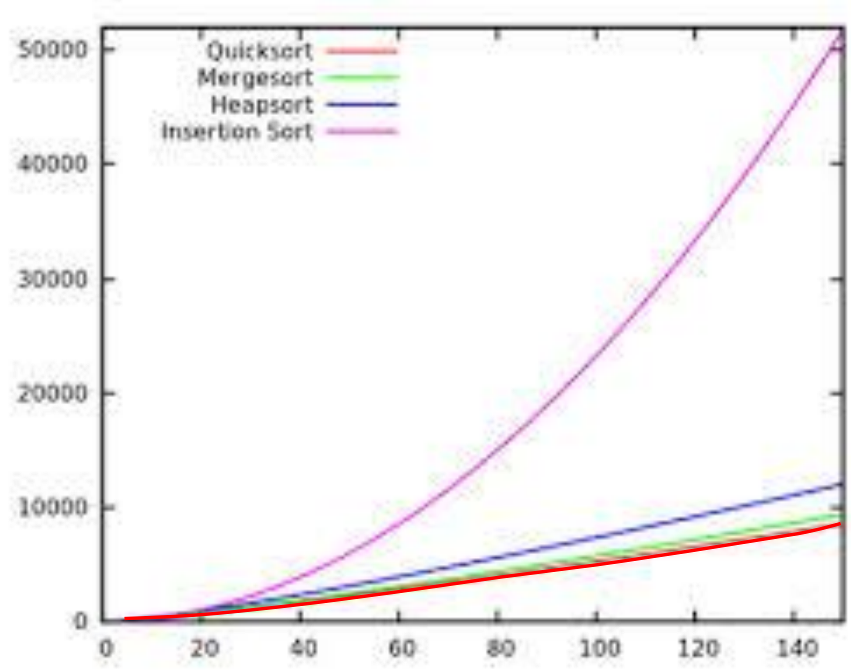
7.Quicksort

中國文化大學
資訊工程學系
副教授 張耀鴻
112學年度第2學期

Outline

7.1 Quick Sort 快速排序
7.2 Analysis of Quick Sort 快速排序分析
7.3 Randomized Quick Sort 隨機快速排序
7.4 Analysis of Randomized Quick Sort 隨機快速排序分析



7.1 Quicksort

- Proposed by C.A.R. Hoare in 1962.
- *Divide-and-conquer* algorithm. 屬於各個擊破法
- Sorts "in place" (like insertion sort, but not like merge sort). 就地排序
- Very practical (with tuning). 實用性高

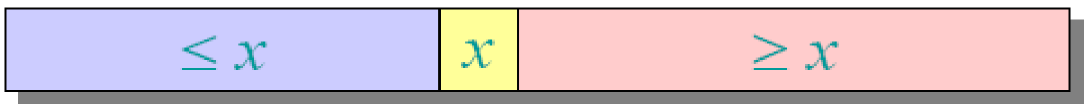
➢ Important results: 重要結果
  - ✓ Worst case $\Theta(n^2)$
  - ✓ Average case $\Theta(n \lg n)$
  - ✓ Hidden constants are small 隱藏的常數很小

Divide and conquer

Quicksort an *n*-element array: QUICKSORT(N) 演算法

1. *Divide*: Partition the array into two subarrays around a ***pivot x*** such that

Elements in lower subarray ≤ *x* ≤ elements in upper subarray.



2. *Conquer*: Recursively sort the two subarrays.

3. *Combine*: Trivial.

Key: ***Linear-time partitioning subroutine.***

1. **分割**: 選一個「基準值」*x*, 將陣列分割成左右兩邊, 使得 $left \le x \le right$
2. **各個擊破**: 左右兩邊再分別遞迴呼叫 QUICKSORT
3. **合併**: **不需要(已自動合併)**

關鍵步驟: 第1步的 ***分割子程序必須屬於線性O(N)***

Partitioning subroutine

分割子程序

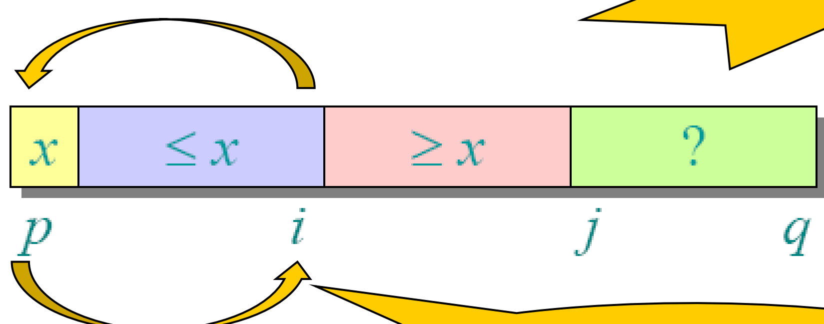
PARTITION(A, p, q) ▷ $A[p \,..\, q]$
  $x \leftarrow A[p]$ ▷ pivot = $A[p]$
  $i \leftarrow p$
  for $j \leftarrow p + 1$ **to** q
    do if $A[j] \le x$
      then $i \leftarrow i + 1$
        exchange $A[i] \leftrightarrow A[j]$
  exchange $A[p] \leftrightarrow A[i]$
  return i

1. 選第一個當 Pivot $O(1)$

2. 接著從第二個比到最後一個
如果比 Pivot 小就往前搬 $O(n)$

Running time = $O(n)$
for n elements

invariant:

