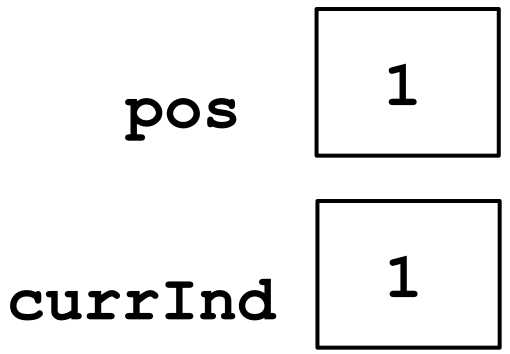
Sorting Data

Mystery sort

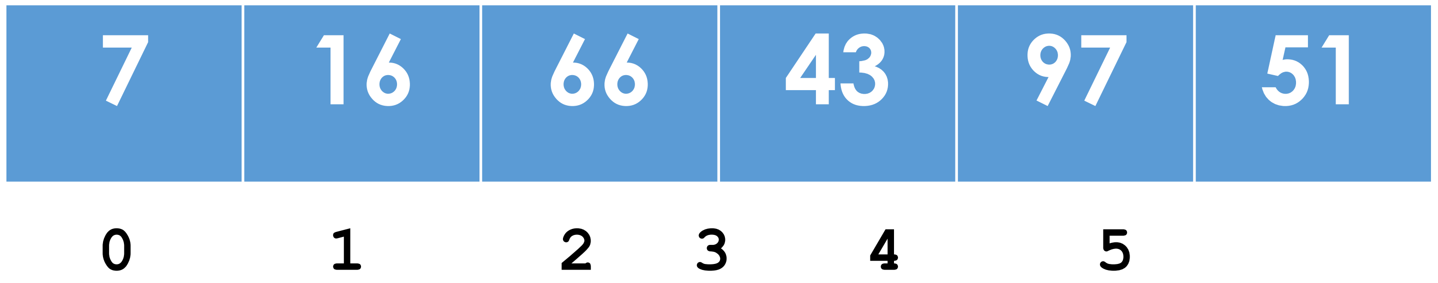
By the end of this video you will be able to…

- Trace code and describe its high-level function
- Describe alternate algorithms for sorting

```
public static void mysterySort( int[] vals )     {

   int currInd;

   for ( int pos=1; pos < vals.length ; pos++ ) {

      currInd = pos ;

      while ( currInd > 0 && vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }

   }

}
```



```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```



pos	1
currInd	1

```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```

7	16	66	43	97	51
0	1	2	3	4	5

pos $\boxed{2}$

currInd $\boxed{2}$

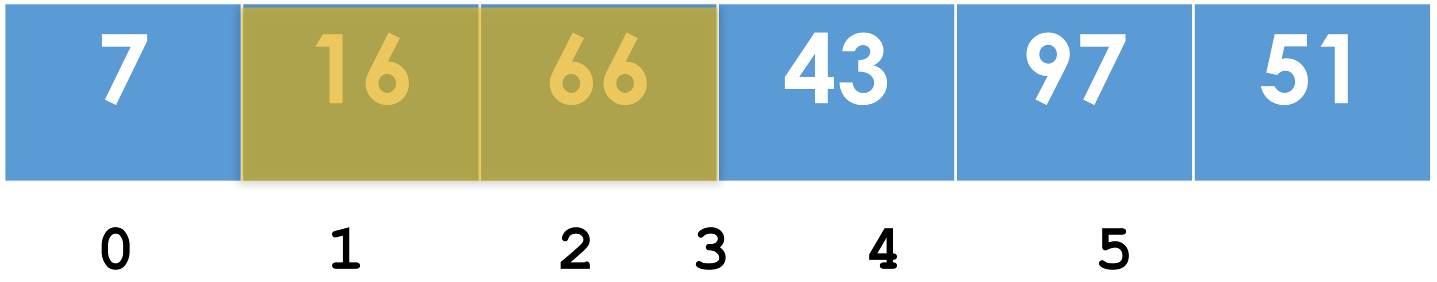
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```

