Sorting Comparison Report

1st Chatchanan Boonpa
65070501014
Computer Engineering Department
KMUTT
Bangkok, Thailand
chatchanan.boon@mail.kmutt.ac.th

2nd Natchanon Phattamanuruk
65070501018
KMUTT
Bangkok, Thailand
natchanon.phat@mail.kmutt.ac.th

3rd Nobphadon Hankittikanjana
65070501032
Computer Engineering Department
KMUTT
Bangkok, Thailand
nobphadon.hank@kmutt.ac.th

4th Nawin Tosilanon
65070501033
Computer Engineering Department
KMUTT
Bangkok, Thailand
nawin.tosi@mail.kmutt.ac.th

5th Nutchanon Boonyato
65070501075
Computer Engineering Department
KMUTT
Bangkok, Thailand
nutchanon.boon@mail.kmutt.ac.th

I. บทนำ

A. ที่มาและความสำคัญ

ศึกษาและเปรียบเทียบประสิทธิภาพของวิธีการจัดเรียงข้อมูล [1] (Sorting Algorithms) ทั้ง 5 แบบได้แก่ Bubble sort, Insertion sort, Merge sort, และ Quick sort โดยจะทดลองเพื่อหาเวลาในการจัดเรียงข้อมูลเฉลี่ย, เวลาในการจัดเรียงข้อมูลที่ดีที่สุดและแย่ที่สุด, พื้นที่ความจำสูงสุดขณะจัดเรียงข้อมูล, และความคงที่ของการจัดเรียงข้อมูล

B. วัตถุประสงค์

เพื่อศึกษาและเปรียบเทียบประสิทธิภาพของวิธีการจัดเรียงข้อมูล (Sorting Algorithms) วิธีการต่าง ๆ

C. ขอบเขต

ทดลองหาเวลาในการจัดเรียงข้อมูลและพื้นที่ความจำสูงสุดที่ใช้ของวิธีการจัดเรียงข้อมูล (Sorting Algorithms) ทั้ง 5 แบบได้แก่ Bubble sort, Insertion sort, Merge sort, และ Quick sort ด้วยชุดข้อมูลขนาดแตกต่างกันบนแล็ปท็อปที่แตกต่างกัน 2 เครื่อง เพื่อนำผลลัพธ์มาเปรียบเทียบและสรุปผลการทดลอง

D. ประโยชน์ที่คาดว่าจะได้รับ

ทราบถึงประสิทธิภาพการทำงานของวิธีการจัดเรียงข้อมูล (Sorting Algorithms) แต่ละแบบ อ้างอิงจากชุดข้อมูลที่ใช้ในการทดลองและผลลัพธ์เวลาการจัดเรียงข้อมูล, พื้นที่ความจำสูงสุดที่ใช้, และความคงที่ของการจัดเรียงข้อมูล

II. เอกสารและงานที่เกี่ยวข้อง

วิธีการจัดเรียงข้อมูล (Sorting Algorithms) ที่นำมาศึกษามีทั้งหมด 5 แบบดังนี้

A. Selection sort

เป็นอัลกอริทึม [2] ในการเรียงลำดับที่ง่ายและมีประสิทธิภาพ ซึ่งทำงานโดยการเลือกสมาชิกที่เล็กที่สุด (หรือใหญ่ที่สุด) จากส่วนที่ยังไม่เรียงลำดับของlist แล้วย้ายไปยังส่วนที่เรียงลำดับแล้วของlist และจะทำการเรียงจนเรียงครบทุกตัว

```
void selectionSort(int arr[],int n){
  int i,j,min;
  for(i=0;i<n-1;i++){
      min = i;
      for(j=i+1;j<n;j++){
          if(arr[j]<arr[min]){
              min = j;
          }
      }
      swap(&arr[i],&arr[min]);
  }
}
```

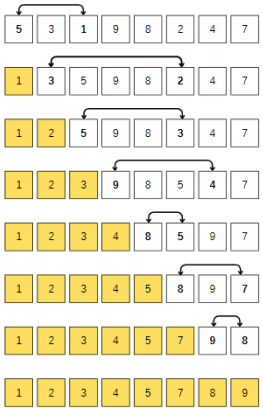

Fig. 1. ตัวอย่างการทำ Selection sort.

B. Bubble sort

เป็นอัลกอริทึม [3] ในการเรียงลำดับที่ง่ายที่สุด ซึ่งทำงานโดยการสลับสมาชิกที่อยู่ติดกันหากข้อมูลนั้นๆอยู่ในตำแหน่งที่ไม่ถูกต้อง อัลกอริทึมนี้ไม่เหมาะสำหรับชุดข้อมูลขนาดใหญ่เนื่องจากเวลาที่ใช้ในกรณีเฉลี่ยและกรณีที่แย่ที่สุดมีtime complexityที่สูง

```
void bubbleSort(int arr[],int n){
    for(int i =1; i<n; i++){
        for(int j=0; j<n-i; j++){
            if(arr[j]>arr[j+1]){
                swap(&arr[j],&arr[j+1]);
            }
        }
    }
}
```

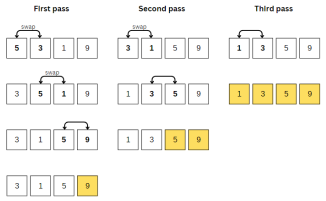

Fig. 2. ตัวอย่างการทำ Bubble sort.

