

บทที่ 6

การเรียงลำดับ (Sorting)

ทำไมต้องเรียงลำดับข้อมูล? เรียงลำดับข้อมูลจะแน่นเพื่อตัดเกรด, เรียงลำดับข้อมูลจะแน่นเพื่อจัดลำดับที่เรียนแก่นักเรียนที่สอบแอดมิชชัน, เรียงลำดับข้อมูลเพื่อการค้นหาแบบไบนารี, เรียงลำดับก่อนการพิมพ์รายงาน เพื่อจัดกลุ่มและสรุปรายงาน เมื่อข้อมูลที่จะเรียงลำดับมีจำนวนน้อยเช่นนักเรียนในห้องเรียน 200 คน เพื่อตัดเกรด เราก็จะไม่เห็นว่าจะเสียเวลาสักเท่าไร แต่ถ้าเป็นข้อมูลแอดมิชชันของนักเรียน 300,000 คน ข้อมูลประชากรไทยประมาณ 70,000,000 คน เวลาที่ใช้จัดเรียงข้อมูลปริมาณมากๆ อาจจะเป็นหลายๆชั่วโมง ดังนั้นต้องหาขั้นตอนวิธีที่มีประสิทธิภาพดีๆ เพื่อให้การจัดเรียงข้อมูลทำได้รวดเร็วยิ่งขึ้น ในบทนี้เราจะนำเสนอขั้นตอนวิธีในการเรียงลำดับวิธีต่างๆ พร้อมกับการวิเคราะห์และเปรียบเทียบประสิทธิภาพของแต่ละวิธี โดยจะยกเว้นวิธีเรียงลำดับแบบเลือก(selection sort) และการเรียงลำดับแบบผสาน (merge sort) ที่ได้ยกตัวอย่างแล้วในบทที่ 1











6.1 การเรียงลำดับแบบฟอง (Bubble Sort)

การเรียงลำดับแบบฟองจากน้อยไปมาก จากตัวอย่างการเข้าแถวของนักเรียนเรียงลำดับจากตัวเตี้ยไปสูง โดยจะเปรียบเทียบนักเรียน 2 คนที่อยู่ติดกัน ถ้าคนที่ 1 สูงกว่าคนที่ 2 ก็ให้ 2 คนสลับที่กัน จากนั้นเปรียบเทียบคนที่ 2 กับ คนที่ 3 ไปเรื่อยๆ จนถึงท้ายแถว (คนที่ n) ตามรูป 6.1 และ 6.2 จะได้ว่าคนที่อยู่ท้ายแถวสูงที่สุดแล้ว และเริ่มทำการเปรียบเทียบใหม่จนกระทั่งถึงคนที่ $n-1$ ทำอีกจนครบ $n-1$ รอบ ก็จะเรียงลำดับกันทุกคน แนวคิดของการเรียงลำดับแบบฟอง คือ คนที่ตัวเตี้ยก็จะค่อยๆ ขยับขึ้นไปหัวแถว ในขณะที่คนตัวสูงก็จะขยับไปอยู่ท้ายแถว

ตัวอย่าง 6.1 การเข้าแถวของนักเรียนอนุบาลวันเริ่มเปิดเทอมซึ่งยืนคละกั้นไป คุณครูประจำชั้นต้องการให้เข้าแถวเรียงลำดับจากตัวเตี้ยไปสูง โดยมีนักเรียน 10 คน ที่ความสูง

$$H = \{100, 120, 130, 140, 95, 115, 135, 118, 145, 132\}$$











$$\text{Name} = \{\text{"A"}, \text{"B"}, \text{"C"}, \text{"D"}, \text{"E"}, \text{"F"}, \text{"G"}, \text{"H"}, \text{"I"}, \text{"J"}\}$$

1	2	3	4	5	6	7	8	9	10
									
A-100	B-120	C-130	D-140	E-95	F-115	G-135	H-118	I-145	J-132











วิธีทำ ตามรูป 6.1

6-2 อัลกอริทึม











1. เทียบนักเรียนตำแหน่งที่ 1 กับ 2 ถ้านักเรียนตำแหน่งที่ 1 เดียวกันไม่ต้องแลกที่กัน
2. เทียบนักเรียนตำแหน่งที่ 2 กับ 3 ถ้านักเรียนตำแหน่งที่ 2 เดียวกันไม่ต้องแลกที่กัน
3. เทียบนักเรียนตำแหน่งที่ 3 กับ 4 ถ้านักเรียนตำแหน่งที่ 3 เดียวกันไม่ต้องแลกที่กัน
4. เทียบนักเรียนตำแหน่งที่ 4 กับ 5 ถ้านักเรียนตำแหน่งที่ 4 สูงกว่านักเรียนตำแหน่งที่ 5 ให้สลับที่กัน