7	16	66	43	97	51
0	1	2	3	4	5

pos 2

currInd 2

```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```



pos	3
currInd	3

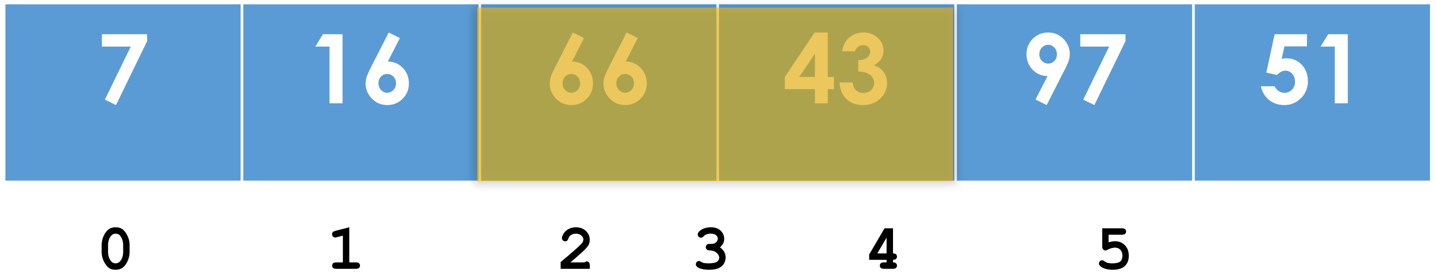
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```

7	16	66	43	97	51
0	1	2	3	4	5

pos 3

currInd 3

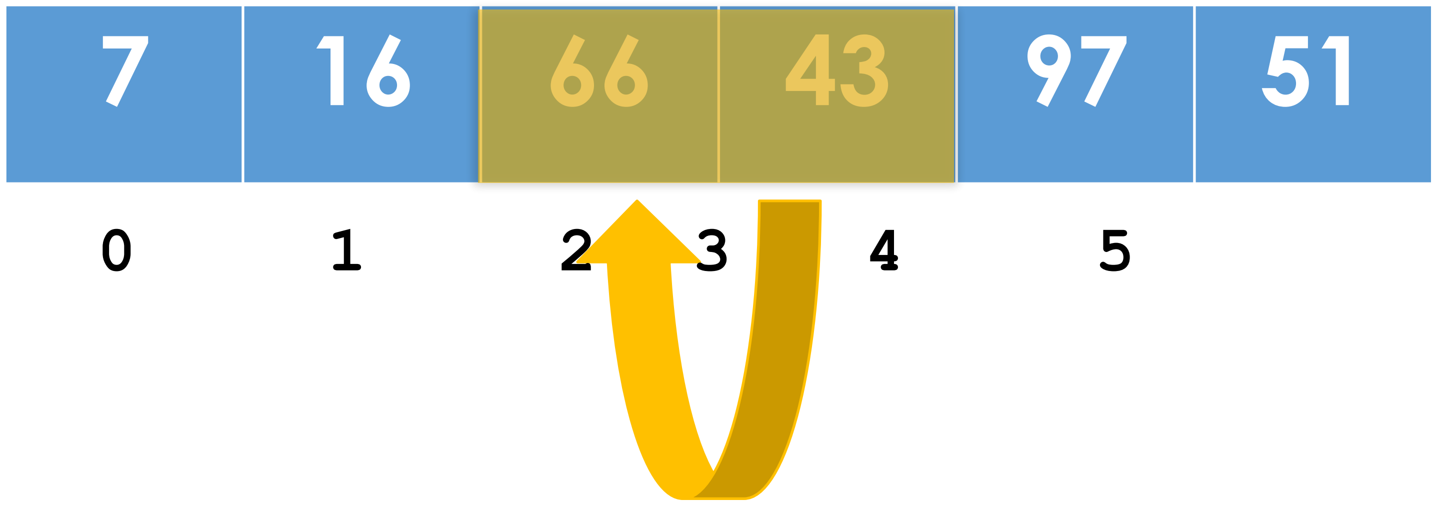
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0  &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```



pos 3

currInd ~~3~~

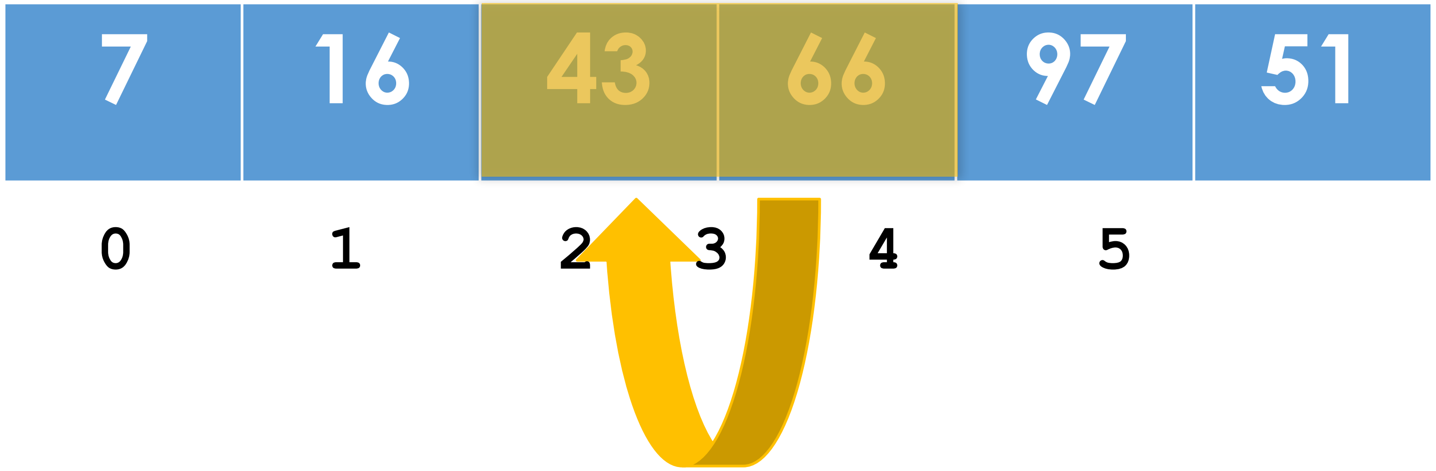
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd – 1;
      }
   }
}
```



pos 3

currInd 2

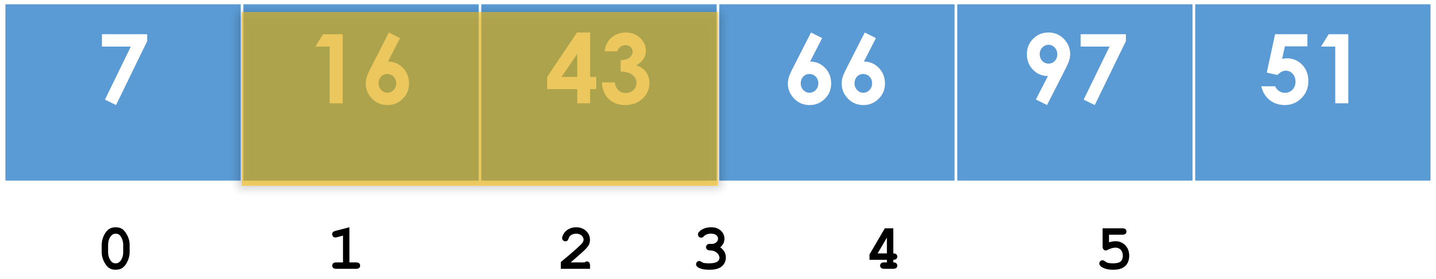
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd – 1;
      }
   }
}
```



pos 3

currInd 2

