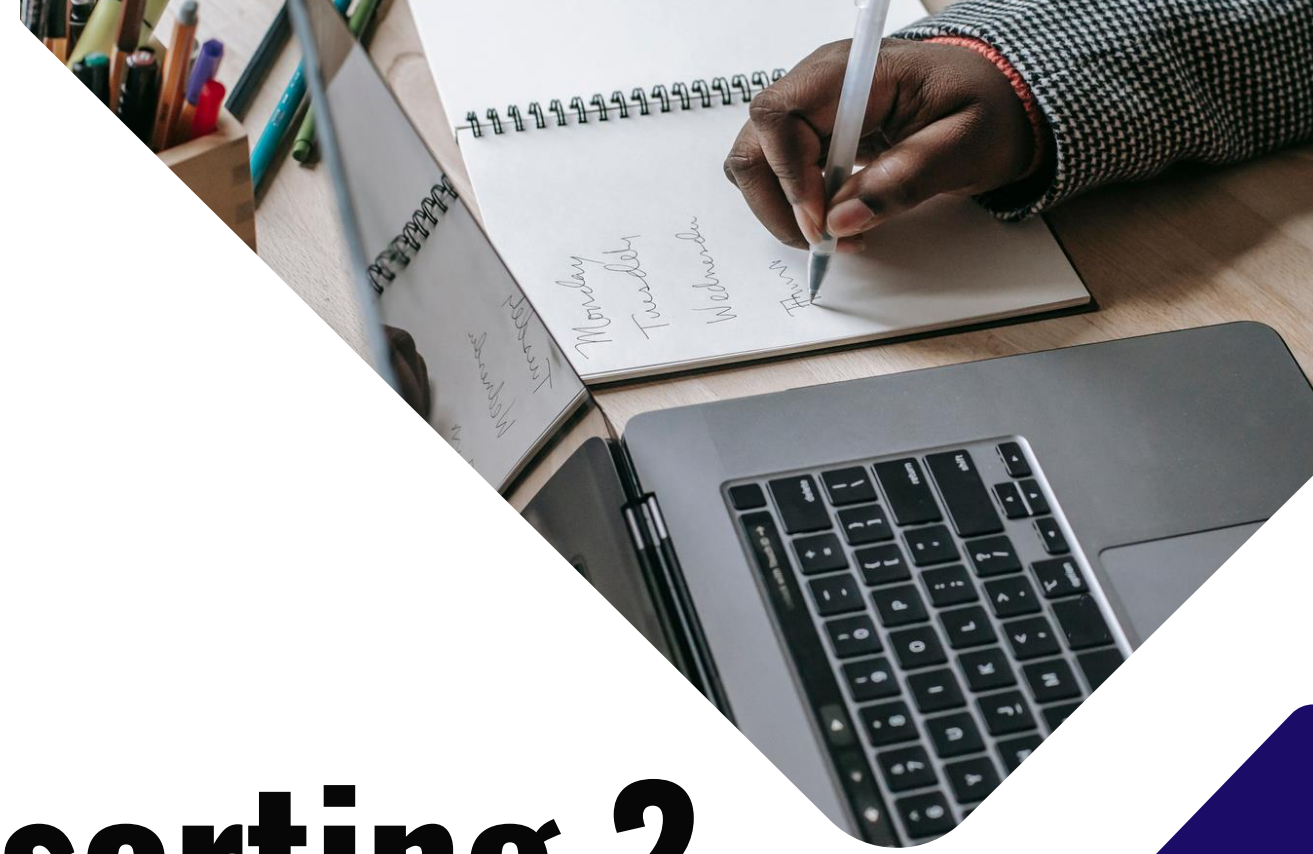
SORT

By KruNueng



What is sorting ?