1	2	3	4	5	6	7	8	9	10
									
A-100	B-120	C-130	D-140	E-95	F-115	G-135	H-118	I-145	J-132

5. เทียบนักเรียนตำแหน่งที่ 5 กับ 6 ถ้านักเรียนตำแหน่งที่ 5 สูงกว่านักเรียนตำแหน่งที่ 6 ให้สลับที่กัน

1	2	3	4	5	6	7	8	9	10
									
A-100	B-120	C-130	E-95	D-140	F-115	G-135	H-118	I-145	J-132











6. เทียบนักเรียนตำแหน่งที่ 6 กับ 7 ถ้านักเรียนตำแหน่งที่ 6 สูงกว่านักเรียนตำแหน่งที่ 7 ให้สลับที่กัน

1	2	3	4	5	6	7	8	9	10
									
A-100	B-120	C-130	E-95	F-115	D-140	G-135	H-118	I-145	J-132

.....











รูป 6.1 รอบแรกของการเรียงลำดับแบบฟอง

7. เทียบไปเรื่อยจนถึงนักเรียนตำแหน่งที่ 9 กับ ตำแหน่งที่ 10 จะได้นักเรียนตำแหน่งที่ 10 สูงที่สุดในแถว ตามรูป 6.2

1	2	3	4	5	6	7	8	9	10
									
A-100	B-120	C-130	E-95	F-115	G-135	H-118	D-140	J-132	I-145

รูป 6.2 เสร็จรอบแรกของการเรียงลำดับแบบฟอง

8. เริ่มรอบที่ 2 โดยเทียบนักเรียนตำแหน่งที่ 1 กับ 2, 2 กับ 3, 3 กับ 4, ... , 8 กับ 9 ถ้าเทียบแล้วคนถัดไปเตี้ยกว่าให้สลับที่กัน เมื่อครบรอบที่ 2 จะได้คนที่ $n-1$ สูงเป็นที่ 2 จากท้ายแถว
9. ทำเช่นนี้ไปเรื่อยๆ จนถึงรอบที่ 9 จะได้แถวนักเรียนที่เรียงจากเตี้ยไปสูง ตามรูป 6.3

1	2	3	4	5	6	7	8	9	10
									
E-95	A-100	F-115	H-118	B-120	C-130	J-132	G-135	D-140	I-145

รูป 6.3 เรียงลำดับแบบฟองแล้วเสร็จ

โปรแกรม 6.1 วิธีเรียงลำดับแบบฟอง

```

public class Sort1
{
public static void main(String argv[])
{   int aa[]={0,100,120,130,140,95,115,135,118,145,132};
    System.out.println("The original data before sorting");
    print(aa);
    BubbleSort(aa);
    System.out.println("The data after sorting");
    print(aa);
}

public static void print(int[] a)
{   for(int i=1;i<= a.length-1;i++)
        System.out.print(a[i]+" ");
    System.out.println();
}

public static void Swap(int[] a, int e1, int e2)
{   int tmp = a[e1];
    a[e1]= a[e2]; a[e2]= tmp;
    print(a);
}

public static void BubbleSort(int[] data )
{
    int out, in;
    for(out = data.length-1 ; out > 1; out-- )
        for(in = 1; in < out ; in++)
            if( data[in] > data[in+1] )
                {System.out.println ("round" + out+"\tswap at :"+in );
                 Swap( data, in,in+1); }
} // end BubbleSort()
}

