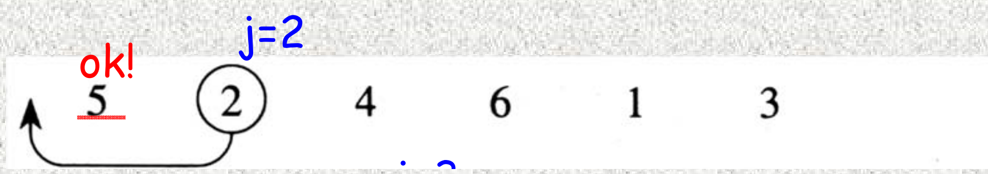
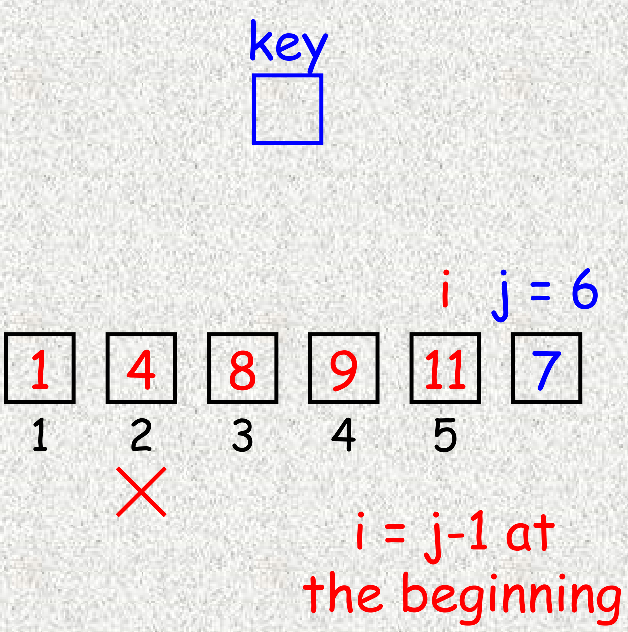
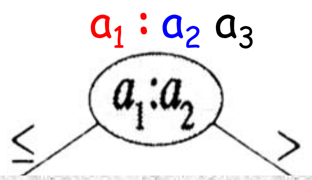
Insertion Sort