Sorting Algorithms

- Bubble Sort
- Quick Sort
- Selection Sort
- Heap Sort
- Insertion Sort
- Redix Sort
- Merge Sort
- Bucket Sort

Bubble Sort

รูปแบบวิธีการเปรียบเทียบและสลับข้อมูลทีละคู่ โดยทำซ้ำกระบวนการนี้ไปเรื่อย ๆ จนกระทั่งข้อมูลทั้งหมดเรียงลำดับ

- เริ่มต้นที่เปรียบเทียบค่าของ element ที่ index 0 กับ element ที่ index 1
- ถ้า element ที่ index 0 มีค่ามากกว่า index 1 ให้สลับค่า element ทั้งสอง
- ทำกระบวนการเปรียบเทียบและสลับเรื่อย ๆ จนกระทั่งไปถึงตำแหน่งสุดท้ายของ array

Bubble Sort

Workshop : Bubble Sort

ไฟล์ : 1_bubble_sort

ปัญหา : ให้นักเรียนเขียนโปรแกรมจัดเรียงข้อมูล bubble sort

ข้อมูลรับเข้า : บรรทัดแรก n คือ จำนวนข้อมูลที่ต้องการรับ

บรรทัดสองถึง n+1 จำนวนเต็มที่ มีค่า 0<ตัวเลข<100

ผลลัพธ์ : ตัวเลขที่เรียงจากน้อยไปหามาก

ตัวอย่าง :

```
5
15
9
-7
25
62

Unsorted array: 15  9  -7  25  62

Sorted array: -7  9  15  25  62
```

Selection Sort

รูปแบบวิธีการเปรียบเทียบข้อมูลทั้งหมด เพื่อค้นหาค่า element ที่น้อยที่สุด ใน subarray แล้วนำมาสลับตำแหน่งกับ element ที่อยู่ที่ตำแหน่งแรกของ subarray

ทำแบบนี้ซ้ำ ๆ และสลับค่า element กับ element ที่อยู่ที่ตำแหน่งถัดไปของ subarray

Selection Sort

Workshop : Selection Sort

ไฟล์ : 2_selection_sort

ปัญหา : ให้นักเรียนเขียนโปรแกรมจัดเรียงข้อมูล selection sort

ข้อมูลรับเข้า : บรรทัดแรก n คือ จำนวนข้อมูลที่ต้องการรับ
บรรทัดสองถึง n+1 จำนวนเต็มที่ มีค่า 0<ตัวเลข<100

ผลลัพธ์ : ตัวเลขที่เรียงจากน้อยไปหามาก

ตัวอย่าง :

```
5
15
9
-7
25
62

Unsorted array: 15  9  -7  25  62

Sorted array: -7  9  15  25  62
```

Insertion Sort

อัลกอริทึมที่เรียงลำดับข้อมูลโดยการเรียงข้อมูลทีละ element ของ array เริ่มจาก element ที่ Index 0 ลงใน subarray ใหม่ ที่เรียงลำดับไว้ โดยการ "แทรก" element ใหม่ลงไปในตำแหน่งที่ถูกต้อง

Insertion Sort

Workshop : Insertion Sort

ไฟล์ : 3_insertion_sort

ปัญหา : ให้นักเรียนเขียนโปรแกรมจัดเรียงข้อมูล insertion sort

ข้อมูลรับเข้า : บรรทัดแรก n คือ จำนวนข้อมูลที่ต้องการรับ

บรรทัดสองถึง n+1 จำนวนเต็มที่ มีค่า 0<ตัวเลข<100

ผลลัพธ์ : ตัวเลขที่เรียงจากน้อยไปหามาก

ตัวอย่าง :

```
5
15
9
-7
25
62

Unsorted array: 15  9  -7  25  62

Sorted array: -7  9  15  25  62
```

Workshop :

ปัญหา : Sorting A-Z

ข้อมูลรับเข้า : บรรทัดแรก คือ จำนวนตัวอักษรที่จะจัดเรียง 2<n<99
บรรทัดสอง คือ ตัวอักษร A-Z มีจำนวนอักขระเท่ากับ n จำนวน โดยสามารถป้อนตัวอักษรซ้ำกันได้

ผลลัพธ์ : มี 1 บรรทัด เรียงตัวอักษรจาก A-Z โดยแต่ละตัวจะเว้น 1 ช่องว่าง

ตัวอย่าง :

ข้อมูลเข้า	ผลลัพธ์
8 COMPUTER	C E M O P R T U

พัก 15 นาที

Merge Sort

อัลกอริทึมแบบ "Divide and Conquer" ที่แบ่งข้อมูลเป็นสองส่วน และเรียงลำดับแต่ละส่วน สุดท้ายนำมารวมกลับเข้าด้วยกัน

1. การแบ่ง (Divide):

- แบ่ง array ที่ไม่เรียงลำดับเป็นสองส่วนเท่า ๆ กัน
- แบ่งจนกระทั่ง subarray มีขนาดเป็น 1

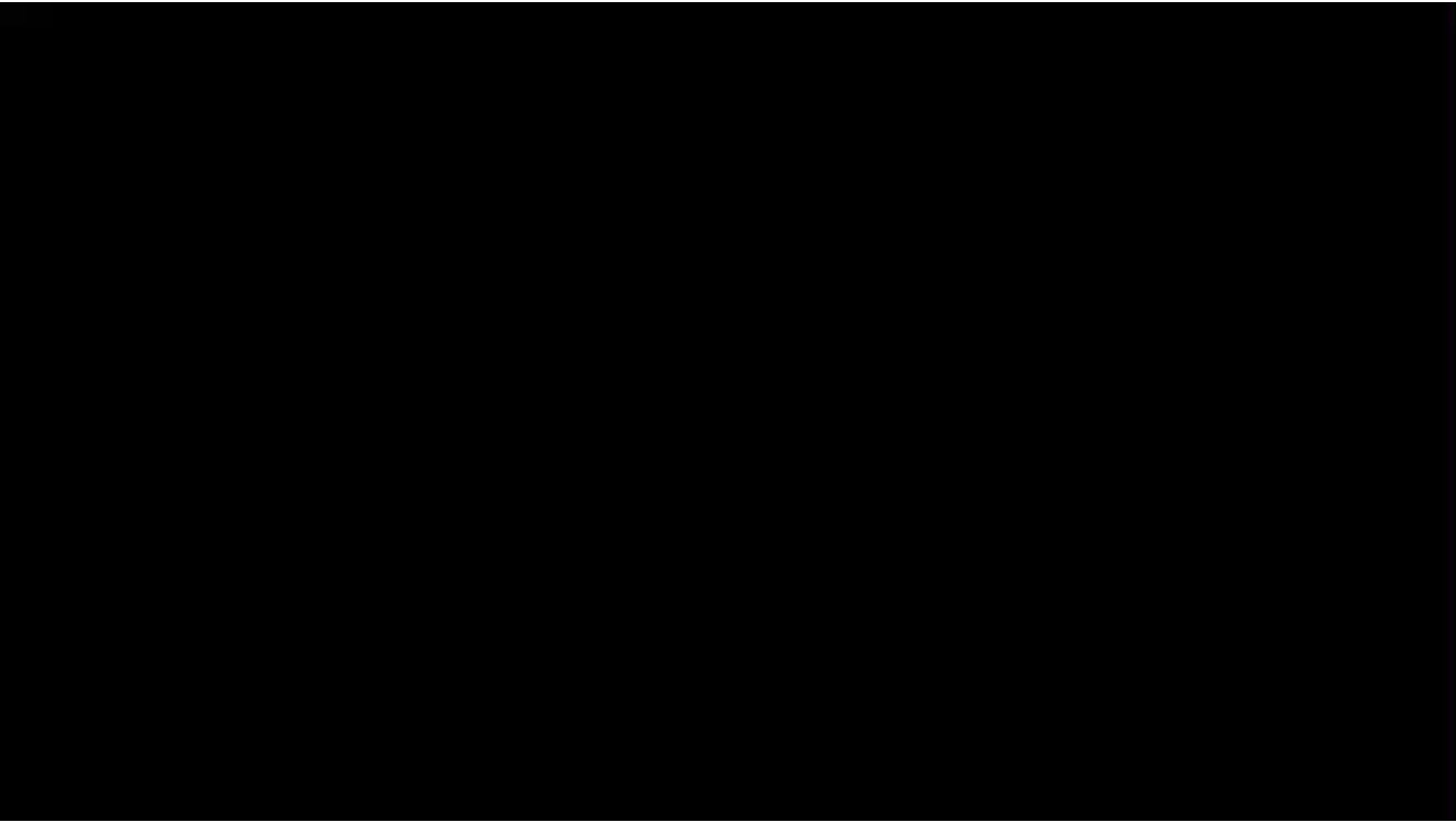
2. การเรียงลำดับ (Conquer):

- เมื่อ subarray มีขนาดเป็น 1 นั้นถือว่าเรียงลำดับแล้ว
- สำหรับ subarrays ที่มีขนาดมากกว่า 1 เรียงลำดับ subarrays แต่ละตัวแยกจากกัน โดยการเรียกใช้งาน Merge Sort ซ้ำ

3. การผสาน (Merge):

- นำ subarrays ที่เรียงลำดับแล้วมาผสานเข้าด้วยกันเพื่อให้ได้ array เรียงลำดับ
- ในขณะที่ทำการผสาน เปรียบเทียบ element ที่อยู่ใน subarrays และนำ element ที่น้อยที่สุดมาวางต่อกัน
- ทำไปเรื่อย ๆ จนกว่าจะผสาน subarrays ทั้งหมด

Merge Sort



Workshop : Merge Sort

ไฟล์ : 4_merge_sort

ปัญหา : ให้นักเรียนทดลองใช้โปรแกรม merge sort และตรวจสอบผลลัพธ์ที่ได้

ข้อมูลรับเข้า :

```
10
99
62
78
-54
43
36
81
5
25
15
```

ผลลัพธ์ : ?

Quick Sort

อัลกอริทึมแบบ "Divide and Conquer" ที่ใช้การเลือกจุดที่แบ่ง (pivot) เพื่อแบ่งข้อมูลออกเป็นสองส่วน ได้แก่ ทางซ้ายเป็นข้อมูลที่น้อยกว่า และทางขวาเป็นข้อมูลที่มากกว่า

Quick Sort

arr

Workshop : Quick Sort

ไฟล์ : 5_quick_sort

ปัญหา : ให้นักเรียนทดลองใช้โปรแกรม quick sort และตรวจสอบผลลัพธ์ที่ได้

ข้อมูลรับเข้า :

```
9
49
7
28
-32
63
0
11
2
88
```

ผลลัพธ์ : ?

Heap Sort

- อัลกอริทึมการเรียงลำดับที่ใช้โครงสร้างข้อมูล Heap เพื่อทำการเรียงลำดับข้อมูล
- Heap เป็นโครงสร้างข้อมูลที่มีลักษณะเป็น binary tree ที่สอดคล้องกับเงื่อนไข "Heap Property" ที่สามารถเรียงลำดับได้ในลักษณะที่ต้องการ (เช่น Max Heap หรือ Min Heap)

Heap Sort

<u>ขั้นตอนการทำงานของ Heap Sort:</u>

1. **สร้าง Max Heap:**
   - โดยเริ่มจากการสร้าง complete binary tree จากนั้นปรับ Heap Property โดยให้ node ที่มากที่สุดอยู่ที่ root
2. **สลับและ Heapify:**
   - สลับค่าของ root กับข้อมูลสุดท้ายของ array เพื่อเอาค่าที่มากที่สุดไปไว้ที่ตำแหน่งสุดท้าย
   - ลดขนาดของ Heap (ไม่พิจารณาตำแหน่งสุดท้ายที่เป็น Max Heap) และทำ Heapify และปรับ Heap Property
3. **ทำซ้ำขั้นตอน 2 จนกว่า Heap จะมีขนาดเป็น 1:**
   - ทำซ้ำขั้นตอน 2 โดยลดขนาดของ Heap ทุกรอบ จนกว่า Heap จะมีขนาดเป็น 1 (คือทุก element จะถูกเรียงลำดับ)

Heap Sort

Workshop : Heap Sort

ไฟล์ : 6_heap_sort

ปัญหา : ให้นักเรียนทดลองใช้โปรแกรม heap sort และตรวจสอบผลลัพธ์ที่ได้

ข้อมูลรับเข้า :

```
9
49
7
28
-32
63
0
11
2
88
```

ผลลัพธ์ : ?

Radix Sort

อัลกอริทึมที่เรียงลำดับข้อมูลตามตัวเลขที่เป็น digit

1. หาค่าที่มากที่สุด (max):

- ในการทำ Radix Sort ต้องหาค่าที่มากที่สุดใน array เพื่อทราบจำนวน digits ที่จะใช้ในการเรียงลำดับ

2. ทำ counting sort สำหรับทุกรอบของ digits:

- เริ่มจาก digit ต่ำสุด (units place) ไปสูงสุด (most significant digit) ของตัวเลข
- ในแต่ละรอบ, ทำการ counting sort โดยให้ digit นั้น ๆ เป็นตัวกำหนดในการเรียงลำดับ
- ทำให้ array ถูกเรียงลำดับตาม digit ที่กำหนด

3. ทำซ้ำขั้นตอน 2 จนกว่าทุกรอบของ digits จะถูกทำ:

- ทำซ้ำขั้นตอน 2 โดยให้ digit เพิ่มขึ้นตามลำดับ (units place, tens place, hundreds place, และต่อไป) จนกว่าจะถึง most significant digit

Radix Sort

Workshop : Radix Sort

ไฟล์ : 7_radix_sort

ปัญหา : ให้นักเรียนทดลองใช้โปรแกรม radix sort และตรวจสอบผลลัพธ์ที่ได้

ข้อมูลรับเข้า :

```
8
7
13
49
27
94
21
68
52
```

ผลลัพธ์ : ?

