204700
Data Structure and Programming Languages

Jakarin Chawachat

Outline

- การค้นหา (Searching)

  การค้นหาแบบเรียงลำดับ (Sequential Search)

  การค้นหาแบบทวิภาค (Binary Search)
- การเรียงลำดับ (Sorting)

  การเรียงลำดับแบบเลือก (Selection Sort)

  การเรียงลำดับแบบฟอง (Bubble Sort)

  การเรียงลำดับแบบแทรก (Insertion Sort)

  การเรียงลำดับแบบผสาน (Merge Sort)

  การเรียงลำดับแบบเร็ว (QuickSort)

Searching

Problem : Searching in Array

Input: A คือ ตารางขนาด 1xn ของตัวเลขจำนวน n ค่า

x คือ ตัวเลขที่ต้องการค้นหา

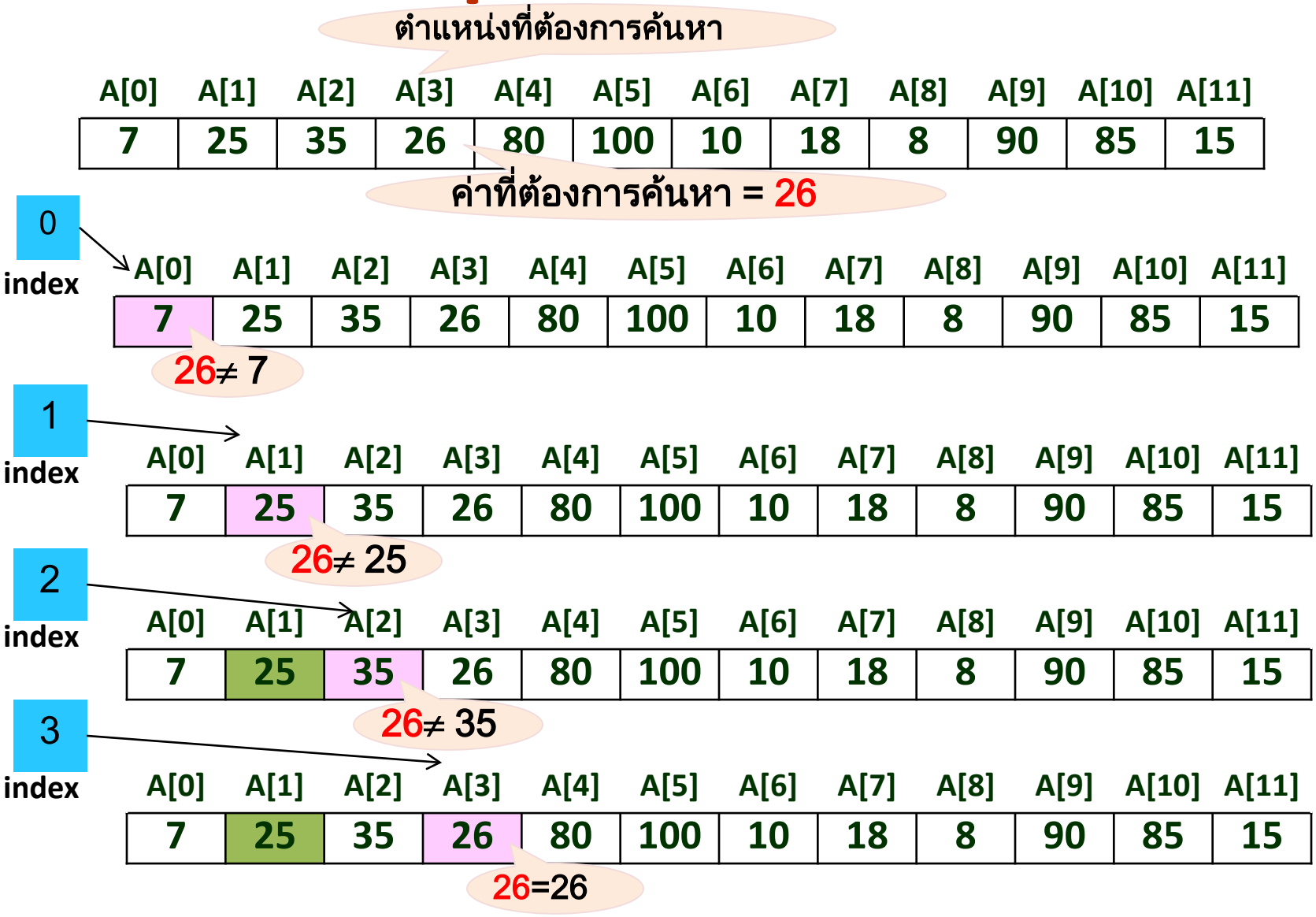
Output: ตำแหน่งของ x ในตาราง A

Perform operation find(k)

