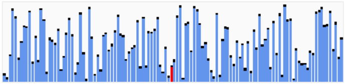
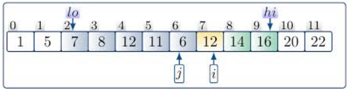
THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Algorithms and Data Structures

Lecture 10 Quicksort

Jiamou Liu
The University of Auckland





Divide and Conquer Sorting Algorithms

Mergesort and **Quicksort**: Both are Divide and conquer algorithms

- Split the input list into two sublists
- Recursively sort each sublists
- Combine the sorted sublists

Mergesort

- Splitting is very easy
- The comparisons are done during combining

Quicksort

- The comparisons are done during splitting
- Combining is very easy

Quicksort

- Choose a pivot element and partition the list into two sublists:
  - **Left sublist** with elements $\leq$ pivot
  - **Right sublist** with elements $>$ pivot.
- Recursively sort left and right sublists
- Combine the sorted sublists

Quicksort

- Choose a pivot element and partition the list into two sublists:
    - **Left sublist** with elements $\leq$ pivot
    - **Right sublist** with elements $>$ pivot.
- Recursively sort left and right sublists
- Combine the sorted sublists

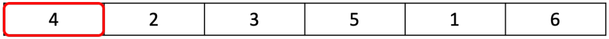
Question. How to choose the pivot element?

Use the first entry as pivot element for a basic presentation.

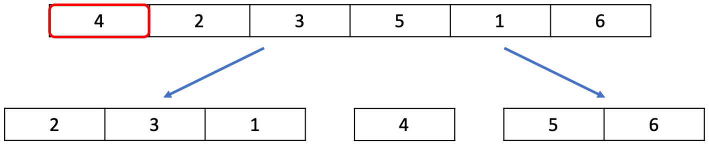
Example.

4	2	3	5	1	6

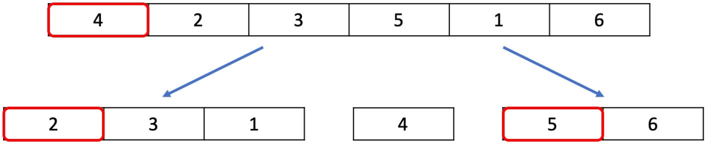
Example.



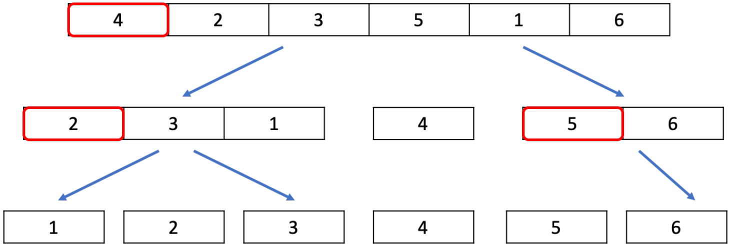
Example.



Example.



Example.



Algorithm 1 Quicksort - basic

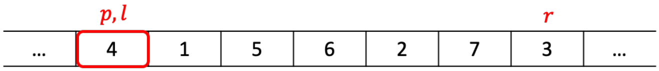
```
function QUICKSORT(list $a[0..n-1]$, integer $i$, integer $j$)
    if $i < j$ then
        $q \leftarrow$ PARTITION$(a, i, j)$          ▷ put pivot in correct position
        QUICKSORT$(a, i, q-1)$                       ▷ sort left half
        QUICKSORT$(a, q+1, j)$                       ▷ sort right half
```

Linear time partitioning - Hoare’s method

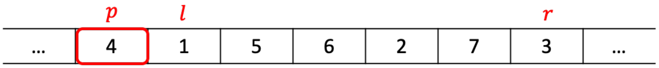
Example.

	i						j	
...	4	1	5	6	2	7	3	...

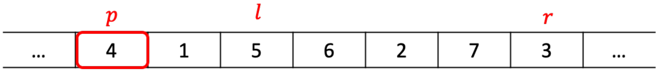
Example.



Example.

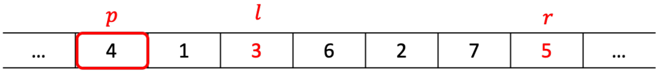


Example.

