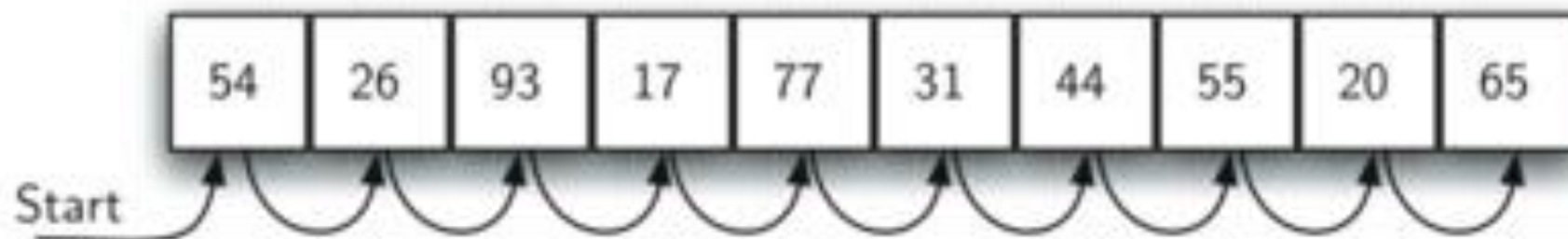


การค้นหา Searching

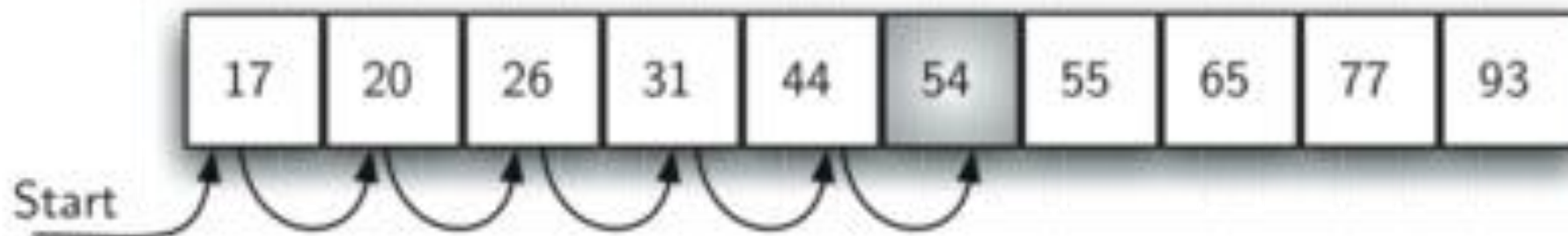
ค้นหาแบบตามลำดับ



ค้นหาแบบตามลำดับ

```
def sequentialSearch(alist, item):  
    pos = 0  
    found = False  
  
    while pos < len(alist) and not found:  
        if alist[pos] == item:  
            found = True  
        else:  
            pos = pos+1  
  
    return found  
  
testlist = [1, 2, 32, 8, 17, 19, 42, 13, 0]  
print(sequentialSearch(testlist, 3))  
print(sequentialSearch(testlist, 13))
```

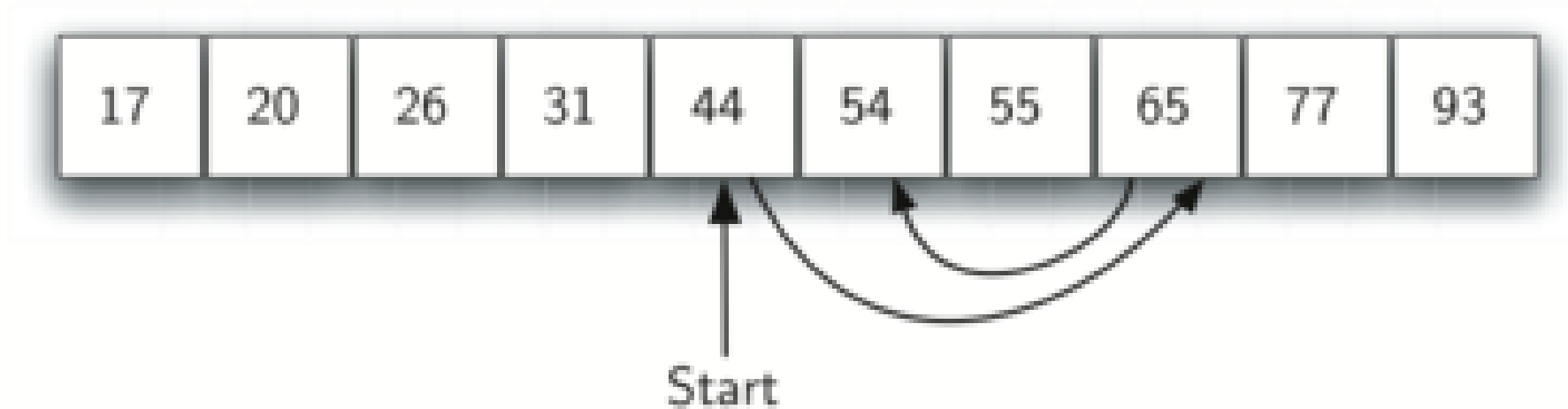
ถ้า list ที่ต้องการค้นหาเรียงลำดับ



ถ้า list ที่ต้องการค้นหาเรียงลำดับ

```
1 def orderedSequentialSearch(alist, item):
2     pos = 0
3     found = False
4     stop = False
5     while pos < len(alist) and not found and not stop:
6         if alist[pos] == item:
7             found = True
8         else:
9             if alist[pos] > item:
10                 stop = True
11             else:
12                 pos = pos+1
13
14     return found
15
```

