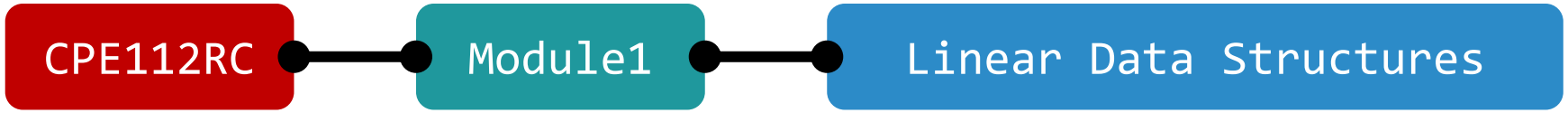


Lecture5: Basic Sorting Algorithm

Dr. Prapong Prechaprapranwong

Computer Engineering

King Mongkut's University of Technology Thonburi

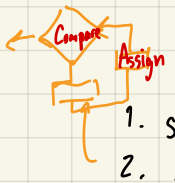
Time efficiency → Time complexity
└ อัตราเวลาที่ใช้ใน อัลกอริทึม

Space efficiency → Space complexity
└ ปัจจุบันเน้นหนักกว่า

การวัด Measuring Ru
└ มันคนละ 1 เครื่องได้ผลคนละแบบ

เราจะลด factor ที่เป็น bias

ฉะนั้นเราจะคำนวณโดยใช้ RAM model (คำนวณคร่าวๆ)
- single processor
- sequential execution - มันต่อเนื่อง
- 1 unit time for arithmetical and logical
- 1 unit time for assignment and turn



	cost	No. of iteration
1. sum ← 0	1	1
2. For i=1 to n do	2 → Assign + Compare	n+1 (ทำ n+1 ครั้ง)
3. sum ← sum+a;	2 → Assign + Arith	n (ทำ n ครั้ง)
4. return sum	1	1

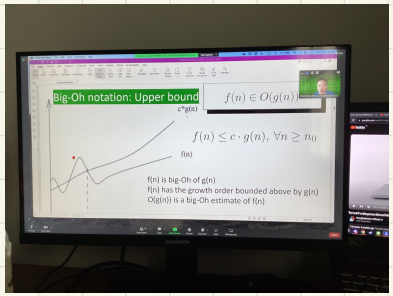
∴ total = 1 + 2(n+1) + 2n + 1 = 4n+4 ← บางครั้งเรามี human error

Algorithm running time ∝ input's size

ไม่ได้ดูเวลา แต่เราดูอัตราการเติบโต

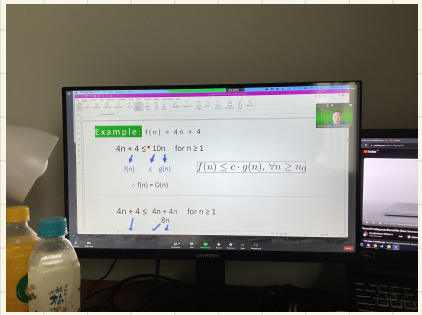
Asymptotic Notation

- Big-Oh Notation (O) - upper bound
- Big-Omega Least bound
- Big-Theta Average bound



ณ ตำแหน่ง n_0 มัน $f(n) \le c \cdot g(n)$

จะบอกว่า มันมี Big-O เป็น g(n)



Test $4n + 4 \le 10n$; $n \ge 1$
$f(n)$ c $g(n)$

$\therefore f(n) = O(n)$
└ ค่า f(n) มี Big-O เป็น n

หรือ $4n + 4 \le 4n + 4n$; $n \ge 1$
$f(n)$ c $g(n)$
$8n$

$\therefore f(n) = O(n)$

ลองอีกอัน
$4n+4 \le 4n^2+4n^2$
$8n^2$
c $g(n)$

$\therefore f(n) = O(n^2)$

สรุป ∴ ค่า Big-O คือ ทุกค่าเลย แต่ค่า Big-O ที่มีความหมายที่สุด หรือมีประโยชน์ นำมาใช้ คือ ค่าที่ใกล้ f(n) มากที่สุด

สรุป Big-Oh notation : Upper bound → หรือเรียกว่า Worst Case

F(n) มีค่าไม่เกิน เส้น O(n) และใกล้เคียงน้อยที่สุด

Case ไหนควรใช้ DS แบบ Linear หรือ Non-Linear โดยเก็บข้อมูลรวบรวมมาทำอะไร
ที่ต้องมี Algorithm Searching & Sorting เพื่อทำอะไรกับข้อมูล