```

outputs

```

The original data before sorting
100 120 130 140 95 115 135 118 145 132
round10 swap at :4
100 120 130 95 140 115 135 118 145 132
round10 swap at :5
100 120 130 95 115 140 135 118 145 132
round10 swap at :6
100 120 130 95 115 135 140 118 145 132
round10 swap at :7
100 120 130 95 115 135 118 140 145 132
round10 swap at :9
100 120 130 95 115 135 118 140 132 145
round9 swap at :3
100 120 95 130 115 135 118 140 132 145
round9 swap at :4
100 120 95 115 130 135 118 140 132 145
round9 swap at :6
100 120 95 115 130 118 135 140 132 145
round9 swap at :8
100 120 95 115 130 118 135 132 140 145
round8 swap at :2
100 95 120 115 130 118 135 132 140 145
round8 swap at :3
100 95 115 120 130 118 135 132 140 145
round8 swap at :5
100 95 115 120 118 130 135 132 140 145
round8 swap at :7
100 95 115 120 118 130 132 135 140 145
round7 swap at :1
95 100 115 120 118 130 132 135 140 145
round7 swap at :4
95 100 115 118 120 130 132 135 140 145
The data after sorting
95 100 115 118 120 130 132 135 140 145

```

ประสิทธิภาพของการเรียงลำดับแบบฟอง

ในรอบที่ 1 จะมีการเปรียบเทียบ $n-1$ ครั้ง

ในรอบที่ 2 จะมีการเปรียบเทียบ $n-2$ ครั้ง

⋮

ในรอบที่ $n-1$ จะมีการเปรียบเทียบ 1 ครั้ง

ซึ่งเมื่อรวมการเปรียบเทียบทุกรอบเข้าด้วยกันจะเป็น

$$1+2+3+\dots+(n-2)+(n-1) = \frac{n(n-1)}{2} = O(n^2)$$

เมื่อพิจารณาการสลับที่ จะแตกต่างกันตามลักษณะของข้อมูล

กรณีที่แย่ที่สุด คือ ในกรณีที่ข้อมูลอยู่ในลำดับที่ตรงข้าม เช่น ต้องการเรียงลำดับจากน้อยไปมาก แต่ข้อมูลเป็นการเรียงลำดับจากมากไปน้อย จะมีการสลับที่ทุกครั้งที่เปรียบเทียบ นั่นคือ รอบที่ 1 มี

การสลับที่ $n-1$ ครั้ง รอบที่ 2 มีการสลับที่ $n-2$ ครั้ง ไปเรื่อย ๆ จนรอบที่ $n-1$ มีการสลับที่เพียง 1 ครั้ง จึงมีการสลับที่ทั้งหมด $O(n^2)$

กรณีที่ดีที่สุด คือ ข้อมูลเรียงลำดับตามต้องการอยู่แล้ว จะไม่มีการสลับที่เลย

กรณีเฉลี่ย คือ ข้อมูลมีลักษณะสุ่ม จะมีการสลับที่เป็นครึ่งหนึ่งของกรณีที่ย่ำที่สุด คือ $\frac{n(n-1)}{4} = O(n^2)$

Complexity: จำนวนครั้งของการเปรียบเทียบและจำนวนครั้งของการสลับที่ของการเรียงลำดับแบบฟอง = $O(n^2)$











6.2 การเรียงลำดับแบบแทรก (Insertion sort)

แนวคิดของการเรียงลำดับแบบแทรก เปรียบเสมือนการถือไพ่อยู่ในมือ โดยไพ่ใบแรกเป็นหลัก แล้วหยิบไพ่ใบที่ 2 มาเปรียบเทียบกับกับใบแรก ถ้าแต้มน้อยกว่าก็จะแทรกไพ่ไปข้างหน้าสุด จากนั้นดูไพ่ใบที่ 3 โดยจะต้องนำมาเปรียบเทียบกับไพ่ใบที่ 1 กับใบที่ 2 เพื่อจะได้แทรกลงในตำแหน่งที่ถูกต้อง และทำเช่นนี้จนไพ่ทุกใบเรียงลำดับแล้ว