การค้นหาแบบแบ่งครึ่ง (binary search)



การค้นหาแบบแบ่งครึ่ง

```
1 def binarySearch(alist, item):
2     first = 0
3     last = len(alist)-1
4     found = False
5
6     while first<=last and not found:
7         midpoint = (first + last)//2
8         if alist[midpoint] == item:
9             found = True
10        else:
11            if item < alist[midpoint]:
12                last = midpoint-1
13            else:
14                first = midpoint+1
15
16    return found
```

การค้นหาแบบแบ่งครึ่ง แก้ปัญหาแบบเวียนเกิด

```
1 def binarySearch(alist, item):
2     if len(alist) == 0:
3         return False
4     else:
5         midpoint = len(alist)//2
6         if alist[midpoint]==item:
7             return True
8         else:
9             if item<alist[midpoint]:
10                return binarySearch(alist[:midpoint],item)
11            else:
12                return binarySearch(alist[midpoint+1:],item)
13
```


การเรียงลำดับ

Sorting

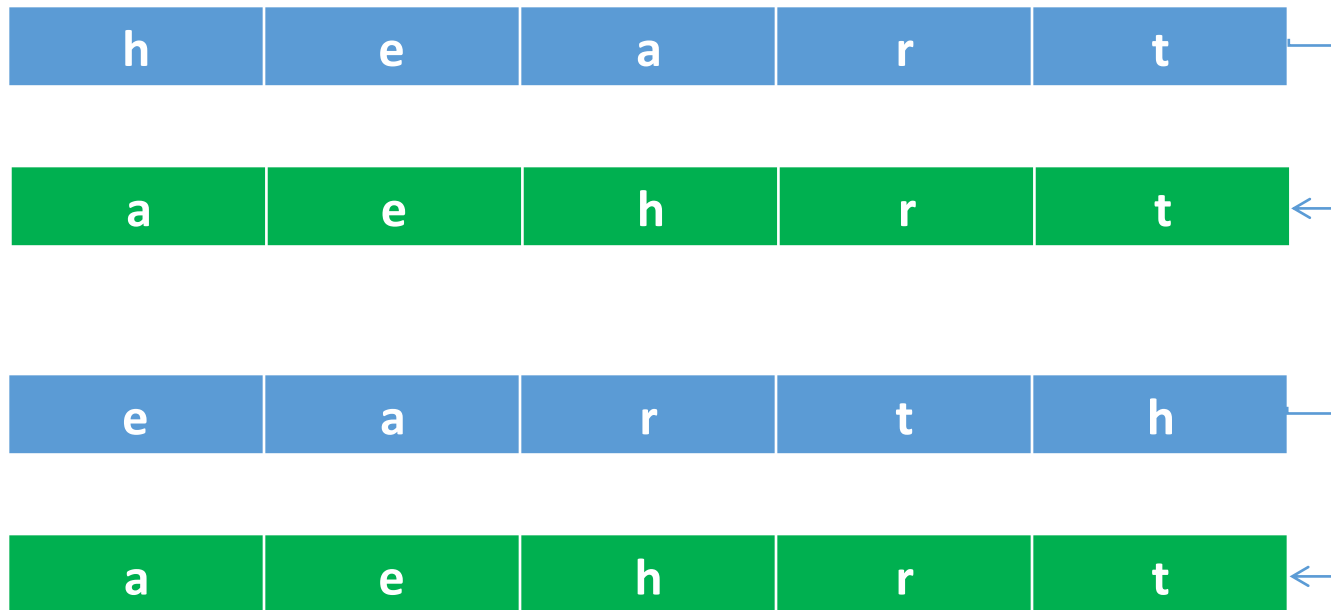
การเรียงลำดับ

54	26	93	17	77	31	44	55	20	65
----	----	----	----	----	----	----	----	----	----

17	20	26	31	44	54	55	65	77	93