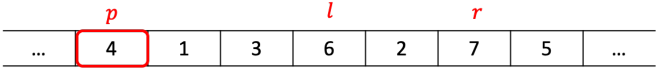


Example.

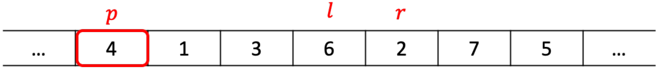
Swap 5 and 3



Example.

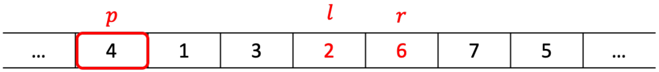


Example.

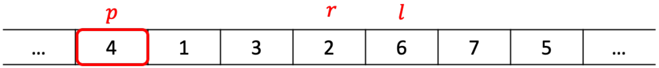


Example.

Swap 2 and 6

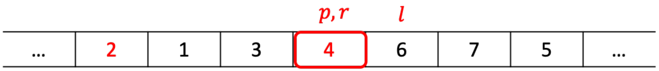


Example.

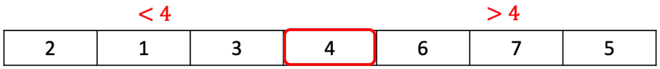


Example.

Swap pivot



Example.



Linear time partitioning - Hoare's method

Given a list L and a pivot p of L, partition so that all elements to the left of L are $\leq p$ and all to the right are $> q$.

- Start with pointers at opposite ends of the list. Stop when pointers cross
- At each step,
  - Increment the left pointer until we reach an element $> p$
  - Decrement the right one until we reach an element $\leq p$
- Swap these elements and continue. When pointers cross, swap right pointer with pivot

Algorithm 2 Partition - Hoare's method.

```
function PARTITION(list $a[0..n-1]$, integer $i$, integer $j$)
    $p \leftarrow a[i]$                                ▷ pivot element
    $l \leftarrow i$                                   ▷ left pointer
    $r \leftarrow j + 1$                               ▷ right pointer
    while True do
        repeat
            $l \leftarrow l + 1$                       ▷ find big element
        until $a[l] \geq p$
        repeat
            $r \leftarrow r - 1$                       ▷ find small element
        until $a[r] \leq p$
        if $l < r$ then
            swap($a, l, r$)                            ▷ swap big and small elements
        else
            swap($a, i, r$)                            ▷ put pivot in correct place
            return $r$
```

Quicksort analysis

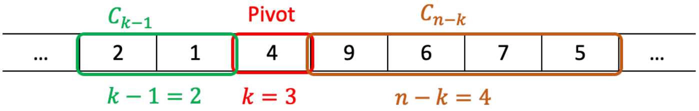
Let C_n denote the number of comparisons performed by Quicksort over a list with n elements.
Assume pivot is k-th element in the sorted list.

C_n consists of:

- Number of comparisons for the left sublist, C_{k-1}
- Number of comparisons for the right sublist, C_{n-k}
- Number of comparisons for Partition, $n-1$

Recurrence relation:

$$C_n = C_{k-1} + C_{n-k} + n - 1$$

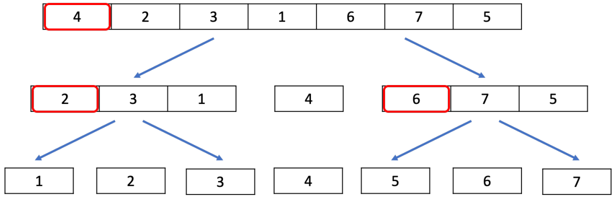


Quicksort analysis - Best case

Question. Which inputs give the best case?

Quicksort analysis - Best case

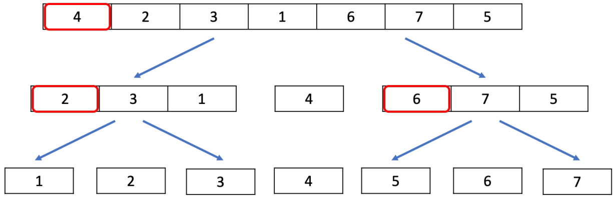
Question. Which inputs give the best case?



Quicksort analysis - Best case

Question. Which inputs give the best case?

When the pivot is the median element, both sublists have equal length.

