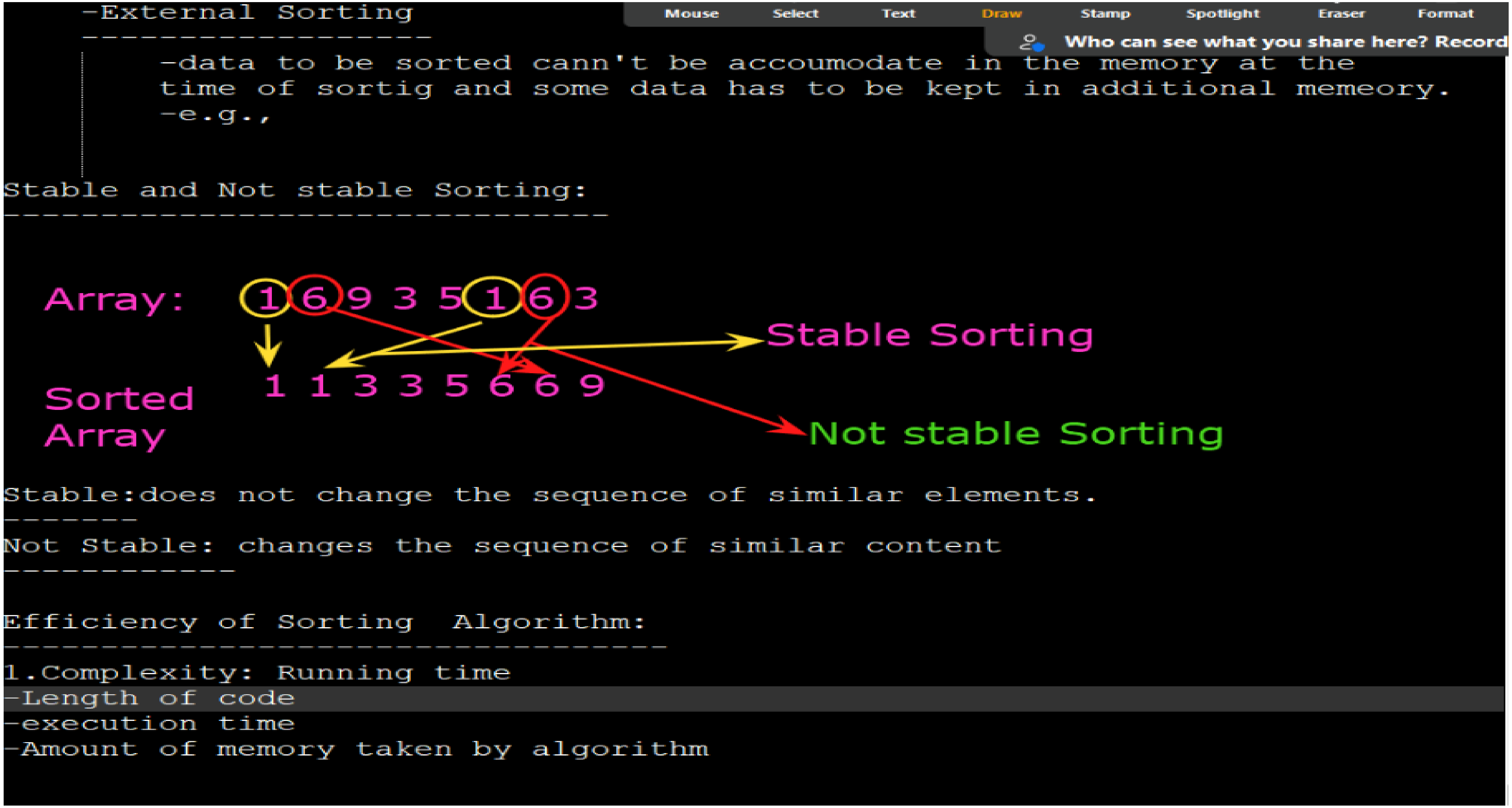
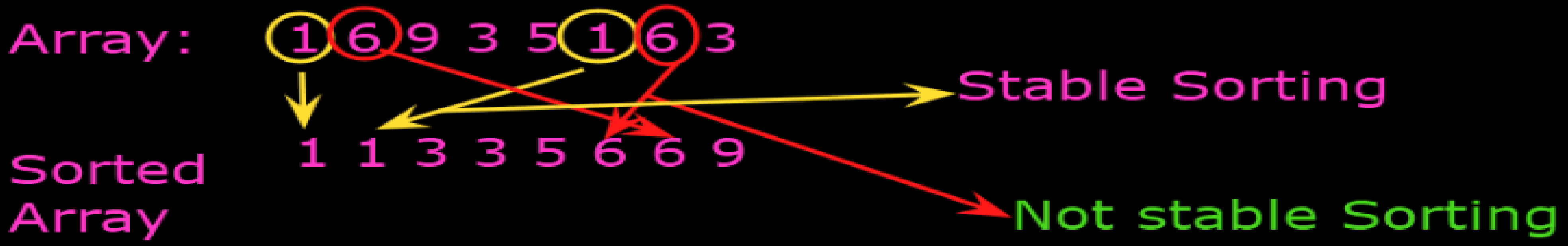
Algorithms & Data Structure

Kiran Waghmare

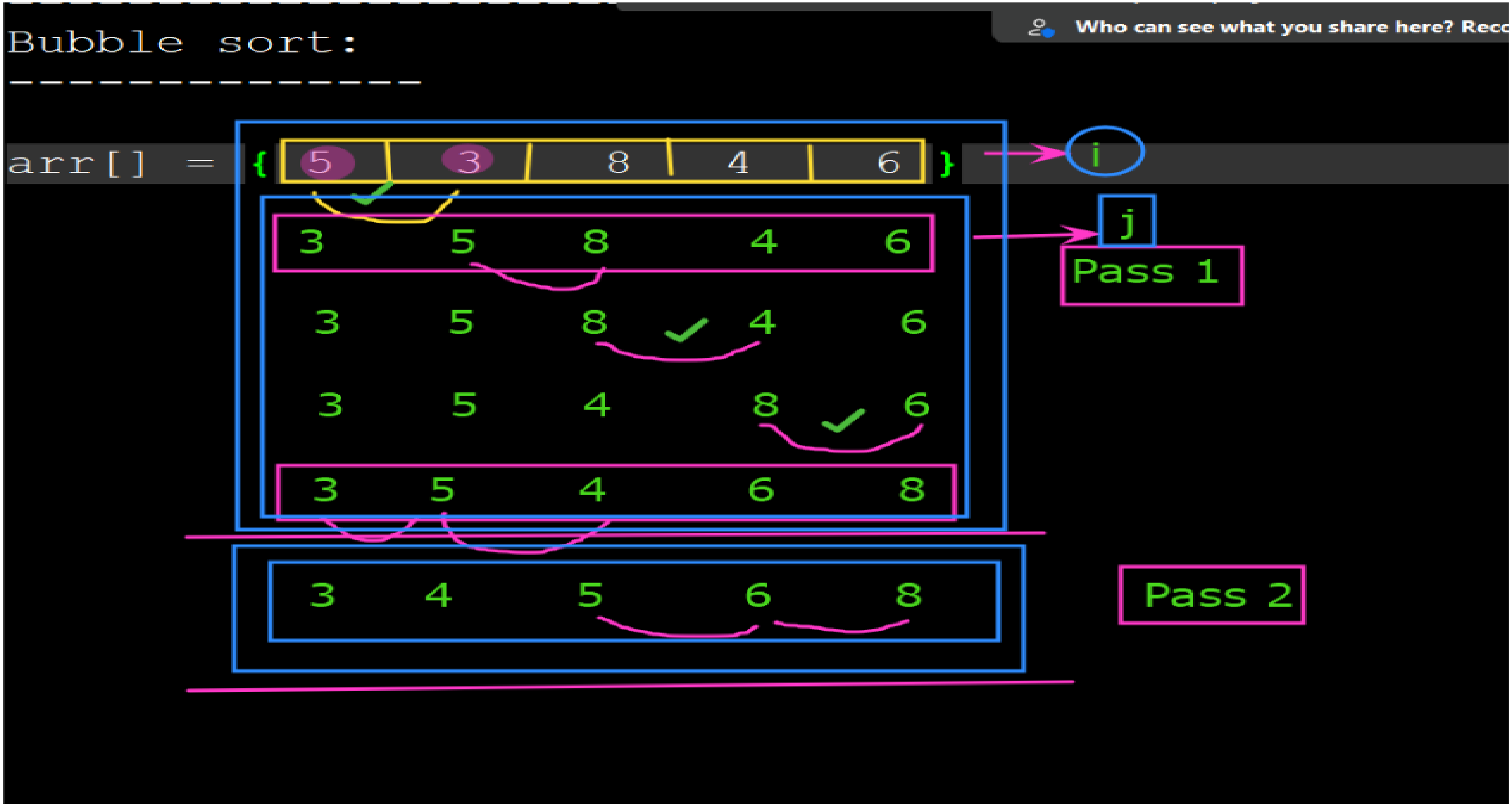


Who can see what you share here? Record

```
    -External Sorting
    -------------------
        -data to be sorted cann't be accoumodate in the memory at the
        time of sortig and some data has to be kept in additional memeory.
        -e.g.,


Stable and Not stable Sorting:
--------------------------------
```



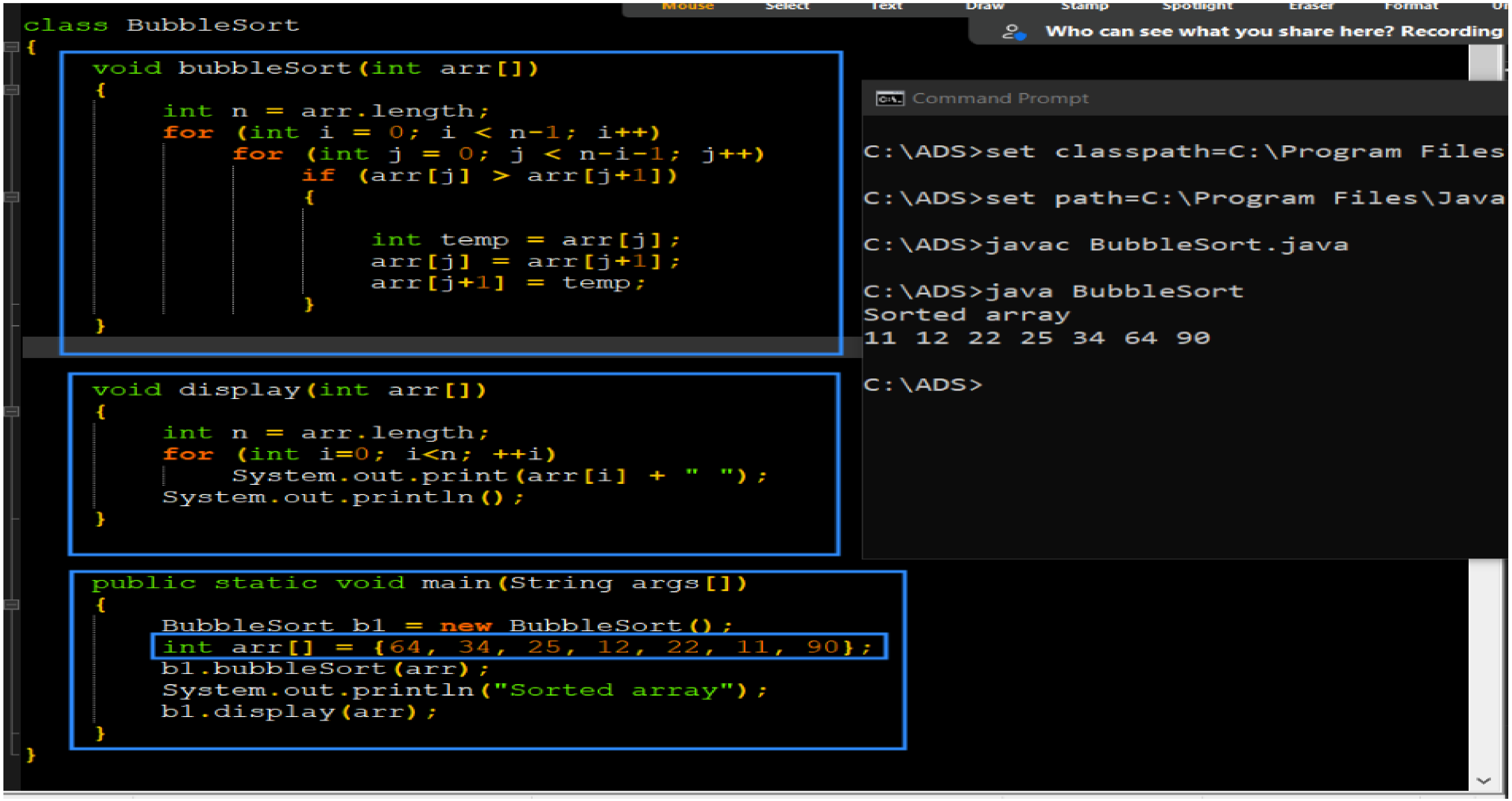
```
Stable:does not change the sequence of similar elements.