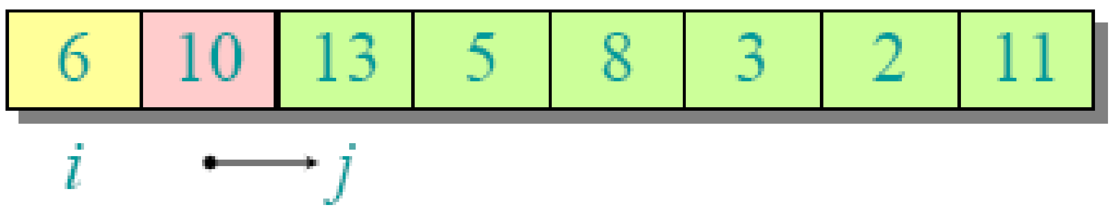


3. 最後第i個再跟第1個交換 $O(1)$

Example of partitioning

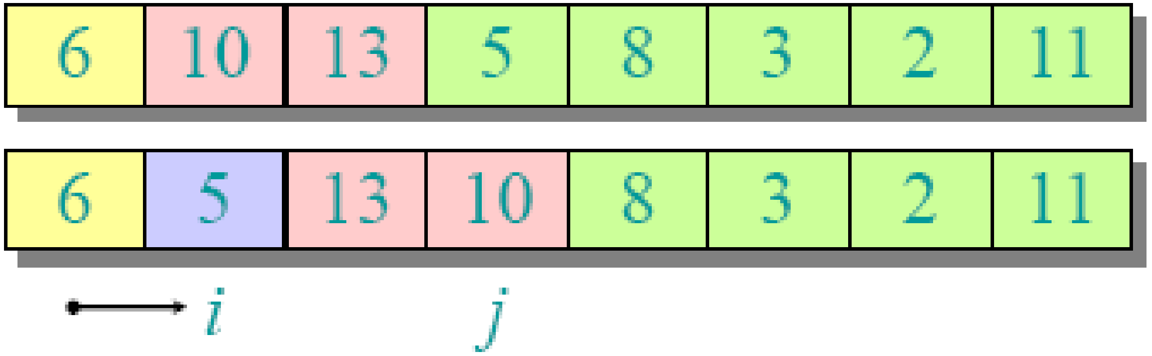
6	10	13	5	8	3	2	11
i	j						

Example of partitioning

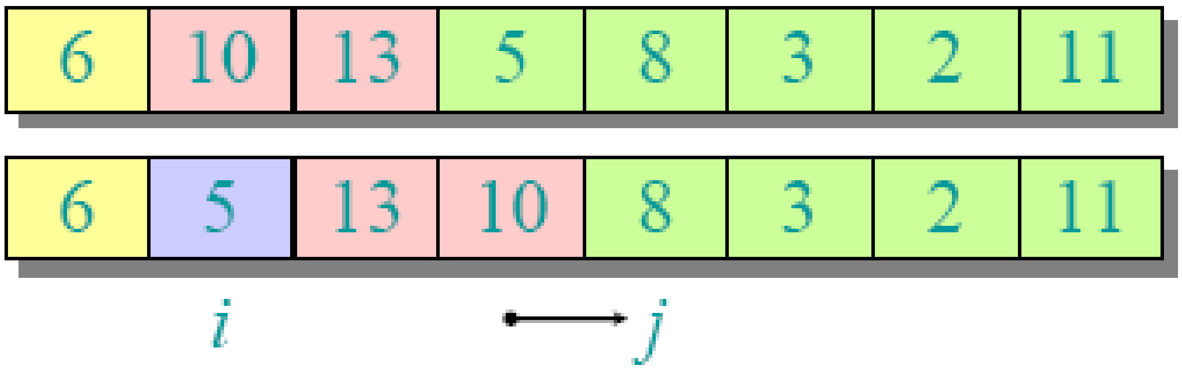


Example of partitioning

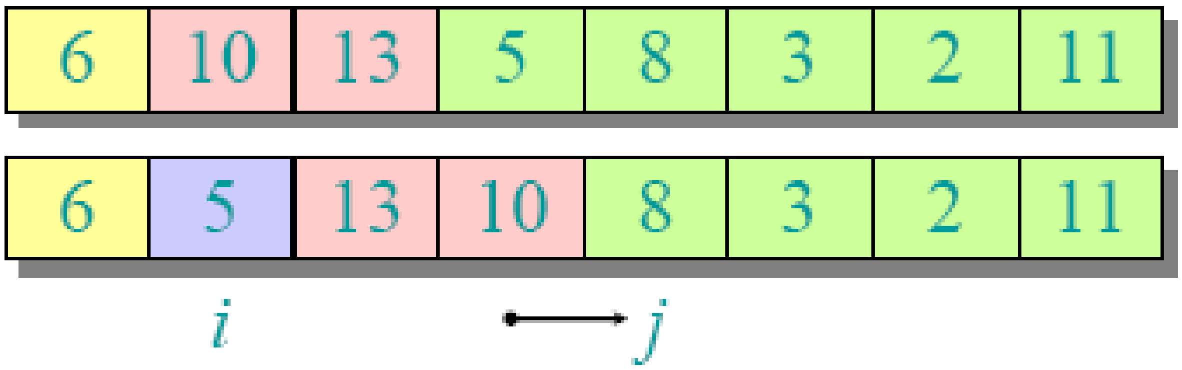
Example of partitioning



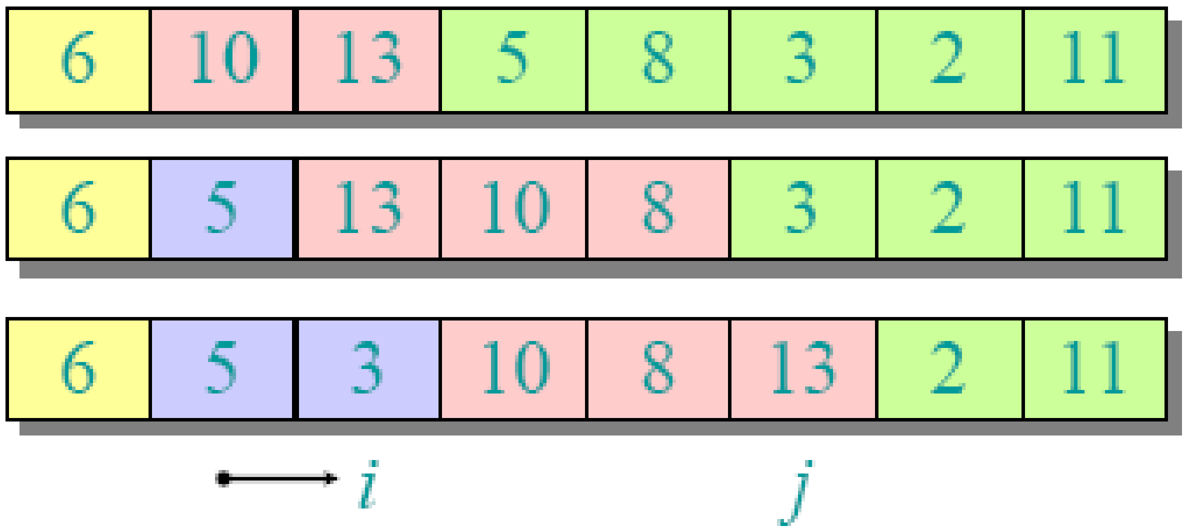
Example of partitioning



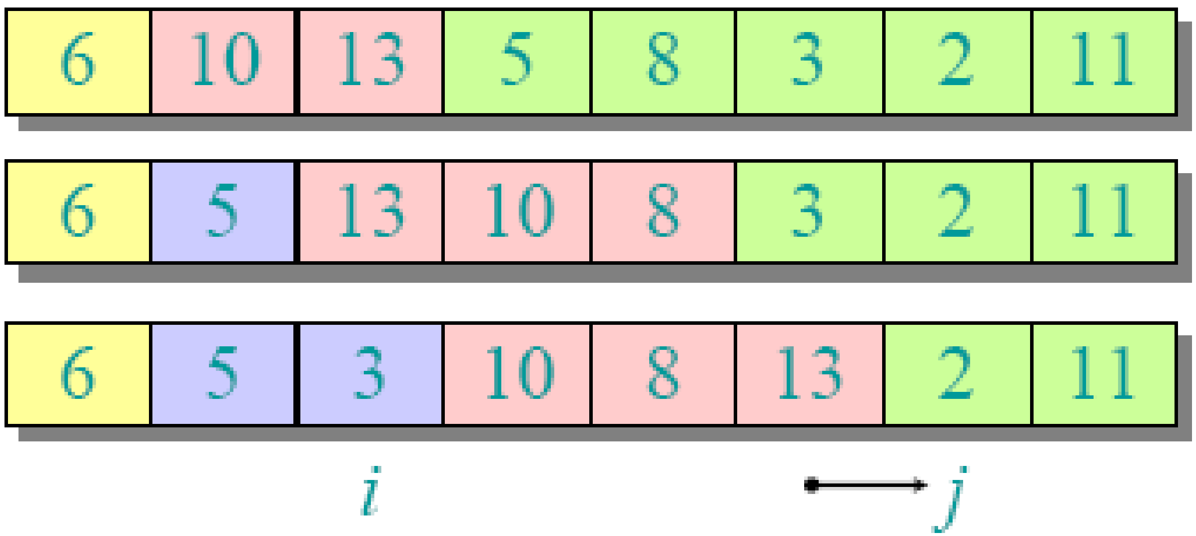
Example of partitioning



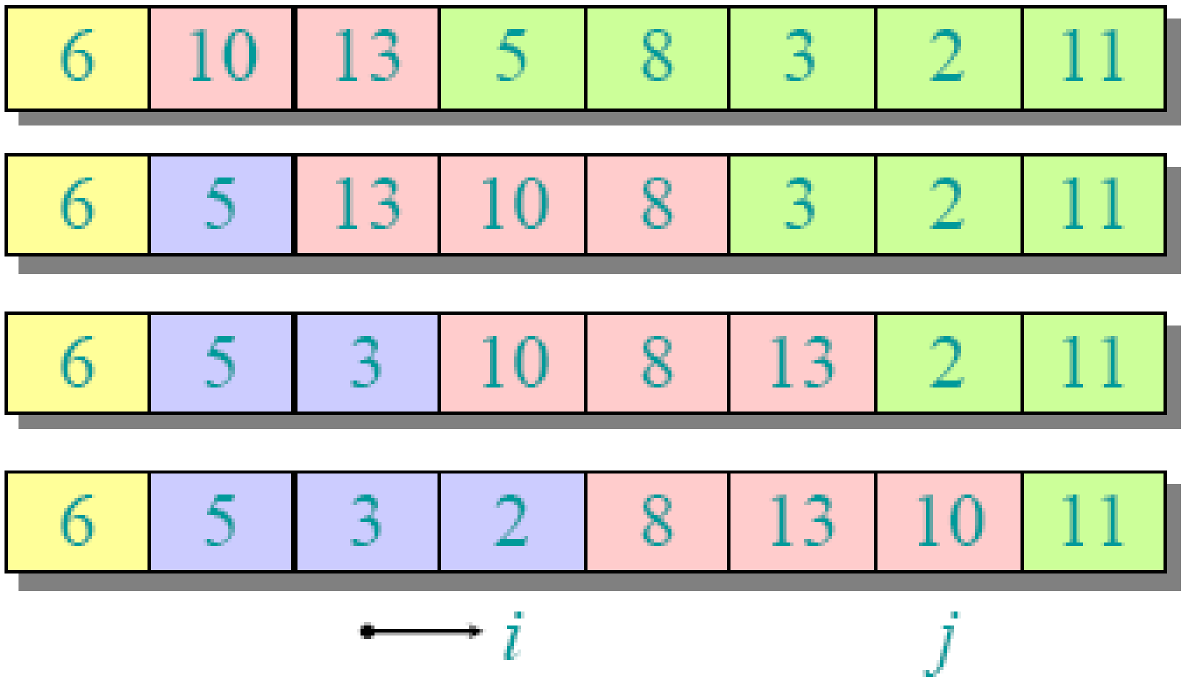
Example of partitioning



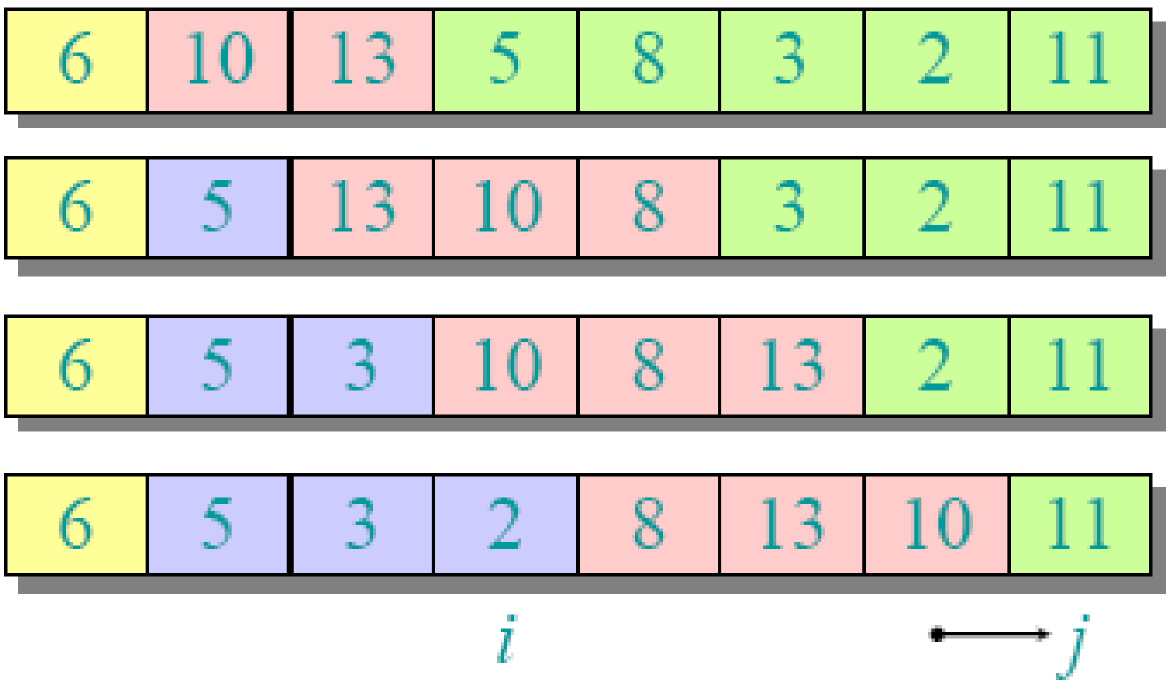
Example of partitioning



Example of partitioning



Example of partitioning



Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11

i $\qquad\qquad\qquad \longrightarrow j$

Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11
2	5	3	6	8	13	10	11

$$left \le x \le right$$

PARTITION C++ implementation

```cpp
unsigned partition(vector<int> & A, unsigned p, unsigned q)
{
    int x = A[p]; // pivot