การหาตำแหน่งของหน่วยความจำที่เก็บข้อมูลที่ต้องการค้นหา

แจ้งเมื่อไม่สามารถหาข้อมูลได้พบเช่น กรณีหาไม่พบ จะคืนค่า NULL

Sequential Search

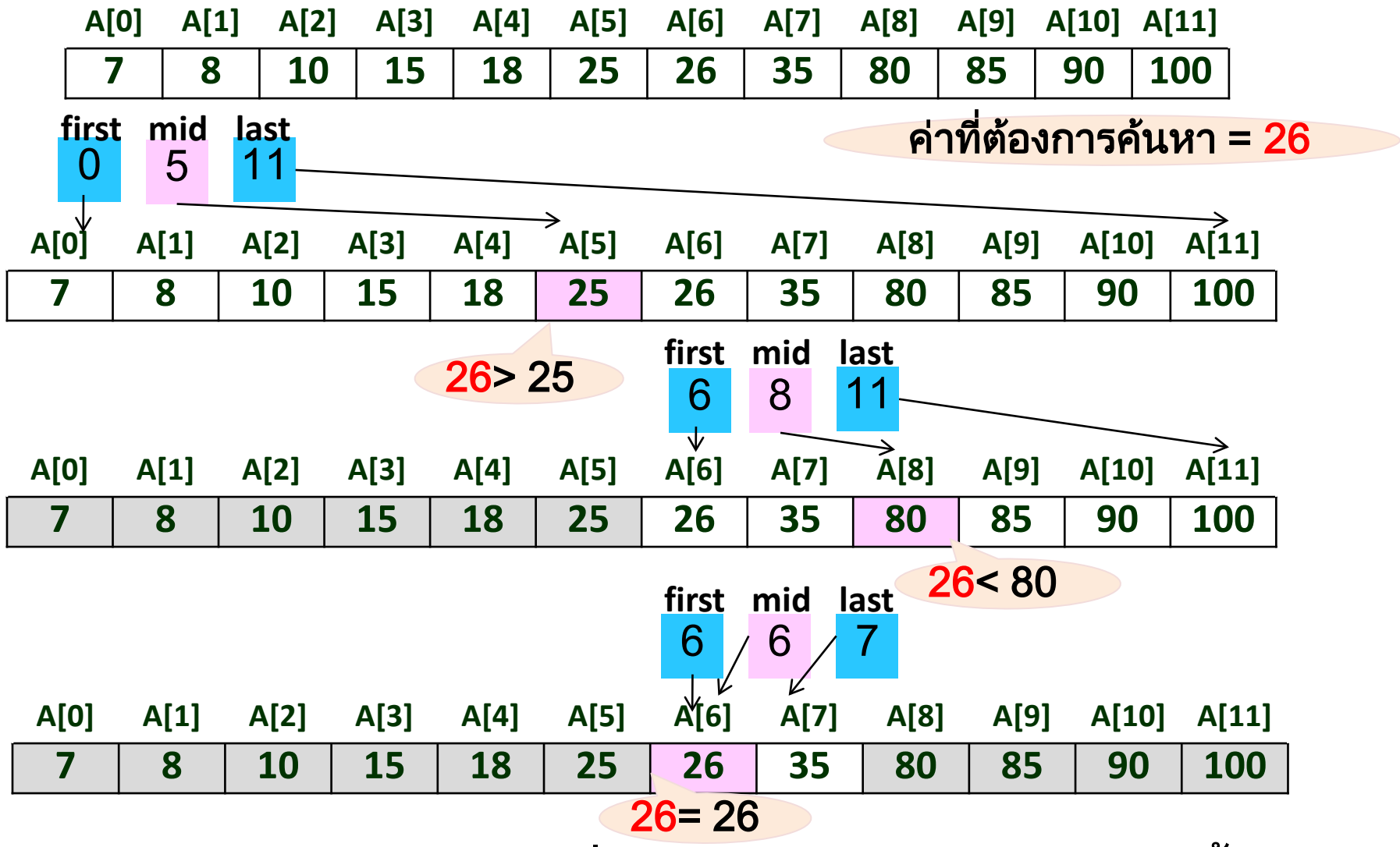


- ค้นหา 26 เจอในตำแหน่งที่ 4 ด้วยการเปรียบเทียบ 4 ครั้ง

Sequential Search

```
static int sequential_search(int numbers[], int size, int key) {
        int i = 0;
        while (i < size) {
                if (key == numbers[i]) {
                        // คืนค่าว่าพบข้อมูลอยู่ในตำแหน่งใดของโครงสร้างข้อมูล
                        return i;
                } else {
                        i = i + 1;
                 }
         }
        return -1;
}
```

Binary Search



- ค้นหา 26 เจอในตำแหน่งที่ 7 ด้วยการเปรียบเทียบค่า 3 ครั้ง

Assignment