----	----	----	----	----	----	----	----	----	----

Anagram detection

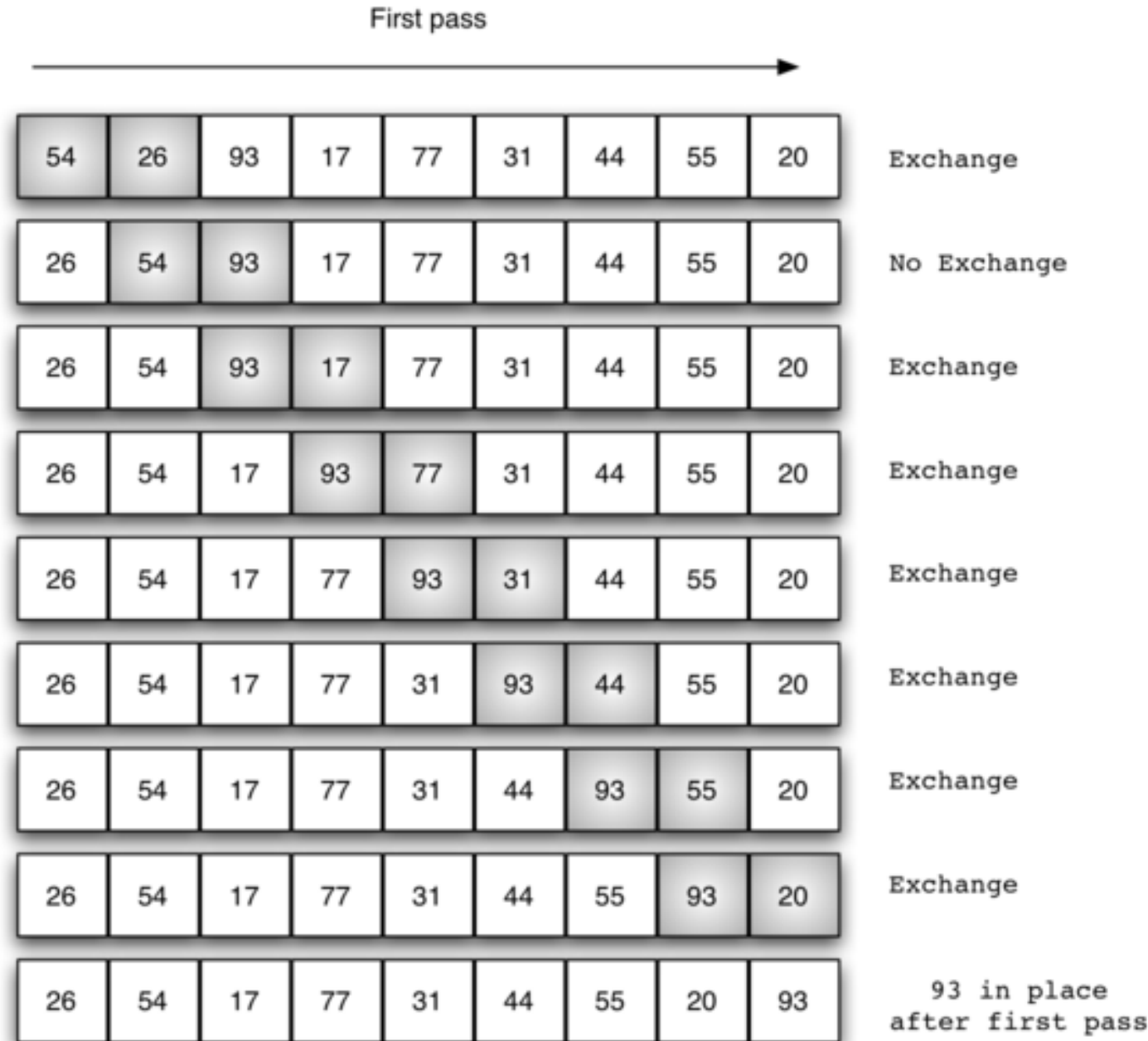
- เรียงลำดับแล้วเปรียบเทียบเป็นคู่ๆ



Bubble sort

- เปรียบเทียบเป็นคู่ๆ
- คู่ไหนไม่ตรงลำดับให้สลับ
- รอบแรก n ตัวเปรียบเทียบ $n - 1$ ครั้ง
- รอบสอง $n - 1$ ตัวเปรียบเทียบ $n - 2$ ครั้ง
- ...
- ทั้งหมด $n - 1$ รอบ

<http://interactivepython.org/runestone/static/pythonds/SortSearch/TheBubbleSort.html>



Bubble sort

```
1 def bubbleSort(alist):
2     for passnum in range(len(alist)-1,0,-1):
3         for i in range(passnum):
4             if alist[i]>alist[i+1]:
5                 temp = alist[i]
6                 alist[i] = alist[i+1]
7                 alist[i+1] = temp
8
9 alist = [54,26,93,17,77,31,44,55,20]
10 bubbleSort(alist)
11 print(alist)
12
```

Bubble sort

เรียงลำดับเสร็จแล้วยังต้องทำต่อ

```
1 def bubbleSort(alist):
2     for passnum in range(len(alist)-1,0,-1):
3         for i in range(passnum):
4             if alist[i]>alist[i+1]:
5                 temp = alist[i]
6                 alist[i] = alist[i+1]
7                 alist[i+1] = temp
8
9 alist = [54,26,93,17,77,31,44,55,20]
10 bubbleSort(alist)
11 print(alist)
12
```