    unsigned i = p; // index to be swapped
    for (unsigned j=p+1; j<=q; ++j) {
        if (A[j] < x) {
            ++i;
            swap(A[i], A[j]);
        }
    }
    swap(A[p], A[i]);
    return i;
} // end of partition
```

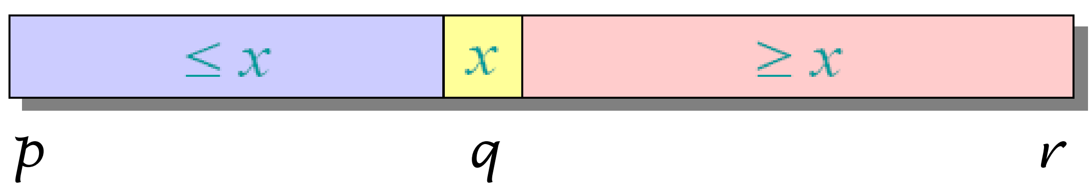
PARTITION(A, p, r)

$x = A[r]$
$i = p - 1$
for $j = p$ **to** $r - 1$
  if $A[j] \leq x$
    $i = i + 1$
    exchange $A[i]$ with $A[j]$
exchange $A[i + 1]$ with $A[r]$
return $i + 1$

Pseudocode for quicksort

```
QUICKSORT(A, p, r)
    if p < r
        then q ← PARTITION(A, p, r)
            QUICKSORT(A, p, q–1)
            QUICKSORT(A, q+1, r)
```

Initial call: QUICKSORT$(A, 1, n)$



QUICK_SORT C++ implementation

CLRS implementation:

```
void quick_sort(vector<int> & A, unsigned p, unsigned r)
{
    if (p < r) {
        unsigned q = partition(A, p, r);
        quick_sort(A, p, (q>0)?(q-1):0);
        quick_sort(A, q+1, r);
    }
} // end of quick_sort
```

QUICKSORT(A, p, r)

if $p < r$

    $q = $ PARTITION(A, p, r)