ตัวอย่าง 6.2 การเข้าแถวของนักเรียนอนุบาลวันเริ่มเปิดเทอมซึ่งยืนคละกันไป คุณครูประจำชั้นต้องการให้เข้าแถวเรียงลำดับจากตัวเตี้ยไปสูง โดยมีนักเรียน 10 คน ที่ความสูง

$$H = \{100, 120, 130, 140, 95, 115, 135, 118, 145, 132\}$$

$$\text{Name} = \{“A”, “B”, “C”, “D”, “E”, “F”, “G”, “H”, “I”, “J”\}$$

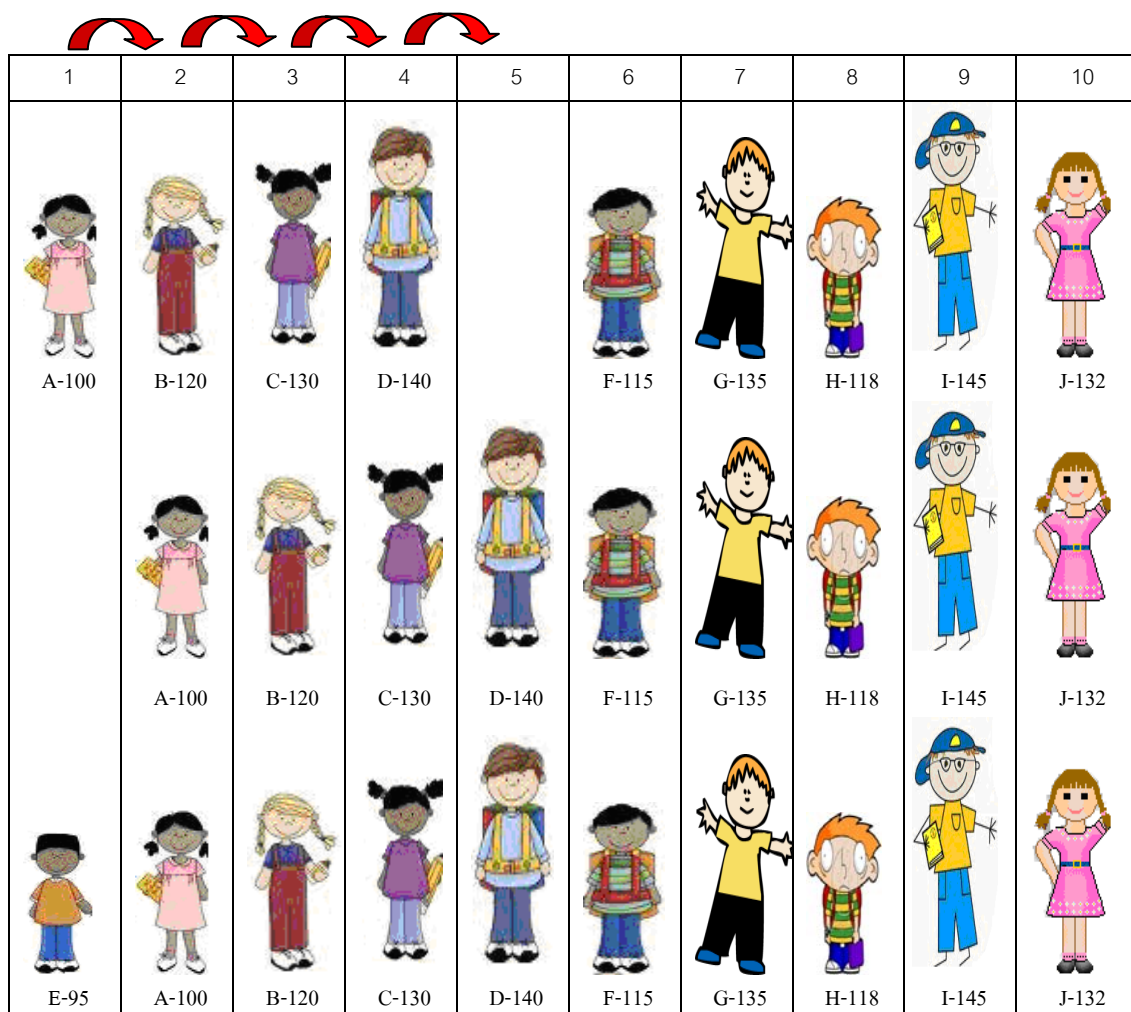
1	2	3	4	5	6	7	8	9	10
									
A-100	B-120	C-130	D-140	E-95	F-115	G-135	H-118	I-145	J-132

วิธีทำ

- ให้นักเรียนตำแหน่งที่ 1 (A-100 ซม.) ยืนเป็นหลักไว้
- ดูว่านักเรียนตำแหน่งที่ 2 (B-120 ซม.) เตี้ยกว่านักเรียนตำแหน่งที่ 1 ? ถ้าไม่เตี้ยกว่าก็ไม่ต้องย้ายตำแหน่ง
- ดูว่านักเรียนตำแหน่งที่ 3 (C-130 ซม.) เตี้ยกว่านักเรียนตำแหน่งที่ 1, 2 ? ถ้าไม่เตี้ยกว่าก็ไม่ต้องย้ายตำแหน่ง