Content

- Searching
- Sorting
  - Bubble Sort
  - Selection Sort
  - Insertion Sort

การเลือก ค่า มาทำให้มี ประโยชน์

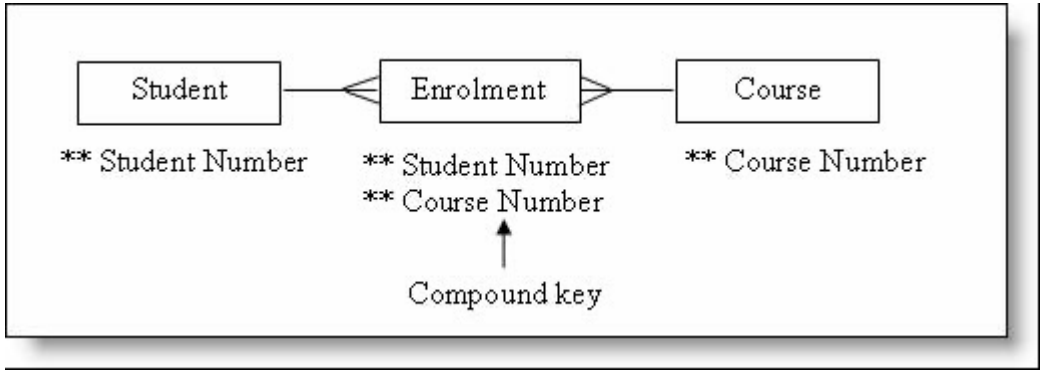
Searching is the process of selecting particular information from a collection of data based on specific criteria.

A key or **A Target** is a unique value used to identify the data elements of a collection.

↳ ค่าที่เราอยากจะได้ (identity)

- In collections containing simple types such as integers or reals, the values themselves are the keys.
  ↳ ค้นหาตัวเลขนั้นๆ โดยตรง คือ target เลย
- For collections of complex types, a specific data component has to be identified as the key.
  dictionary - เราอยากได้ค่าชิ้น แต่เราต้องใส่ค่า key เพื่อค้นหาและโยงไปค่าแท้จริง
- In some instances, a key may consist of multiple components, which is also known as a **compound key**.

↳ เราใช้ key มากกว่า 1 key ในการหาข้อมูล

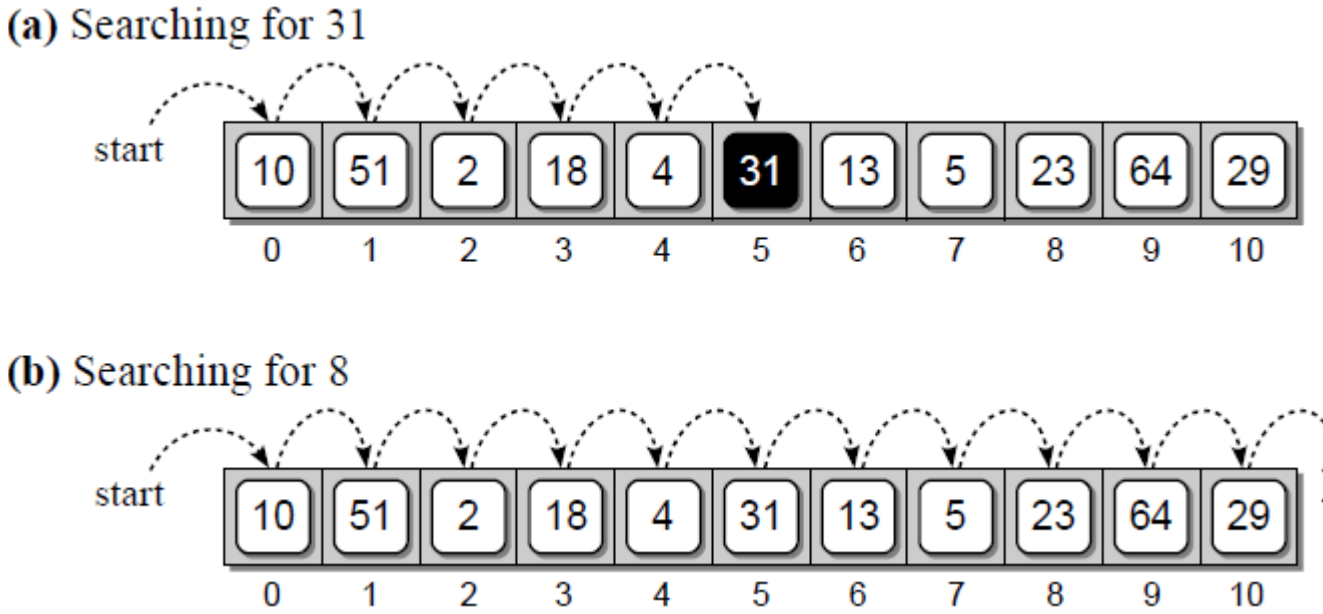


Sequential search or **Linear search** is the standard approach to search for a target value is to use a loop to examine every element, until either finding the target or exhausting the data set

```
def sequential_search(seq,key):
    for i in range(len(seq))
        if seq[i] == key: return True
return False
```

```
def sequential_search(seq, key):
    for item in seq:
        if item == key: return True
    return False
```