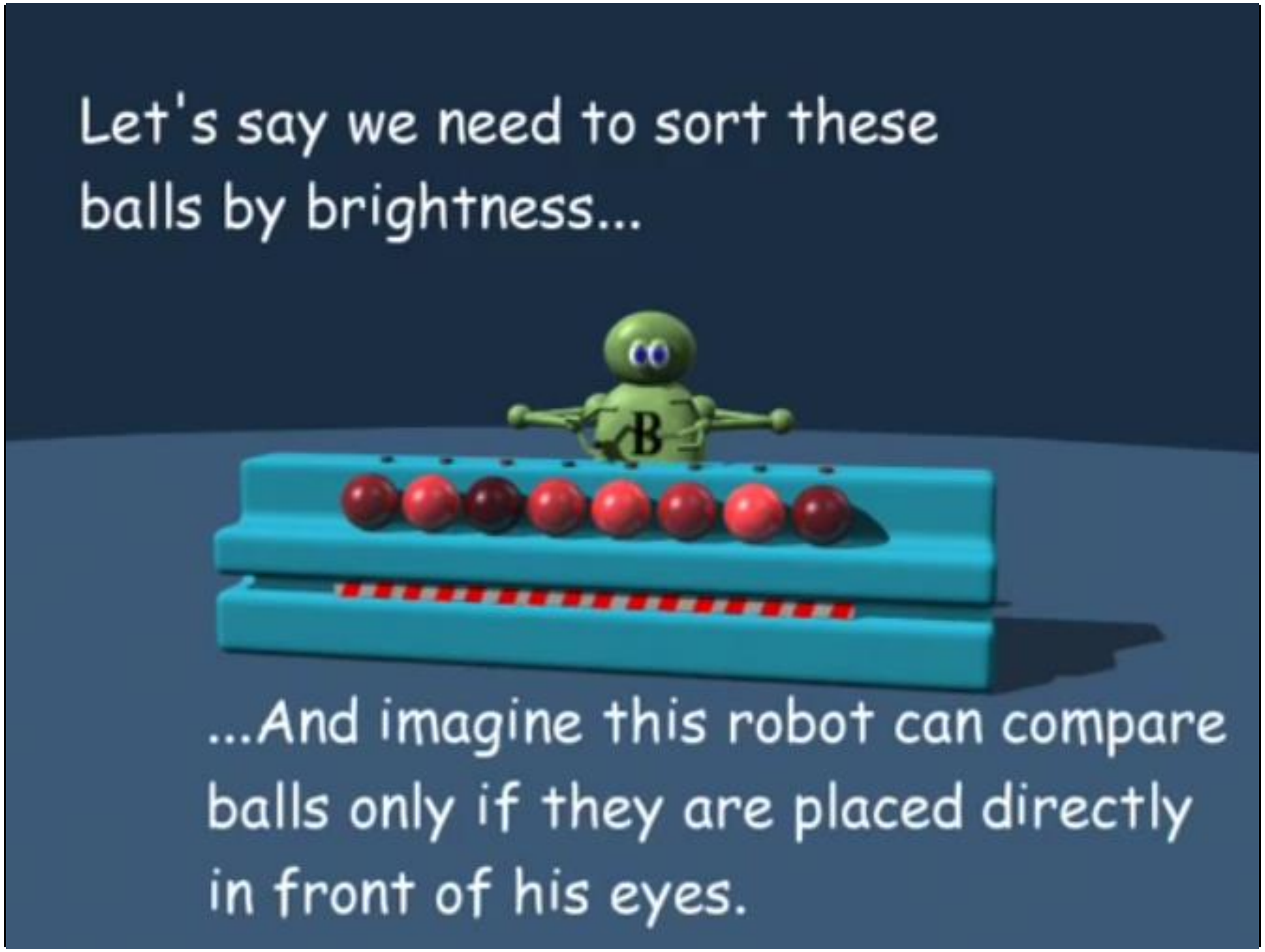
    QUICKSORT$(A, p, q-1)$

    QUICKSORT$(A, q+1, r)$

C++ STL implementation:

```
qsort(A.data(), A.size(), sizeof(int), [](const void * a, const void * b) { return ( *(int*)a - *(int*)b ); } );
```

QuickSort vs. BubbleSort



Exercise 7.1-1

- 請問以下陣列經過 PARTITION 運算後的結果為何?

  $A = \{13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11\}$

[ANS]:

$A = \{ 11, 9, 5, 12, 8, 7, 4, 2, 6, \underline{13}, 21, 19\}$

7.2 Analysis of quicksort

- Assume all input elements are distinct.
  假設每個元素皆不同
- In practice, there are better partitioning algorithms for when duplicate input elements may exist.
  實務上若有相同元素可用更好的分割法
- Let $T(n)$ = worst-case running time on an array of n elements.

令$T(n)$ = worst-case 執行時間

Worst-case of quicksort 最壞情況

- Input sorted or reverse sorted.
  輸入元素已排序或已反向排序
- Partition around min or max element.
  以最大或最小元素為基準值做分割
- One side of partition always has no elements.
  一邊永遠沒有元素

$$
\begin{aligned}
T(n) &= T(0) + T(n-1) + \Theta(n) \\
&= \Theta(1) + T(n-1) + \Theta(n) \\
&= T(n-1) + \Theta(n) \\
&= \Theta(n^2) \quad \textbf{\textit{(arithmetic series)}}
\end{aligned}
$$

Worst-case recursion tree最壞情況

$$T(n) = T(0) + T(n-1) + cn$$

Worst-case recursion tree 最壞情況

$$T(n) = T(0) + T(n-1) + cn$$

$$T(n)$$

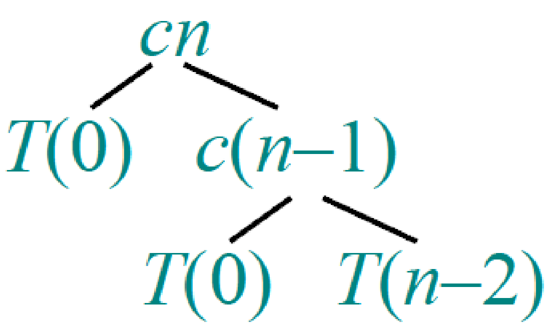
Worst-case recursion tree最壞情況

$$T(n) = T(0) + T(n-1) + cn$$

$$\begin{array}{ccc} & cn & \\ \swarrow & & \searrow \\ T(0) & & T(n-1) \end{array}$$