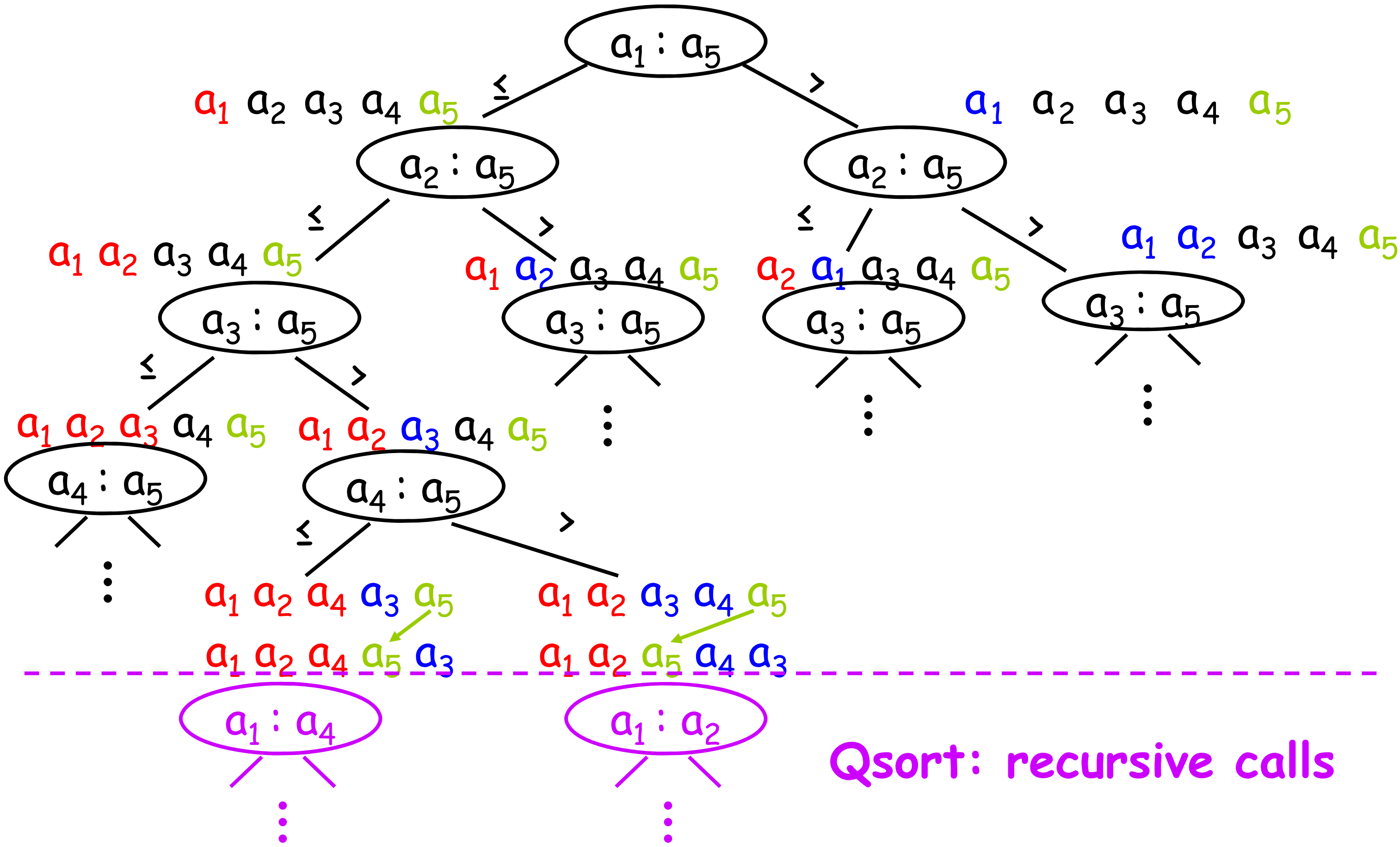




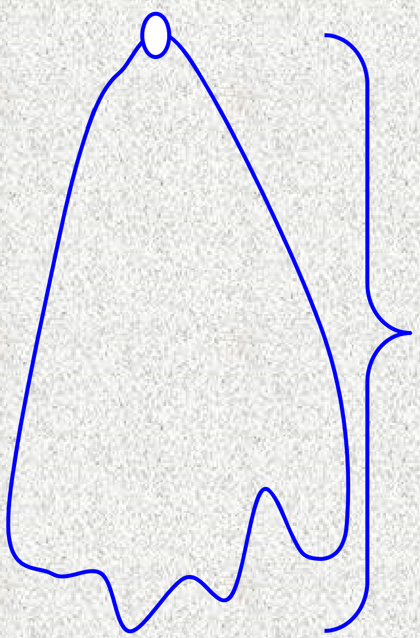
8-1z

Partition of Qsort

8-1a



a comparison sort A with worst-case time T(n)



height
$h = T(n)$

a decision tree DT
(given n)

* T has at most 2^h leaves
* T must have at least $n!$ leaves

$\Rightarrow$ $2^h \geq n!$

$\Rightarrow$ $h \geq \lg_2 (n!)$

$O(n^2)$

$O(n^{1.5})$

$O(n\lg n)$

$\theta(n\lg n)$

$\Omega(n\lg n)$

$\Omega(n\lg\lg n)$

$\Omega(n\lg^* n)$

$\Omega(n)$

O - upper bound:
algorithm 的品質

$O = \Omega$ $\Rightarrow$ optimal

Ω - lower bound:
problem 的困難度

* Ω 很重要，但也很困難，所以好的結果不多

Example: sorting

$O(n^2)$ (insertion sort, ...)

⋮

$O(n \lg n)$ (merge sort, ...)

optimal ⇦ $O = \Omega$

$\Omega(n \lg n)$ (decision tree)

$\Omega(n)$ (output size: trivial)

Example: matrix multiplication

$O(n^3)$ (naive)

$O(n^{2.81})$ (Strassen)

⋮

$O(n^{2.374})$ (2011)