-------
Not Stable: changes the sequence of similar content
------------

Efficiency of Sorting  Algorithm:
-----------------------------------
1.Complexity: Running time
-Length of code
-execution time
-Amount of memory taken by algorithm
```



Who can see what you share here? Rec

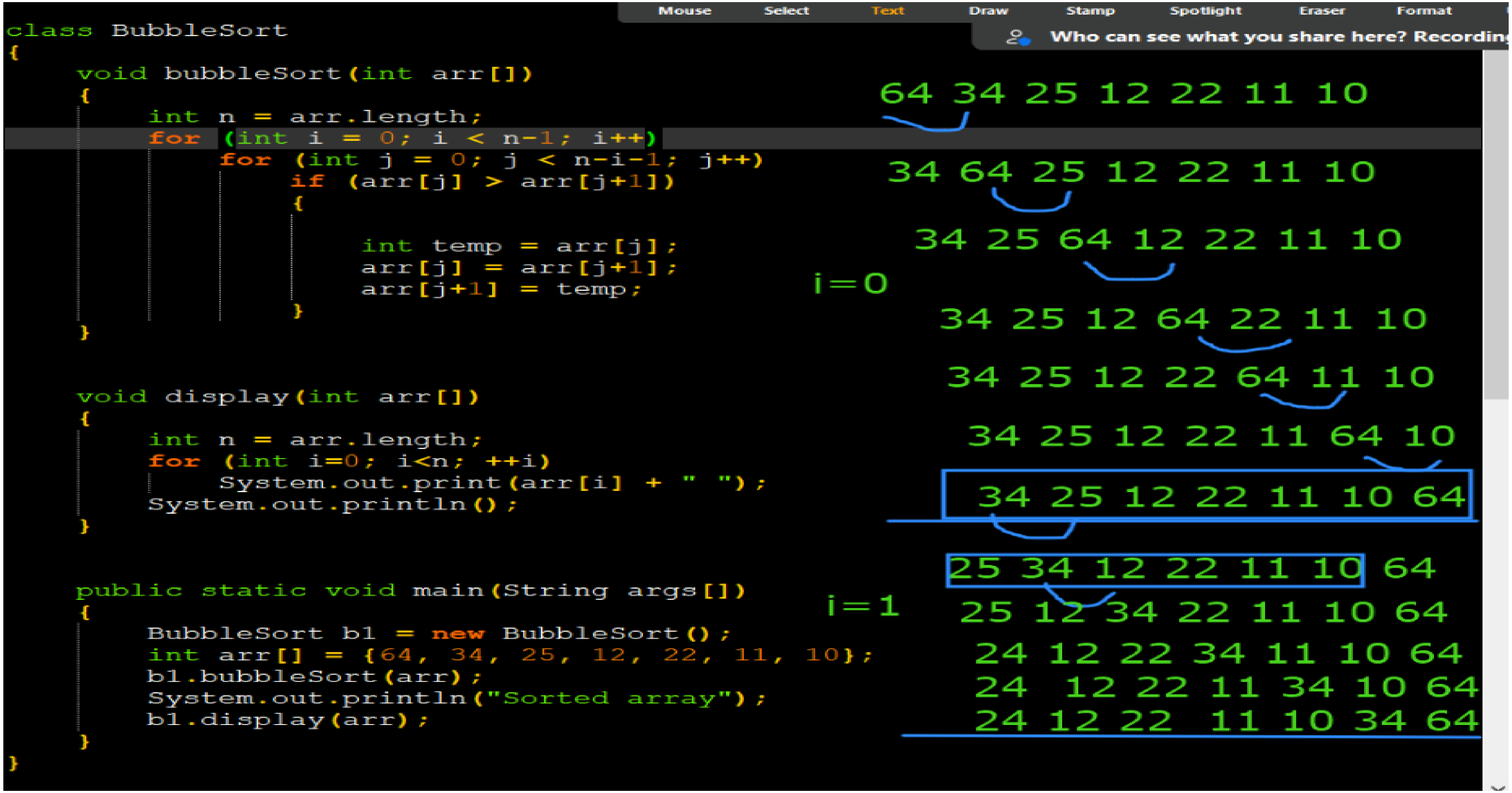
Bubble sort:
----

arr[] = { 5 3 8 4 6 }



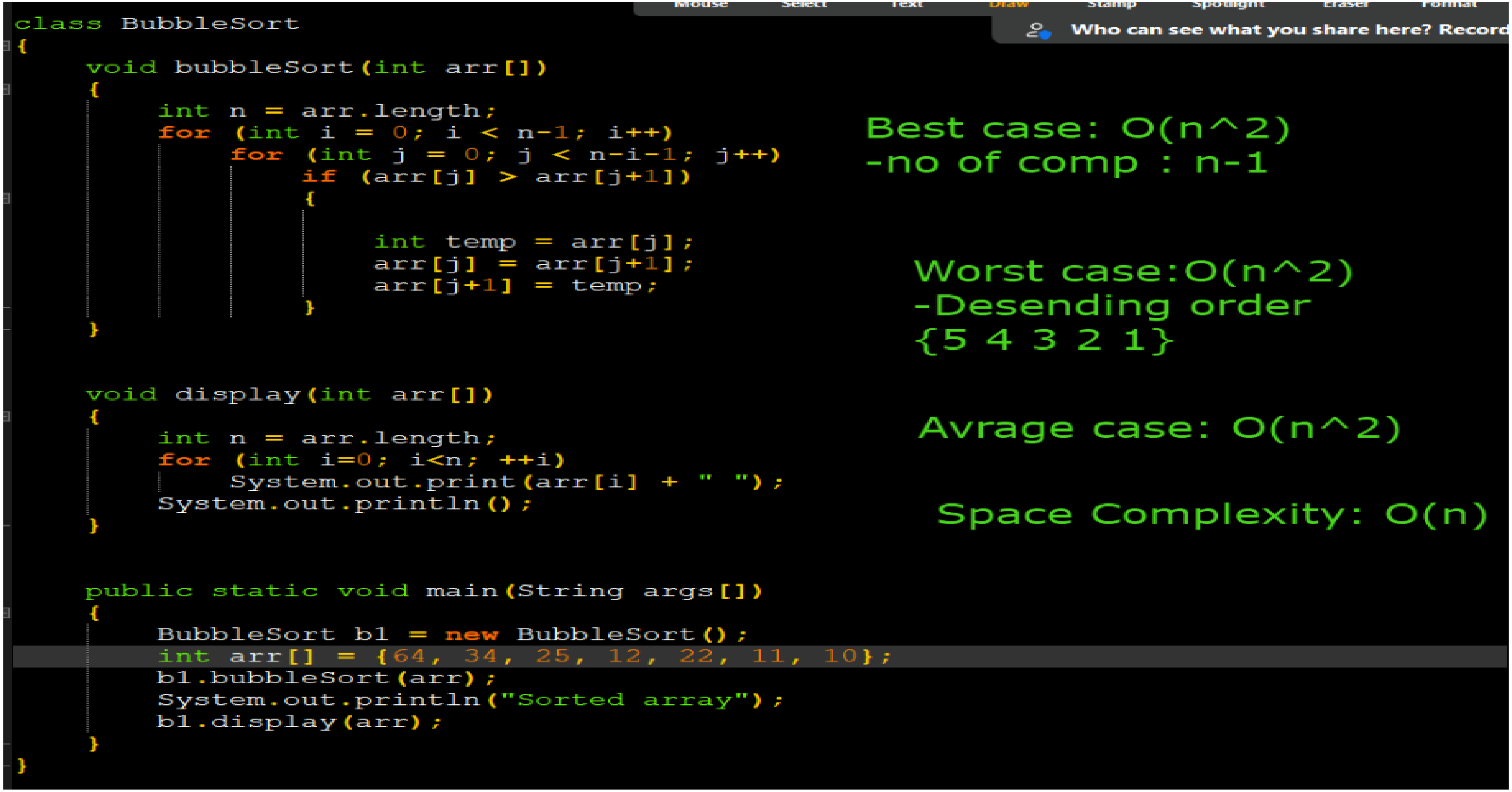
```java
class BubbleSort
{
    void bubbleSort(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (arr[j] > arr[j+1])
                {

                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
    }

    void display(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    public static void main(String args[])