Bucket Sort

อัลกอริทึมที่แบ่งข้อมูลออกเป็นกลุ่มหลาย ๆ กลุ่ม (bucket) แล้วเรียงลำดับแต่ละกลุ่ม จากนั้นรวมผลลัพธ์กลับเข้าด้วยกัน

Bucket Sort

Workshop : Bucket Sort

ไฟล์ : 8_bucket_sort

ปัญหา : ให้นักเรียนทดลองใช้โปรแกรม bucket sort และตรวจสอบผลลัพธ์ที่ได้

ข้อมูลรับเข้า :

```
9
7
13
49
27
94
21
68
52
99
```

ผลลัพธ์ : ?

การเปรียบเทียบประสิทธิภาพของอัลกอริทึมการเรียงลำดับ

Bubble Sort:

- ความเร็ว: O(n^2)
- การใช้พื้นที่ในหน่วยความจำ: ใช้พื้นที่คงที่

Selection Sort:

- ความเร็ว: O(n^2)
- การใช้พื้นที่ในหน่วยความจำ: ใช้พื้นที่คงที่

Insertion Sort:

- ความเร็ว: O(n^2)
- การใช้พื้นที่ในหน่วยความจำ: ใช้พื้นที่คงที่

Merge Sort:

- ความเร็ว: O(n log n)
- การใช้พื้นที่ในหน่วยความจำ: ต้องใช้พื้นที่เพิ่มเติม

Heap Sort:

- ความเร็ว: O(n log n)
- การใช้พื้นที่ในหน่วยความจำ: ใช้พื้นที่คงที่

Radix Sort:

- ความเร็ว: O(nk) โดย k คือ จำนวน digits ในค่าที่มากที่สุด
- การใช้พื้นที่ในหน่วยความจำ: ใช้พื้นที่เพิ่มเติม

Quick Sort:

- ความเร็ว: O(n^2) ในกรณีสุดแย่ (ถ้าใช้ Pivot ที่ไม่ดี), แต่มีโอกาส O(n log n) ในกรณีโดยเฉลี่ย
- การใช้พื้นที่ในหน่วยความจำ: ใช้พื้นที่คงที่, แต่อาจใช้พื้นที่**เพิ่มเติมในกรณีแบบ recursive**

Bucket Sort:

- ความเร็ว: ขึ้นอยู่กับวิธีการเรียงลำดับที่ใช้ในแต่ละ bucket (โดยทั่วไปจะใช้ O(n^2) หรือ O(n log n) กับขนาดของ bucket)
- การใช้พื้นที่ในหน่วยความจำ: ใช้พื้นที่เพิ่มเติม

Merge Sort = Heap Sort < Quick Sort = Bucket Sort < Radix Sort < Insertion Sort = Selection Sort = Bubble Sort

Workshop :

ปัญหา : Sorting 0-9 and A-Z

ข้อมูลรับเข้า : บรรทัดแรก คือ จำนวนตัวอักษรที่จะจัดเรียง 2<n<99
บรรทัดสอง คือ ตัวอักษร 0-9 และ A-Z มีจำนวนอักขระเท่ากับ n จำนวน โดยสามารถป้อนตัวอักษรซ้ำกันได้

ผลลัพธ์ : มี 2 บรรทัด
บรรทัดแรก เรียงตัวอักษรจาก 0-9
บรรทัดสอง เรียงตัวอักษรจาก A-Z
*** โดยแต่ละตัวจะเว้น 1 ช่องว่าง ***

ตัวอย่าง :

ข้อมูลเข้า	ผลลัพธ์
15 C4O58M6P97UT5ER	4 5 5 6 7 8 9 C E M O P R T U

Question ?

Thank You