Searching in array or python list

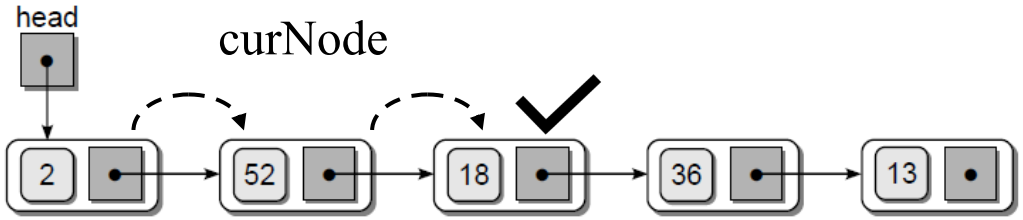


Sequential search in linked list

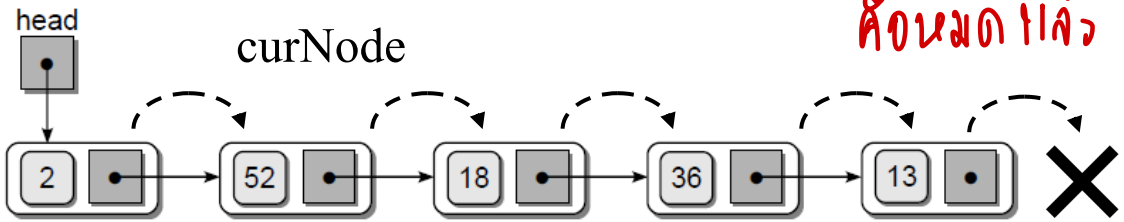
```
def search(self,target):
    curNode = self._head
    while curNode is not None and curNode._item != target:
        curNode = curNode._next
    return curNode is not None
```

Searching in linked list

(a) Searching for 18



(b) Searching for 23



curNode = None คือหมดแล้ว

กรณีที่แย่ที่สุดใน Linear search คือ

Assuming the sequence contains n items, the sequential search or linear search has a worst case time of O(n).

แต่ Average time คือ สุ่มมาแล้วเจอ

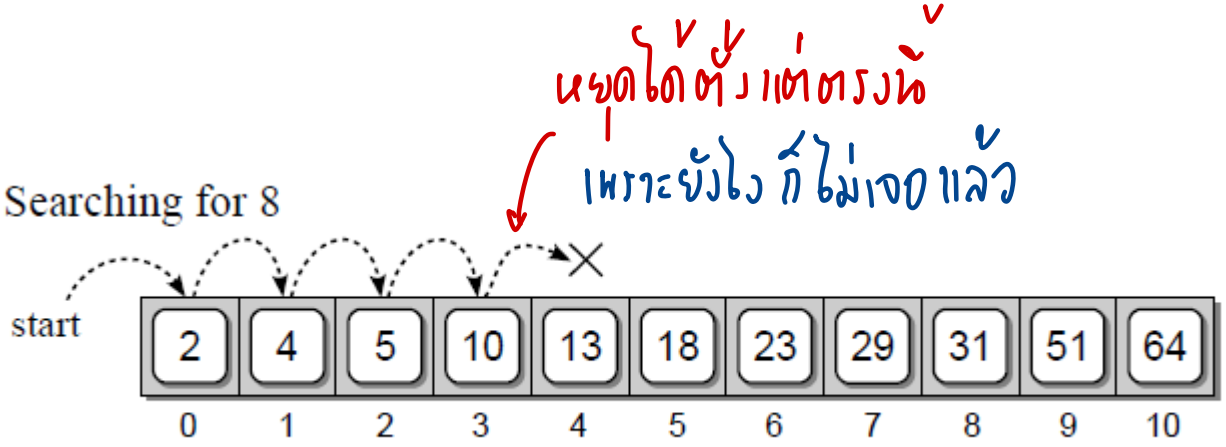
↳ มันขึ้นกับ n และสถานการณ์ → เช่น มีล้านตัว แต่หาเจอตัวแรก Best case time คือ O(1)

Searching a sorted sequence

เราสามารถ Improve worst case ได้โดยการ sort ข้อมูลก่อน

A linear search on a sorted sequence works in the same fashion as that for the unsorted sequence, with one exception. It's possible to terminate the search early when the value is not in the sequence instead of always having to perform a complete traversal.

```
def ordered_sequential_search(seq, key):
    [redacted]
    for item in seq:
        if item > key: return False
        if item == key: return True
    return False
```



ถึงการ sort ข้อมูล ก็ต้องเสียเวลาด้วย
บางครั้ง ถึง search ไปเลย อาจจะเร็วกว่า

Sorting is the process of arranging or ordering a collection of items such that each item and its successor satisfy a prescribed relationship.