- ให้ทำ Binary Search

Sorting

- การเรียงลำดับข้อมูล
  - จากน้อยไปมาก (Ascending)
  - จากมากไปน้อย (Descending)
- คำนึงถึงความถูกต้องและความรวดเร็ว

Problem : Sorting in Array

Input: A sequence of n numbers (a1, a2, …, an)

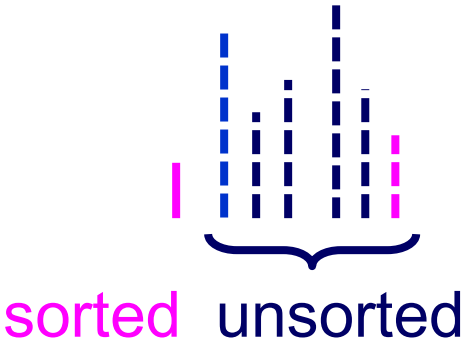
Output: The permutation (reordering) of the input sequence such as (á1, á2, …, án) such that (á1<= á2<= … <= án)

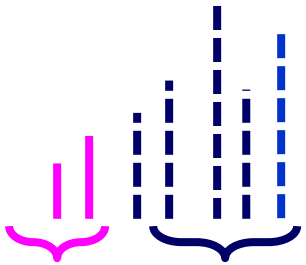
๑

Selection Sort

อัลกอริทึมการเรียงลำดับแบบเลือก

1. สมมติตำแหน่งแรกในชุดข้อมูลที่เหลืออยู่เป็นสมาชิกที่ถูกเลือก
2. เปรียบเทียบกับข้อมูลที่เหลือทุกตัวเพื่อหาข้อมูลที่มีค่าน้อยที่สุด(หรือมากที่สุด)
3. นำข้อมูลที่มีค่าน้อยที่สุด(จากข้อ 2)สลับกับสมาชิกที่ถูกเลือก
4. เปรียบเทียบต่อไปจนหมดชุดข้อมูลที่มี