Short bubble sort

```
1 def shortBubbleSort(alist):
2     exchanges = True
3     passnum = len(alist)-1
4     while passnum > 0 and exchanges:
5         exchanges = False
6         for i in range(passnum):
7             if alist[i]>alist[i+1]:
8                 exchanges = True
9                 temp = alist[i]
10                alist[i] = alist[i+1]
11                alist[i+1] = temp
12            passnum = passnum-1
13
14 alist=[20,30,40,90,50,60,70,80,100,110]
15 shortBubbleSort(alist)
16 print(alist)
17
```

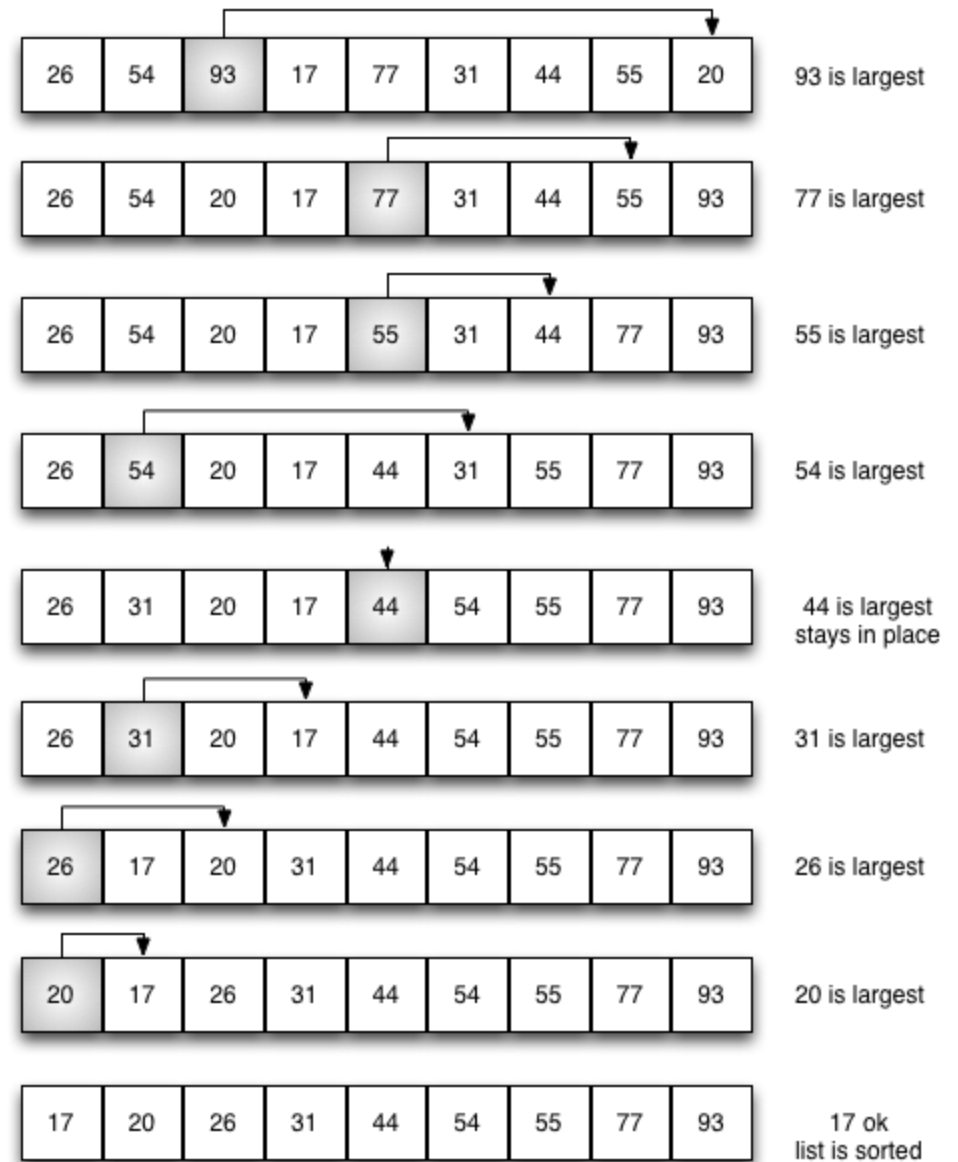
Short bubble sort

มีการสลับที่คือยังมีการเรียงลำดับอยู่

```
1 def shortBubbleSort(alist):
2     exchanges = True
3     passnum = len(alist)-1
4     while passnum > 0 and exchanges:
5         exchanges = False
6         for i in range(passnum):
7             if alist[i]>alist[i+1]:
8                 exchanges = True
9                 temp = alist[i]
10                alist[i] = alist[i+1]
11                alist[i+1] = temp
12        passnum = passnum-1
13
14 alist=[20,30,40,90,50,60,70,80,100,110]
15 shortBubbleSort(alist)
16 print(alist)
17
```