    {
        BubbleSort b1 = new BubbleSort();
        int arr[] = {64, 34, 25, 12, 22, 11, 90};
        b1.bubbleSort(arr);
        System.out.println("Sorted array");
        b1.display(arr);
    }
}
```

```
C:\ADS>set classpath=C:\Program Files

C:\ADS>set path=C:\Program Files\Java

C:\ADS>javac BubbleSort.java

C:\ADS>java BubbleSort
Sorted array
11 12 22 25 34 64 90

C:\ADS>
```



```
class BubbleSort
{
    void bubbleSort(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (arr[j] > arr[j+1])
                {

                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
    }


    void display(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }


    public static void main(String args[])
    {
        BubbleSort b1 = new BubbleSort();
        int arr[] = {64, 34, 25, 12, 22, 11, 10};
        b1.bubbleSort(arr);
        System.out.println("Sorted array");
        b1.display(arr);
    }
}
```

```
      64 34 25 12 22 11 10
      34 64 25 12 22 11 10
      34 25 64 12 22 11 10
i=0
      34 25 12 64 22 11 10
      34 25 12 22 64 11 10
      34 25 12 22 11 64 10
      34 25 12 22 11 10 64

      25 34 12 22 11 10 64
i=1   25 12 34 22 11 10 64
      24 12 22 34 11 10 64
      24 12 22 11 34 10 64
      24 12 22 11 10 34 64
```



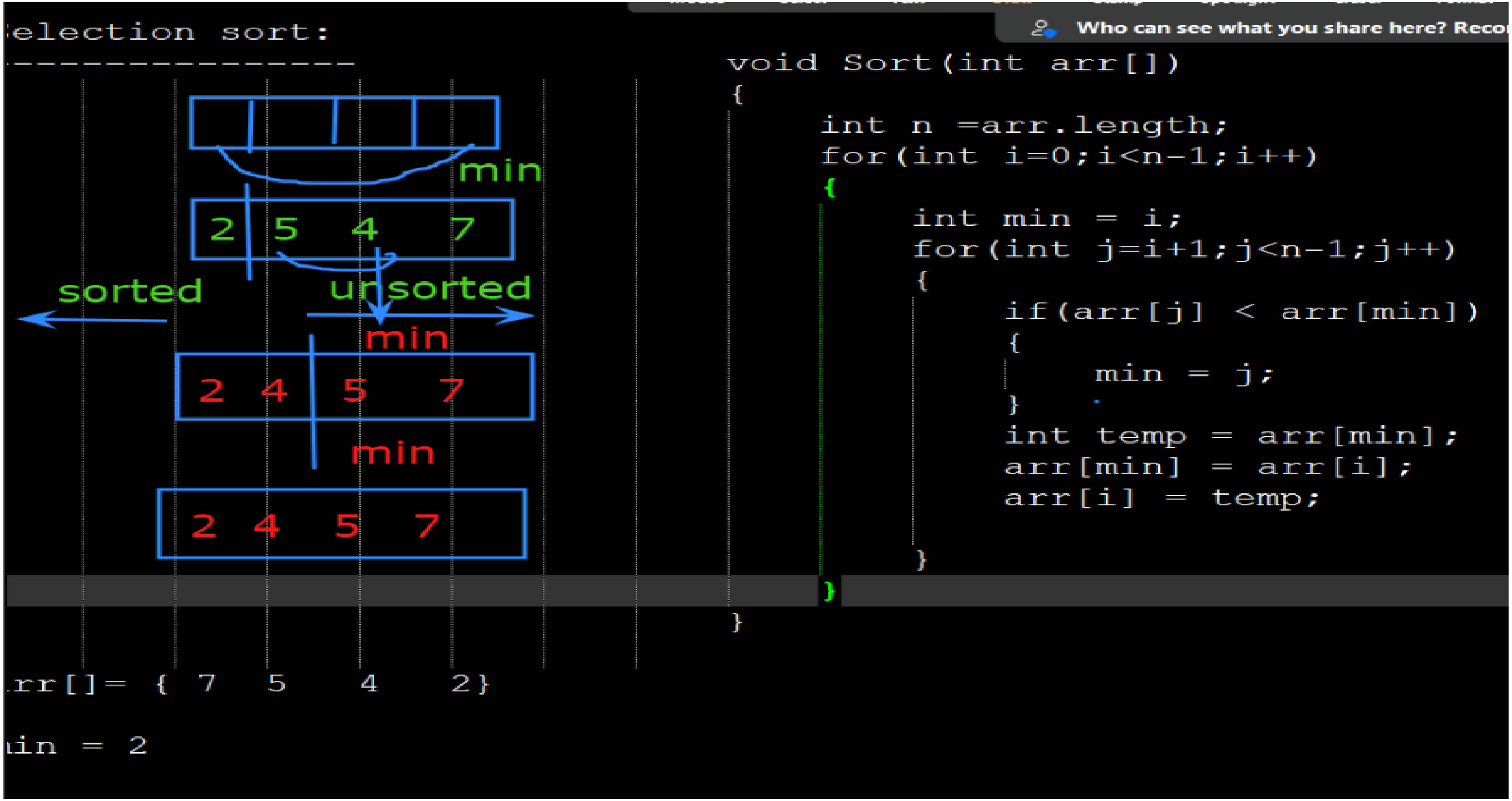
```