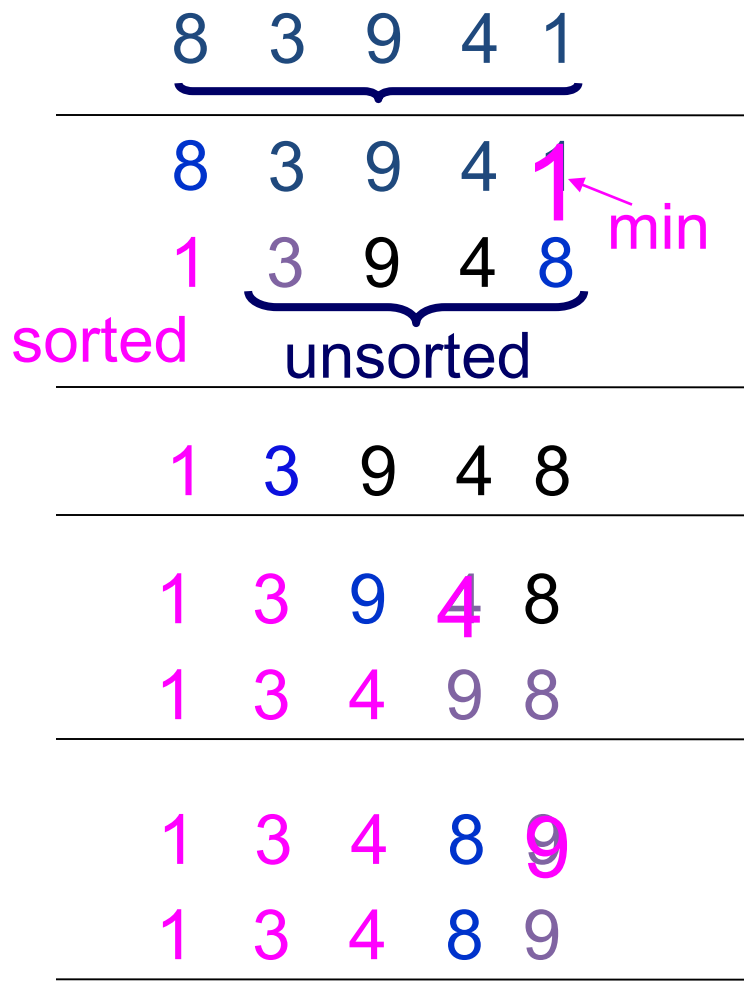




๑

Selection Sort

1. สมมติตำแหน่งแรกในชุดข้อมูลที่เหลืออยู่เป็นสมาชิกที่ถูกเลือก
2. เปรียบเทียบกับข้อมูลที่เหลือทุกตัวเพื่อหาข้อมูลที่มีค่าน้อยที่สุด
3. นำข้อมูลที่มีค่าน้อยที่สุดสลับกับสมาชิกที่ถูกเลือก
4. เปรียบเทียบต่อไปจนหมดชุดข้อมูลที่มี



Selection Sort

```
static void selection_sort(int numbers[], int array_size){
        for (int i = 0; i < array_size-1; i++)  {
                int min = i;
                for (int j = i+1; j < array_size; j++){
                        if (numbers[j] < numbers[min]){
                                min = j;
                        }
                }
                int temp = numbers[i];
                numbers[i] = numbers[min];
                numbers[min] = temp;
        }
}
```

ตัวสุดท้ายไม่ต้องทำ(แค่ n-1 รอบ)