C. Insertion sort

เป็นอัลกอริทึม [4] ในการเรียงลำดับอย่างง่าย โดยทำงานคล้ายกับวิธีการเรียงการ์ดเล่นในมือ โดยอาร์เรย์ (Array) จะถูกแบ่งออกเป็นส่วนที่เรียงลำดับและส่วนที่ยังไม่เรียงลำดับ ค่าจากส่วนที่ยังไม่เรียงลำดับจะถูกเลือกและวางไว้ในตำแหน่งที่ถูกต้องในส่วนที่เรียงลำดับ

```
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int temp = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > temp) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = temp;
    }
}
```

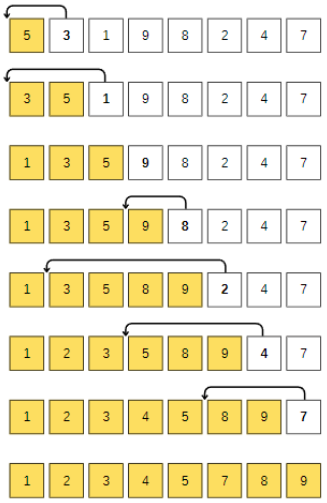

Fig. 3. ตัวอย่างการทำ Insertion sort.

D. Quick sort

เป็นอัลกอริทึม [5] ในการเรียงลำดับที่ใช้กลยุทธ์แบบแบ่งแยกและครอบครอง (Divide and Conquer algorithm) โดยการเลือกสมาชิกหนึ่งตัวเป็น Pivot และ Partitions array ที่กำหนดออกจากกันโดยวางตัว Pivot ในตำแหน่งที่ถูกต้องใน Array ที่เรียงลำดับอย่างถูกต้อง

```
int HoarePartition(int arr[],int l,int r)
{
    int pivot = arr[l];
    int i = l;
    int j = r+1;

    do
    {
        do
        {
            i++;
        }
        while(arr[i]<pivot);
        do
        {
            j--;
        }
        while(arr[j]>pivot);
        swap(&arr[i],&arr[j]);
    }
    while(i<j);
    if(i!=j)
    {
        swap(&arr[i],&arr[j]);
    }
    swap(&arr[l],&arr[j]);
    return j;
}

void quickSort(int arr[],int l,int r)
{
    if (l<r)
    {
        int s = HoarePartition(arr,l,r);
        quickSort(arr,l,s-1);
        quickSort(arr,s+1,r);
    }
}
```

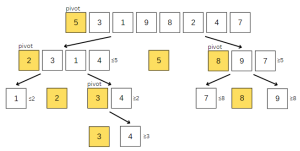

Fig. 4. ตัวอย่างการทำ quick sort.

E. Merge sort

เป็นอัลกอริทึม [6] การเรียงลำดับที่ทำงานโดยการแบ่งอาร์เรย์เป็น อาร์เรย์ย่อย ที่ มี ขนาด เล็ก ลง จาก นั้น เรียง ลำดับ ค่า ใน แต่ละ อาร์เรย์ย่อย แล้วจึงรวมอาร์เรย์ย่อยที่เรียงลำดับกลับมาเพื่อสร้างอาร์เรย์ที่เรียงลำดับใหม่ในขั้นสุดท้าย

```
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}
```

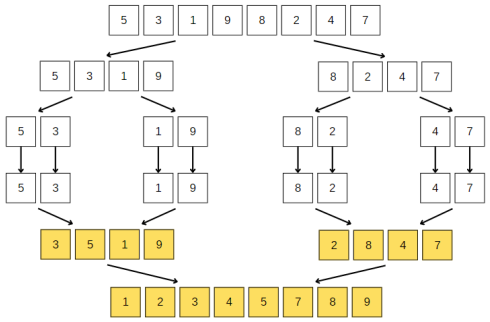

Fig. 5. ตัวอย่างการทำ merge sort.

III. วิธีดำเนินการ

A. เปรียบเทียบประสิทธิภาพการทำงานของวิธีการจัดเรียงข้อมูล

วิธีการจัดเรียงข้อมูล (Sorting Algorithms) แต่ละแบบนั้นมีหลักการทำงานที่แตกต่างกัน ซึ่งจะมีประสิทธิภาพแตกต่างกันเช่นกัน การทดลองจะแบ่งการทดลองและเปรียบเทียบประสิทธิภาพเป็น 4 หมวดหมู่ โดยทุก ๆ การทดลองในแต่ละหัวข้อจะทำการทดลองซ้ำ 3 ครั้งเพื่อป้องกันความผิดพลาดที่อาจจะเกิดขึ้นได้

1) **Average case in time**: การทดลองวัดเวลาที่ใช้โดยเฉลี่ยจากข้อมูลแบบสุ่มซึ่งจะนำเวลามาหาค่าเฉลี่ยและเปรียบเทียบกัน

2) **Ascending order case and Descending order case in time**: การทดลองวัดเวลาที่ใช้จากชุดข้อมูลที่มีการเรียงลำดับจากน้อยไปมาก (Ascending Order) และข้อมูลที่มีการเรียงลำดับจากมากไปน้อย (Descending Order)

3) **Space complexity**: การทดลองวัดพื้นที่ความจำโดยจะวัดการใช้พื้นที่ความจำและจะบันทึกการใช้พื้นที่ความจำสูงสุดขณะที่ทำการจัดเรียงข้อมูลลงในไฟล์ข้อความ (.txt file) ด้วยเครื่องมือ Valgrind [7]