Selection sort

- หาตัวที่มากที่สุดไปวางที่ตำแหน่งสุดท้าย
- ใช้ $n - 1$ รอบในการเรียงลำดับ n ตัว

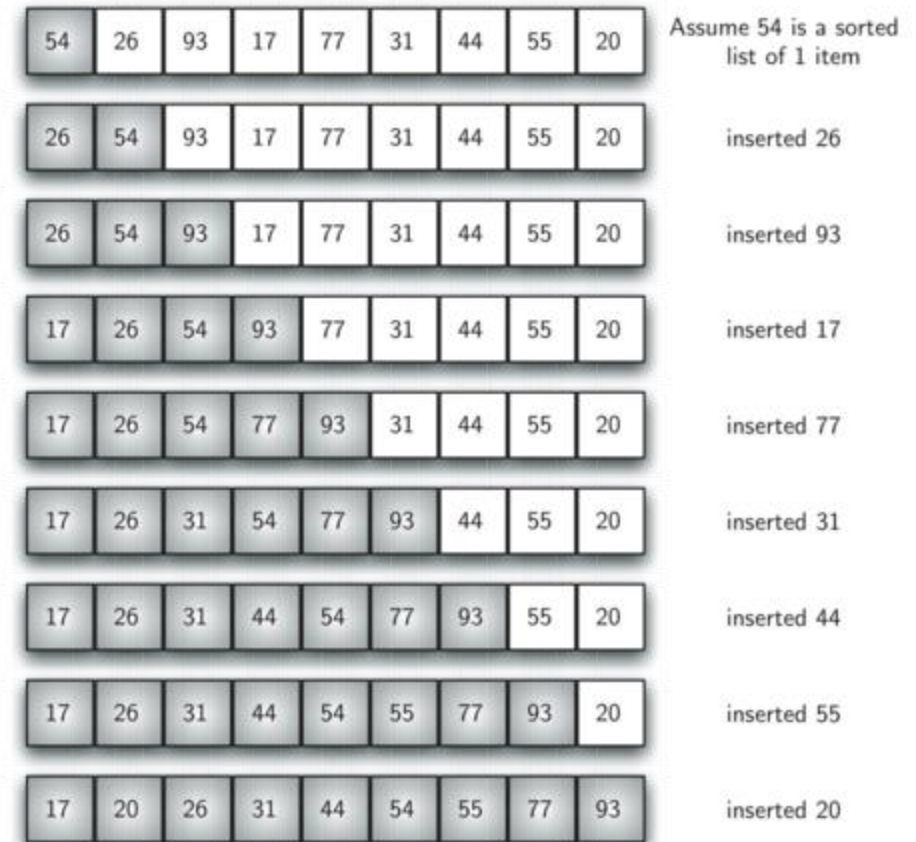


Selection sort

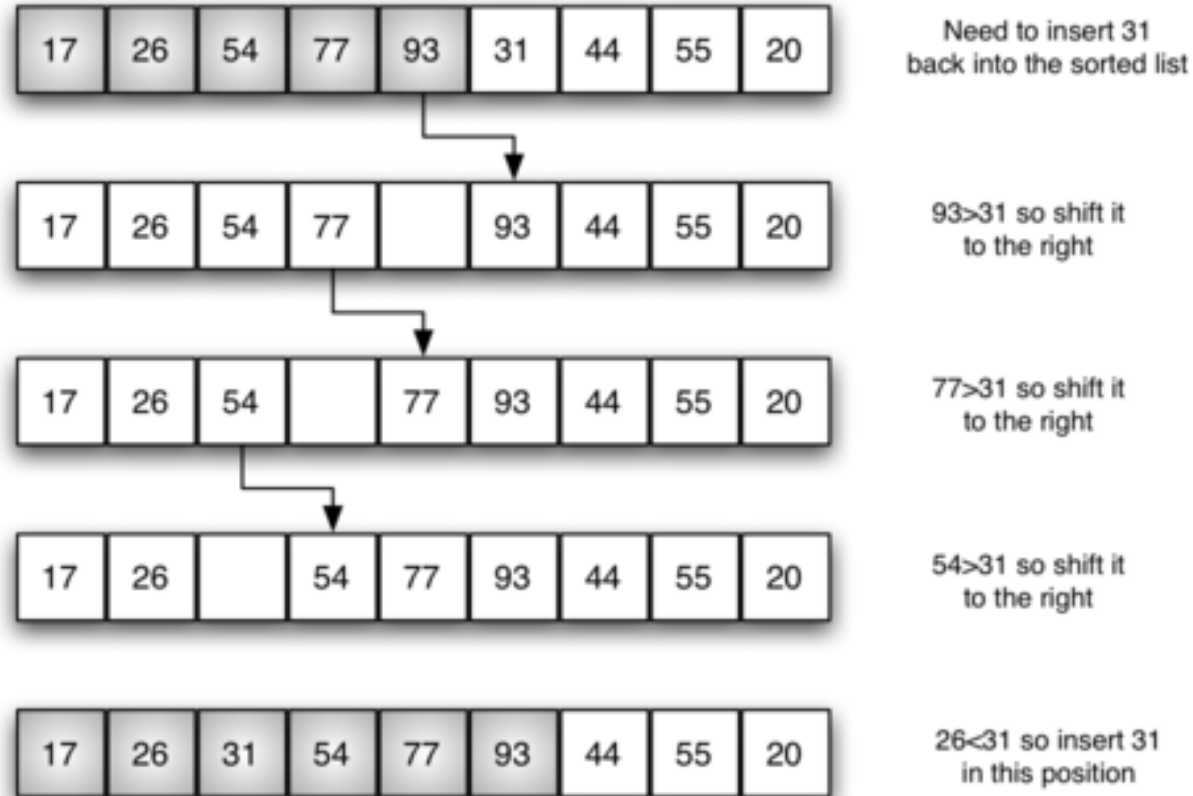
```
1 def selectionSort(alist):
2     for fillslot in range(len(alist)-1,0,-1):
3         positionOfMax=0
4         for location in range(1,fillslot+1):
5             if alist[location]>alist[positionOfMax]:
6                 positionOfMax = location
7
8         temp = alist[fillslot]
9         alist[fillslot] = alist[positionOfMax]
10        alist[positionOfMax] = temp
11
12 alist = [54,26,93,17,77,31,44,55,20]
13 selectionSort(alist)
14 print(alist)
15
```