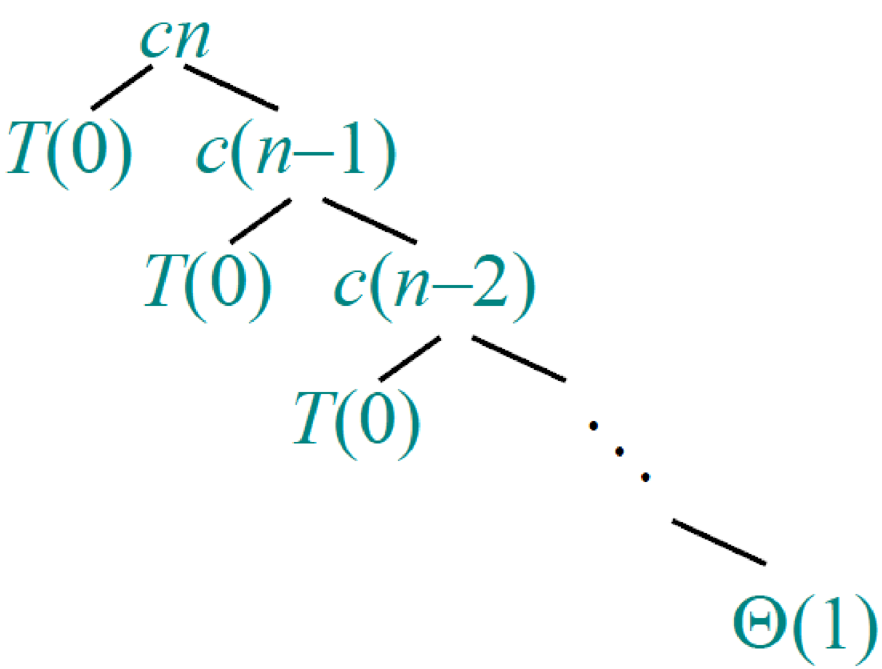
Worst-case recursion tree最壞情況

$$T(n) = T(0) + T(n-1) + cn$$



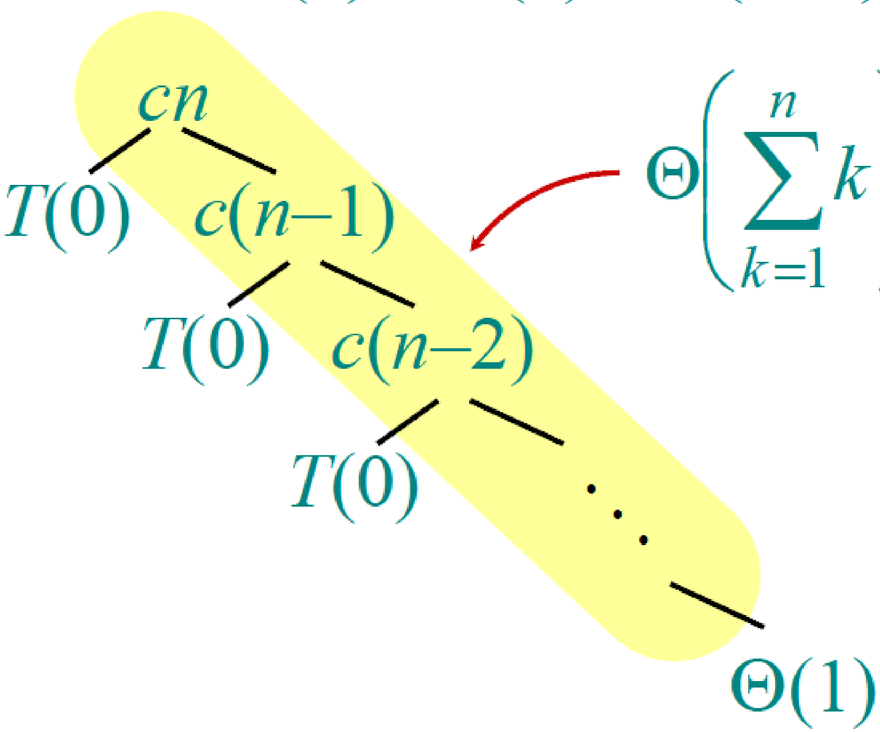
Worst-case recursion tree最壞情況

$$T(n) = T(0) + T(n-1) + cn$$



Worst-case recursion tree最壞情況

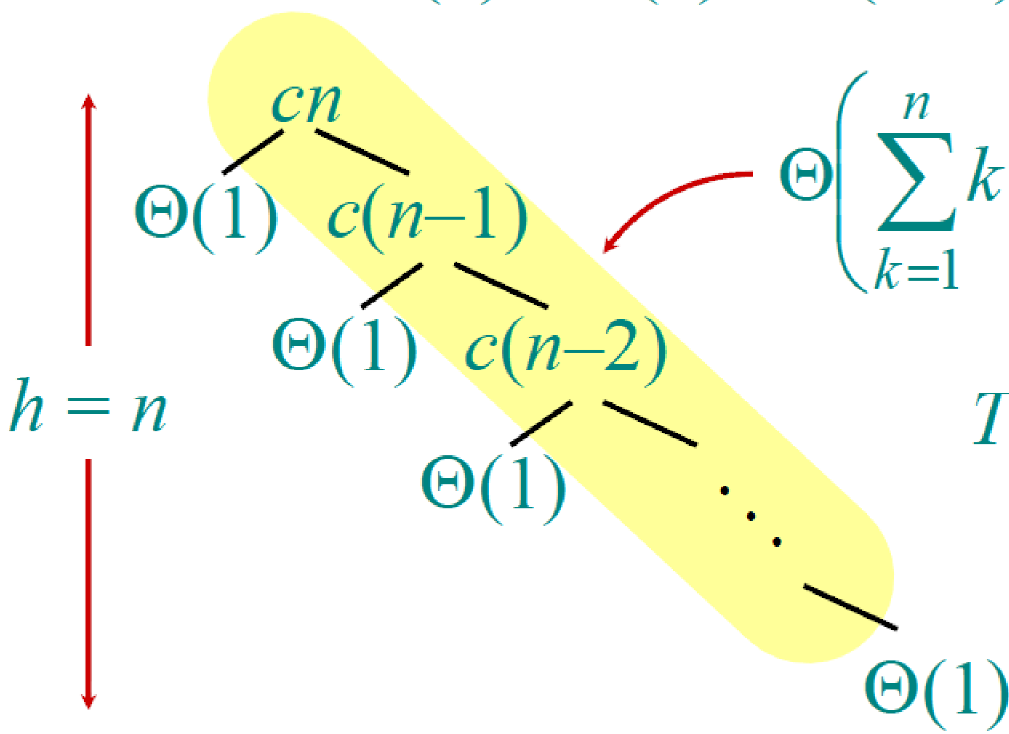
$$T(n) = T(0) + T(n-1) + cn$$



$$\Theta\left(\sum_{k=1}^{n} k\right) = \Theta\left(n^2\right)$$

Worst-case recursion tree最壞情況

$$T(n) = T(0) + T(n-1) + cn$$



$$\Theta\left(\sum_{k=1}^{n} k\right) = \Theta\left(n^2\right)$$

$$T(n) = \Theta(n) + \Theta(n^2)$$
$$= \Theta(n^2)$$

Best-case analysis (For intuition only!)

最佳情況

If we're lucky, PARTITION splits the array evenly:

假設PARTITION很幸運地將array恰好切一半

$T(n) = 2T(n/2) + \Theta(n)$

$= \Theta(n \lg n)$ (same as merge sort)

What if the split is always $1/10:9/10$?

若每次分割都分成左:右=1/10:9/10

$T(n) = T(1/10n) + T(9/10n) + \Theta(n)$

What is the solution to this recurrence?

求解以上遞迴式

Analysis of “almost-best” case

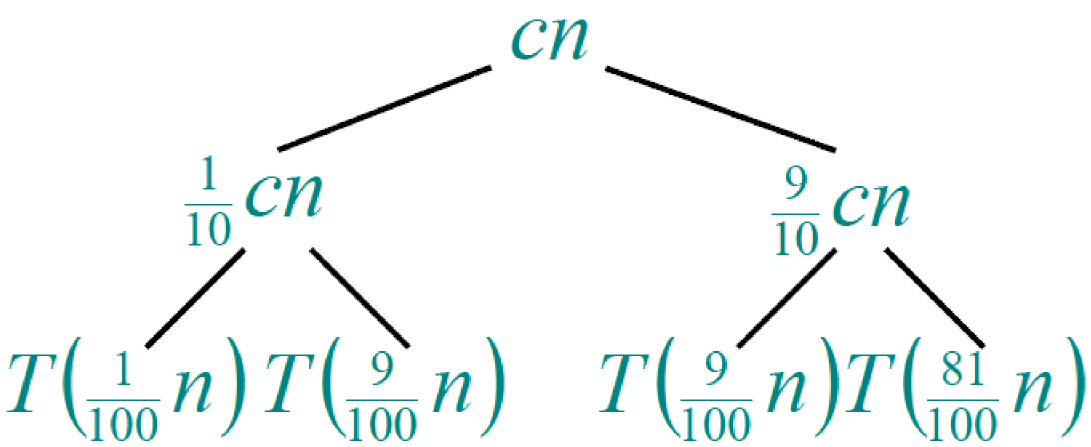
$T(n)$

Analysis of “almost-best” case

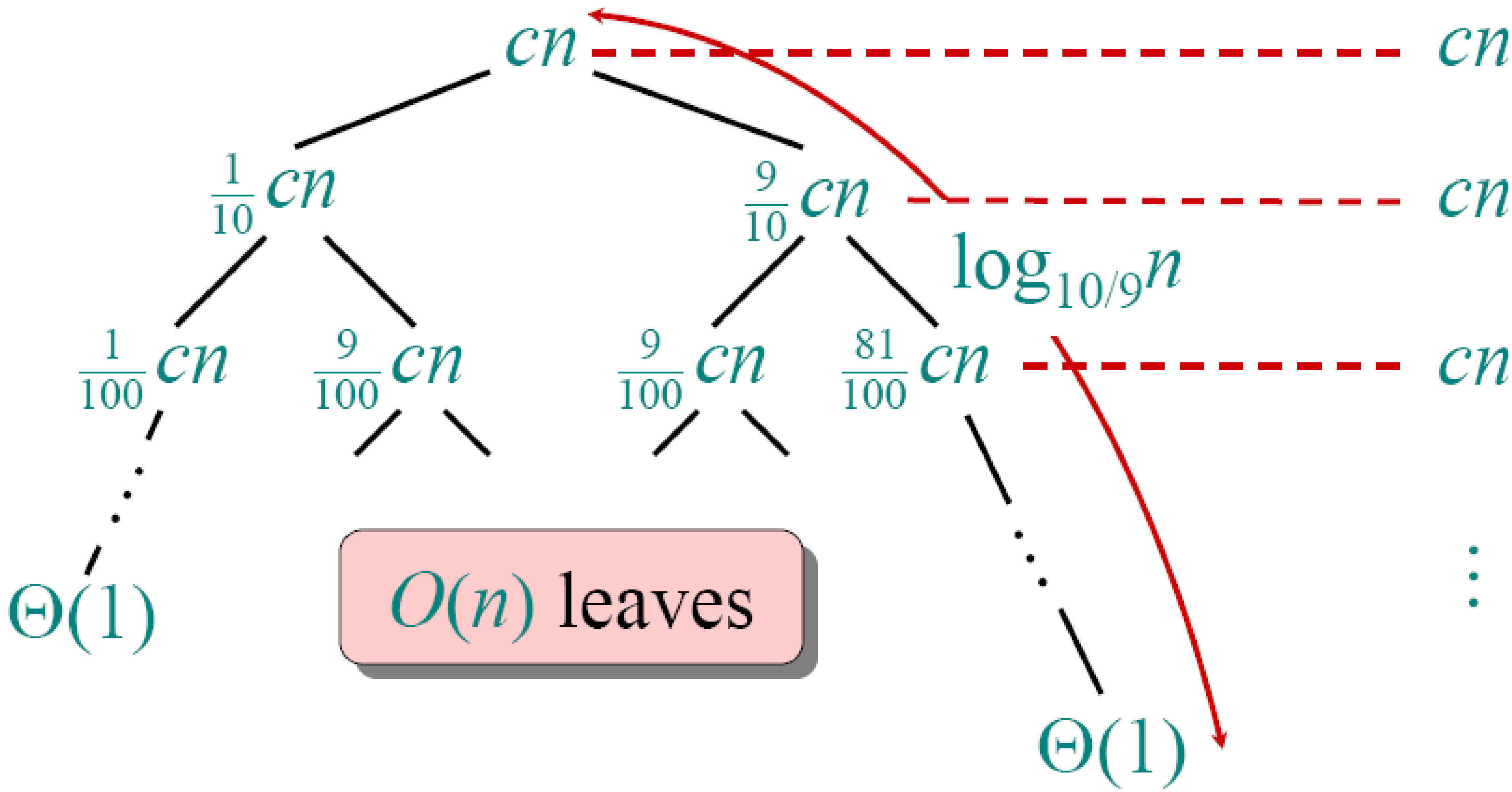
$$cn$$

$$T\left(\frac{1}{10}n\right) \qquad T\left(\frac{9}{10}n\right)$$

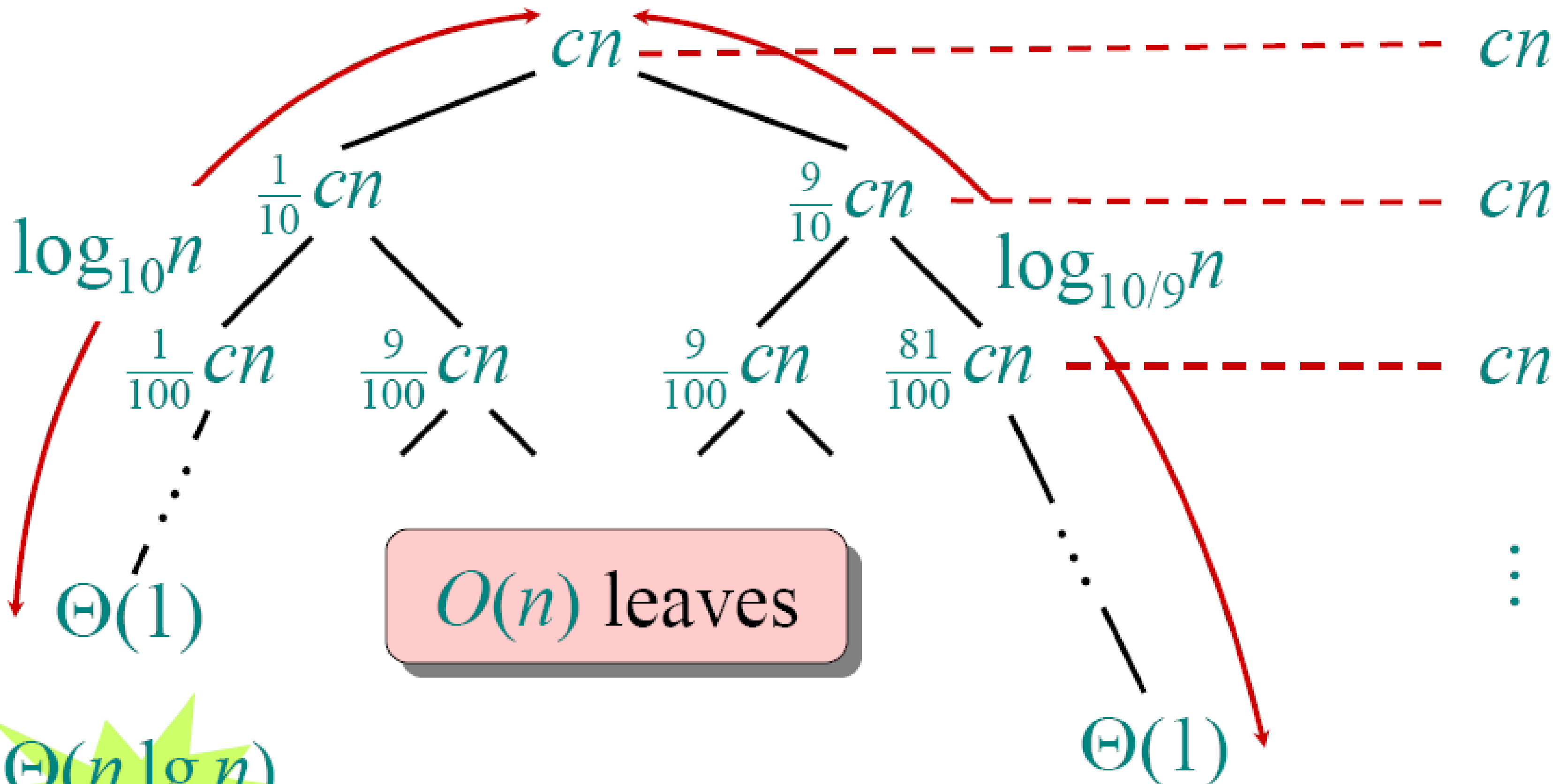
Analysis of "almost-best" case



Analysis of "almost-best" case



Analysis of “almost-best” case



$\Theta(n \lg n)$

Lucky!

$$cn \log_{10} n \le T(n) \le cn \log_{10/9} n + O(n)$$

More intuition

進一步觀察

Suppose we alternate lucky, unlucky, lucky, unlucky, lucky, ….

假設分割時運氣時好時壞

$L(n) = 2U(n/2) + \Theta(n)$ **lucky**

$U(n) = L(n-1) + \Theta(n)$ **unlucky**

Solving:

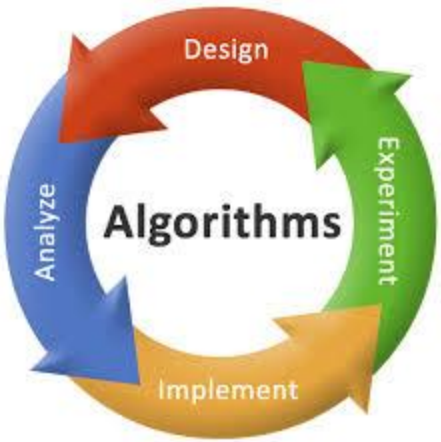
$$\begin{aligned} L(n) &= 2U(n/2) + \Theta(n) \\ &= 2(L(n/2 - 1) + \Theta(n/2)) + \Theta(n) \\ &= 2L(n/2 - 1) + \Theta(n) \\ &= \Theta(n \lg n) \end{aligned}$$

How can we make sure we are usually lucky?

如何保證分割時通常都很好運?

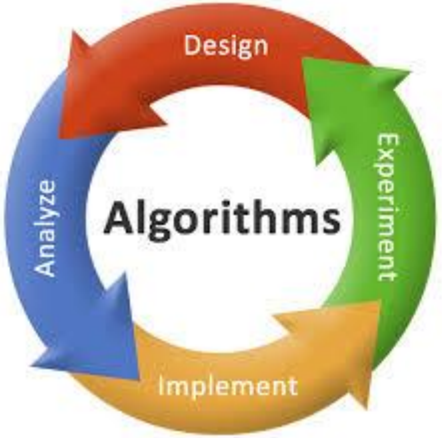
Exercise 7.2-2

- 當陣列*A*中元素的值都相同時, QUICKSORT 演算法執行時間複雜度以 Θ 表示是多少?



Exercise 7.2-3

- 證明當陣列*A*中元素已排序時, QUICKSORT 演算法執行時間為 $\Theta(n^2)$



7.3 Randomized quicksort

IDEA: Partition around a ***random*** element.
想法：隨機取一個基準元素做PARTITION

- Running time is *independent of the input* order. 執行時間與輸入無關
- No assumptions need to be made about the input distribution. 無需事先得知輸入分佈
- No specific input elicits the worst-case behavior. 賭最壞情況幾乎不會發生
- The worst case is determined only by the output of a random-number generator.
  最壞情況僅跟亂數產生步驟有關

RANDOMIZED_PARTITION(A,p,r)

$i \leftarrow RANDOM(p,r)$

exchange $A[p] \leftrightarrow A[i]$

return PARTITION(A,p,r)

隨機分割: RANSOMIZED_PARTITION

1. 呼叫RANDOM 隨機選一個 *Pivot*
2. 把*Pivot*和第1個元素交換
3. 呼叫 PARTITION 子程序

RANDOMIZED_QUICKSORT(A,p,r)

if $p < r$

then

$q \leftarrow RANDOMIZED\_PARTITION(A, p, r)$

RANDOMIZED_QUICKSORT(A,p,q)

RANDOMIZED_QUICKSORT(A,$q+1$,r)

隨機快速排序:

RANDOMIZED_QUICKSORT(N)

1. 呼叫RANSOMIZED_PARTITION
2. RANDOMIZED_QUICKSORT(左)
3. RANDOMIZED_QUICKSORT(右)

C++ RANDOMIZED_QUICKSORT implementation

```cpp
unsigned randomized_partition(vector<int> & A, unsigned p, unsigned r)
{
    unsigned i = random(p, r);
    swap(A[p], A[i]);
    return partition(A, p, r);
} // end of randomized_partition
```

RANDOMIZED-PARTITION(A, p, r)

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

```cpp
void randomized_quicksort(vector<int> & A, unsigned p, unsigned r)
{
    if (p < r) {
        unsigned q = randomized_partition(A, p, r);
        randomized_quicksort(A, p, (q>0)?(q-1):0);
        randomized_quicksort(A, q+1, r);
    }
} // end of randomized_quicksort
```

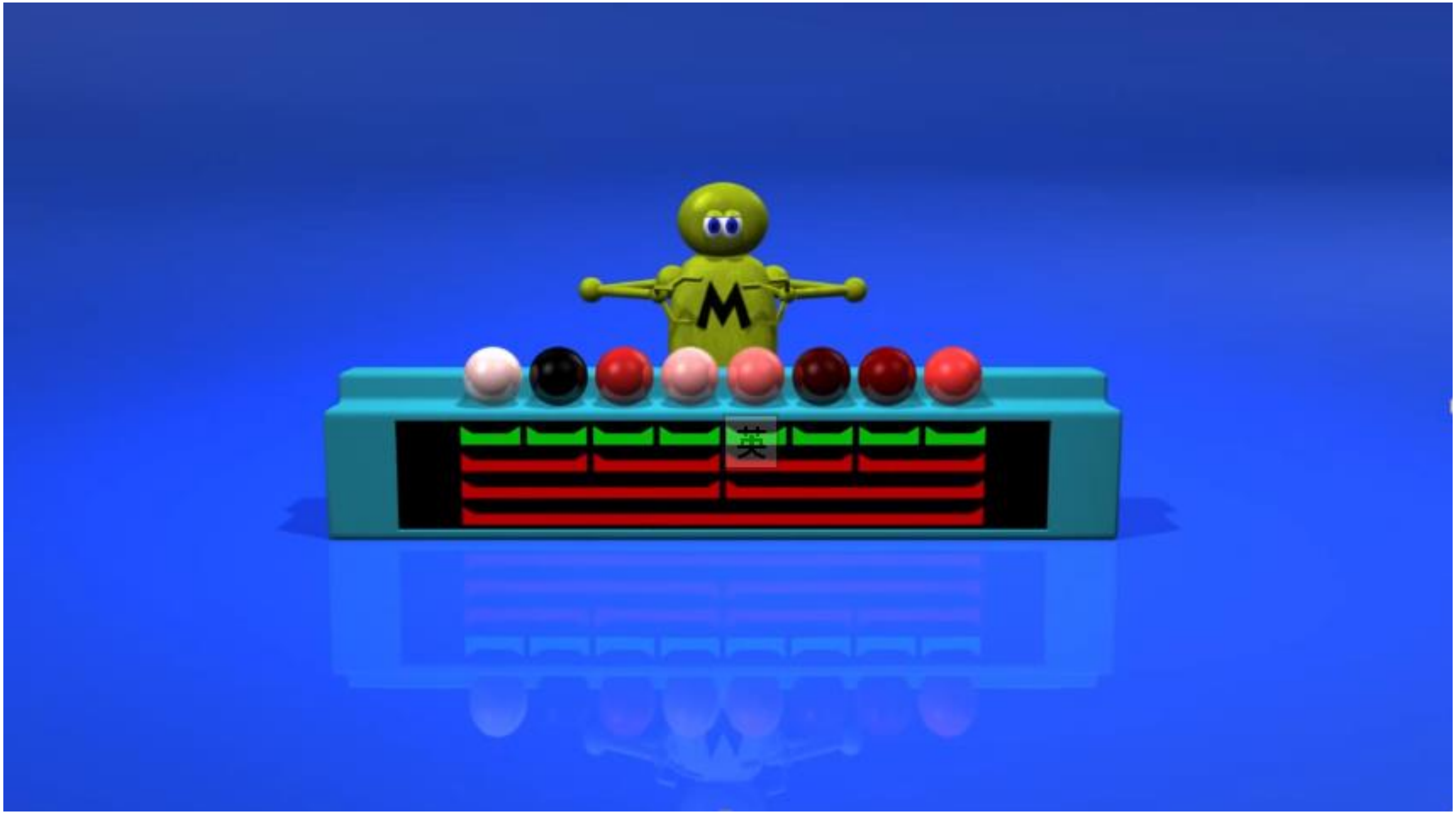
RANDOMIZED-QUICKSORT(A, p, r)

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

```cpp
unsigned random(unsigned p, unsigned r)
{
    std::default_random_engine generator = my_engine();
    std::uniform_int_distribution<unsigned> distribution(p, r); // define the range
    return distribution(generator); // generate a number
} // end of random
```