4) **Stability**: การทดลองความคงที่ [8] (Stability) จะทดลองโดยการกำหนดลำดับ (Order) ให้กับข้อมูลที่ซ้ำกันเป็นลำดับ $1, 2, 3, \ldots$ และตรวจสอบลำดับหลังจากที่นำมาจัดเรียงแล้วว่ามีลำดับเหมือนเดิมหรือไม่ โดยการเรียงลำดับนั้นจะคงที่หากมีข้อมูลที่ซ้ำกันมีการเรียงลำดับเหมือนเดิม โดยจะนับจำนวนของชุดข้อมูลที่เรียงลำดับไม่คงที่จากชุดข้อมูลทั้งหมดเพื่อนำมาสรุปว่าการจัดเรียงข้อมูลแต่ละแบบนั้นคงที่หรือไม่

B. การกำหนดเงื่อนไขการทดลอง

1) **การกำหนดชุดข้อมูล**: ทดลองโดยใช้ชุดข้อมูลสุ่มที่มีจำนวนของข้อมูล N โดยชุดข้อมูลจะมีขนาดแตกต่างกันทั้งหมด 10 แบบ ซึ่งคือ N = 1000, 5000, 10000, 15000, 20000, 30000, 40000, 50000, 75000, และ 100000 โดยจะทำการสุ่มชุดข้อมูลของแต่ละ N เป็นจำนวน 100 ชุดข้อมูล และแต่ละข้อมูลจะมีค่าตั้งแต่ -100000 ถึง 100000 นอกจากนี้ยังมีการทดลองชุดข้อมูลที่มีการเรียงลำดับจากน้อยไปมาก (Ascending Order) และเรียงลำดับจากมากไปน้อย (Descending Order) สำหรับแต่ละขนาด N และสำหรับการทดลองความคงที่ของข้อมูลจะใช้ทุกชุดข้อมูลขนาด N = 4 โดยที่ข้อมูลมีค่าตั้งแต่ 1 ถึง 4 ได้แก่ $1111, 1112, 1113, \ldots, 4444$ ซึ่งชุดข้อมูลทั้งหมดที่ถูกสร้างขึ้นจากโปรแกรมสร้างชุดข้อมูลจะถูกเก็บไว้ในรูปของไฟล์ข้อความ (.txt File)

2) **การ กำหนด ระบบ สำหรับ การ ทดลอง**: ทำการ ทดลอง ด้วย แล็ปท็อป 2 เครื่องโดยแต่ละเครื่องมีสเปกดังนี้

- Laptop A : AMD Ryzen 5 5600H 3.30GHz, RAM 16 GB
- Laptop B : Intel Core i5-7300HQ 2.50GHz, RAM 16 GB

ซึ่งจะทำการทดลองบน Virtual Machine: VMware โดยจำลองระบบปฏิบัติการ Ubuntu 20.04.3 ด้วยการตั้งค่า CPU Processors = 4 core และ RAM = 4 GB

C. โปรแกรมที่ใช้ในการทดลอง

การทำงานแต่ละครั้งโปรแกรมจะทำการทดลองชุดข้อมูลขนาด N และเลือกวิธีการเรียงข้อมูล 1 แบบ โดยจะแบ่งขั้นตอนการทำงานดังนี้

1) **การอ่านข้อมูล**: การทดลองจะใช้ชุดข้อมูลที่ถูกเก็บไว้ในไฟล์ข้อความ (.txt File) และโปรแกรมจะอ่านข้อมูลลงในอาร์เรย์เพื่อนำไปจัดเรียงข้อมูลต่อไป

2) **การ จับ เวลา และ จัด เรียง ข้อมูล**: โปรแกรม จะ จับ เวลา ณ ปัจจุบัน ไว้ในตัวแปร $time\_start$ จากนั้นจะทำการจัดเรียงข้อมูล และจะจับเวลาอีกครั้งเมื่อทำการจัดเรียงข้อมูลเสร็จแล้วเก็บไว้ในตัวแปร $time\_end$ โดยจะนำค่า $time\_end - time\_start$ มาเก็บไว้ในตัวแปร $totalTime$

ซึ่งโปรแกรมจะทำซ้ำไปในทุกชุดข้อมูลสุ่ม M ชุดที่ทำการทดลอง และจะได้ค่า $totalTime/M$ เป็นเวลาเฉลี่ยของชุดข้อมูลขนาด N ของวิธีการเรียงข้อมูลที่เลือก

3) **ชุดคำสั่งสำหรับการทดลอง**: การทดลองจะดำเนินการด้วย Shell Script ซึ่งจะทำการเลือกวิธีการจัดเรียงข้อมูลที่ต้องการในแต่ละขนาดชุดข้อมูล N โดยจะทำการล้างข้อมูลแคชทุกครั้งที่ทำการทดลอง จากนั้นจะบันทึกเวลาที่ใช้ในการจัดเรียงข้อมูลและพื้นที่ความจำสูงสุดที่ใช้ระหว่างการจัดเรียงข้อมูล

ในส่วนของการทดลองความคงที่จะใช้โปรแกรมในการเช็คลำดับ (Order) ของข้อมูลหลังการจัดเรียงข้อมูลจากชุดข้อมูล และจะนับจำนวนของชุดข้อมูลที่เรียงลำดับไม่คงที่จากชุดข้อมูลทั้งหมดของแต่ละวิธีการจัดเรียงข้อมูล