Insertion sort

- ให้ส่วนที่เรียงลำดับแล้วอยู่ด้านหน้า
 - เริ่มที่ **1** ตัว
- ตัวถัดไปจะถูกใส่ในส่วนด้านหน้าให้ถูกตำแหน่ง



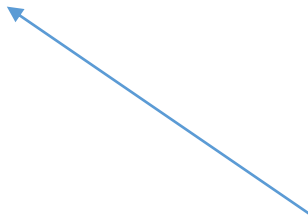
Insertion sort



Insertion sort

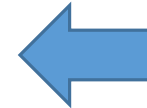
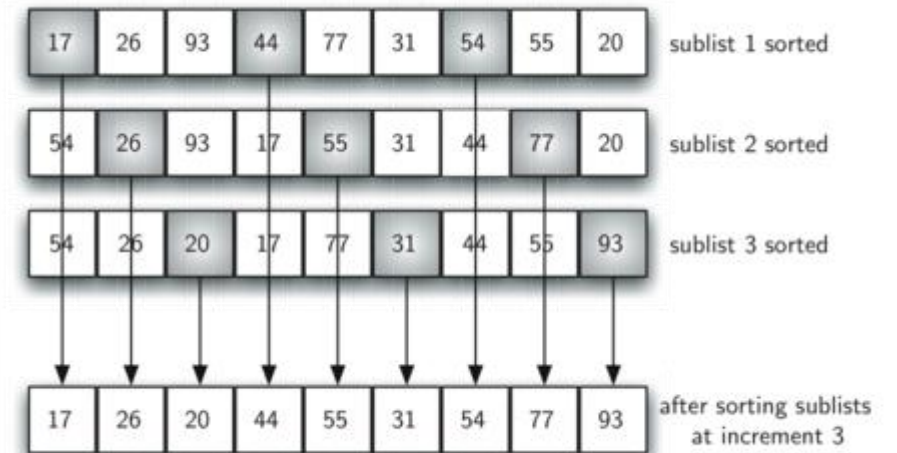
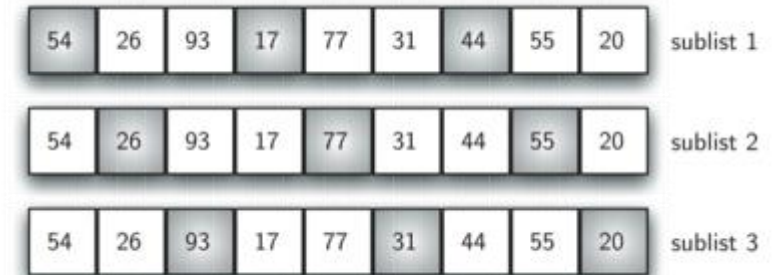
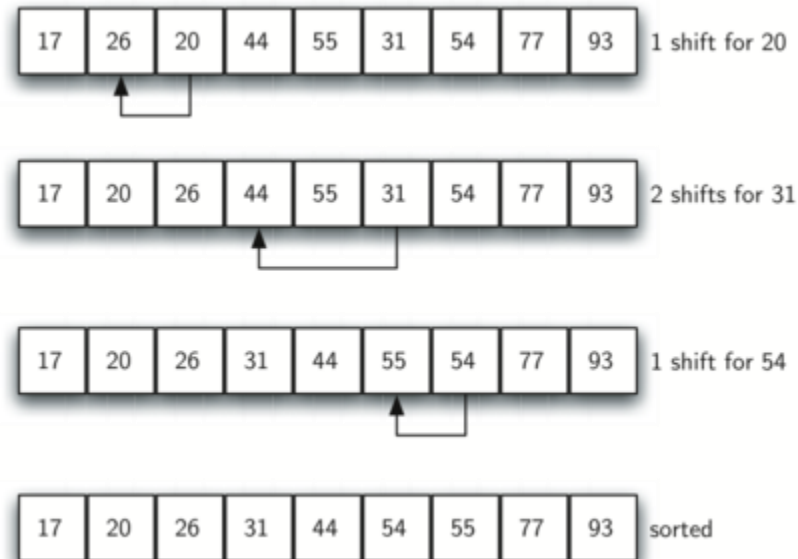
```
1 def insertionSort(alist):
2     for index in range(1,len(alist)):
3
4         currentvalue = alist[index]
5         position = index
6
7         while position>0 and alist[position-1]>currentvalue:
8             alist[position]=alist[position-1]
9             position = position-1
10
11         alist[position]=currentvalue
12
13 alist = [54,26,93,17,77,31,44,55,20]
14 insertionSort(alist)
15 print(alist)
16
```