```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0  &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```

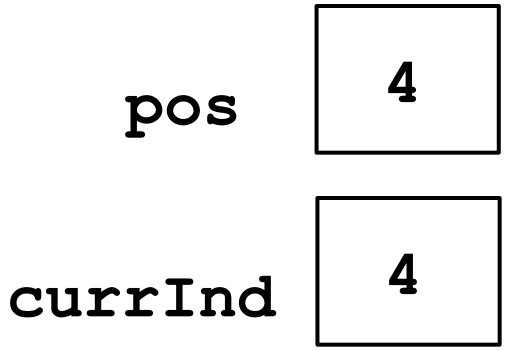


pos $\boxed{4}$

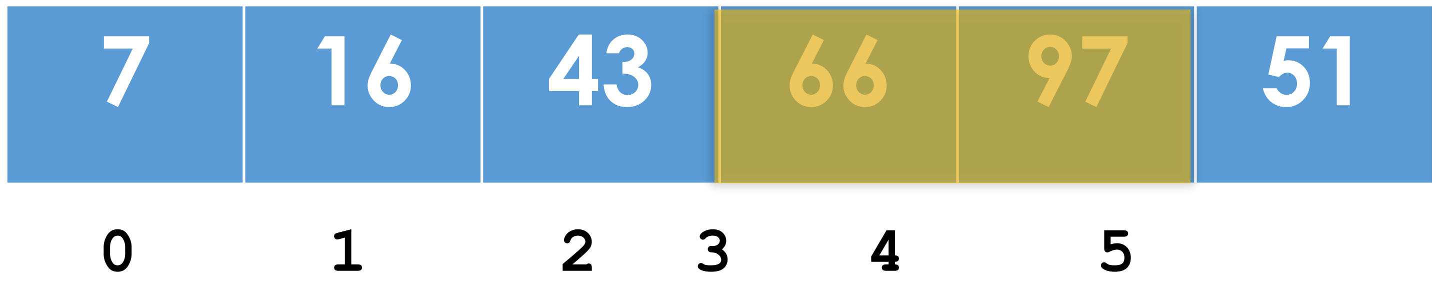
currInd $\boxed{4}$

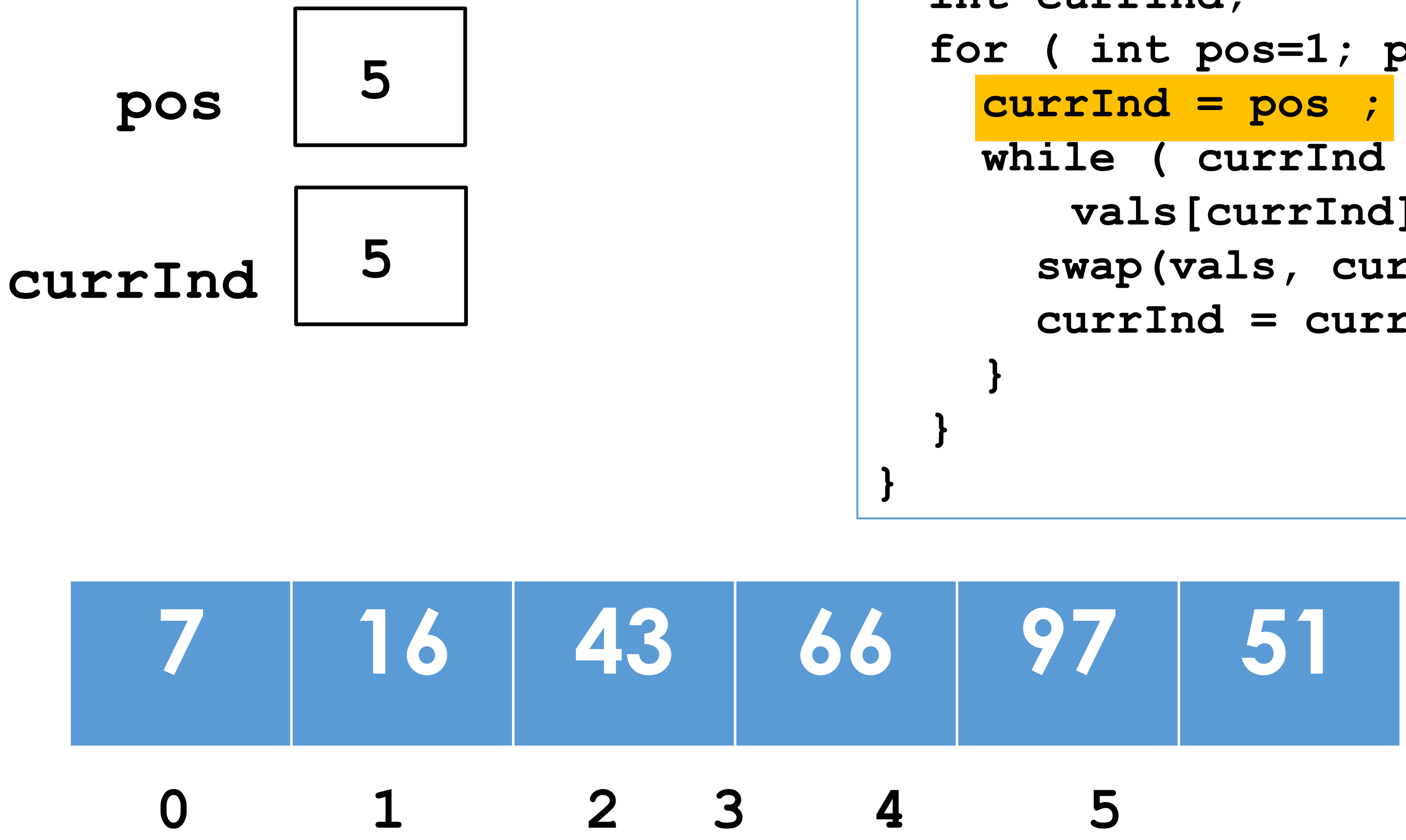
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```