D. เปรียบเทียบผลลัพธ์การทดลองของวิธีการจัดเรียงข้อมูลแบบต่าง ๆ

จากการทดลองประสิทธิภาพการทำงานของวิธีการจัดเรียงข้อมูลแต่ละแบบจากทั้ง 4 หมวดหมู่ นำผลที่ได้มาเปรียบเทียบเพื่อหาข้อดีและข้อเสียที่แตกต่างกันของแต่ละวิธีการ

IV. ผลการทดลอง

จากการทดลองวิธีการจัดเรียงข้อมูลทั้ง 5 วิธีด้วยชุดข้อมูลต่าง ๆ ได้ผลลัพธ์ดังนี้

A. Average case in time

ผลลัพธ์การทดลองหาเวลาที่ใช้ในการจัดเรียงข้อมูลจากชุดข้อมูลสุ่มด้วยแล็ปท็อปทั้ง 2 เครื่อง

- **Laptop A**

Type of Sort	N			
	1000	5000	10000	15000
Bubble	1602.47	36017.51	150594.31	343840.06
Selection	718.32	15264.65	60569.64	136615.91
Insertion	439.15	9872.67	40391.41	90690.13
Merge	104.83	586.52	1188.14	1799.84
Quick	68.34	382.17	752.29	1185.80

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A (microsecond)

Type of Sort	N		
	20000	30000	40000
Bubble	579931.14	1568857.58	3206193.00
Selection	243891.64	243891.65	968066.97
Insertion	160013.58	361848.23	645034.20
Merge	2478.52	3758.05	5074.16
Quick	1607.35	2479.06	3320.53

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A (microsecond)

Type of Sort	N		
	50000	75000	100000
Bubble	5208819.12	12580726.04	23458008.98
Selection	1537450.66	3455491.09	6033037.89
Insertion	1029781.11	2277165.12	4104637.69
Merge	6409.31	9919.32	13667.00
Quick	4287.84	6507.80	8892.87

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A (microsecond)

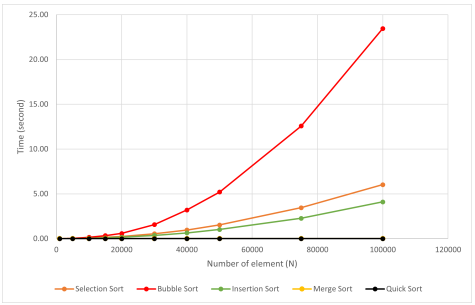

Fig. 6. กราฟแสดงผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A

- **Laptop B**

Type of Sort	N			
	1000	5000	10000	15000
Bubble	2928.36	81014.59	359652.70	828466.28
Selection	1422.06	33846.79	133864.41	300073.22
Insertion	839.70	20512.68	80235.46	180654.97
Merge	144.90	856.18	1829.51	2836.35
Quick	112.08	654.61	1391.26	2192.69

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

Type of Sort	N		
	20000	30000	40000
Bubble	1516579.05	3500436.23	6357350.68
Selection	526971.75	1197126.69	2126447.51
Insertion	319992.34	526971.75	1250297.45
Merge	3723.33	5850.32	7953.87
Quick	2938.56	4585.84	6347.63

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

Type of Sort	N		
	50000	75000	100000
Bubble	9741454.72	22489463.68	40220396.66
Selection	3314472.95	7433016.50	13320089.96
Insertion	1989372.89	4486320.64	7842612.93
Merge	10057.07	15514.00	21087.62
Quick	7980.05	12411.44	17164.38

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

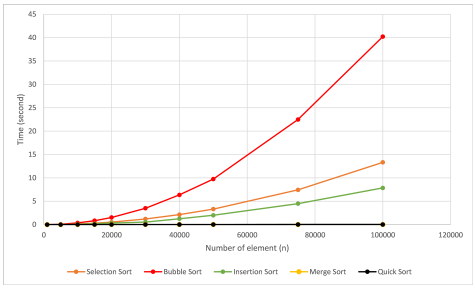

Fig. 7. กราฟแสดงผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B

B. Ascending order case and Descending order case in time

ผลลัพธ์จากการทดลองหาเวลาที่ใช้ในการจัดเรียงข้อมูลจากชุดข้อมูลที่เรียงลำดับจากน้อยไปมาก (Ascending Order) และชุดข้อมูลจากมากไปน้อย (Descending Order) ด้วยแล็ปท็อปทั้ง 2 เครื่อง

- **Ascending order case : Laptop A**

Type of Sort	N			
	1000	5000	10000	15000
Bubble	760.20	18118.87	65875.47	150960.50
Selection	558.65	15558.40	53151.34	140523.42
Insertion	2.66	12.83	23.85	41.57
Merge	67.67	363.45	741.48	1253.87
Quick	256.89	6185.63	21471.63	46481.91

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A (microsecond)

Type of Sort	N		
	20000	30000	40000
Bubble	267560.72	623450.67	1090713.80
Selection	235209.81	523666.00	940785.14
Insertion	57.20	82.87	105.82
Merge	1621.25	2691.55	3118.13
Quick	86874.26	199540.45	341266.87

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A (microsecond)

Type of Sort	N		
	50000	75000	100000
Bubble	1701585.63	3854854.40	6923047.72
Selection	1466096.33	3253644.95	6017677.75
Insertion	153.06	177.98	225.75
Merge	4392.05	5816.48	7678.43
Quick	524048.15	1204912.49	2195993.20