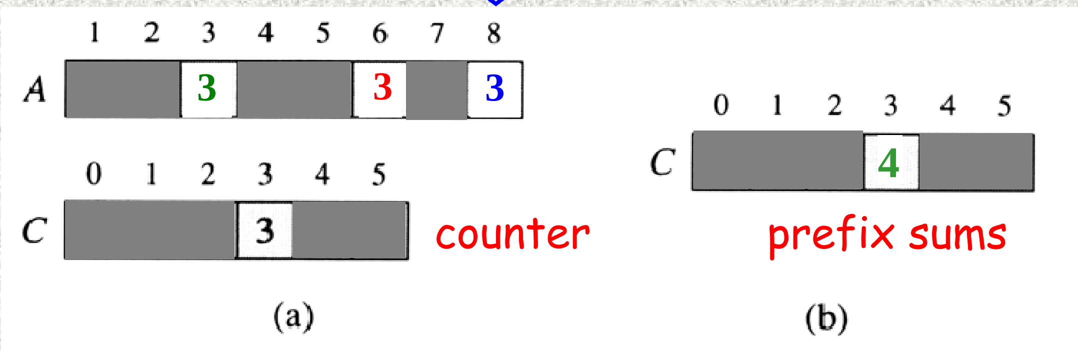
$O(n^{2.3728})$ (2014)

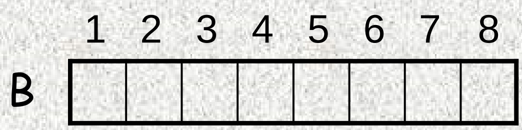
? (an important open problem)

$\Omega(n^2)$ (output size: trivial)

scan A from left-to-roght

* 7 is the position for the last "3"
* positions for "3" are 5, 6, 7





n integers, $k = 10^{12}$

2	3	1	6	7	5	1	9	9	2	6	7

Counting sort: $k = 10^{12}$
$O(n + k) = O(n+10^{12})$

Radix sort: $d = 2$, $k = 10^6$

Radix sort: $d = 4$, $k = 10^3$

Radix sort: $d = 12$, $k = 10$

n strings of length 9

s	t	r	a	n	g	e	r	s

Radix sort: $d = 9$, $k = 26$
(a ~ z)

n integers, $k = 10^{12}$

2	3	1	6	7	5	1	9	9	2	6	7

Counting sort: $k = 10^{12}$

$O(n + k) = O(n+10^{12})$

Radix sort: $d = 2$, $k = 10^6$

$O(d(n+k)) = O(2(n + 10^6))$

Radix sort: $d = 4$, $k = 10^3$

$O(4(n + 10^3))$

Radix sort: $d = 12$, $k = 10$

$O(12(n + 10))$

What is the best d?

$\Rightarrow$ take d s.t. $k = n$!!!

n integers, $k = n^2$

2 lg n bits

1	1	1	0	1	1	0	0	1	0	1	1

lg n bits

$k = 2^{\lg n} = n$

Counting sort: $k = n^2$

$O(n + n^2) = O(n^2)$

Radix sort: $d = 2$, $k = n$

$O(2(n + n)) = O(n)$

n integers, $k = n^d$

d lg n bits

1	1	1	0	1	1	0	0	1	0	1	1	1	1	1	0	1	1	0	0	1	0	1	1

lg n bits

$k = 2^{\lg n} = n$

Integer sort: $k = n^d$

$O(n + k) = O(n + n^d)$

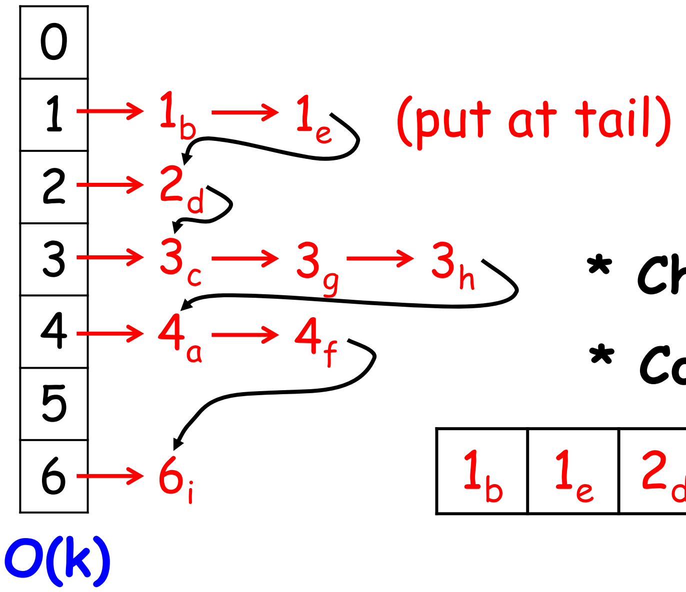
Radix sort: d, $k = n$

$O(d(n + k)) = O(dn)$

If d is a constant, it needs linear time !