เลื่อนไปทางขวา



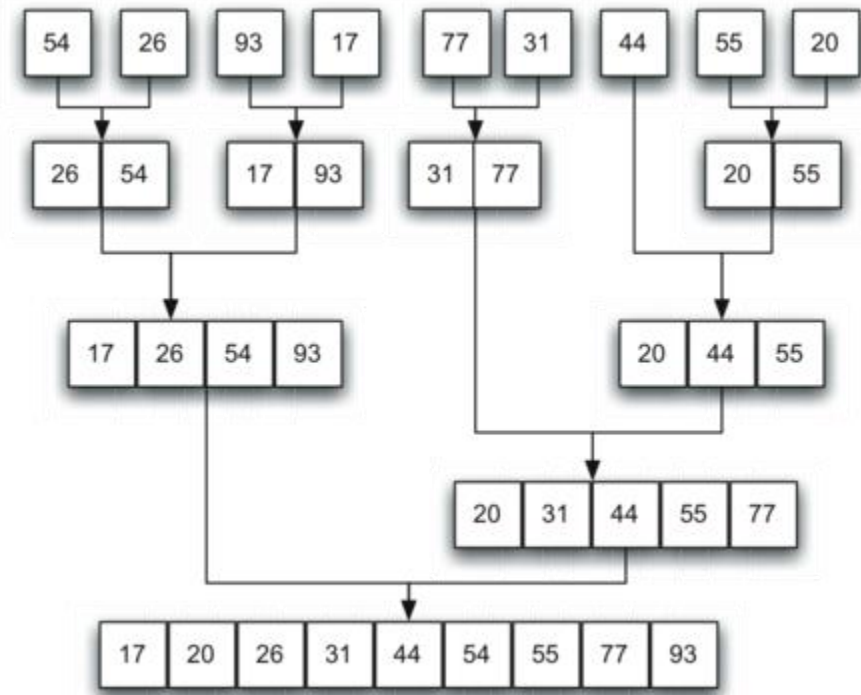
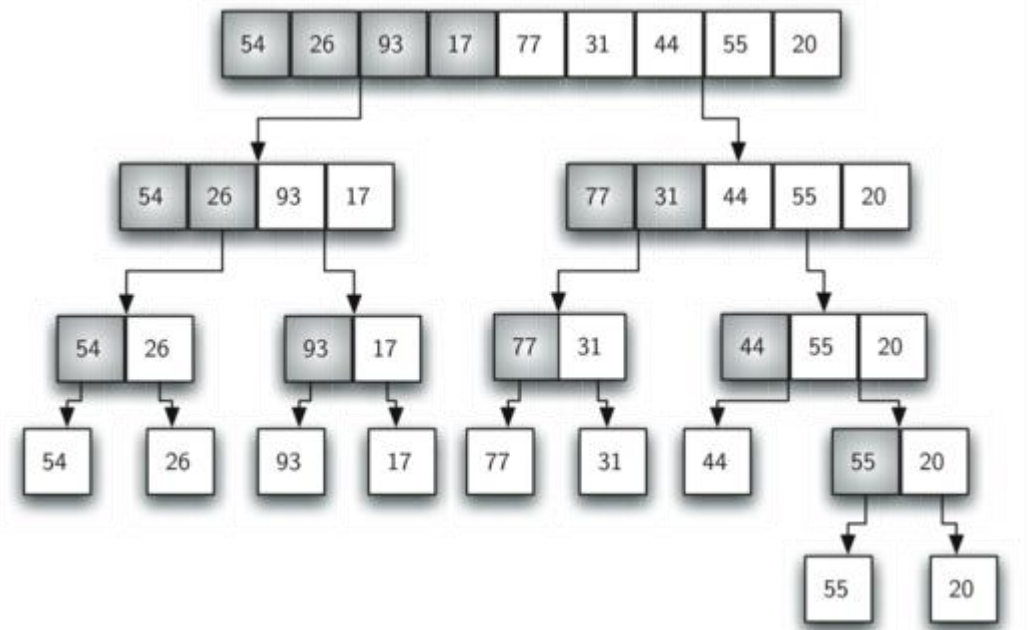
Shell sort

- Insertion sort แบบกระโดดข้าม
- รวมผลจาก insertion sort แบบกระโดดข้าม
- Insertion sort แบบปกติ