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A (microsecond)

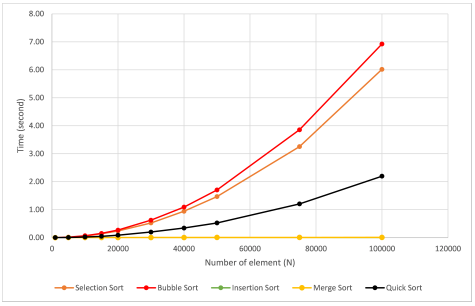

Fig. 8. กราฟแสดงผลลัพธ์เวลาการจัดเรียงข้อมูลจากชุดข้อมูลที่เรียงจากน้อยไปมากของแล็ปท็อป A

- **Ascending order case : Laptop B**

Type of Sort	N			
	1000	5000	10000	15000
Bubble	1227.23	33485.46	127033.25	291660.91
Selection	1343.95	34746.80	136804.21	311953.28
Insertion	3.91	19.12	44.81	60.51
Merge	85.70	542.93	1081.09	1690.59
Quick	1319.39	34341.14	129850.67	294802.27

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

Type of Sort	N		
	20000	30000	40000
Bubble	505569.50	1160314.78	2017697.46
Selection	524814.77	1178291.81	2105411.51
Insertion	82.37	116.30	155.75
Merge	2289.58	3559.74	4730.42
Quick	527759.79	1152527.83	2118595.93

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

Type of Sort	N		
	50000	75000	100000
Bubble	3124416.80	6938942.81	12598964.75
Selection	3319395.64	7430248.32	12993097.92
Insertion	201.45	312.43	406.64
Merge	6233.74	9897.35	13234.14
Quick	3254752.17	7187371.89	12772642.67

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

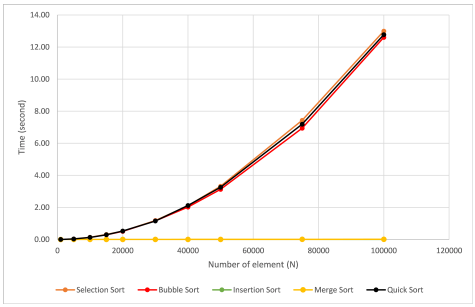

Fig. 9. กราฟแสดงผลลัพธ์เวลาการจัดเรียงข้อมูลจากชุดข้อมูลที่เรียงจากน้อยไปมากของแล็ปท็อป B

- **Descending order case : Laptop A**

Type of Sort	N			
	1000	5000	10000	15000
Bubble	2055.76	51986.39	208401.18	461510.29
Selection	791.64	18352.65	64152.16	147084.85
Insertion	830.93	24223.04	76353.41	186579.21
Merge	64.50	352.93	701.92	1224.27
Quick	326.71	10984.79	36980.14	83928.73

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A (microsecond)

Type of Sort	N		
	20000	30000	40000
Bubble	837479.42	1950734.01	3378757.18
Selection	256264.32	599911.64	1071614.97
Insertion	336952.36	730563.13	1368697.30
Merge	1363.24	2163.67	2972.10
Quick	164520.37	343463.55	601911.25

Descending A

Type of Sort	N		
	50000	75000	100000
Bubble	5489123.39	12351653.09	22111150.66
Selection	1646552.30	3698890.98	6601119.29
Insertion	2077832.97	4744223.76	8387402.88
Merge	4771.25	6639.46	8198.07
Quick	954652.31	2200297.17	3924263.96

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป A (microsecond)

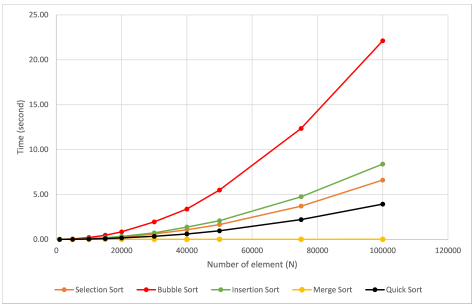

Fig. 10. กราฟแสดงผลลัพธ์เวลาการจัดเรียงข้อมูลจากชุดข้อมูลที่เรียงจากมากไปน้อยของแล็ปท็อป A

- **Descending order case : Laptop B**

Type of Sort	N			
	1000	5000	10000	15000
Bubble	3922.25	99601.37	404029.99	903127.47
Selection	1261.43	32454.77	132097.25	285926.56
Insertion	1614.96	41181.27	165899.89	365378.57
Merge	85.95	520.19	1074.87	1640.55
Quick	1342.47	32135.36	133751.25	293968.02

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

Type of Sort	N		
	20000	30000	40000
Bubble	1620964.12	3577122.08	6285883.91
Selection	514159.70	1150045.72	2006054.00
Insertion	633729.17	1428242.27	2535407.49
Merge	2347.66	3646.74	4895.24
Quick	531162.40	1204943.84	2068472.49

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

Type of Sort	N		
	50000	75000	100000
Bubble	9741021.12	22011478.16	39006424.99
Selection	3161026.60	7087901.70	12719346.73
Insertion	3991014.03	8873875.69	15792727.93
Merge	5950.10	9457.62	13153.38
Quick	3272428.48	7332230.29	12800830.57

ตารางผลลัพธ์เวลาการจัดเรียงข้อมูลของแล็ปท็อป B (microsecond)