$\boxed{4_a}$ 1_b 3_c 2_d 1_e 4_f 3_g 3_h 6_i

* put all items into buckets $O(n)$

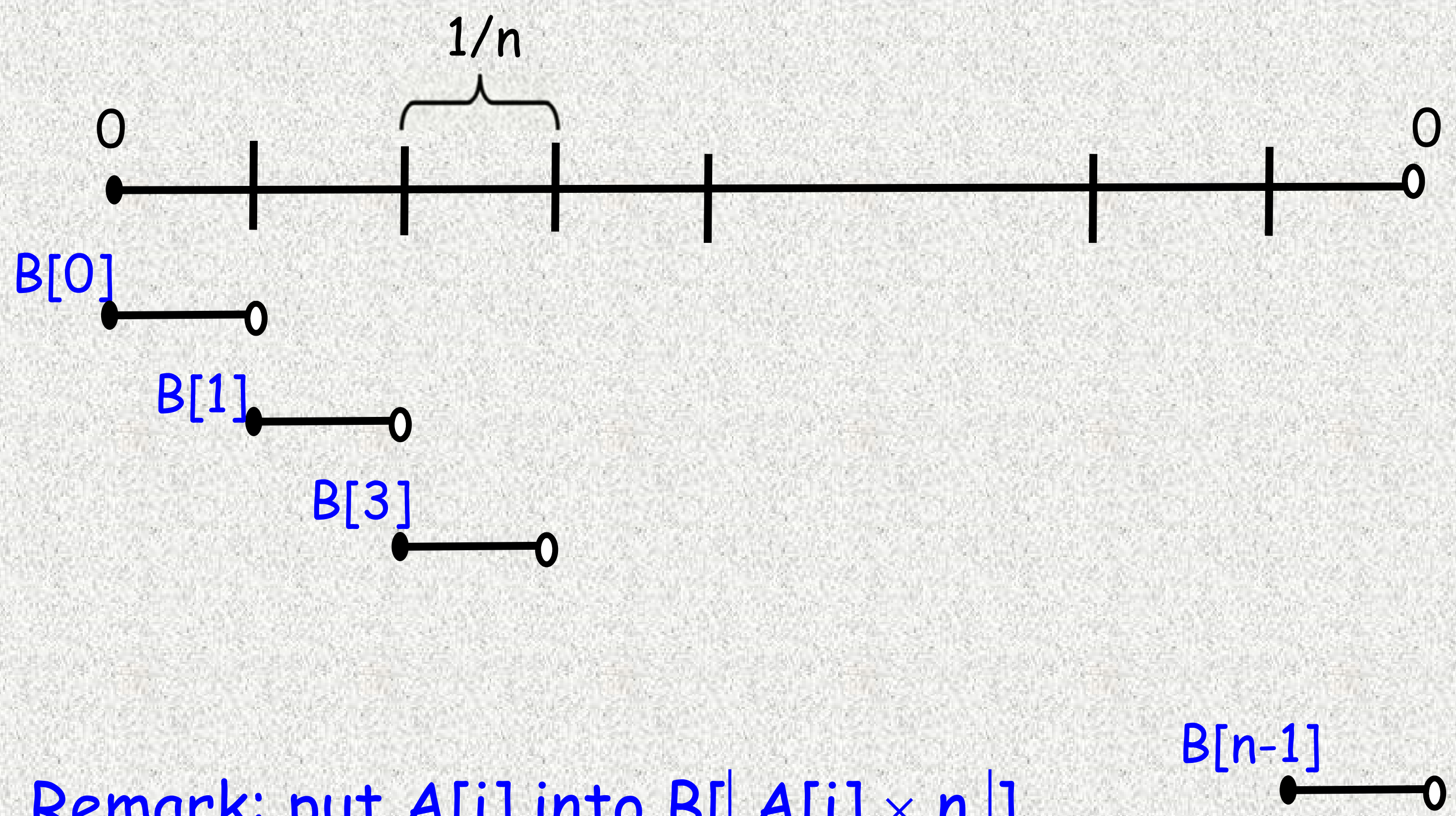


* Chain all lists into one $O(k)$

* Convert into array $O(n)$

1_b	1_e	2_d	3_c	3_g	3_h	4_a	4_f	6_i

Time: $O(n + k)$



Remark: put $A[i]$ into $B[\lfloor A[i] \times n \rfloor]$