7	16	43	66	97	51
0	1	2	3	4	5

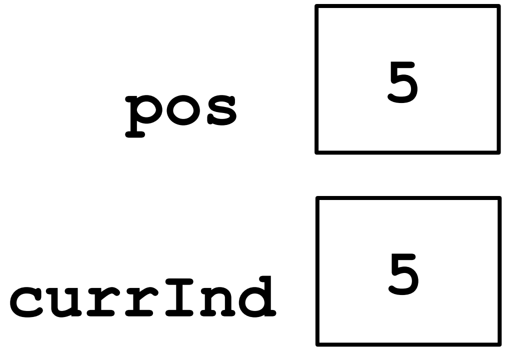


```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0  &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```

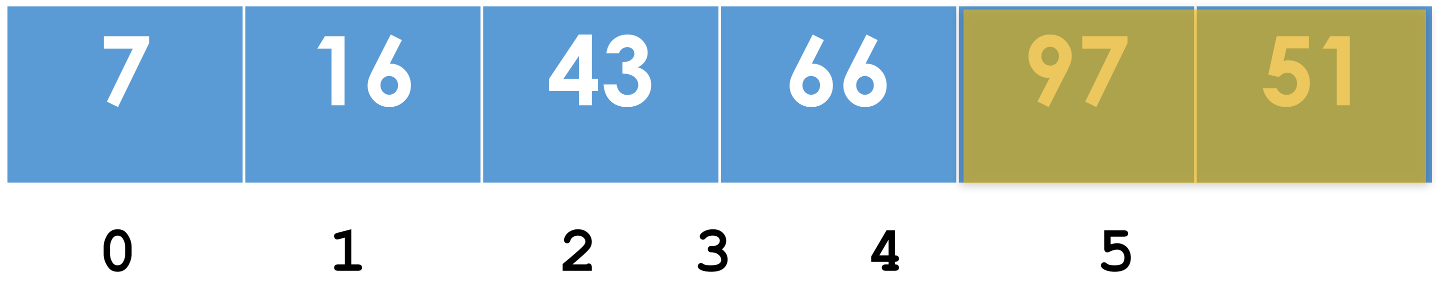




```
public static void mysterySort( int[] vals )    {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```



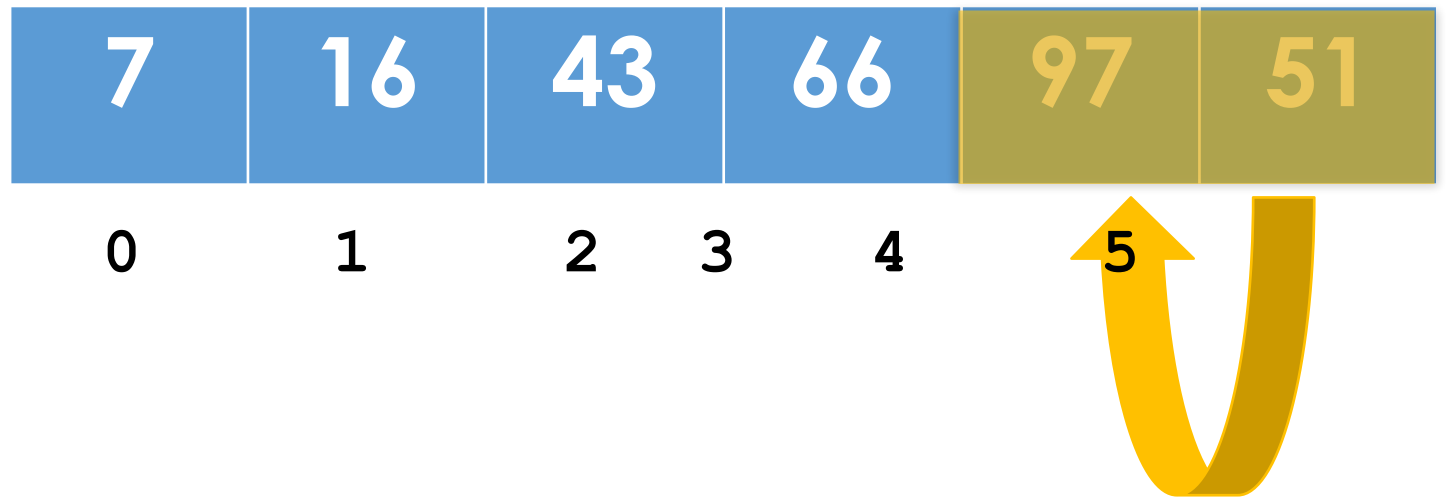
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```