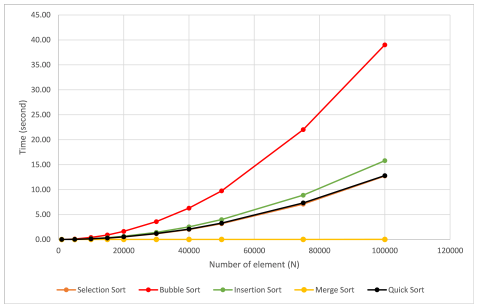

Fig. 11. กราฟแสดงผลลัพธ์เวลาการจัดเรียงข้อมูลจากชุดข้อมูลที่เรียงจากมากไปน้อยของแล็ปท็อป B

C. Space complexity

ผลลัพธ์จากการทดลองหาพื้นที่ความจำสูงสุดที่ใช้ในขณะที่จัดเรียงข้อมูล โดยที่แล็ปท็อป A และแล็ปท็อป B ให้ผลลัพธ์ที่ตรงกัน

Type of Sort	N			
	1000	5000	10000	15000
Bubble	8368	24368	44088	64088
Selection	8368	24368	44088	64088
Insertion	8368	24368	44368	64368
Merge	8736	40736	80736	120736
Quick	8368	24368	44368	64368

ตารางการใช้พื้นที่ความจำสูงสุดของแต่ละวิธีการจัดเรียงข้อมูล

Type of Sort	N		
	20000	30000	40000
Bubble	84088	124088	164088
Selection	84088	124088	164088
Insertion	84368	124368	164368
Merge	160736	240736	320736
Quick	84368	124368	164368

ตารางการใช้พื้นที่ความจำสูงสุดของแต่ละวิธีการจัดเรียงข้อมูล

Type of Sort	N		
	50000	75000	100000
Bubble	204088	304088	404088
Selection	204088	304088	404088
Insertion	204368	304368	404368
Merge	400736	600736	800736
Quick	204368	304368	404368

ตารางการใช้พื้นที่ความจำสูงสุดของแต่ละวิธีการจัดเรียงข้อมูล

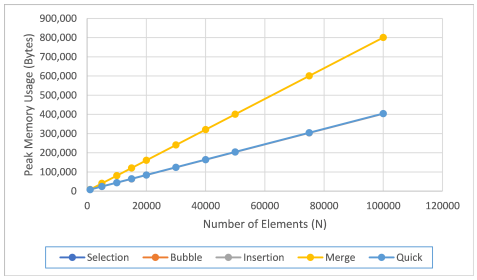

Fig. 12. กราฟแสดงผลการใช้พื้นที่ความจำสูงสุดของแต่ละวิธีการจัดเรียงข้อมูล

D. Stability

ตัวอย่างของข้อมูลในชุดข้อมูลที่ใช้การทดลอง

original data		bubble sort		insertion sort		merge sort	
data	order	data	order	data	order	data	order
1	1	1	1	1	1	1	1
2	1	1	2	1	2	1	2
2	2	2	1	2	1	2	1
1	2	2	2	2	2	2	2

[a]ตัวอย่างข้อมูลชุดข้อมูลที่ Stable

original data		selection sort		quick sort	
data	order	data	order	data	order
1	1	1	1	1	2
2	1	1	2	1	1
2	2	2	2	2	1
1	2	2	1	2	2

[a]ตัวอย่างข้อมูลชุดข้อมูลที่ไม่ Stable

```
Number of non stable case of Bubble sort: 0
Number of non stable case of Selection sort: 56
Number of non stable case of Insertion sort: 0
Number of non stable case of Merge sort: 0
Number of non stable case of Quick sort: 166
```

Fig. 13. กราฟแสดงผลลัพธ์จำนวนชุดข้อมูลที่ไม่คงที่

V. สรุปและอภิปรายผล

A. Average case in time

จากการทดลองวัดระยะเวลาการจัดเรียงด้วยวิธีการจัดเรียงได้แก่ Quick sort, Merge sort, Insertion sort, Selection sort และ Bubble sort ด้วยการใช้ชุดข้อมูลที่ได้จากการสุ่มค่าและชุดข้อมูลที่มีจำนวนข้อมูลต่างกัน10รูปแบบ โดยที่แต่ละรูปแบบมีชุดข้อมูลที่มีค่าต่างกัน 100 ชุด และนำข้อมูลทั้งหมดที่ได้มาทำการทดลองและหาค่าเฉลี่ยได้ข้อสังเกตดังนี้

การจัดเรียงแบบ Quick sort และ Merge sort เป็นอัลกอริทึมที่ใช้ระยะเวลาในการจัดเรียงน้อยที่สุดในทุกขนาดของชุดข้อมูลซึ่งเป็นไปตามทฤษฎีที่ว่าในการจัดเรียงข้อมูลแบบ Quick และ Merge sort จะมี Time complexity ในกรณีของข้อมูลที่สุ่มเป็น $O(nlogn)$ เพราะการจัดเรียงทั้ง 2 แบบนี้มีใช้กลยุทธ์ในการจัดการกับปัญหาแบบ Divide and conquer ที่จะทำการแบ่งปัญหาเป็นปัญหาย่อยและจัดการแต่ละปัญหาย่อย สำหรับการจัดเรียงแบบ Insertion และ Selection sort ใช้เวลาในการจัดเรียงมากขึ้นมาตามลำดับ และการจัดเรียงแบบ Bubble sort เป็นอัลกอริทึมที่ใช้เวลาในการจัดเรียงมากที่สุด โดยที่อัลกอริทึมทั้ง 3 อัลกอริทึมมีการใช้เวลาในการจัดเรียงที่มากกว่าแบบ Quick และ Merge sort ตามดังทฤษฎีที่กล่าวว่า Time complexity ของทั้ง3อัลกอริทึมมีค่าเป็น $O(n^2)$ เพราะการจัดเรียงทั้ง 3 ใช้หลักการแก้ปัญหาแบบ Decrease and conquer (by constant) ที่จะลดปัญหาลงตามค่าคงที่และจัดการกับปัญหา