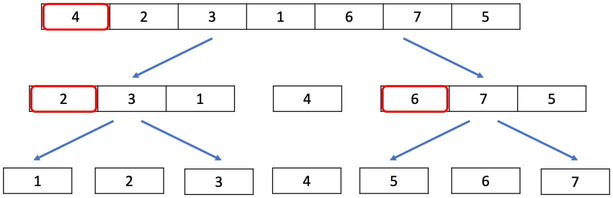


Quicksort analysis - Best case

Question. Which inputs give the best case?

When the pivot is the median element, both sublists have equal length.

- $\approx \log n$ levels of recursion
- Each level of recursion takes $\Theta(n)$ time
- Therefore $\Theta(n \log n)$



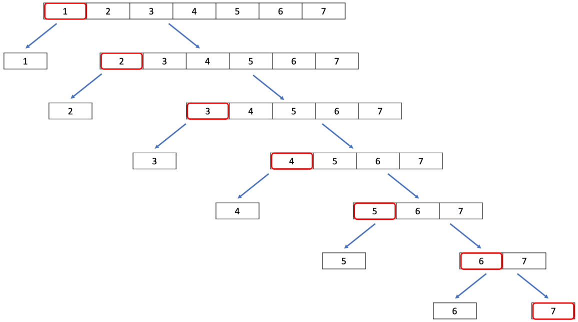
Quicksort analysis - Worst case

Question. Which inputs give the worst case?

Quicksort analysis - Worst case

Question. Which inputs give the worst case?

When the list is already sorted, one side of the pivot is always empty, and the other side is $n - 1$

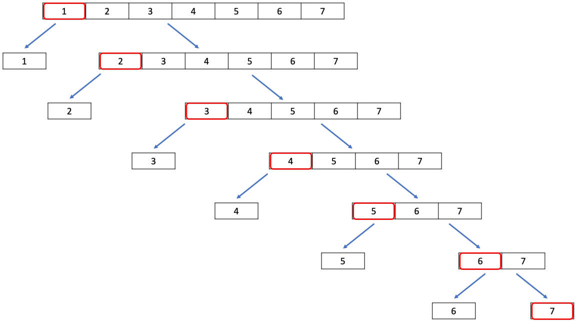


Quicksort analysis - Worst case

Question. Which inputs give the worst case?

When the list is already sorted, one side of the pivot is always empty, and the other side is $n - 1$

- n levels of recursion
- Each level of recursion takes $\Theta(n)$ time
- Therefore $\Theta(n^2)$



Quicksort analysis - Average case

Equal probability assumption

Assume all input permutations appear with equal probability, i.e., for $1 \le j \le n$, $k = j$ with probability $\frac{1}{n}$.

Recurrence relation: $C_n = C_{k-1} + C_{n-k} + n - 1$

Let a_n be the average of C_n.

$$\begin{aligned} a_n &= \frac{1}{n}\left(\underbrace{(a_0 + a_{n-1})}_{k=1} + \underbrace{(a_1 + a_{n-2})}_{k=2} + \ldots + \underbrace{(a_{n-2} + a_1)}_{k=n-1} + \underbrace{(a_{n-1} + a_0)}_{k=n}\right) + n - 1 \\ &= \frac{2}{n}(a_0 + a_1 + \ldots + a_{n-2} + a_{n-1}) + n - 1 \\ &= \frac{2}{n}\sum_{j=0}^{n-1} a_j + n - 1 \end{aligned}$$

Eliminate the history

$$a_n = \frac{2}{n}\sum_{j=0}^{n-1} a_j + n - 1$$

$$na_n = 2\sum_{j=0}^{n-1} a_j + n(n-1) \tag{1}$$

$$(n-1)a_{n-1} = 2\sum_{j=0}^{n-2} a_j + (n-1)(n-2) \tag{2}$$

(1) − (2)

$$na_n - (n-1)a_{n-1} = \left(2\sum_{j=0}^{n-1} a_j + n(n-1)\right) - \left(2\sum_{j=0}^{n-2} a_j + (n-1)(n-2)\right)$$

$$\begin{aligned} na_n - (n-1)a_{n-1} &= 2a_{n-1} + 2(n-1) \\ na_n &= (2+n-1)a_{n-1} + 2(n-1) \\ na_n &= (n+1)a_{n-1} + 2(n-1) \end{aligned}$$