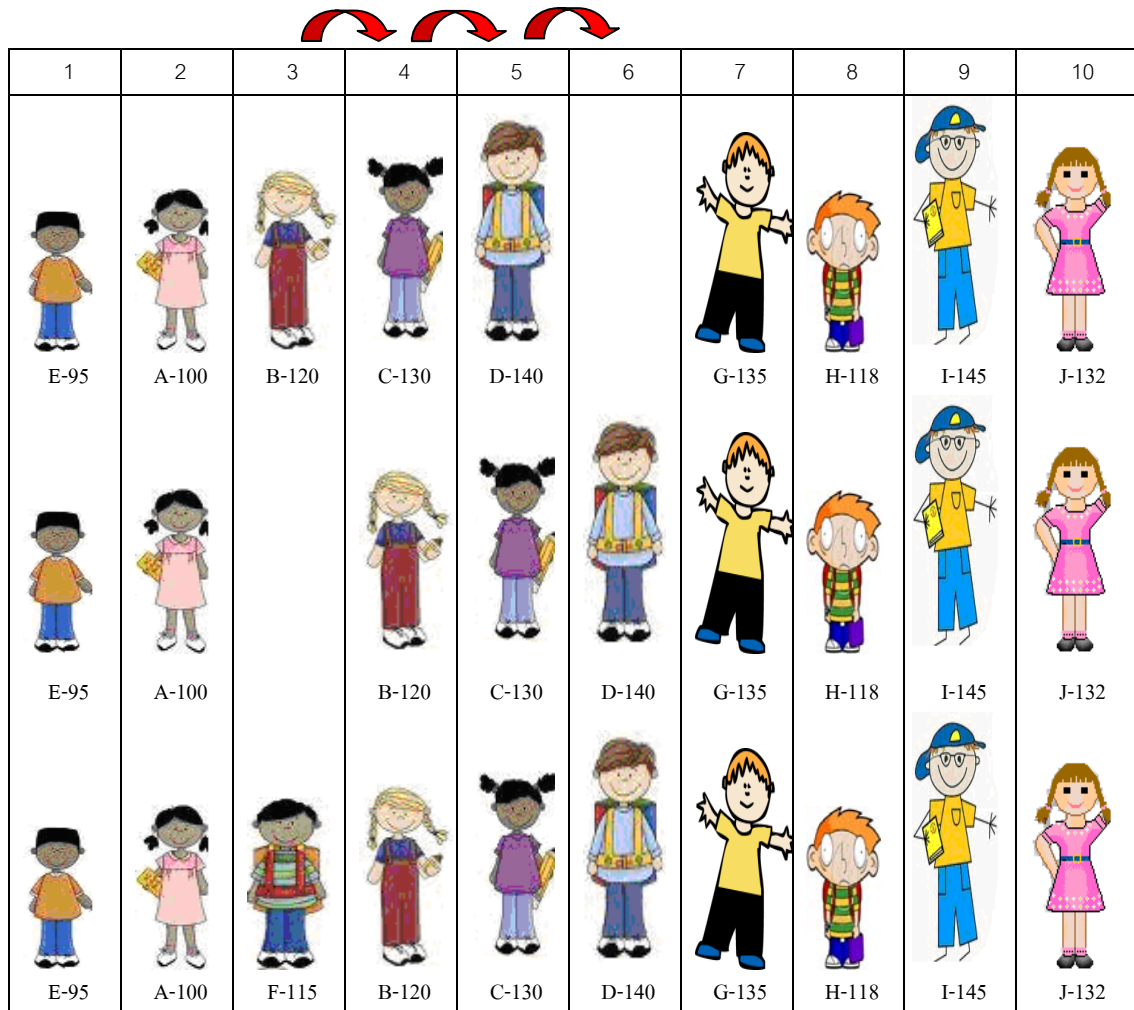
4. คู่ว่านักเรียนตำแหน่งที่ 4(D-140 ซม.) เดี่ยว่านักเรียนตำแหน่งที่ 1, 2, 3 ? ถ้าไม่เดี่ยวากก็ไม่ต้องย้ายตำแหน่ง

