

Sorting Algorithms

Selection Sort

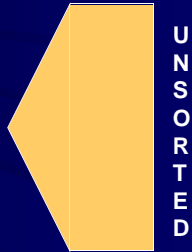
values [0]	36
[1]	24
[2]	10
[3]	6
[4]	12

Divides the array into two parts: already sorted, and not yet sorted.

On each pass, finds the smallest of the unsorted elements, and swap it into its correct place, thereby increasing the number of sorted elements by one.

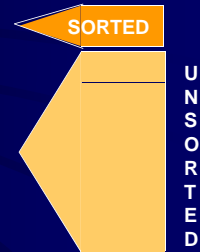
Selection Sort: Pass One

values [0]	36
[1]	24
[2]	10
[3]	6
[4]	12



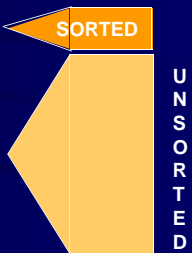
Selection Sort: End Pass One

values [0]	6
[1]	24
[2]	10
[3]	36
[4]	12



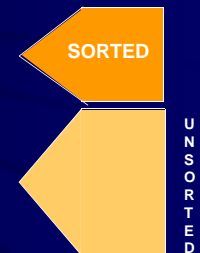
Selection Sort: Pass Two

values [0]	6
[1]	24
[2]	10
[3]	36
[4]	12

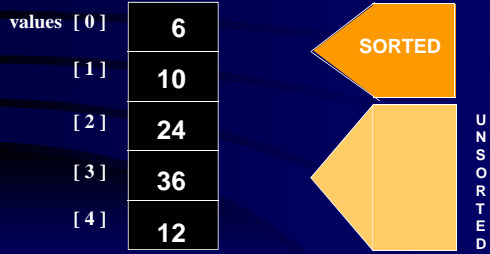


Selection Sort: End Pass Two

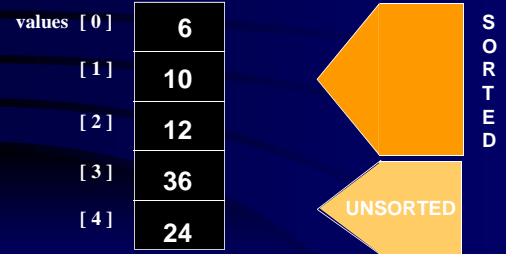
values [0]	6
[1]	10
[2]	24
[3]	36
[4]	12



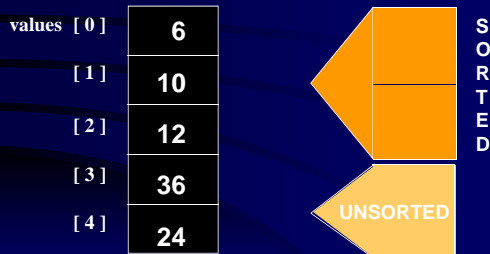
Selection Sort: Pass Three



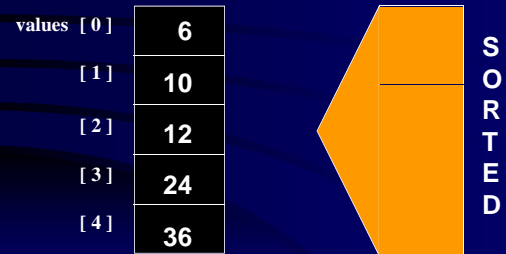
Selection Sort: End Pass Three



Selection Sort: Pass Four



Selection Sort: End Pass Four



Code for Selection Sort

```
void SelectionSort (int[] values)
// Post: Sorts array values[0 . . numValues-1 ]
// into ascending order by key
{
    int endIndex = values.Length - 1 ;
    for (int current=0;current<endIndex;current++)
        Swap (values, current,
            MinIndex(values,current,endIndex));
}
```

Selection Sort code (contd)

```
int MinIndex(int[] values, int start, int end)
// Post: Function value = index of the smallest value
// in values [start] . . values [end].
{
    int indexOfMin = start ;
    for(int index =start + 1 ; index <= end ; index++)
        if (values[ index] < values [indexOfMin])
            indexOfMin = index ;
    return indexOfMin;
}
```

Bubble Sort

values [0]	36
[1]	24
[2]	10
[3]	6
[4]	12

Compares neighboring pairs of array elements, starting with the last array element, and swaps neighbors whenever they are not in correct order.

On each pass, this causes the smallest element to "bubble up" to its correct place in the array.

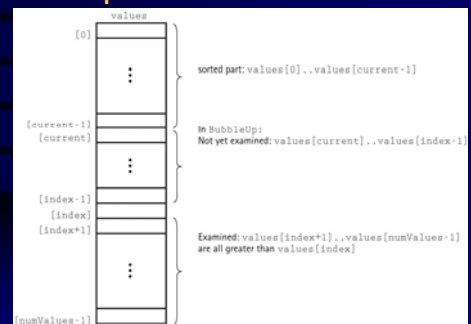
Bubble Sort: Pass One

36	36	36	6
24	24	6	36
10	6	24	24
6	10	10	10
12	12	12	12

Bubble Sort: Pass Two

6	6	6
36	36	10
24	10	36
10	24	24
12	12	12

Snapshot of BubbleSort



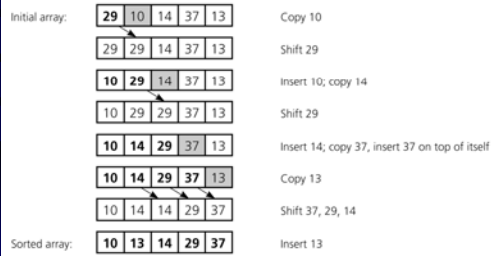
Code for Bubble Sort

```
void BubbleSort(int[] values)
{
    int current = 0;
    while (current < values.Length - 1)
    {
        BubbleUp(values, current, values.Length-1);
        current++;
    }
}
```

Bubble Sort code (contd.)

```
void BubbleUp(int[] values,
              int startIndex, int endIndex)
// Post: Adjacent pairs that are out of
// order have been switched between
// values[startIndex]..values[endIndex]
// beginning at values[endIndex].
{
    for (int index = endIndex;
         index > startIndex; index--)
        if (values[index] < values[index-1])
            Swap(values, index, index-1);
}
```


Insertion Sort



Code for Insertion Sort

```
void InsertionSort(int[] values)
// Post: Sorts array values[0 . . numValues-1] into
// ascending order by key
{
    for (int curSort=0;curSort<values.Length;curSort++)
        InsertItem ( values , 0 , curSort );
}
```

Insertion Sort code (contd.)

```
void InsertItem (int[] values, int start, int end )
{
    bool finished = false ;
    int current = end ;
    bool moreToSearch = ( current != start );

    while (moreToSearch && !finished ) {
        if (values[current] < values[current - 1]){
            Swap(values[current], values[current - 1]);
            current--;
            moreToSearch = ( current != start );
        }
        else
            finished = true ;
    }
}
```