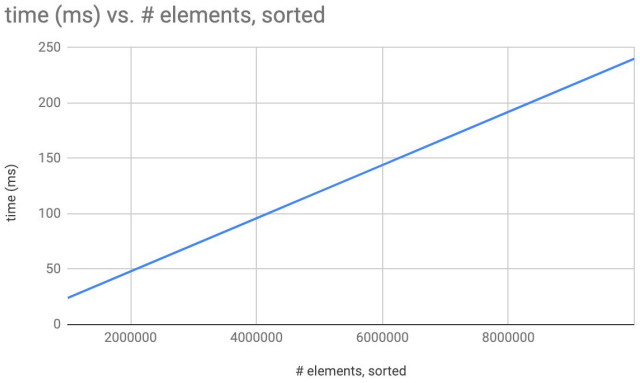
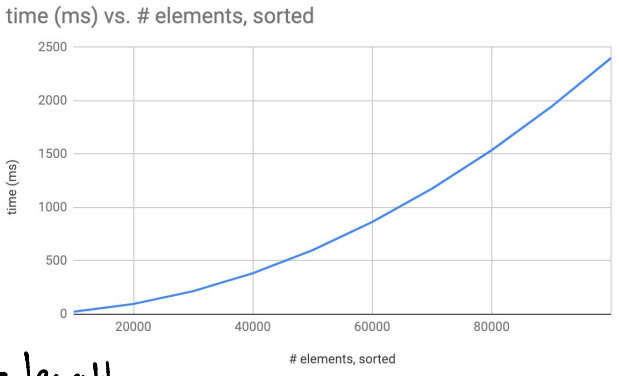
```
class Sort1 {
  public static boolean isSorted1(int[] arr) {
    for(int i = 0; i < arr.length - 1; i += 1) {
      if(arr[i] > arr[i + 1]) { return false; }
    }
    return true;
  }
}
```




```
class Sort2 {
  public static boolean isSorted2(int[] arr) {
    for(int i = 0; i < arr.length; i += 1) {
      for(int j = i + 1; j < arr.length; j += 1) {
        if(arr[i] > arr[j]) { return false; }
      }
    }
    return true;
  }
}
```



n = arr.length

```
boolean isSorted1(int[] arr) {

  for(int i = 0;

      i < arr.length - 1;

      i += 1) {

    if(arr[i] > arr[i + 1]) {

      return false;
    }
  }

  return true;
}
```

```
// # of times evaluated
// in sorted order          unordered at index k, k+1
//    1                     1
//    n - 1                 k - 1
//    n - 1                 k - 1
//    n - 1                 k - 1
//    0                     1
//
//    1                     0
```

$C * (1 + 3(n-1) + 1)$

```
boolean isSorted2(int[] arr) {

  for(int i = 0;

      i < arr.length;

      i += 1) {

    for(int j = i + 1;

        j < arr.length;

        j += 1) {

      if(arr[i] > arr[j]) {

        return false;
      }
    }
  }
  return true;
}
```

```
// # of times evaluated
// in sorted order          unordered at index k, k+1
//    1
//    n
//    n
//    n                                   n(n-1)/2
// (n-1) + (n-2) + (n-3) .....
//    n(n-1)/2
//    "
//    0
//
//    1
```

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

$C * (1 + 3n + (\frac{n(n-1)}{2}) * 3 + 1)$

```
boolean find( String[] theList, String toFind ) {

    for ( int i = 0;
          i < theList.length;
          i += 1 ) {
        if ( theList[i].equals( toFind )) {
            return true;
        }
    }
    return false;
}
```

```
// # of times evaluated
// toFind NOT FOUND      toFind FIRST      toFind at index k
//        1                   1                  1
//        n                   1                  k
//        n                   0                  k
//        n                   1                  k
//        0                   1                  1
//        1                   0                  0
```

WORST CASE

BEST CASE

AVERAGE CASE?

```
boolean find( String[] theList, String toFind ) {

    boolean found = false;
    for ( int i = 0;
          i < theList.length;
          i += 1 ) {
        if ( theList[i].equals( toFind )) {
            found = true;
        }
    }
    return found;
}
```

```
// # of times evaluated
// toFind NOT FOUND      toFind FIRST      toFind at index k
//        1                   1                  1
//        1                   1                  1
//        n                   n                  n
//        n                   n                  n
//        n                   n                  n
//        0                   1                  1
//        1                   1                  1
```

n + n + n ...