B. Ascending order case and Descending order case in time

จากการทดลองวัดระยะเวลาการจัดเรียงข้อมูลจากชุดข้อมูลที่เรียงจากน้อยไปมากและจากมากไปน้อยได้ข้อสังเกตดังนี้

- **ชุดข้อมูลที่เรียงจากน้อยไปมาก** : ในวิธีการจัดเรียงข้อมูลแบบ Bubble sort, Selection sort, และ Insertion sort จะใช้เวลาน้อยที่สุดเมื่อชุดข้อมูลเรียงลำดับจากน้อยไปมากเนื่องมาจากในระหว่างการจัดเรียงมีการสลับข้อมูล (Swap) ที่น้อยหรือในกรณีของการจัดเรียงแบบ Insertion sort นั้นจะมี Time complexity ลดลง จาก $O(n^2)$ เป็น $O(N)$ ซึ่งหากนำการจัดเรียงแบบ Bubble sort มาปรับแก้ไขก็จะได้ Time complexity เป็น $O(N)$ เช่นกัน สำหรับวิธีการจัดเรียงแบบ Merge sort จะได้ผลลัพธ์ที่เทียบเท่ากับชุดข้อมูลสุ่มเนื่องจากเป็นวิธีการจัดเรียงที่ไม่เกิดการ (Swap) และสาเหตุที่การจัดเรียงแบบ Quick sort ได้ผลลัพธ์ที่ใช้ระยะเวลานานกว่าชุดข้อมูลสุ่มเป็นผลมาจากการแบ่งส่วน (Partition) ขณะจัดเรียงข้อมูลจะสามารถแยกได้ครั้งละ 1 ข้อมูล ส่งผลให้มี Time complexity เป็น $O(n^2)$
- **ชุดข้อมูลที่เรียงจากมากไปน้อย** : สำหรับชุดข้อมูลที่เรียงจากมากไปน้อยจะให้ผลในทางตรงกันข้ามสำหรับวิธีการจัดเรียงแบบ Bubble sort, Selection sort, และ Insertion sort เนื่องจากเกิดการสลับ (Swap) มากกว่าการการจัดเรียงชุดข้อมูลสุ่ม โดยสำหรับวิธีการจัดเรียงแบบ Merge sort และ Quick sort ยังคงได้ผลลัพธ์เทียบเท่ากับชุดข้อมูลที่เรียงจากน้อยไปมากด้วยเหตุผลเดิม

C. Space complexity

จากการทดลองวัดการใช้พื้นที่ความจำสูงสุดขณะการจัดเรียงข้อมูลพบว่าทุกวิธีการจัดเรียงข้อมูลจะมี Space complexity เป็น $O(N)$ โดยมีค่าคงที่ (Constant) อยู่ที่ 4088 และ 4368 ไบต์ แต่ในส่วนของวิธีการจัดเรียงข้อมูลแบบ Merge sort ถึงแม้ว่าจะมี Space complexity เป็น $O(N)$ แต่มีการใช้พื้นที่ความจำสูงสุงเป็น $2n$ และมีค่าคงที่ (Constant) เป็น 736 ไบต์

D. Stability

จากการทดลองการหาความคงที่ (Stability) ของแต่ละวิธีการจัดเรียงข้อมูล โดยการตรวจสอบลำดับของข้อมูลที่เหมือนกันก่อนและหลังการจัดเรียงข้อมูล พบว่าหากมีการเปลี่ยนแปลงลำดับของข้อมูลที่ซ้ำกัน จะสรุปได้ว่าวิธีการจัดเรียงข้อมูลนั้นจะไม่มีความคงที่ (Unstable) ในการทดลองจะใช้ชุดข้อมูลที่เป็นตัวเลขตั้งแต่ 1 ถึง 4 เช่น 1111, 1221, 1113, 3211 และอื่นๆ โดยจะมีจำนวนชุดข้อมูลทั้งหมด 4^4 ชุด ซึ่งผลการทดลองแสดงให้เห็นว่าจากทั้งหมด 5 วิธีการจัดเรียงข้อมูล มี 2 วิธีการจัดเรียงข้อมูล ที่ไม่มีความคงที่ (Stability) คือวิธีการจัดเรียงข้อมูลแบบ Selection sort ที่มีจำนวนชุดข้อมูลที่ไม่มีความคงที่ (Unstable) เป็น 56 ชุดข้อมูล และวิธีการจัดเรียงข้อมูลแบบ Quick sort ที่มีจำนวนชุดข้อมูลที่ไม่มีความคงที่ (Unstable) 166 ชุดข้อมูล จากทั้งหมด 256 ชุดข้อมูล เหตุผลที่ Selection sort ไม่มีความคงที่ (Unstable) เนื่องจากว่าในการสลับ (Swap) ข้อมูลจะทำให้ลำดับของตัวเลขเปลี่ยนแปลงและหากตัวเลขตัวนั้นมีตัวซ้ำ Selection sort จะข้ามการสลับตัวเลขนั้นไป ส่วน Quick sort จะไม่สนใจลำดับของตัวเลขที่ซ้ำกัน จึงทำให้ไม่มีความคงที่ (Unstable)