ขั้นตอนที่ ๒ หา min

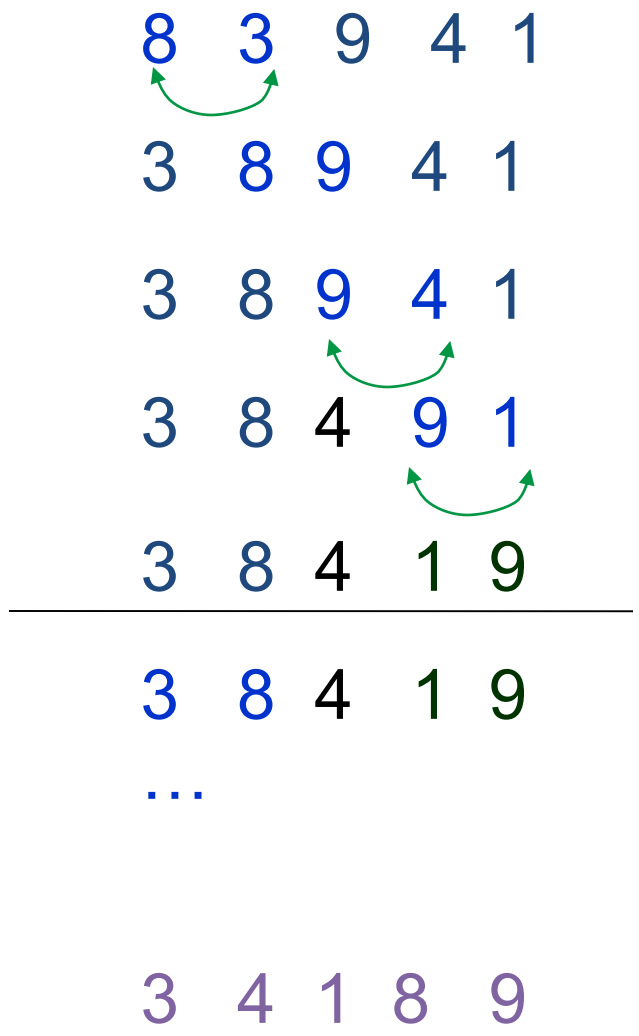
ขั้นตอนที่ ๓ สลับค่า

๒

Bubble Sort

อัลกอริทึมการเรียงลำดับแบบฟอง

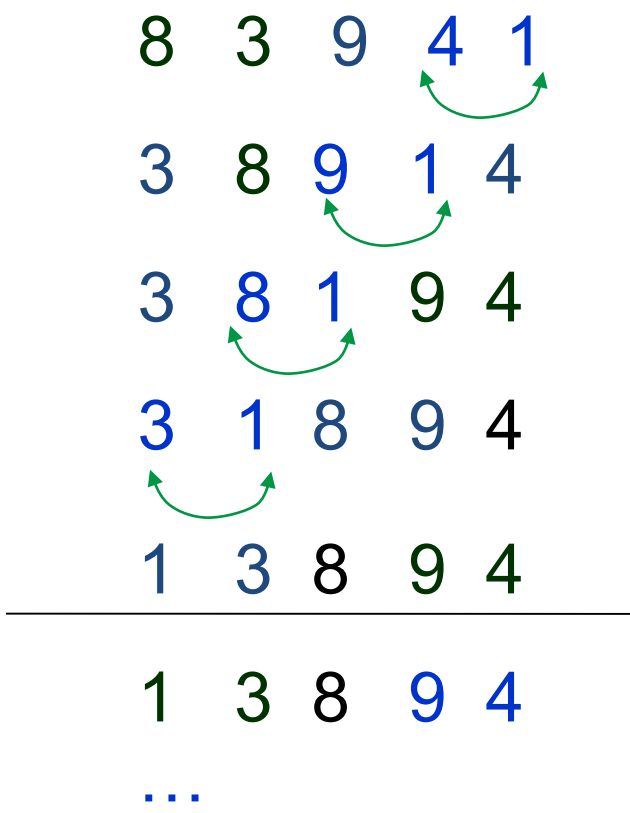
1. เปรียบเทียบข้อมูลที่ติดกัน ถ้าข้อมูลตัวแรกมากกว่าตัวที่สอง ให้ทำการสลับตำแหน่งกัน
2. เปลี่ยนคู่ข้อมูล โดยเปรียบเทียบกับข้อมูลใหม่ที่ติดกันไปทีละคู่
   - ถ้าตัวแรกมากกว่าตัวที่สองให้ทำการสลับตำแหน่งกัน
   - วนทำซ้ำจนครบทุกคู่ข้อมูล



๒

Bubble Sort

```
static void bubble_sort(int numbers[], int array_size) {
    int i, j, temp;
    for (i = (array_size - 1); i >= 0; i--){
        for (j = 1; j <= i ; j++){
            if (numbers[j-1] > numbers[ j ]) {
                temp = numbers[ j-1 ];
                numbers[ j-1 ] = numbers[ j ];
                numbers[ j ] = temp;
            }
        }
    }
}
```



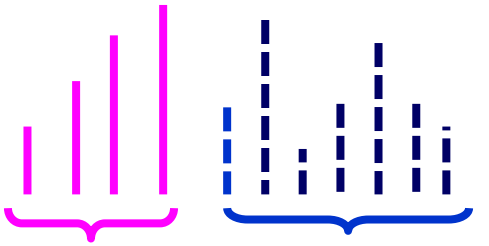
อัลกอริทึมนี้ ให้ผลลัพธ์เหมือนอัลกอริทึม ใน slide ก่อนหน้านี้หรือไม่ ?