5. คู่ว่านักเรียนตำแหน่งที่ 5(E-95 ซม.) เดี่ยว่านักเรียนตำแหน่งที่ 1 เด็กชาย E จะต้องยืนตำแหน่งที่ 1 ดังนั้นต้องเว้นตำแหน่งที่ 1 ให้อับเด็กชาย E เพื่อให้เด็กชาย E เข้ามายืนแทนได้โดยไม่เหยียบเท้าเด็กหญิง A ด้วยการดึงเด็กชาย E (ตำแหน่งที่ 5) ออกจากแถว แล้วขยับเด็กชาย D ตำแหน่งที่ 4 ไปแทนตำแหน่งที่ 5, ขยับ A,B,C เลื่อนไปตามๆกัน จนตำแหน่งที่ 1 ว่าง ค่อยเอาเด็กชาย A เข้าไปในแถวในตำแหน่งที่ 1 ตามรูป 6.4



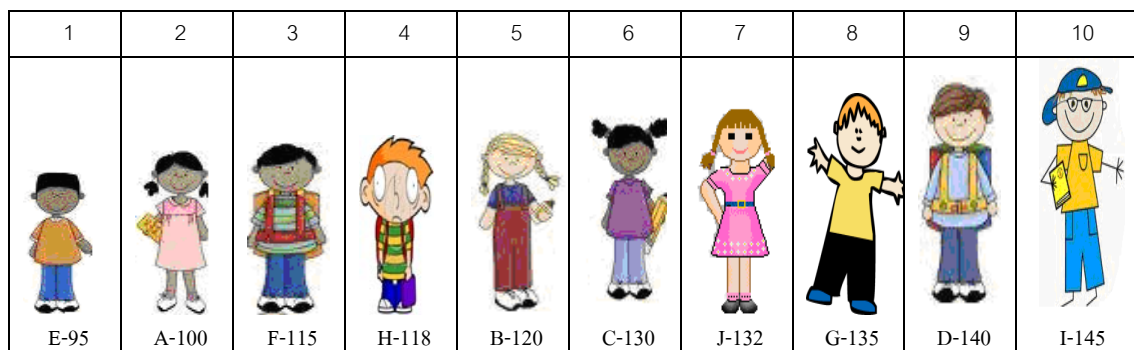
รูป 6.4 ดึงนักเรียนตำแหน่งที่ 5 ออกจากแถว ขยับนักเรียนคนที่ 4, 3, 2, 1 ไปทางขวาตามลำดับ จัดคนที่ดึงออกกลับไปแถวในตำแหน่งที่ 1

6. เทียบเด็กชาย F กับนักเรียนในตำแหน่ง 1, 2, 3 จะต้องย้ายเด็กชาย F ไปที่ตำแหน่ง 3 ทำวิธีการเดียวกับ ข้อ 5 ขยับ B, C, D ไปไว้ตำแหน่ง 4, 5, 6 จนตำแหน่งที่ 3 ว่าง ค่อยเอาเด็กชาย F เข้าไปในแถวในตำแหน่งที่ 3 ตามรูป 6.5



รูป 6.5 ดึงนักเรียนตำแหน่งที่ 6 ออกจากแถว ขยับนักเรียนคนที่ 5, 4, 3 ไปทางขวาตามลำดับ จัดคนที่ดึงออกกลับไปแถวในตำแหน่งที่ 3

7. ทำเช่นนี้ไปเรื่อยๆ จนถึงเด็กหญิง J ที่ยืนในตำแหน่งที่ 10 ก็จะเข้าแถวตามลำดับจากเตี้ยไปสูงตามรูป 6.6