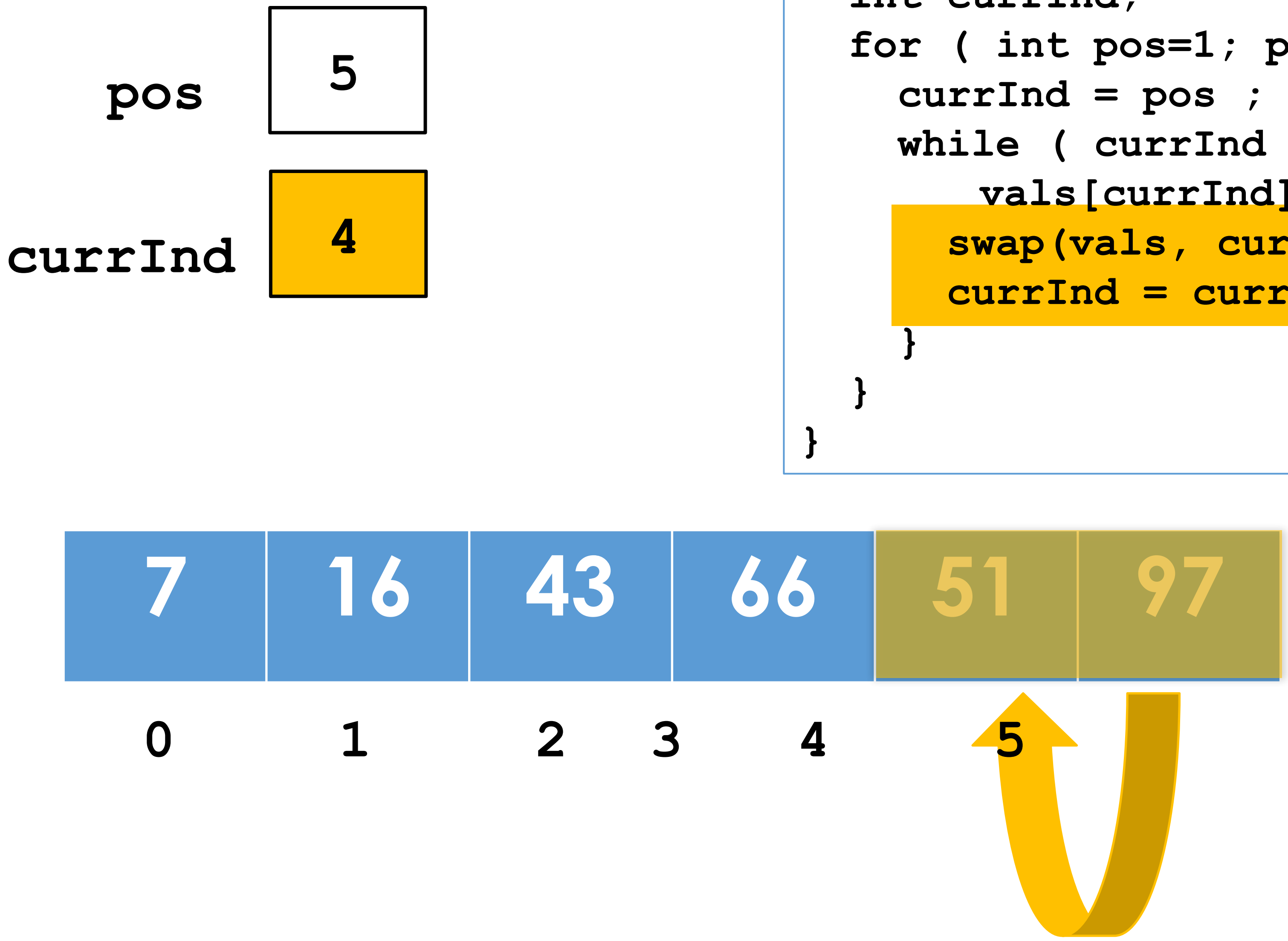
pos 5

currInd ~~5~~

```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd – 1;
      }
   }
}
```



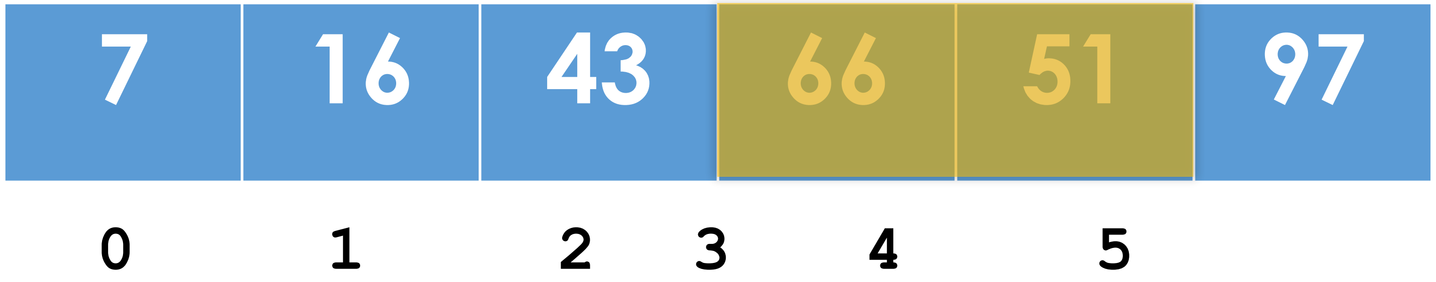
```
public static void mysterySort( int[] vals )     {
  int currInd;
  for ( int pos=1; pos < vals.length ; pos++ ) {
    currInd = pos ;
    while ( currInd > 0 &&
        vals[currInd] < vals[currInd-1] ) {
      swap(vals, currInd, currInd-1);
      currInd = currInd - 1;
    }
  }
}
```

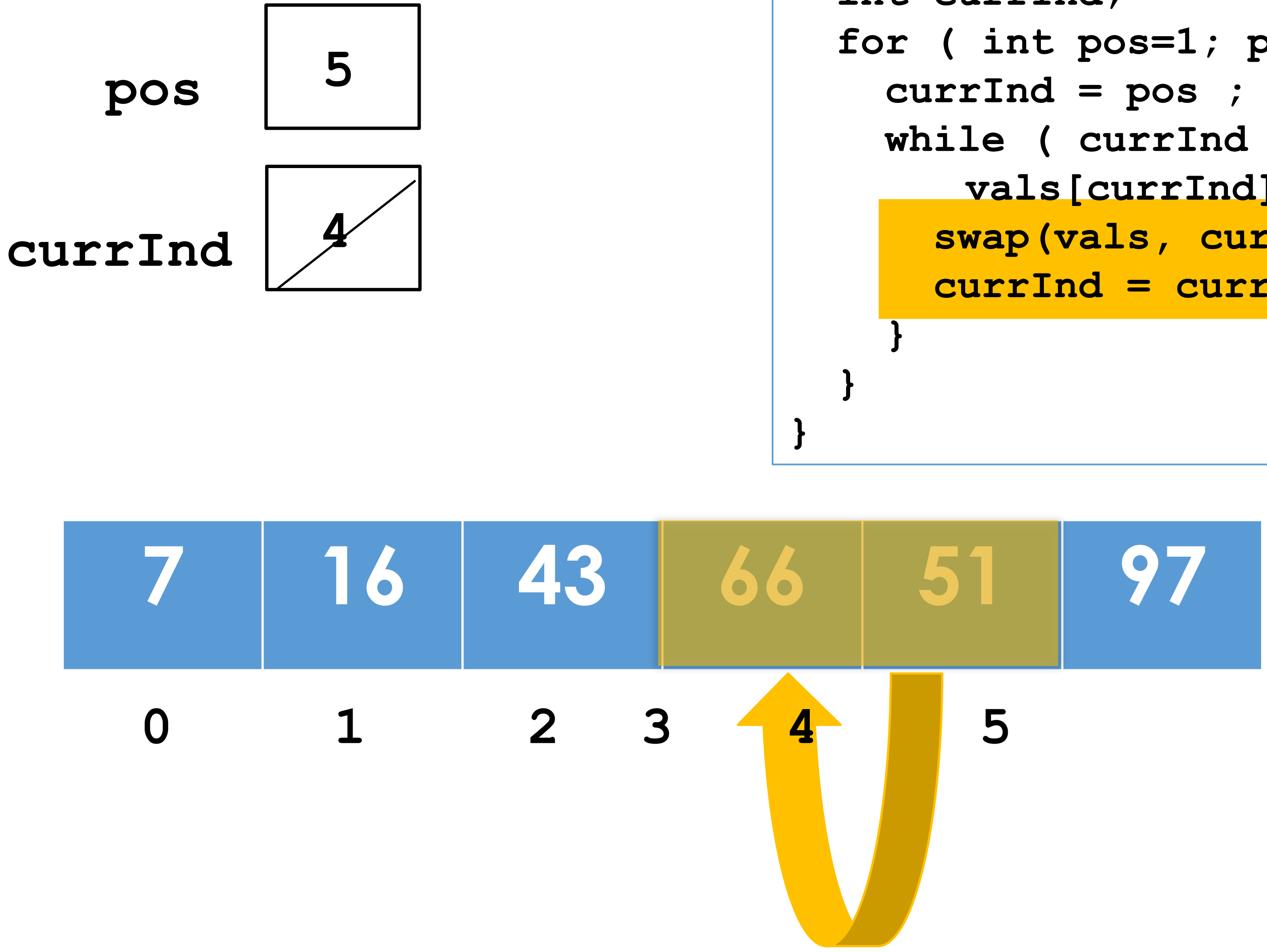


pos 5

currInd 4

```
public static void mysterySort( int[] vals )    {
  int currInd;
  for ( int pos=1; pos < vals.length ; pos++ ) {
    currInd = pos ;
    while ( currInd > 0  &&
        vals[currInd] < vals[currInd-1] ) {
      swap(vals, currInd, currInd-1);
      currInd = currInd – 1;
    }
  }
}
```



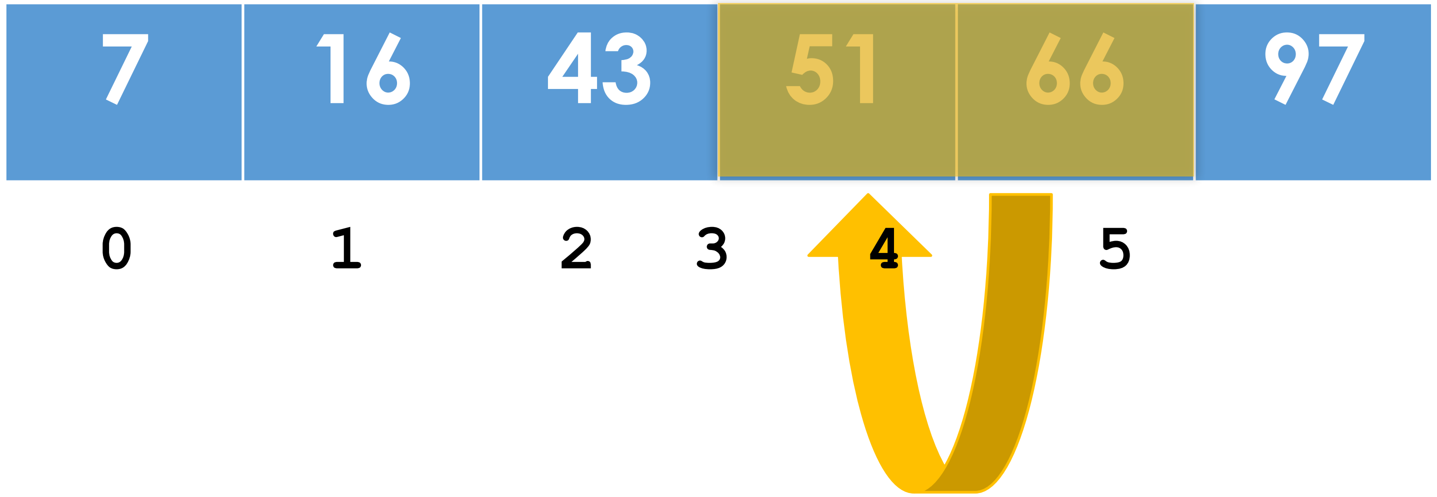


```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```

pos 5

currInd 3

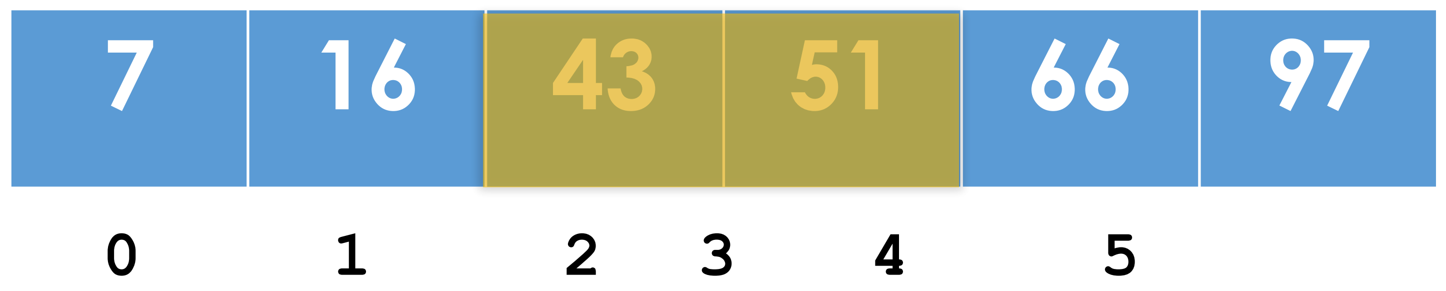
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0 &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd - 1;
      }
   }
}
```

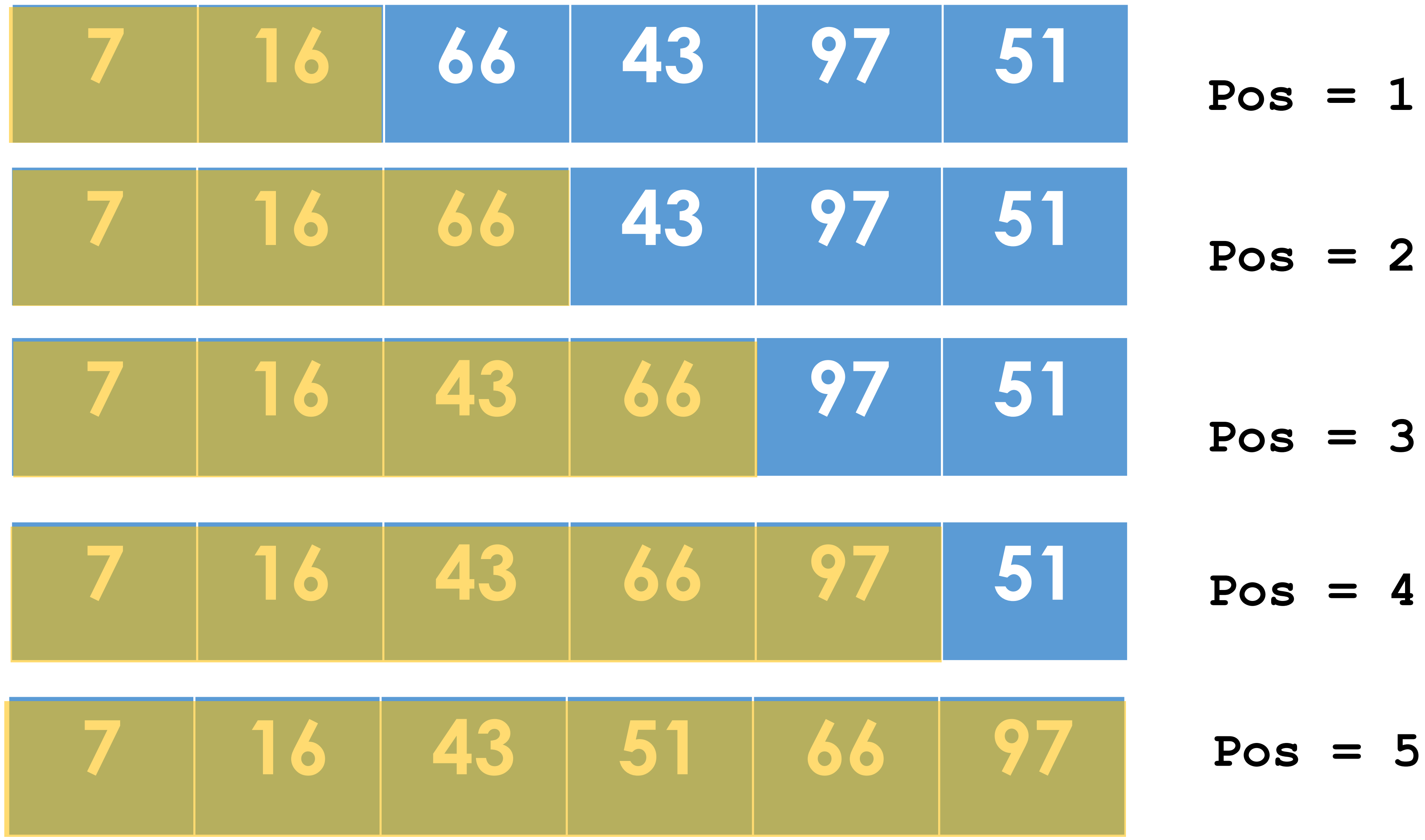


pos 5

currInd 3

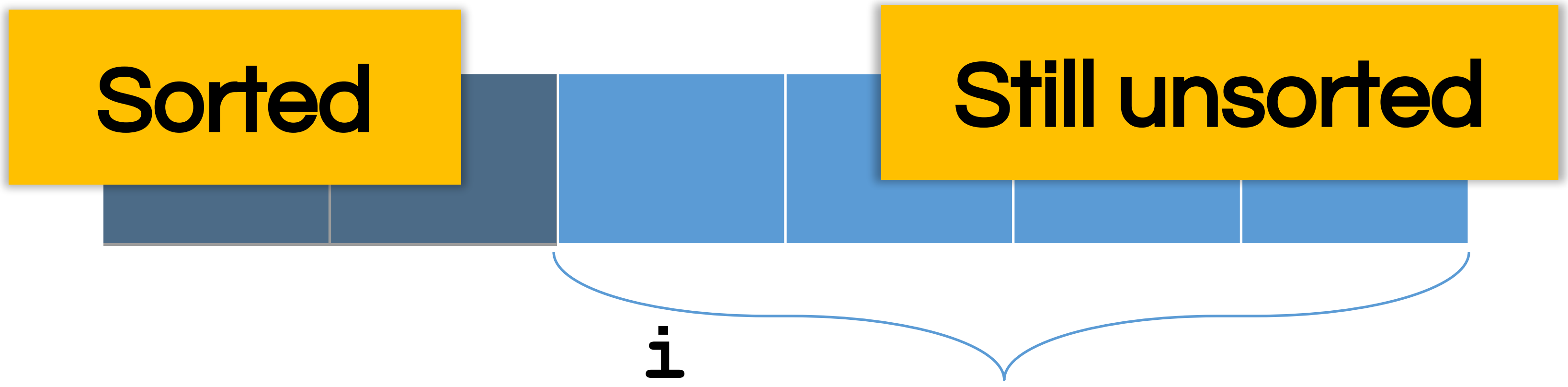
```
public static void mysterySort( int[] vals )     {
   int currInd;
   for ( int pos=1; pos < vals.length ; pos++ ) {
      currInd = pos ;
      while ( currInd > 0  &&
           vals[currInd] < vals[currInd-1] ) {
         swap(vals, currInd, currInd-1);
         currInd = currInd – 1;
      }
   }
}
```




7 16 66 43 97 51
Pos = 1
7 16 66 43 97 51
Pos = 2
7 16 43 66 97 51
Pos = 3
7 16 43 66 97 51
Pos = 4
7 16 43 51 66 97
Pos = 5

Mystery Sort: Basic Algorithm

For each **position** `i` from `1` to `length-1`