มีหลากหลาย Algorithm ที่มากกว่า 20 แบบ

- Bubble Sort
- Selection Sort
- Insertion Sort
- Count Sort
- Heap Sort → โดด Algorithm
- Merge Sort
- Quick Sort
- Radix Sort
- Bucket Sort

Quadratic $O(n^2)$

Linear $O(n)$

Loglinear $O(n \log n)$

Bubble Sort

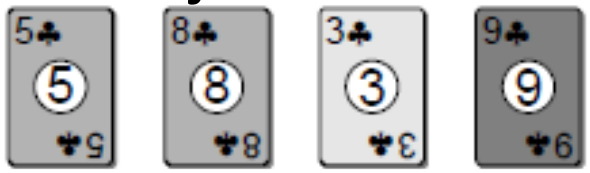
```
1 # Sorts a sequence in ascending order using the bubble sort algorithm.
2 def bubbleSort( seq ):
3         n = len( seq ) - 1
4         # Perform n-1 bubble operations on the sequence
5         for i in range( n , 0 , -1) :
6         # Bubble the largest item to the end
7                 for j in range(i) :
8                   if seq[j] > seq[j + 1] : # swap the j and j+1 items
9                           tmp = seq[j]
10                          seq[j] = seq[j + 1]
11                          seq[j + 1] = tmp
```

n = 3

i = 3

j = 0..2

i = 2 j = 0..1



```
1 # Sorts a sequence in ascending order using the bubble sort algorithm.
2 def bubbleSort( seq ):
3         n = len( seq ) - 1
4         # Perform n-1 bubble operations on the sequence
5         for i in range( n , 0 , -1) :
6         # Bubble the largest item to the end
7                 for j in range(i) :
8                   if seq[j] > seq[j + 1] : # swap the j and j+1 items
9                         tmp = seq[j]
10                        seq[j] = seq[j + 1]
11                        seq[j + 1] = tmp
```

Python

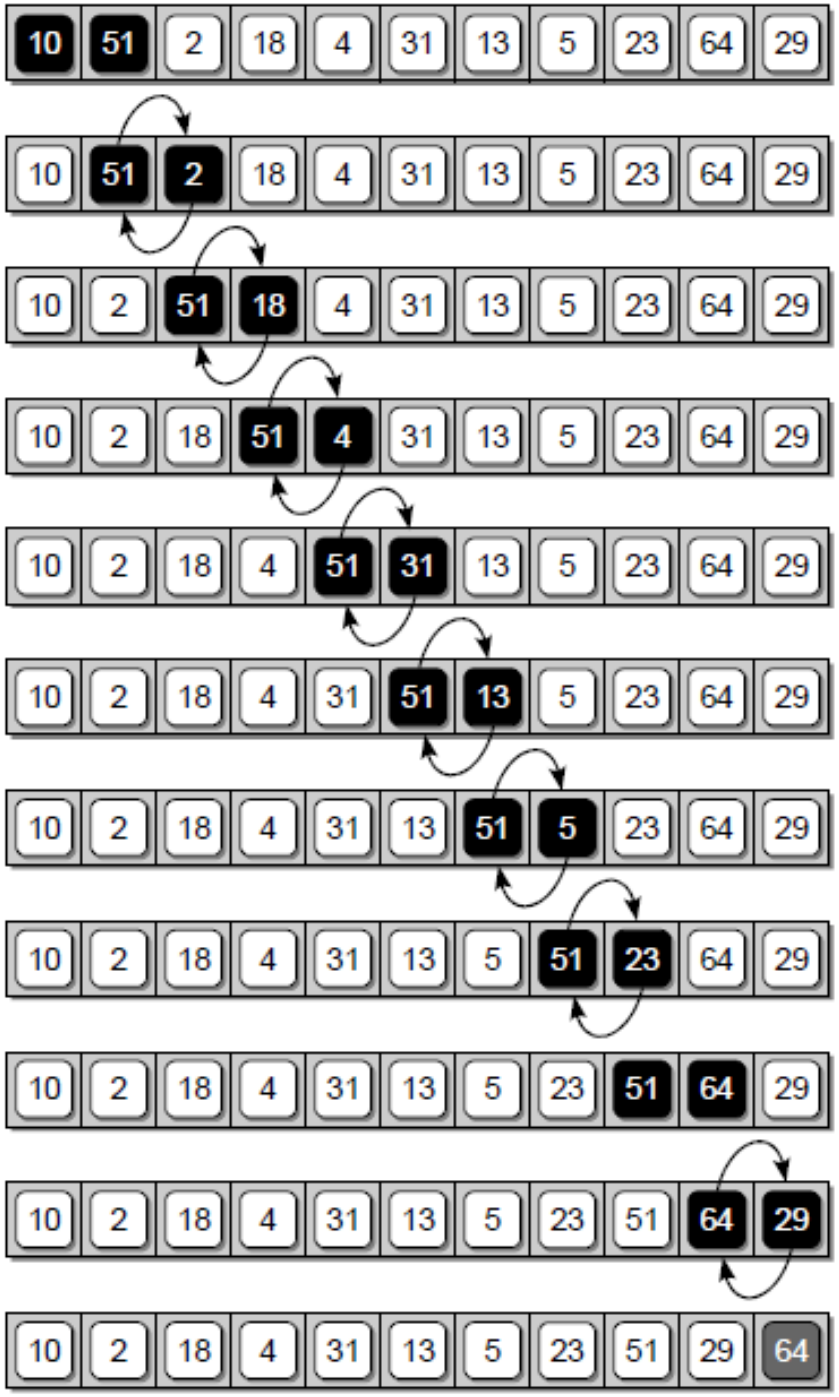
seq[j], seq[j+1] = seq[j+1], seq[j]

i = i j = 0

มองว่าเป็นฟองอากาศในแก้ว ลอยขึ้นมาเรื่อยๆ

หนึ่งใน อัลกอริทึมที่ไม่มีประสิทธิภาพ มากที่สุด

Bubble sort is considered one of the most inefficient sorting algorithms due to the total number of swaps required.