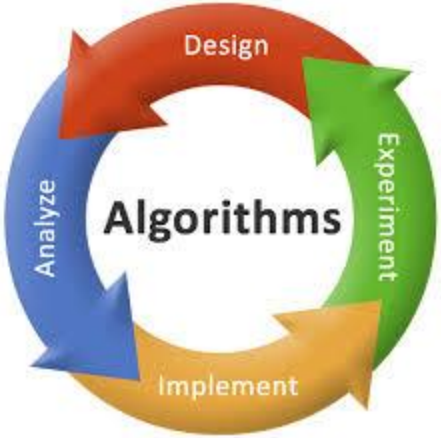
```cpp
std::default_random_engine & my_engine()
{
    static std::default_random_engine e{std::random_device{}()};
    return e;
} // end of my_engine
```

QuickSort vs. MergeSort



Exercise 7.3-1

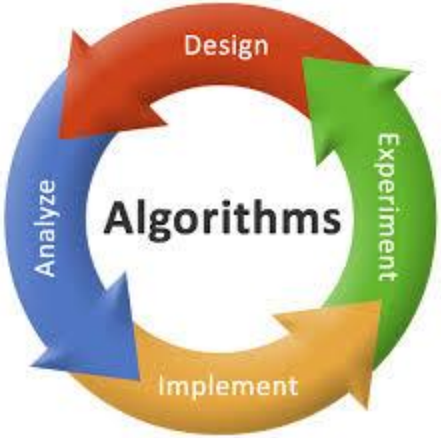
- 請將RANDOMIZED_QUICKSORT 演算法worst case執行時間複雜度以Θ表示。



Exercise 7.3-2

- 當RANDOMIZED_QUICKSORT執行時,

1. 在最佳狀況時 RANDOM函式會被呼叫幾次?
2. 在最差狀況時 RANDOM函式會被呼叫幾次?



7.4 Randomized quicksort analysis 分析隨機快速排序法

Let $T(n)$ = the **random variable** for the running time of randomized quicksort on an input of size n, assuming random numbers are independent.

令T(n)為隨機快速排序n 個元素的執行時間之隨機變數

For $k = 0, 1, \ldots, n-1$, 定義指標隨機變數 X_k

define the ***indicator random variable***

$X_k = \begin{cases} 1 & \text{若 PARTITION 成 } k : n-k-1 \text{ split,} \\ 0 & \text{otherwise.} \end{cases}$ (在第k個元素分割) (不在第k個元素分割)

$E[X_k] = \Pr\{X_k = 1\} = 1/n$, since all splits are equally likely, assuming elements are distinct.

因為分割在第幾個元素的機會都一樣,

所以X_k 的期望值= $1/n$

Analysis (continued)

$$T(n) = \begin{cases} T(0) + T(n-1) + \Theta(n) & \text{if } 0 : n-1 \text{ split,} \\ T(1) + T(n-2) + \Theta(n) & \text{if } 1 : n-2 \text{ split,} \\ \vdots & \\ T(n-1) + T(0) + \Theta(n) & \text{if } n-1 : 0 \text{ split,} \end{cases}$$

$$= \sum_{k=0}^{n-1} X_k \left(T(k) + T(n-k-1) + \Theta(n)\right).$$

Calculating expectation

$$T(n) = \sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \Theta(n))$$

等號兩邊取期望值

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k \big(T(k) + T(n-k-1) + \Theta(n)\big)\right]$$

期望值線性特性:
E[X+Y] = E[X] + E[Y]

$$= \sum_{k=0}^{n-1} E\big[X_k \big(T(k) + T(n-k-1) + \Theta(n)\big)\big]$$

期望值為線性;
E[XY] = E[X] E[Y]

$$= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(k) + T(n-k-1) + \Theta(n)]$$

每個 X_k 隨機選項為獨立
$E[X_k] = 1/n$.

$$= \frac{1}{n}\sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n}\sum_{k=0}^{n-1} E[T(n-k-1)] + \frac{1}{n}\sum_{k=0}^{n-1} \Theta(n)$$

前兩項加總式其實一樣.

$$= \frac{2}{n}\sum_{k=1}^{n-1} E[T(k)] + \Theta(n)$$

Hairy recurrence

$$E[T(n)] = \frac{2}{n}\sum_{k=2}^{n-1} E[T(k)] + \Theta(n)$$

(k = 0, 1 項可被 $\Theta(n)$ 吸收.)

Prove: $E[T(n)] \le an\lg n$ for constant $a > 0$.

- 選一個夠大的 a 值, 使得 $an\lg n$ 大於 $E[T(n)]$ for sufficiently small $n \ge 2$.

Use fact: $\displaystyle\sum_{k=2}^{n-1} k\lg k \le \tfrac{1}{2}n^2\lg n - \tfrac{1}{8}n^2$ (exercise).

Substitution method

$$E[T(n)] \leq \frac{2}{n} \sum_{k=2}^{n-1} ak \lg k + \Theta(n)$$

取代歸納假設. *(假設* $E[T(n)] \leq an \lg n$ *)*

Substitution method

$$E[T(n)] \leq \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$

$$\leq \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n)$$

Use fact.

Use fact: $\sum_{k=2}^{n-1} k\lg k \leq \frac{1}{2}n^2\lg n - \frac{1}{8}n^2$ (exercise).

Substitution method

$$E[T(n)] \le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$

$$\le \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n)$$

$$= an\lg n - \left(\frac{an}{4} - \Theta(n)\right)$$

移項將上式轉換成以 ***desired* – *residual*** 表示

(把欲證明項在一起, 再減掉其餘項)

Substitution method

$$
\begin{aligned}
E[T(n)] &\le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n) \\
&= \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n) \\
&= an\lg n - \left(\frac{an}{4} - \Theta(n)\right) \\
&\le an\lg n,
\end{aligned}
$$

當 *a* *夠大時*, *an*/4 支配 Θ(*n*) 項 (使得 *an*/4 - Θ(*n*) > 0).

Exercise 7.4-1

- 證明 $\sum_{k=2}^{n-1} k\lg k \le \frac{1}{2}n^2\lg n - \frac{1}{8}n^2$

[ANS]:

$$\sum_{k=1}^{n-1} k\lg k = \sum_{k=1}^{\lceil n/2\rceil-1} k\lg k + \sum_{k=\lceil n/2\rceil}^{n-1} k\lg k$$

拆分求和(詳附錄A)

$$\le \sum_{k=1}^{\lceil n/2\rceil-1} k\lg n + \sum_{k=\lceil n/2\rceil}^{n-1} k\lg(n-1)$$

每項分別取最大

$$\le \sum_{k=1}^{\lceil n/2\rceil-1} k(\lg n - 1) + \sum_{k=\lceil n/2\rceil}^{n-1} k\lg n$$

前後項最大對調

$$= (\lg n - 1)\sum_{k=1}^{\lceil n/2\rceil-1} k + \lg n \sum_{k=\lceil n/2\rceil}^{n-1} k$$

最大項與K值無關

$$= \lg n \sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2\rceil-1} k \le \frac{n(n-1)\lg n}{2} - \frac{1}{2}\left(\frac{n}{2}-1\right)\frac{n}{2}$$

$$\le \frac{n^2\lg n}{2} - \frac{n^2}{8}\text{, if } n \ge 2.$$

Quicksort in practice *實務上應用*

- Quicksort is a great general-purpose sorting algorithm. 絶佳通用排序演算法
- Quicksort is typically over twice as fast as merge sort. 通常比 MergeSort 快一倍
- Quicksort can benefit substantially from ***code tuning***. 藉由程式碼最佳化可大幅提昇效率
- Quicksort behaves well even with caching and virtual memory. 在快取和虛擬記憶體環境中效率表現尤佳

Summary

- **QUICKSORT**
  - Worst-case $\Theta(n^2)$
  - Expected $\Theta(n \lg n)$
  - Divide & Conquer
    - **PARTITION**
- **RANDOMIZED_QUICKSORT**
  - **RANDOMIZED PARTITION**
- Analysis of Quicksort Algorithm
  - Worst-case $\Theta(n^2)$
  - Average-case $\Theta(n \lg n)$