๓

Insertion Sort

อัลกอริทึมการเรียงลำดับแบบแทรก

1. ข้อมูลนำเข้าเป็นสมาชิกตัวถัดไป
2. เปรียบเทียบข้อมูลนำเข้ากับผลลัพธ์ปัจจุบัน
3. แทรกข้อมูลนำเข้าในตำแหน่งที่เหมาะสม
4. วนทำซ้ำขั้นตอนที่ 2 สำหรับข้อมูลนำเข้าตัวถัดไป



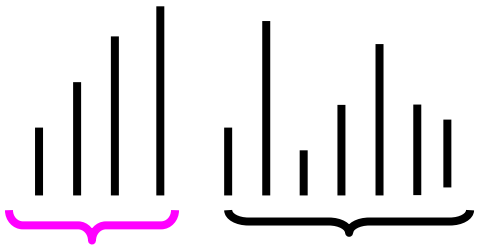
1)sorted 2)not yet examined

3 8 9 4 1

key

3 4 8 9 1

Insertion Sort



1)sorted 2)not yet examined

➔starting

8 3 9 1 4

key

1)sorted 2)not yet examined

8 3 9 4 1
3 8 9 4 1
3 8 9 4 1

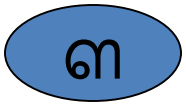
---

3 8 9 4 1

---

3 8 9 4 1
3 8 4 9 1
3 4 8 9 1
3 4 8 9 1

....



Insertion Sort

```
static void insertion_sort(int numbers[], int array_size) {
        int i, j, index;
        for (i = 1; i < array_size; i++) {
                index = numbers[i];
                j = i;
                while ((j > 0) && (numbers[j-1] > index)) {
                        numbers[j] = numbers[j-1];
                        j = j - 1;
                }
                numbers[j] = index;
        }
}
```