(1) − (2)

$$na_n - (n-1)a_{n-1} = \left(2\sum_{j=0}^{n-1} a_j + n(n-1)\right) - \left(2\sum_{j=0}^{n-2} a_j + (n-1)(n-2)\right)$$

$$na_n - (n-1)a_{n-1} = 2a_{n-1} + 2(n-1)$$

$$na_n = (2+n-1)a_{n-1} + 2(n-1)$$

$$na_n = (n+1)a_{n-1} + 2(n-1)$$

Divide $n(n+1)$ on both sides, we get

$$\frac{a_n}{n+1} = \frac{2(n-1)}{n(n+1)} + \frac{a_{n-1}}{n} = \frac{a_{n-1}}{n} + \frac{4}{n+1} - \frac{2}{n}$$

Telescoping

$$\begin{aligned}
\frac{a_n}{n+1} &= \frac{a_{n-1}}{n} + \frac{4}{n+1} - \frac{2}{n} \\
\frac{a_{n-1}}{n} &= \frac{a_{n-2}}{n-1} + \frac{4}{n} - \frac{2}{n-1} \\
\frac{a_{n-2}}{n-1} &= \frac{a_{n-3}}{n-2} + \frac{4}{n-1} - \frac{2}{n-2} \\
\ldots &= \ldots + \ldots - \ldots \\
\frac{a_2}{3} &= \frac{a_1}{2} + \frac{4}{3} - \frac{2}{2}
\end{aligned}$$

Telescoping

$$
\begin{aligned}
\frac{a_n}{n+1} &= \frac{a_{n-1}}{n} + \frac{4}{n+1} - \frac{2}{n} \\
\frac{a_{n-1}}{n} &= \frac{a_{n-2}}{n-1} + \frac{4}{n} - \frac{2}{n-1} \\
\frac{a_{n-2}}{n-1} &= \frac{a_{n-3}}{n-2} + \frac{4}{n-1} - \frac{2}{n-2} \\
\ldots &= \ldots + \ldots - \ldots \\
\frac{a_2}{3} &= \frac{a_1}{2} + \frac{4}{3} - \frac{2}{2}
\end{aligned}
$$

Sum all up and cancel out the common terms, we have

$$\frac{a_n}{n+1} = \frac{a_1}{2} + 4\left(\frac{1}{n+1} + \ldots + \frac{1}{3}\right) - 2\left(\frac{1}{n} + \ldots + \frac{1}{2}\right)$$

Harmonic number

The n-th harmonic number is $H_n = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}$.

Recall fact: H_n is $\Theta(\log n)$.

$$\begin{aligned}
\frac{a_n}{n+1} &= \frac{a_1}{2} + 4\left(\frac{1}{n+1} + \ldots + \frac{1}{3}\right) - 2\left(\frac{1}{n} + \ldots + \frac{1}{2}\right) \\
&= 4\left(H_{n+1} - \left(\frac{1}{1} + \frac{1}{2}\right)\right) - 2\left(H_n - \frac{1}{1}\right) \\
&= 4H_{n+1} - 4\left(\frac{1}{1} + \frac{1}{2}\right) - 2H_n + 2 \\
&= 4H_{n+1} - 2H_n - 4\left(\frac{1}{1} + \frac{1}{2}\right) + 2 \\
&= \left(\frac{4}{n+1} + 4H_n\right) - 2H_n - 4 \\
&= \frac{4}{n+1} + 2H_n - 4
\end{aligned}$$

$a_n = (n+1)\left(\frac{4}{n+1} + 2H_n - 4\right)$. Therefore, a_n is $\Theta(n \log n)$.

Further analysis of Quicksort

Question. How can we improve Quicksort?

- Better way of choosing the pivot:
  - Take the median of a sample of (say 3) elements
  - Choose a pivot uniformly at random
- Use insertion sort once the sublists are small

Summary

- Quicksort
- Linear time partitioning method, Hoare's method

	Quicksort
Worst case time	$\Theta(n^2)$
Best case time	$\Theta(n \log n)$
Average case time	$\Theta(n \log n)$

THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND