Sorting Techniques

- Bubble sort
- Insertion sort
- Selection sort

Sorting Algorithm

- Sorting takes an unordered collection and makes it an ordered one.

1	2	3	4	5	6
77	42	35	12	101	5

↓

1	2	3	4	5	6
5	12	35	42	77	101

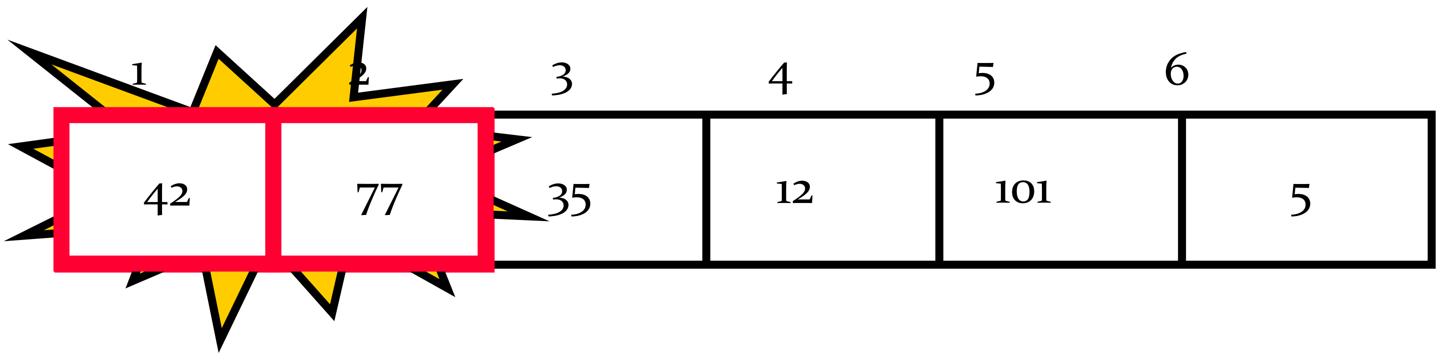
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**

1	2	3	4	5	6
77	42	35	12	101	5

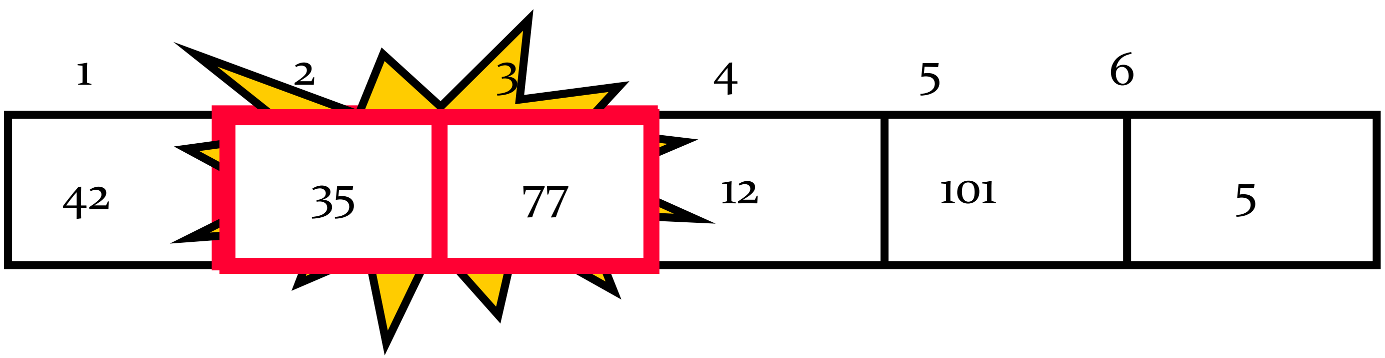
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



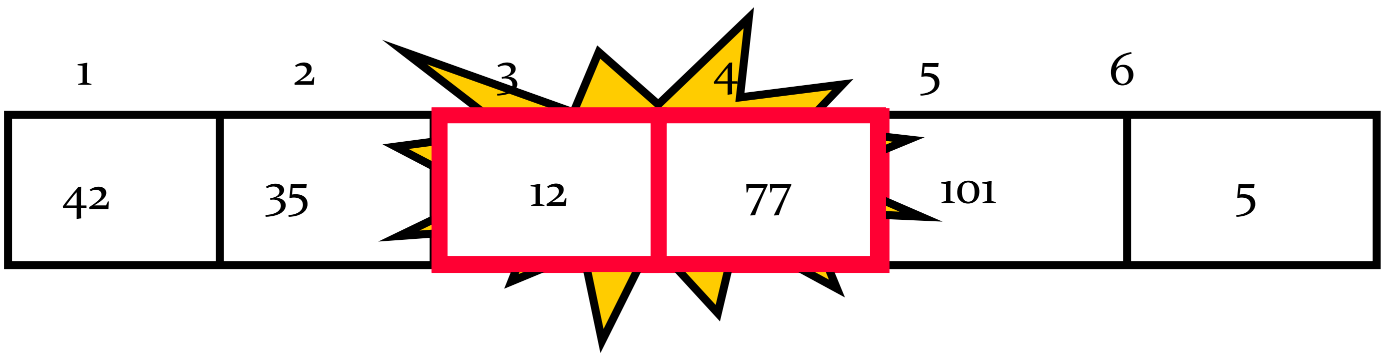
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



"Bubbling Up" the Largest Element

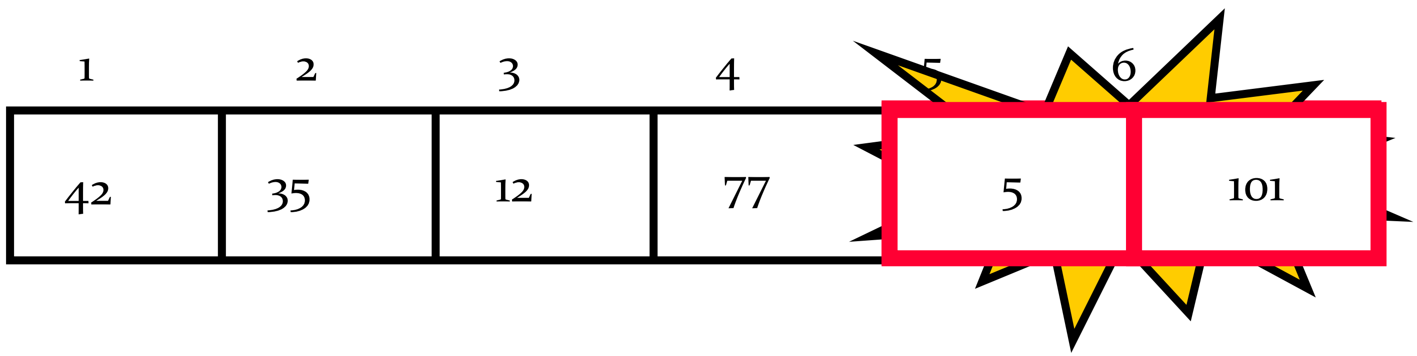
- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**

1	2	3	4	5	6
42	35	12	77	101	5

No need to swap

"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**

1	2	3	4	5	6
42	35	12	77	5	101

Largest value correctly placed

Repeat "Bubble Up" How Many Times?

- If we have N elements…
- And if each time we bubble an element, we place it in its correct location…
- Then we repeat the "bubble up" process N – 1 times.
- This guarantees we'll correctly place all N elements.

“Bubbling” All the Elements

N - 1

1	2	3	4	5	6
42	35	12	77	5	101
35	12	42	5	77	101
12	35	5	42	77	101
12	5	35	42	77	101
5	12	35	42	77	101

Bubble Sort

Algorithm

for i ← 1 to n-1 do

    for j ← 1 to n-i do

        if (A[j+1] < A[j]) swap A[j] and A[j+1] ;

    }

}

Analysis:

In general, if the list has n elements, we will have to do

$(n-1) + (n-2) \dots + 2 + 1 = (n-1)\, n / 2$ comparisons.

$= O(n^2)$

Insertion Sort

INSERTION_SORT (A, N)

1. Set A[0] = $-\infty$.
2. Repeat Step 3 to 5 for K = 2, 3, …, N:
3. Set TEMP = A[K] and PTR = K – 1.
4. Repeat while TEMP < A[PTR]:
   (a) Set A[PTR+1] = A[PTR]
   (b) Set PTR = PTR – 1.
   [End of Loop.]
5. Set A[PTR+1] = TEMP.
   [End of Loop 2.]
6. Return.

CODERINDEED

Insertion Sort Example

- Sort: 34 8 64 51 32 21
- 34 8 64 51 32 21
  - The algorithm sees that 8 is smaller than 34 so it swaps.
- 8 34 64 51 32 21
  - 51 is smaller than 64, so they swap.
- 8 34 51 64 32 21
- 8 34 51 64 32 21 (from previous slide)
  - The algorithm sees 32 as another smaller number and moves it to its appropriate location between 8 and 34.
- 8 32 34 51 64 21
  - The algorithm sees 21 as another smaller number and moves into between 8 and 32.
- Final sorted numbers:
- 8 21 32 34 51 64

CODERINDEED

Insertion Sort Complexity

- This Sorting algorithm is frequently used when n is very small.

- Worst case occurs when array is in reverse order. The inner loop must use K – 1 comparisons.

$$
\begin{aligned}
f(n) &= 1 + 2 + 3 + \dots + (n - 1) \\
&= n(n - 1)/2 \\
&= O(n^2)
\end{aligned}
$$

- In average case, there will be approximately (K – 1)/2 comparisons in the inner loop.

$$
\begin{aligned}
f(n) &= (1 + 2 + 3 + \dots + (n - 1))/2 \\
&= n(n - 1)/4 \\
&= O(n^2)
\end{aligned}
$$

Selection Sort

This algorithm sorts an array A with N elements.

SELECTION(A, N)

1. Repeat steps 2 and 3 for k=1 to N-1:
2. Call MIN(A, K, N, LOC).
3. [Interchange A[k] and A[LOC]]
   Set Temp:= A[k], A[k]:= A[LOC] and A[LOC]:=Temp.
   [End of step 1 Loop.]
4. Exit.

MIN(A, K, N, LOC).

1. Set MIN := A[K] and LOC:= K.
2. Repeat for j=k+1 to N:
   If Min>A[j], then: Set Min:= A[j] and LOC:=J.
   [End of if structure]
3. Return.

CODERINDEED

Selection Sort Example

8	4	6	9	2	3	1

1	4	6	9	2	3	8

1	2	6	9	4	3	8

1	2	3	9	4	6	8

1	2	3	4	9	6	8

1	2	3	4	6	9	8

1	2	3	4	6	8	9

1	2	3	4	6	8	9

Selection Sort Complexity

The number f(n) of comparisons in selection sort algorithm is independent of original order of elements. There are n-1 comparisons during pass 1 to find the smallest element, n-2 comparisons during pass 2 to find the second smallest element, and so on.

Accordingly,

$$
\begin{aligned}
f(n) &= (n-1)+(n-2)+\text{-----}+2+1 \\
&= n(n-1)/2 \\
&= O(n^2)
\end{aligned}
$$

The f (n) holds the same value $O(n^2)$ both for worst case and average case.

Comparing the Algorithms

	Best Case	Average Case	Worst Case
• Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$
• Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
• Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$

Thank You