n = 10, i = 10, j = 0..9

n = 10, i = 9, j = 0..8

n = 10, i = 8, j = 0..7

n = 10, i = 7, j = 0..6

n = 10, i = 6, j = 0..5

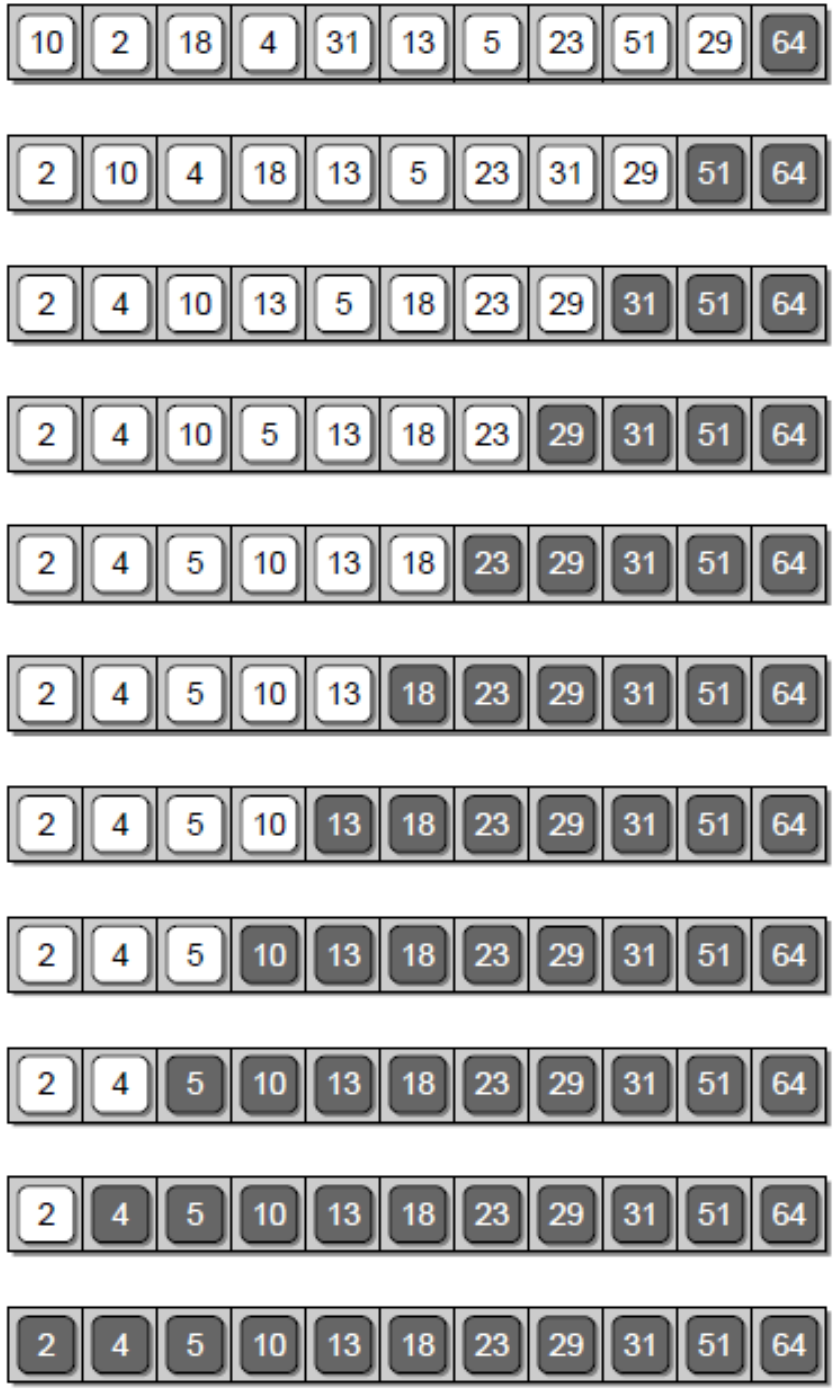
n = 10, i = 5, j = 0..4

n = 10, i = 4, j = 0..3

n = 10, i = 3, j = 0..2

n = 10, i = 2, j = 0..1

n = 10, i = 1, j = 0

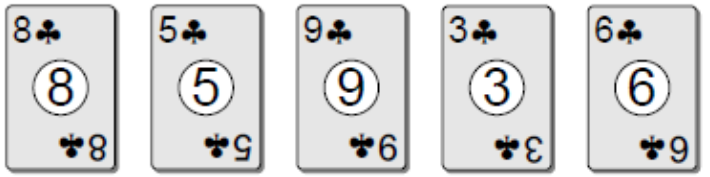


ขั้นตอน Bubble Sort

Selection Sort

คล้ายกับวิธีที่มนุษย์ใช้ sort

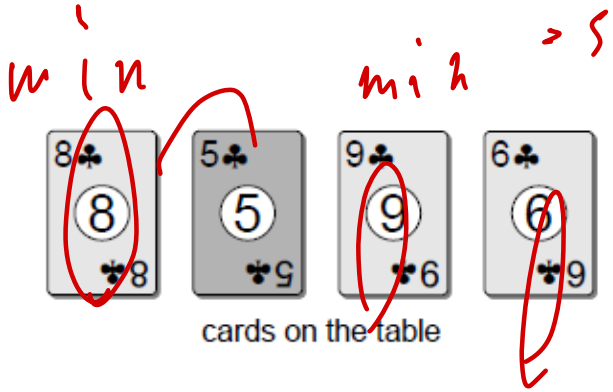
Selection Sort improves on the bubble sort and works in a fashion similar to what a human may use to sort a list of values

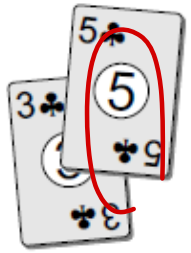


หาค่าที่น้อยสุดออกมาก่อน

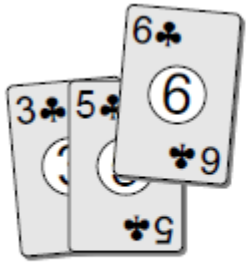
our hand

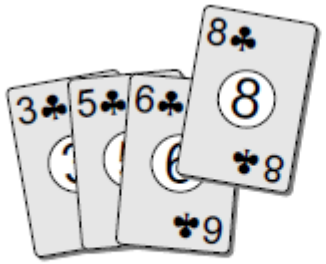
cards on the table

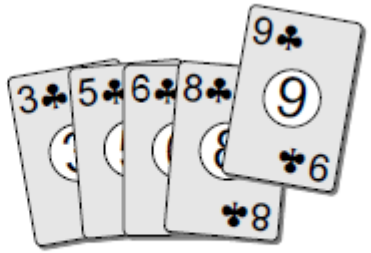

cards on the table


our hand

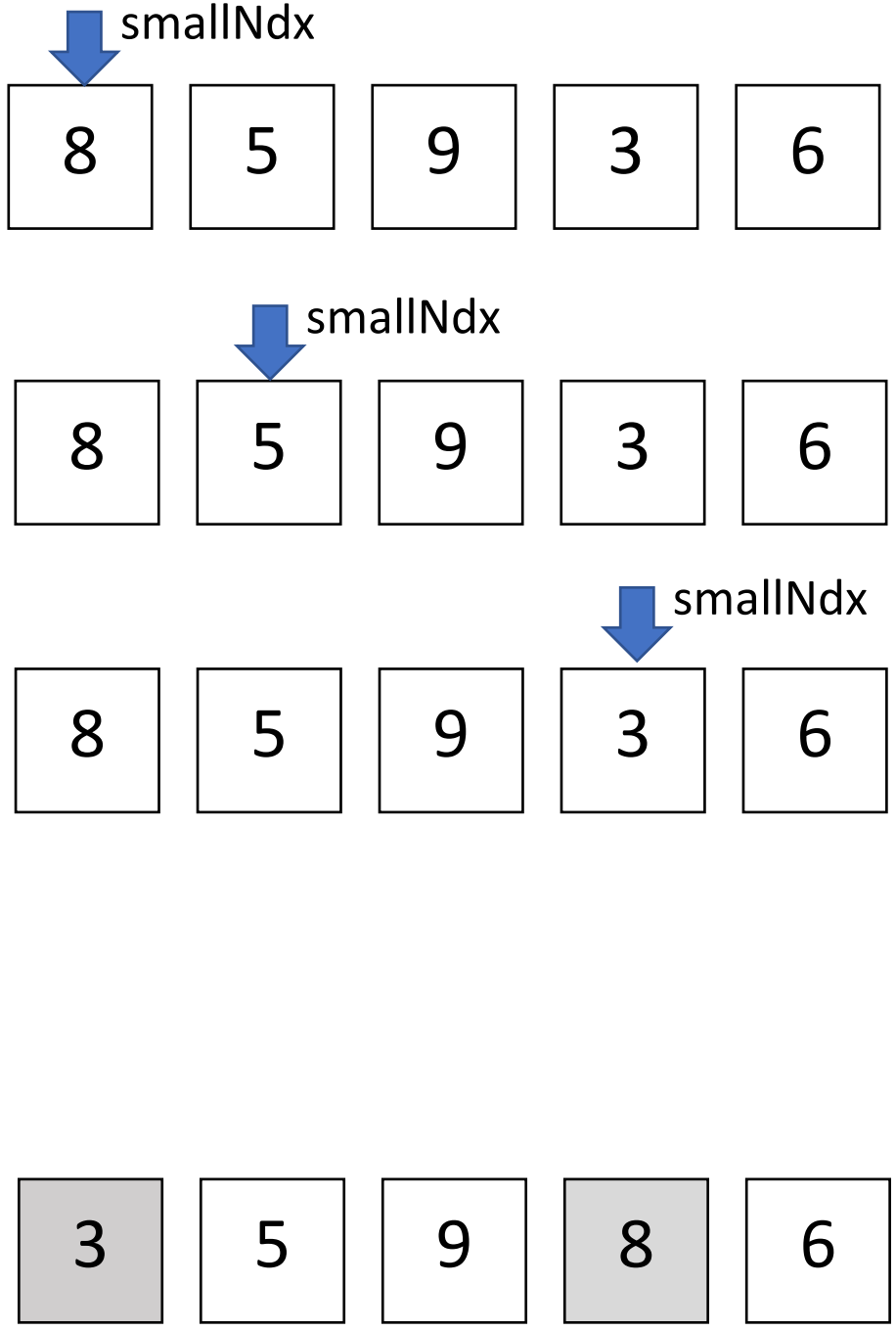

cards on the table


pick up the next
smallest card (6)


pick up the next
smallest card (8)


pickup the last
card (9)


the resulting hand



n = 5

i = 0

smallNdx = 0

j = 4

smallNdx = 3

swap

```
def selectionSort( seq ):
    n = len( seq )
    for i in range( n - 1 ):
        smallNdx = i
        for j in range( i + 1, n ):
            if seq[j] < seq[smallNdx] :
                smallNdx = j
        if smallNdx != i :
            tmp = Seq[i]
            seq[i] = seq[smallNdx]
            seq[smallNdx] = tmp
    return seq
```

ใส่ตัวที่จะ swap มาตน แรก

ใส่จากตัวที่อยู่ ไป สุด

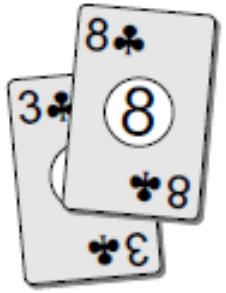

10
51
2
18
4
31
13
5
23
64
29

Insertion Sort

the deck

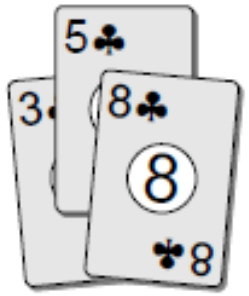
our hand

the deck



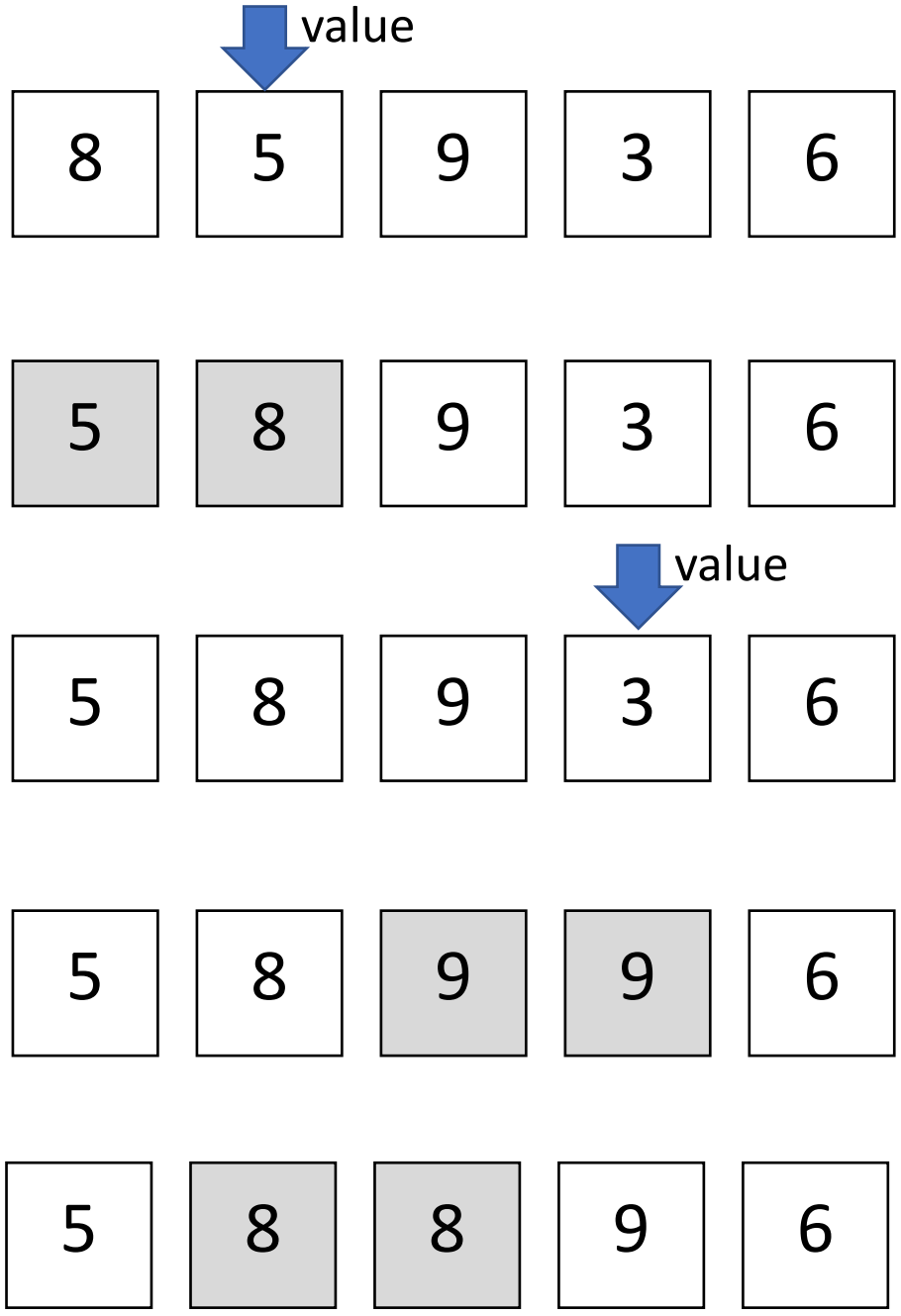
our hand

the deck



our hand

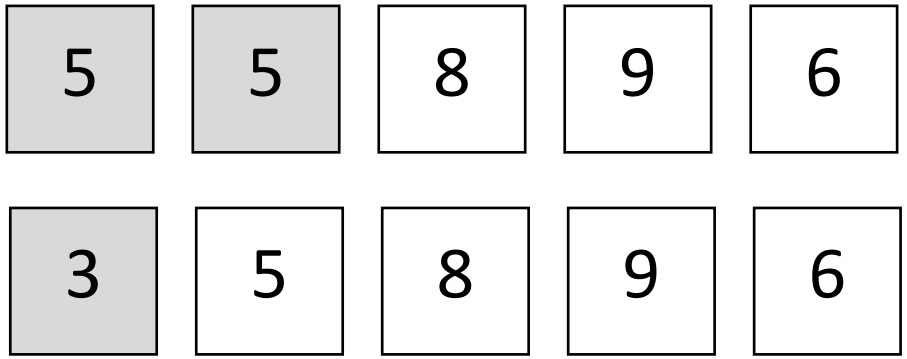
the deck



n = 5
i = 3
value = 3
pos = 3
pos = 0

```
def insertionSort(seq):
    n = len(seq)
    for i in range(1,n):
        value = seq[i]
        pos = i
        while pos > 0 and value < seq[pos -1]:
            seq[pos] = seq[pos - 1]
            pos -= 1
        seq[pos] = value
    return seq
```

ก็อปปี้ค่ามาทับ



Summary

กระบวนการที่ใช้ในการหา ข้อมูล

- **Searching** is the process of selecting specific information from a collection of data
- **Sequential search/Linear search** is the standard approach in searching to examine every element, until either finding the target or exhausting the data set
- The sequential search/linear search has a worst-case time complexity of O(n).
- A linear search can be improved by pre-sorting the sequence then we can early terminate the search if we know that the key is surely not in the sequence
- **Sorting** is the process of arranging or ordering an orderable collection of items