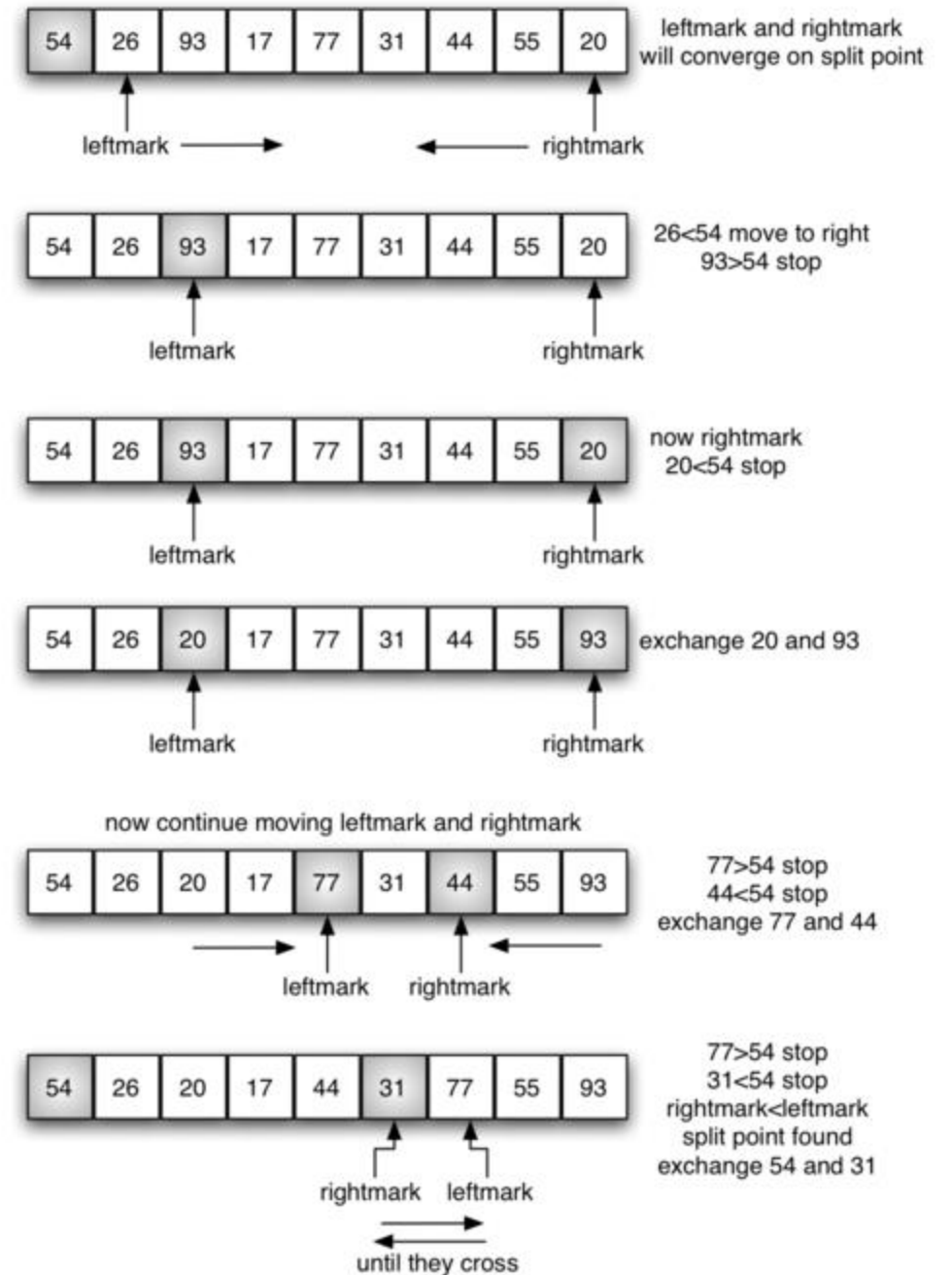
Merge sort

- แบ่งเป็นปัญหาย่อยลงไปเรื่อยๆ
- รวมกลับขึ้นมาเป็นการแก้ปัญหาลใหญ่
 - เปรียบเทียบแล้วเลือกหนุ่ม



Quick sort

- แบ่งปัญหาย่อยด้วยหมด
 - ใช้ตัวแรกของปัญหาย่อยเป็นหมด
- มี **marker 2** ตัวไว้จัดกลุ่ม
 - กลุ่มที่ $<$ ตัวหมด
 - กลุ่มที่ \geq ตัวหมด
- ทั้ง **2** กลุ่มคือปัญหาย่อยที่นำไปแบ่งต่อ



Quick sort

- แบ่งปัญหาย่อยด้วยหมุด
 - ใช้ตัวแรกของปัญหาย่อยเป็นหมุด
- มี **marker 2** ตัวไว้จัดกลุ่ม
 - กลุ่มที่ $<$ ตัวหมุด
 - กลุ่มที่ \geq ตัวหมุด
- ทั้ง 2 กลุ่มคือปัญหาย่อยที่นำไปแบ่งต่อ

