Worst Case Analysis

n power of 2

$C(n) = 2(n/2) + C\_(merge(n))$

$C(1) = 0$.

Worst case

$C\_merge(n) = n-1$

$C\_worst(n) = 2\,C\_worst(n/2) + n-1$.

$C\_worst(1) = 0$

↓

1 element.

Use Master's theorem

$C\_worst(n) = \theta(n \log n)$

Quick Sort

2, 1, 3, 4, 5

P = taken as 2.

every element is compared with Pivot(P)

2 1 3 4 5

↑ i++ ↑ j--

value ↑ pivot --j

eg: 12 14 13 9 5 4.

↑i ↑j Pivot = 12

(i) 12 <= 12 i++.

(ii) 14 > 12. then compare [j 4 < 12]

(iii) Compare i and j i < j [a[i] and a[j] swap]

12 4 13 9 5 14

↑i →i ↑j ↑j

j↑ j--

i↓ i++.

1) 4 < 12 i++.

2) 13 > 12 ⟹ 14 > 12 j--

3) 5 < 12

4) i and j compare.

swap.

12 4 5 9 13 14.

↑i ↑i ↑i ↑ j

↑j ═══

1) 5 < 12 i++

2) 9 < 12 i++

3) 13 > 12 for i

13 > 12 j--

4) compare j and Pivot.

9 < 12. for j.

i and j (compare & swap)

[9 4 5 12 13 14]

1 9 12 4 8 5 P = 1;

↑i ↑ ↑j ↑j ↑j ↑j

9 > 1

5 > 1

8 > 1

4 > 1

12 > 1

9 > 1

1 [9 12 4 8 5]

P = 9.

1 < 9.

12 > 9

5 < 9.

8 < 9

12 < 9

1 9 12 4 8 5

↑i ↑j

[ ]

P = 12.

if (l < r)

S ← Partition (A[l - r])

Quicksort (A[l .... s-1])

Quicksort (A[s+1 .... r])

P ← A[i]

i ← l

j ← r+1

Repeat

repeat
    $i \leftarrow i+1$ until $i >= j$ (or) $A[i] >= p$.
repeat $j \leftarrow j-1$ until $A[j] <= p$.
if $i < j$ then swap $(A[i], A[j])$.
until $i >= j$
swap $(A[i], A[j])$
return j.

Analysis

$C(n) = 2\,C(n/2) + f(n)$.

$f(n) = n$. $f(n) \in \Theta n^d$.

$C(n) = 2C(n/2) + n$.

$a = 2, b = 2, d = 1$.

$[a = b^d]$ $a = 2$. $b = 2$. so $(\Theta n \log_2 n)$.

Worst case.

1 2 3 4 5 $\Rightarrow$ 6 comparison.

$C\_worst(n) = \frac{(n+1)(n+2)}{2} - 3$.

$= \Theta(n^2)$ //

$\Downarrow$

$(n+1) + n \ldots + 3$.

5 comp $\Rightarrow$ 1 [2 3 4 5]

$C\_worst(1) = 0$.

$C\_worst(2) = C\_worst(0) + 3$

Binary Search

```
l ← 1
r ← n
while l <= r.
m ← L (l+r)/2 ]
if k == A[m] return m.
else if K < A[m] r ← m-1
else l ← m+1
return -1.
```