] then
        exchange(A[i], A[j])
        i := i + 1
exchange(A[i], A[r])
return i
```

Partition

take A[r] as pivot!




```
Partition(A, p, r)

i := p

for j := p to r - 1

    if A[j] <= A[r] then

        exchange(A[i], A[j])

        i := i + 1

exchange(A[i], A[r])

return i
```

Partition

take A[r] as pivot!

Partition(A, p, r)

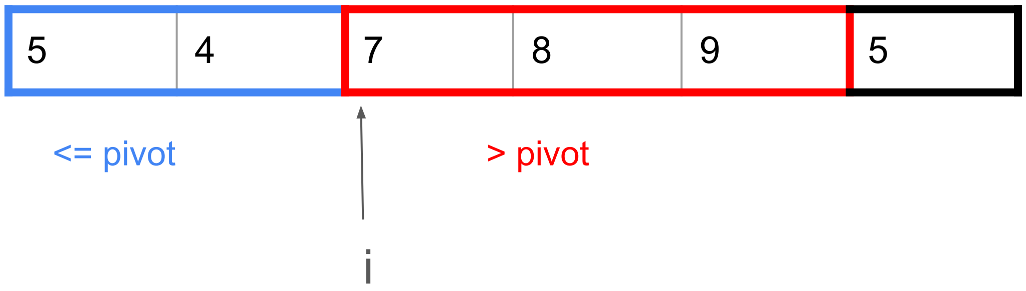
```
i := p
for j := p to r - 1
    if A[j] <= A[r] then
        exchange(A[i], A[j])
        i := i + 1
exchange(A[i], A[r])
return i
```

Partition

Partition(A, p, r)

```
i := p
for j := p to r - 1
    if A[j] <= A[r] then
        exchange(A[i], A[j])
        i := i + 1
exchange(A[i], A[r])
return i
```

take A[r] as pivot!

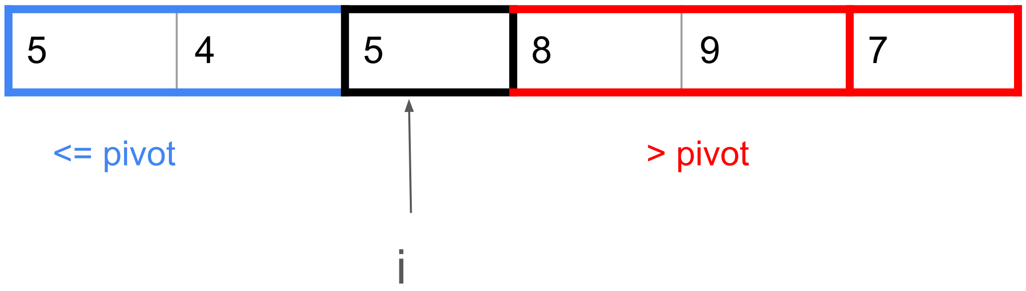


Partition

Partition(A, p, r)

```
i := p
for j := p to r - 1
    if A[j] <= A[r] then
        exchange(A[i], A[j])
        i := i + 1
exchange(A[i], A[r])
return i
```

take A[r] as pivot!

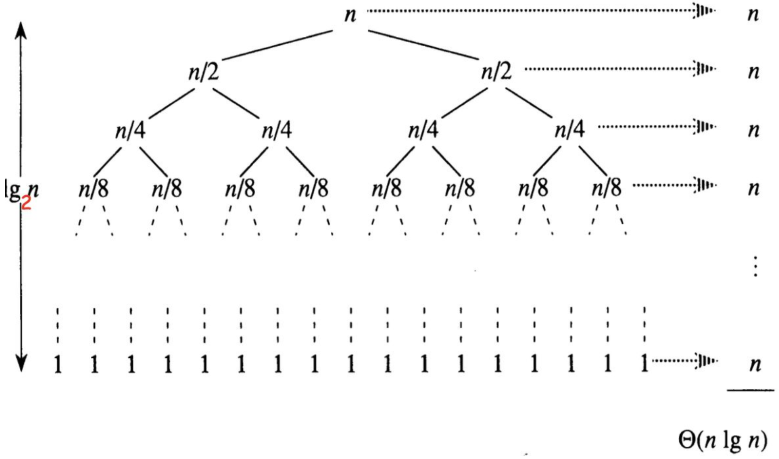


Analysis

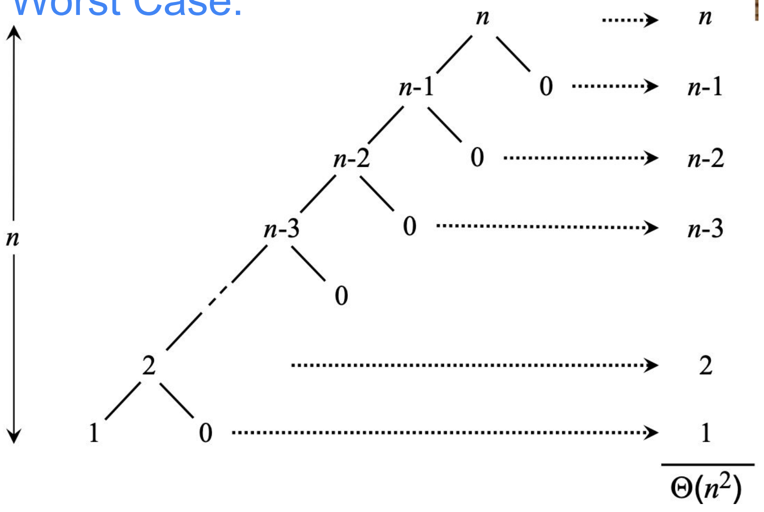
Best Case: choose the medium as pivot

Worst Case: for example A = {1, 2, 3, …, n - 2, n - 1, n}

Best Case:



Worst Case:



Analysis

Average Case:

$$T(N) = (N-1) + \frac{1}{N} \cdot \sum_{i=1}^{N} (T(i-1) + T(N-i))$$

$$= (N-1) + \frac{2}{N} \cdot \sum_{i=1}^{N-1} T(i)$$

For simplicity, assume

$$T(N) = (N+1) + \frac{2}{N} \cdot \sum_{i=1}^{N-1} T(i)$$

$$\Rightarrow N \cdot T(N) = N^2 + N + 2\sum_{i=1}^{N-1} T(i) \qquad ---(1)$$

$$\Rightarrow (N-1) \cdot T(N-1) = (N-1)^2 + N - 1 + 2\sum_{i=1}^{N-2} T(i) \qquad ---(2)$$

(1) - (2), we have

Analysis

Average Case:

$$\Rightarrow N \cdot T(N) = N^2 + N + 2\sum_{i=1}^{N-1} T(i) \qquad ---(1)$$

$$\Rightarrow (N-1) \cdot T(N-1) = (N-1)^2 + N - 1 + 2\sum_{i=1}^{N-2} T(i) \qquad ---(2)$$

(1) - (2), we have

$$N \cdot T(N) = (N+1)T(N-1) + 2N$$

$$\Rightarrow T(N) = \frac{N+1}{N}T(N-1) + 2$$

$$= \frac{N+1}{N} \cdot \frac{N}{N-1}T(N-2) + 2\frac{N+1}{N} + 2$$

$$= \ \ldots$$

$$= \frac{N+1}{2}T(1) + 2(N+1)(\frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{N}) + 2$$

$$= \Theta(N) + \Theta(N) \cdot \Theta(\lg N) + 2$$

$$= \Theta(N \lg N)$$

Randomized-Partition

A simple way to prevent almost sorted or attack

Randomized-Partition(A, p, r)

```
i := Random(p, r)

exchange(A[i], A[r])

return Partition(A, p, r)
```

Quick Select

What is Quick Select?

- Selection in expected linear time
- Quick Select is a prune-and-search algorithm, which relies on a partition operation
- If the pivot is the kth element we're looking for, return it.
- Otherwise, recursively select from the appropriate sublist:
  - If k <= size of left sublist, select from the left sublist
  - Else, select from the right sublist, adjusting k accordingly

Quick Select

QuickSelect(A, p, r, k)

```
if p = r then return A[p]

q := Partition(A, p, r)                         /* prune */

i := q - p + 1

if k = i then return A[q]

else if k < i then

    return QuickSelect(A, p, q - 1, k)      /* search */

else return QuickSelect(A, q + 1, r, k - i) /* search */
```

Analysis

Worst Case: T(N) = O(N) + T(N - 1) = O(N^2)

Average Case:

$$
\begin{aligned}
E(N) &= O(N) + \frac{1}{N}\sum_{i=1}^{N} E(max(i-1, N-i)) \\
&= O(N) + \frac{2}{N}\sum_{i=\lfloor \frac{N}{2} \rfloor}^{N-1} E(i) \\
&= O(N)
\end{aligned}
$$

Practice

Quick Sort Implementation

QuickSort(A, p, r)

https://gist.github.com/LJH-coding/f236b6044c842e67ff58a005208d6f79

NTHUOJ 14148

https://acm.cs.nthu.edu.tw/problem/14148/

find out the k-th highest element in the array

use QuickSelect!

https://gist.github.com/LJH-coding/4420611126cf6dcc9ebe905f923e263c