Algorithm performance

Analysis of selection sort

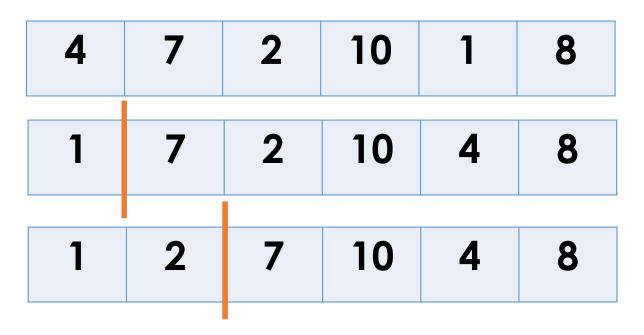
By the end of this video you will be able to...

Analyze the performance of selection sort

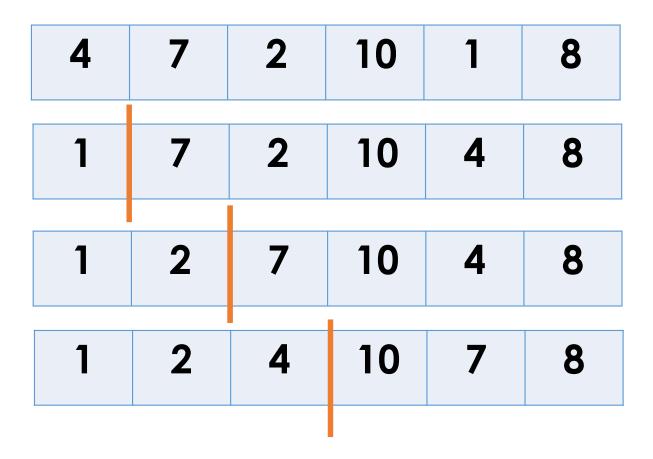


4	7	2	10	1	8
1	7	2	10	4	8











```
public static void selectionSort( int[] vals ) {
  int indexMin;
  for ( int i=0; i < vals.length-1; i++) {
      indexMin = i ;
      for ( int j=i+1; j < vals.length; j++ ) {</pre>
        if ( vals[j] < vals[indexMin] ) {</pre>
            indexMin = j ;
        }
      swap ( vals, indexMin , i );
```

```
public static void selectionSort( int[] vals )
  int indexMin;
  for ( int i=0; i < vals.length-1; i++) {
      indexMin = i ;
      for ( int j=i+1; j < vals.length; j++ ) {</pre>
        if ( vals[j] < vals[indexMin] ) {</pre>
            indexMin = j ;
        }
      swap ( vals, indexMin , i );
```

```
public static void selectionSort( int[] vals )
  int indexMin;
  for ( int i=0; i < vals.length-1; i++) {
      indexMin = i ;
      for ( int j=i+1; j < vals.length; j++ ) {</pre>
        if ( vals[j] < vals[indexMin] ) {</pre>
            indexMin = j ;
        }
      swap ( vals, indexMin , i );
```

```
public static void selectionSort( int[] vals )
  int indexMin;
  for ( int i=0; i < vals.length; i++ ) {
      indexMin = i ;
      for ( int j=i+1; j < vals.length; j++ ) {</pre>
        if ( vals[j] < vals[indexMin] ) {</pre>
            indexMin = j ;
        }
      swap ( vals, indexMin , i );
```

```
public static void selectionSort( int[] vals )
  int indexMin;
  for ( int i=0; i < vals.length; i++ ) {
                                                    OUTER LOOP
                                                       n times
      indexMin = i ;
      for ( int j=i+1; j < vals.length; j++ ) {</pre>
        if ( vals[j] < vals[indexMin] ) {</pre>
            indexMin = j ;
        }
      swap ( vals, indexMin , i );
```

```
public static void selectionSort( int[] vals )
  int indexMin;
  for ( int i=0; i < vals.length; i++ ) {</pre>
      indexMin = i ;
      for ( int j=i+1; j < vals.length; j++ ) {</pre>
        if ( vals[j] < vals[indexMin] ) {</pre>
             indexMin = j ;
                                        INNER LOOP
         }
      swap ( vals, indexMin , i );
```

```
for ( int j=i+1; j < vals.length; j++ ) {
   if ( vals[j] < vals[indexMin] ) {
      indexMin = j ;
   }
}</pre>
```

```
for ( int j=i+1; j < vals.length; j++ ) {
      O(1)
}</pre>
```

```
for ( int j=i+1; j < vals.length; j++ ) {
    O(1)
    INNER FOR LOOP: O ( n-(i+1) )</pre>
```

```
for ( int i=0; i < vals.length; i++ ) {</pre>
   indexMin = i ;
   for ( int j=i+1; j < vals.length: i
     if / --- 1
           INNER FOR LOOP: O (n-(i+1))
   swap ( vals, indexMin , i );
```

```
for ( int i=0; i < vals.length; i++ ) {
      O(1)
      INNER FOR LOOP: O ( n-(i+1) )
      O(1)
}</pre>
```

```
for ( int i=0; i < vals.length; i++ ) {</pre>
               O(1) + O(n-i-1) + O(1)
```

```
for ( int i=0; i < vals.length; i++ ) {</pre>
                     O ( n-i )
```

```
for ( int i=0; i < vals.length; i++ ) {</pre>
                    O ( n-i )
               Outer * inner is
                O(n) * O(n-i)
```

```
for ( int i=0; i < vals.length; i++ ) {</pre>
                     O ( n-i )
               Outer * inner is
                 O(n) * O(n-i)
                But what is O(n-i)?
```

```
for ( int i=0; i < vals.length; i++ ) {</pre>
                     O (n-i)
                Outer * inner is
                 O(n) * O(n-i)
                But what is O(n-i)?
                As i increases, this
                term decreases....
```

```
) n +
```

```
for ( int i=0; i < vals.length; i++ ) {
            O(n-i)</pre>
```

```
n + (n-1) +
```

```
for ( int i=0; i < vals.length; i++ ) {
          O(n-i)</pre>
```

```
)
n + (n-1) + (n-2)
```