รูป 6.6 นักเรียนเข้าแถวเรียงลำดับจากสูงไปเตี้ย

โปรแกรม 6.2 วิธีเรียงลำดับแบบแทรก (insertion sort)

```

import java.io.*;
class InsertSortApp
{ public static int [] a={0,100,120,130,140,95,115,135,118,145,132};      // ref to array a
  public static int nElems = 11;          // number of data items
  //-----
  public static void display()          // displays array contents
  { for(int j=1; j<nElems; j++)        // for each element,
    System.out.print(a[j] + " "); // display it
    System.out.println("");
  }
  //-----
  public static void insertSort()
  {
    int in, out;
    for(out=2; out<nElems; out++)    // out is dividing line
    { int temp = a[out];          // remove marked item
      in = out;                  // start shifts at out
      System.out.print("round="+out+"\t");
      while(in>1 && a[in-1] >= temp) // until one is smaller,
      { System.out.print("shift item "+(in-1)+"\t");
        a[in] = a[in-1];        // shift item to right
        --in;                   // go left one position
      }
      a[in] = temp;              // insert marked item
      System.out.println();
      display();
    } // end for
  } // end insertSort()
  // -----
  public static void main(String[] args) throws IOException
  { System.out.println("Raw data before sorting");
    display();              // display items
    insertSort();
    System.out.println("Array is ascending sort");
    display();              // display them again
  } // end main()
} // end class InsertSortApp

```

Outputs

```
Raw data before sorting
100 120 130 140 95 115 135 118 145 132
round=2
100 120 130 140 95 115 135 118 145 132
round=3
100 120 130 140 95 115 135 118 145 132
round=4
100 120 130 140 95 115 135 118 145 132
round=5 shift item 4  shift item 3  shift item 2  shift item 1
95 100 120 130 140 115 135 118 145 132
round=6 shift item 5  shift item 4  shift item 3
95 100 115 120 130 140 135 118 145 132
round=7 shift item 6
95 100 115 120 130 135 140 118 145 132
round=8 shift item 7  shift item 6  shift item 5  shift item 4
95 100 115 118 120 130 135 140 145 132
round=9
95 100 115 118 120 130 135 140 145 132
round=10  shift item 9  shift item 8  shift item 7
95 100 115 118 120 130 132 135 140 145
Array is ascending sort
95 100 115 118 120 130 132 135 140 145
```

ประสิทธิภาพของการเรียงลำดับแบบแทรก

การเรียงลำดับข้อมูลแบบแทรก จะไม่มีการสลับที่แต่จะมีการเลื่อนข้อมูลเพื่อให้ข้อมูลใหม่แทรกลงไป ดังนั้น ประสิทธิภาพของการเรียงลำดับแบบแทรก จึงพิจารณาจากจำนวนครั้งของการเปรียบเทียบและจำนวนครั้งของการเลื่อนข้อมูล ซึ่งจะแตกต่างกันตามลักษณะของข้อมูล กรณีที่ดีที่สุด คือ ข้อมูลเรียงลำดับตามต้องการอยู่แล้ว จะมีการเปรียบเทียบรอบละ 1 ครั้ง ดังนั้น จึงมีการเปรียบเทียบทั้งหมด $n-1$ ครั้ง และไม่มีการเลื่อนข้อมูลเลย ดังนั้น จำนวนครั้งของการเปรียบเทียบและการเลื่อนข้อมูล $= O(n)$

กรณีที่แย่ที่สุด คือ กรณีที่ข้อมูลอยู่ในลำดับที่ตรงข้าม เช่น ต้องการเรียงลำดับจากน้อยไปมาก แต่ข้อมูลเป็นการเรียงลำดับจากมากไปน้อย รอบที่ 1 จะมีการเปรียบเทียบ 1 ครั้งรอบที่ 2 มีการเปรียบเทียบ 2 ครั้ง ไปเรื่อย ๆ จนรอบที่ $n-1$ มีการเปรียบเทียบ $n-1$ ครั้ง จึงมีการเปรียบเทียบทั้งหมด $O(n^2)$ เนื่องจากมีการเลื่อนข้อมูลทุกครั้งที่เปรียบเทียบ จึงมีการเลื่อนข้อมูลทั้งหมด

$$1+2+3+\dots+(n-2)+(n-1) = \frac{n(n-1)}{2} = O(n^2)$$

ดังนั้น จำนวนครั้งของการเปรียบเทียบและการเลื่อนข้อมูล $= O(n^2)$

กรณีเฉลี่ย คือ ข้อมูลมีลักษณะสุ่ม จะมีการเปรียบเทียบและเลื่อนข้อมูลเป็นครึ่งหนึ่งของกรณีที่แย่ที่สุด คือ $\frac{n(n-1)}{4} = O(n^2)$ ดังนั้น จำนวนครั้งของการเปรียบเทียบและการเลื่อนข้อมูล $= O(n^2)$