E. Summary and Discussion

จากการทดลองเปรียบเทียบประสิทธิภาพของวิธีการจัดเรียงข้อมูล (Sorting Algorithms) ทั้ง 5 แบบ ผลลัพธ์เวลาในการจัดเรียงข้อมูลแสดงให้เห็นว่าวิธีการจัดเรียงข้อมูลแบบ Merge sort และ Quick sort ซึ่งมี Time complexity เป็น $O(nlogn)$ ใช้เวลาน้อยกว่าการจัดเรียงข้อมูลแบบ Bubble sort, Selection sort, และ Insertion sort ซึ่งมี Time complexity เป็น $O(n^2)$ และจากผลลัพธ์การใช้พื้นที่ความจำสูงสุดแสดงให้เห็นว่าทุกวิธีการจัดเรียงข้อมูลมี Space complexity เป็น $O(N)$ แต่วิธีการจัดเรียงข้อมูลแบบ Merge sort จะ

ใช้พื้นที่มากกว่าการจัดเรียงแบบอื่นถึง 1.98 เท่า ด้วยเหตุผลที่กล่าวมา การจัดเรียงข้อมูลแบบ Quick sort จึงเป็นวิธีการจัดเรียงข้อมูลที่เหมาะแก่การนำไปใช้งานกับชุดข้อมูลที่มีขนาดใหญ่มากที่สุดหากชุดข้อมูลนั้นเป็นชุดข้อมูลที่ไม่ได้ถูกเรียงลำดับก่อนแล้ว แต่หากต้องการให้การจัดเรียงข้อมูลนั้นคงที่ การจัดเรียงข้อมูลแบบ Merge sort จึงเป็นตัวเลือกที่เหมาะสมถึงแม้จะใช้พื้นที่ความจำสูงสุดที่มากกว่า หรือหากข้อมูลที่ต้องการจัดเรียงเป็นข้อมูลขนาดเล็กและต้องการวิธีการจัดเรียงข้อมูลที่ไม่ซับซ้อนก็สามารถพิจารณาการจัดเรียงข้อมูลแบบ Bubble sort, Selection sort, และ Insertion sort ได้เช่นกัน

นอกจากนี้จากการเปรียบเทียบผลลัพธ์การใช้เวลาการจัดเรียงข้อมูลของแล็ปท็อป A และ แล็ปท็อป B จะเห็นถึงข้อแตกต่างที่ชัดเจนในการทดลองจัดเรียงชุดข้อมูลจากน้อยไปมาก โดยถึงแม้ผลลัพธ์ของการจัดเรียงข้อมูลแบบ Quick sort ด้วยชุดข้อมูลดังกล่าวจะมี Time complexity เป็น $O(n^2)$ แต่ก็ใช้เวลาน้อยกว่าการจัดเรียงข้อมูลแบบ Bubble sort และ Selection sort เป็นอย่างมาก ซึ่งแตกต่างจากผลลัพธ์ที่ได้จากแล็ปท็อป B ด้วยชุดข้อมูลเดียวกันที่การจัดเรียงแบบ Quick sort จะใช้เวลาเทียบเท่าการจัดเรียงแบบ Bubble sort และ Selection sort ที่เป็นเช่นนี้อาจเป็นผลมาจากการทำงานของ CPU ที่แตกต่างกันหรือสาเหตุอื่น ๆ ซึ่งอาจต้องทำการทดลองเชิงลึกเพิ่มเติมเพื่อให้ทราบถึงสาเหตุของความแตกต่างนี้

References

[1] GeeksforGeeks. (2023d, September 26). Sorting algorithms. Retrieved November 1, 2023, from https://www.geeksforgeeks.org/sorting-algorithms/

[2] GeeksforGeeks. (2023a, May 26). Selection sort data structure and algorithm tutorials. Retrieved November 1, 2023, from https://www.geeksforgeeks.org/selection-sort/

[3] GeeksforGeeks. (2023f, November 21). Bubble Sort data structure and algorithm tutorials. Retrieved November 1, 23 C.E., from https://www.geeksforgeeks.org/bubble-sort/

[4] GeeksforGeeks. (2023b, May 31). Insertion sort data structure and algorithm tutorials. Retrieved November 1, 2023, from https://www.geeksforgeeks.org/insertion-sort/

[5] GeeksforGeeks. (2023e, October 16). QuickSort Data structure and algorithm tutorials. Retrieved November 1, 2023, from https://www.geeksforgeeks.org/quick-sort/

[6] GeeksforGeeks. (2023c, July 6). Merge Sort Data Structure and Algorithms tutorials. Retrieved November 1, 2023, from https://www.geeksforgeeks.org/merge-sort/ Retrieved November 1, 2023, from https://www.geeksforgeeks.org/sorting-algorithms/

[7] Nethercote, N., & Seward, J. 2007. "Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation". Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI). https://doi.org/10.1145/1250734.1250746.

[8] Manwani, C. (2023, July 20). Stable and Unstable Sorting Algorithms. GeeksforGeeks. Retrieved November 18, 2023, from https://www.geeksforgeeks.org/stable-and-unstable-sorting-algorithms/