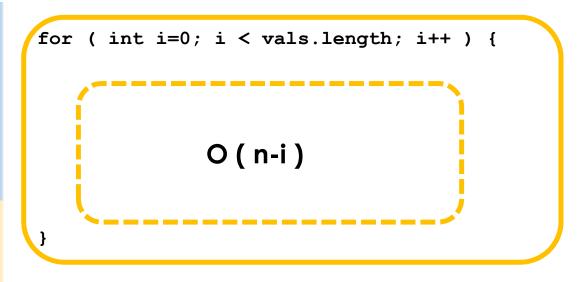
```
for ( int i=0; i < vals.length; i++ ) {
          O(n-i)</pre>
```

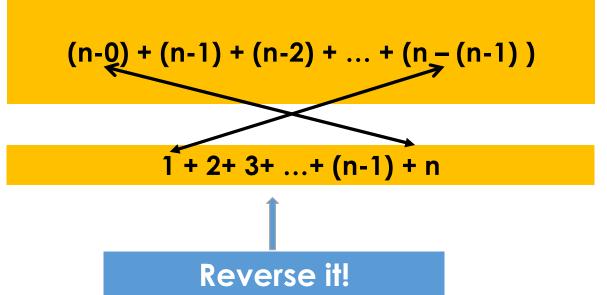
```
)
n + (n-1) + (n-2) + ... + 1
```

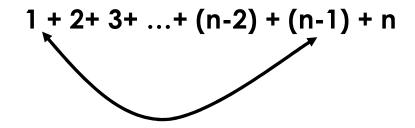
```
for ( int i=0; i < vals.length; i++ ) {
      O(n-i)</pre>
```

Captures the entire runtime!

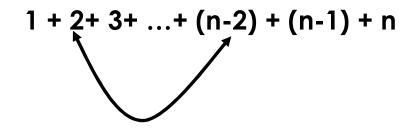
$$(n-0) + (n-1) + (n-2) + ... + (n - (n-1))$$





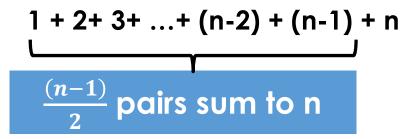


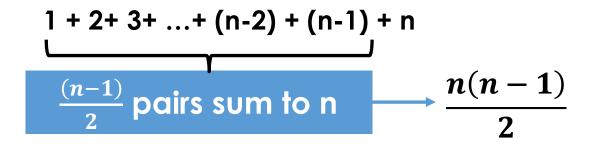
These sum to n



These sum to n

How many total pairs (in the range above) sum to n?





$$\frac{1 + 2 + 3 + ... + (n-2) + (n-1) + n}{\frac{(n-1)}{2} \text{ pairs sum to n}} \xrightarrow{n(n-1)} \frac{n(n-1)}{2}$$

$$=\frac{n(n-1)}{2}+n$$

Fun Fact (Gauss Summation Trick)

$$1+2+3+\cdots+(n-1)=\frac{n(n-1)}{2}$$

Fun Fact (Gauss Summation Trick)

$$1+2+3+\cdots+(n-1)=\frac{n(n-1)}{2}$$

Busy work: add all the numbers from 1 to 100

Fun Fact (Gauss Summation Trick)

$$1+2+3+\cdots+(n-1)=\frac{n(n-1)}{2}$$

Busy work: add all the numbers from 1 to 100

$$=\frac{101(101)}{2}=\frac{10,100}{2}=5,050$$

Wrapping up...

$$=\frac{n(n-1)}{2}+n$$

To finish the analysis, what is the tightest correct classification for the equation above?

IVQ

- What is the tightest correct classification for
- f(n) = n(n-1)/2 + n?
 - O(n)
 - O(n-1)
 - O(n^2)
 - O(n(n-1))

$$= \frac{n(n-1)}{2} + n$$

$$= \frac{n^2 - n}{2} + n$$

$$= \frac{n^2 - n}{2} + \frac{2n}{2}$$

$$= \frac{n^2 + n}{2} = O(n^2)$$

$$= \frac{n(n-1)}{2} + n$$

$$= \frac{n^2 - n}{2} + n$$

$$= \frac{n^2 - n}{2} + \frac{2n}{2}$$

$$= \frac{n^2 + n}{2} = O(n^2)$$

Selection sort's runtime is $O(n^2)$