- **Bubble sort**, **Selection sort**, **Insertion sort** is categorized into **Quadratic sorting**

What should you know

- Understand searching process in each type of data structures
- How to implement Sorting algorithm with python
  - Bubble Sort
  - Selection sort
  - Insertion sort

Average Case ของการ sorting คือ สุ่ม

Best case ของการ sorting คือ เรียงมาแล้ว sorted

Worst Case ของ การ sorting คือ ถูกเรียงลำดับแบบ reversed order

↳ สรุปทั้งหมด 3 กรณี

จริงๆในพวก sort เราไม่ต้อง return ใน def ก็ได้

ถ้าเราไม่ใส่ return มันจะเปลี่ยนแปลงค่าใน List ที่ส่งมาเอง

Scoop Rule - ค่าตัวแปรที่ถูกใส่มา จะถูกมองเป็น Local มันจะไม่ส่งค่ากลับไป

แต่ที่มันเปลี่ยน เพราะ Python ไม่มีแบบ By value, แต่มันจะใช้แบบที่ๆ "Object Reference." " Coring Type

เพราะ Python มี Coring Type โดยใช้แบบ reference และห่อด้วย object-orientend

สรุปคือมัน Reference กลับไปที่ค่าเดิมด้วย

Concept แบบ Professional มาก (Python)

id (seq-A) - ตำแหน่ง Address

seq-A

id(seq-A) → address เดียวกัน

seq-B = selectionSort(seq-A)

print(seq-B)

id(seq-B)

**

Pass แบบ reference เอา variable seq-B มารับ obj ทั้งก้อนที่มาจาก A

มันจะชี้ไปที่ reference obj เดียวกัน

∴ seq-A ก็จะถูกเรียงในตัวเองเช่นกัน

แกน x - จำนวนข้อมูล

แกน y - เวลาที่ใช้ในการ sorting

} โดยเส้นแต่ละเส้นอยู่ในกราฟเดียว

คือ Bubble, Select, Insertion

กราฟเป็น 3 เส้น Compare กราฟซึ่งกันและกัน