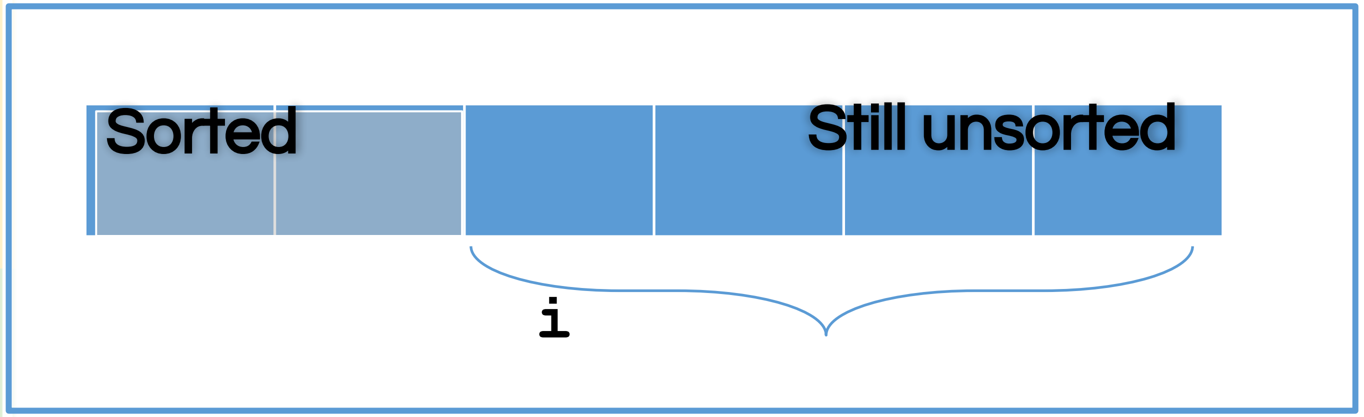


Mystery Sort: Basic Algorithm

For each **position** `i` from `1` to `length-1`

Find correct location of `i`th element relative to first `i-1`

Swap successive pairs to get there



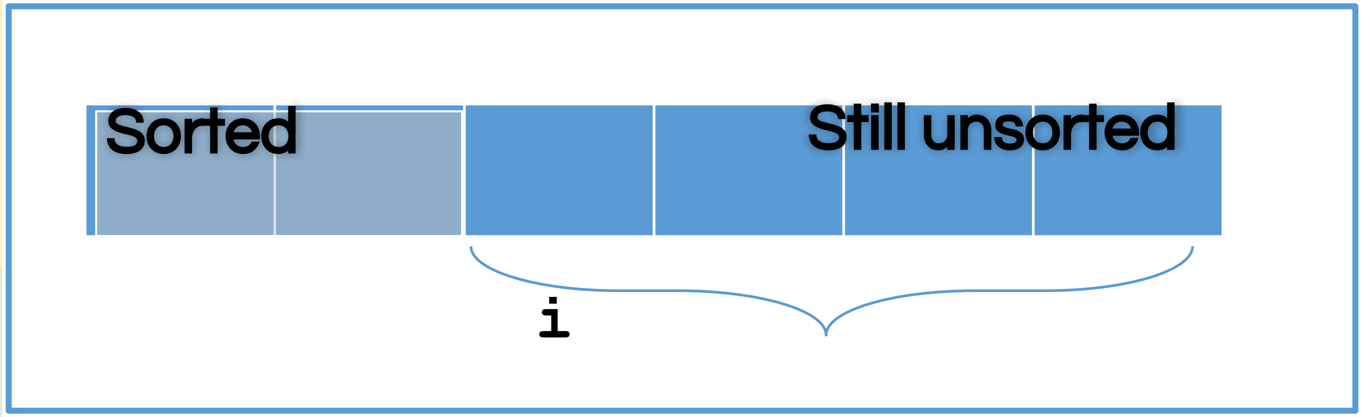
~~Mystery~~ Sort: Basic Algorithm

Insertion Sort

For each **position** `i` from `1` to `length-1`

Find correct location of `i`th element relative to first `i-1`

Swap successive pairs to get there



Thought questions

- How do we know this algorithm works?
- Are there other approaches?
- Can we do better?