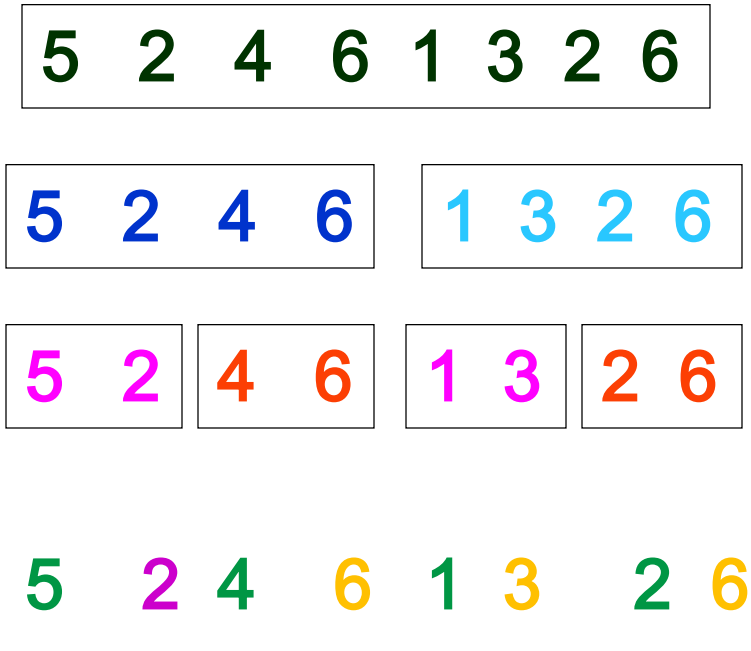
๔

Merge Sort

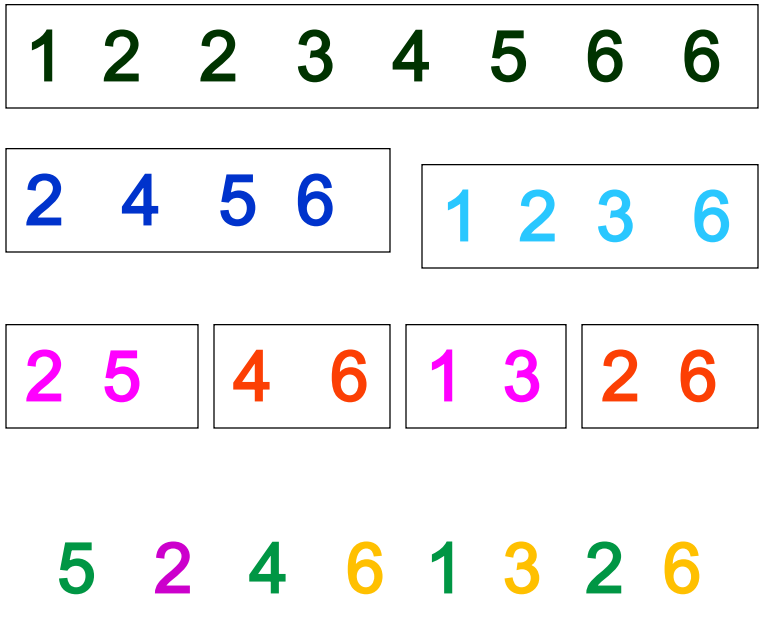
- การเรียงลำดับแบบผสาน ใช้หลักการ **Divide and Conquer**
  - แบ่งข้อมูลออกเป็นส่วนย่อย (**Divide)** สองส่วนเท่า ๆ กัน แล้ววนซ้ำเพื่อแบ่งแยกข้อมูลจนมีข้อมูลเพียงหนึ่ง (ข้อมูลในส่วนย่อยที่สุดของการแบ่งแยก)
  - ทำการเรียงลำดับข้อมูลในส่วนย่อยแต่ละส่วน แล้วจึงจะนำข้อมูลย่อยแต่ละส่วนที่ได้รับการจัดเรียงเรียบร้อยแล้ว มาทำการผสานเพื่อจัดเรียงข้อมูลทั้งหมดอีกครั้งหนึ่ง (**Conquer)**
- **Recursive Algorithm**

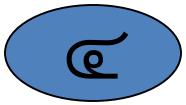
Merge Sort

1) แบ่งแยกเป็น 2 ส่วน



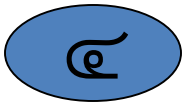
2) ผสาน (merge)





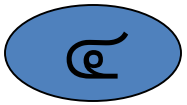
Merge Sort

```
static void merge_sort(int numbers[], int temp[], int left,
int right) {
        int mid;
        if (right > left) {
                mid = (right + left) / 2;
                merge_sort(numbers, temp, left, mid);
                merge_sort(numbers, temp, mid+1, right);
                //เรียกอัลกอริทึมการผสาน
                merge(numbers, temp, left, mid+1, right);
        }
}
```



Merge Sort : ฟังก์ชัน merge()

```
static void merge(int numbers[], int temp[], int left, int mid, int right) {
        int i, left_end, num_elements, tmp_pos;
        left_end = mid - 1;
        tmp_pos = left;
        num_elements = right - left + 1;
        while ((left <= left_end) && (mid <= right)) {
                if (numbers[left] <= numbers[mid]) {
                        temp[tmp_pos] = numbers[left];
                        tmp_pos = tmp_pos + 1;
                        left = left + 1;
                } else {
                        temp[tmp_pos] = numbers[mid];
                        tmp_pos = tmp_pos + 1;
                        mid = mid + 1;
                }
        }
```



Merge Sort : ฟังก์ชัน merge()

```
while (left <= left_end) {
        temp[tmp_pos] = numbers[left];
        left = left + 1;
        tmp_pos = tmp_pos + 1;
}
while (mid <= right) {
        temp[tmp_pos] = numbers[mid];
        mid = mid + 1;
        tmp_pos = tmp_pos + 1;
}
for (i = 0; i < num_elements; i++) {
        numbers[right] = temp[right];
        right = right - 1;
}
}
```

Assignment

Implement sorting algorithms ทั้ง 4 แบบ