```java
public static <T extends Comparable <? super T>> void mysterySort1(List <T> list){
    for (int i = 1; i < list.size(); i++){
        int j = i;
        while (j > 0 && list.get(j - 1).compareTo(list.get(j)) > 0){
            swap(list, j, j - 1);
            j --;
        }
    }
}
```

1 * n
2 * (n-1)
5* (n-1)(n-1)
8 *(n-1) (n-2)
2 *(n-1) (n-2)

In mysterySort1, one of the most important parts in code is

```
swap(list, j,j-1);
j--;
```

therefore, just find how many times they run in worst case. Suppose the size of list is n, these 2 lines run (n-1)(n-2) times. So, the mysterySort1 is $O(n^2)$

```java
public static <T extends Comparable<? super T>> void mysterySort2(List<T> list) {
    int h = 1;
    while (h < list.size() / 3) {
        h = 3 * h + 1;// 1, 4, 13, 40, 121, 364, 1093, ...
    }
    while (h >= 1) { // "h-sort" the list.
        for (int i = h; i < list.size(); i++) {
            int j = i;
            while (j >= h && list.get(j - h).compareTo(list.get(j)) > 0) {
                swap(list, j, j - h);
                j -= h;
            }
        }
        h = h / 3;
    }
}
```

2 * 1
1 * (n+1)
3 * n

1 * (n)(n-h+1)
2 * (n) * (n-h)
1 * (n) * (n-h) *)
8 * (n) * (n-h)
2* (n) * (n-h)

h = 3 * h + 1 = (3^k+1) – 1 / 2

The first while loop is not that important, so we just ignore it. As second while loop, there are some nest loops inside it. However, the outmost while loop can be ignored, so, we just consider for loop and third while loop since

```
swap(list, j,j-h);
j-=h;
```

are generated by these 2 loops. Therefore, we just consider how many times for loop and third while loop run. for loop runs n-h+1 times, third while loop runs n/h +1 times. As result mysterySort2 is $O(n^2)$.

```java
public static <T extends Comparable<? super T>> void mysterySort3(List<T> list) {
    while (!isSorted(list)) {
        Collections.shuffle(list);
    }
}
```

n^n
$O(n) * n^n$

Collections.shuffle() is O(n). Consider the worst case, suppose size of list is n, and every position of list has n possibility. Therefore, the worst case for !isSorted(list) is $O(n^n)$.So, mysterySort3 is $O(n^{n+1})$