class BubbleSort
{
	void bubbleSort(int arr[])
	{
		int n = arr.length;
		for (int i = 0; i < n-1; i++)
			for (int j = 0; j < n-i-1; j++)
				if (arr[j] > arr[j+1])
				{
					
					int temp = arr[j];
					arr[j] = arr[j+1];
					arr[j+1] = temp;
				}
	}

	void display(int arr[])
	{
		int n = arr.length;
		for (int i=0; i<n; ++i)
			System.out.print(arr[i] + " ");
		System.out.println();
	}


	public static void main(String args[])
	{
		BubbleSort b1 = new BubbleSort();
		int arr[] = {64, 34, 25, 12, 22, 11, 10};
		b1.bubbleSort(arr);
		System.out.println("Sorted array");
		b1.display(arr);
	}
}
```

Best case: O(n^2)
-no of comp : n-1

Worst case:O(n^2)
-Desending order
{5 4 3 2 1}

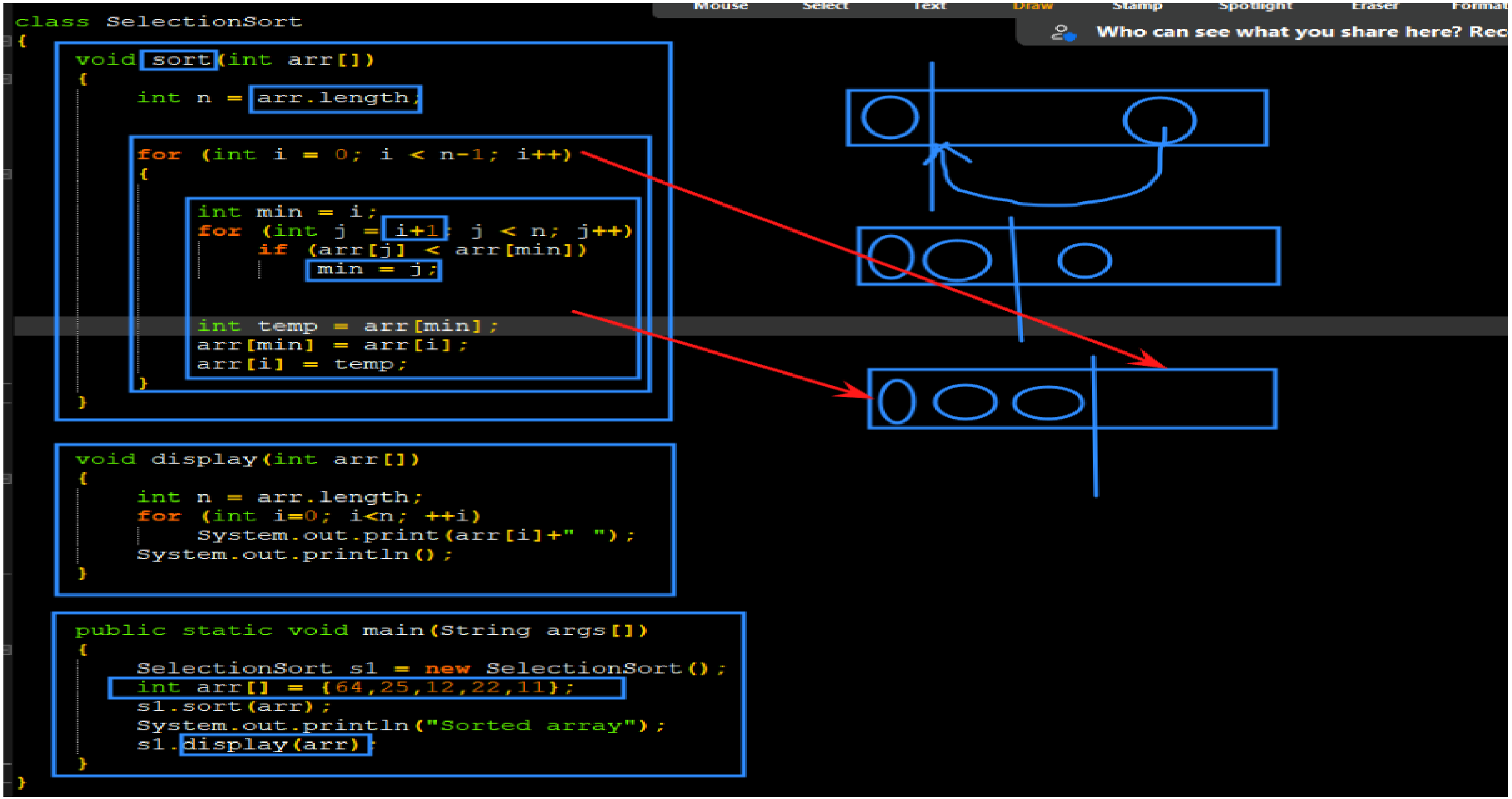
Avrage case: O(n^2)

Space Complexity: O(n)

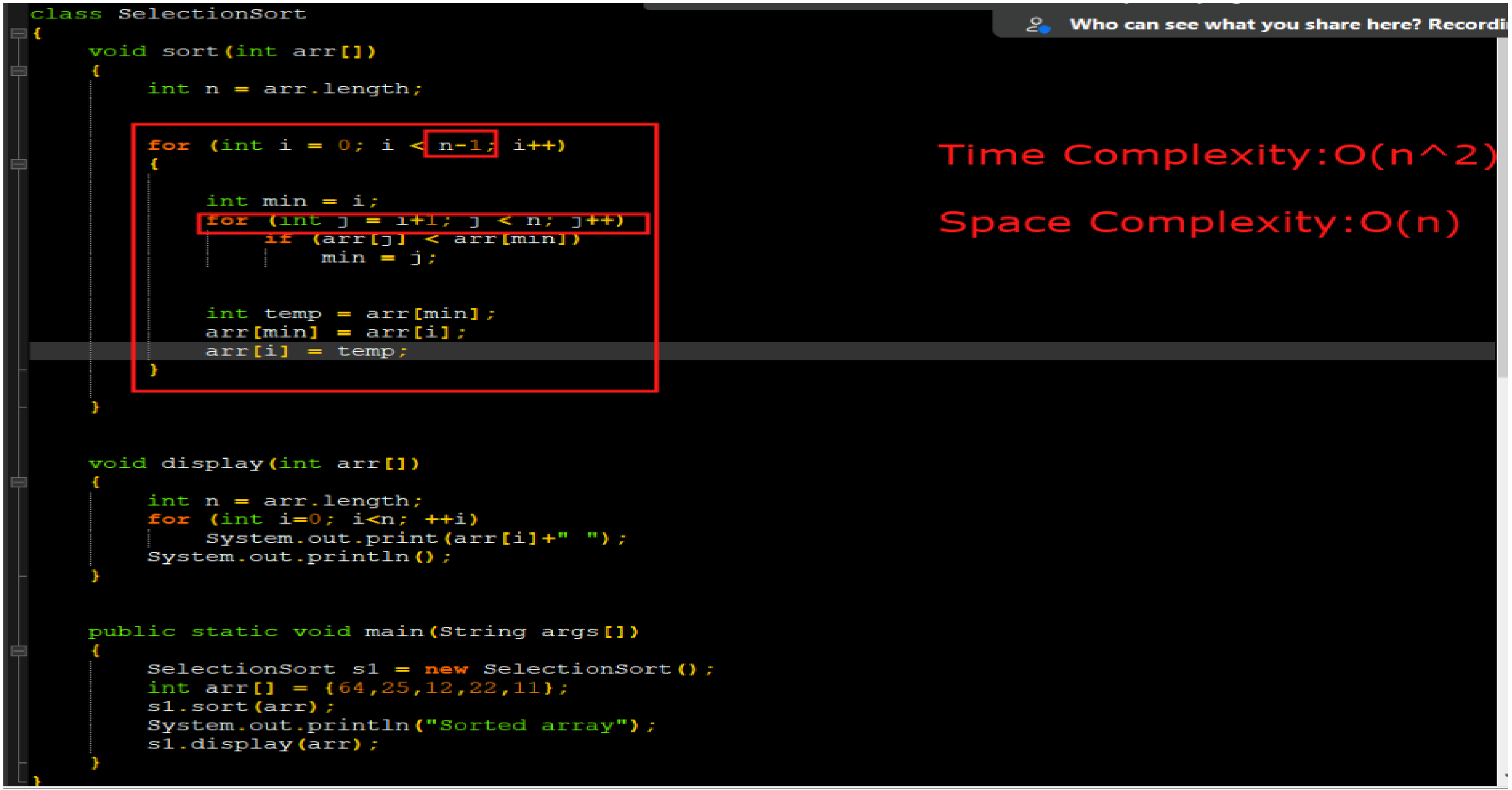


```
election sort:
------------------

rr[]= { 7  5  4  2}

in = 2
```

```
void Sort(int arr[])
{
        int n =arr.length;
        for(int i=0;i<n-1;i++)
        {
                int min = i;
                for(int j=i+1;j<n-1;j++)
                {
                        if(arr[j] < arr[min])
                        {
                                min = j;
                        }
                        int temp = arr[min];
                        arr[min] = arr[i];
                        arr[i] = temp;

                }
        }
}
```



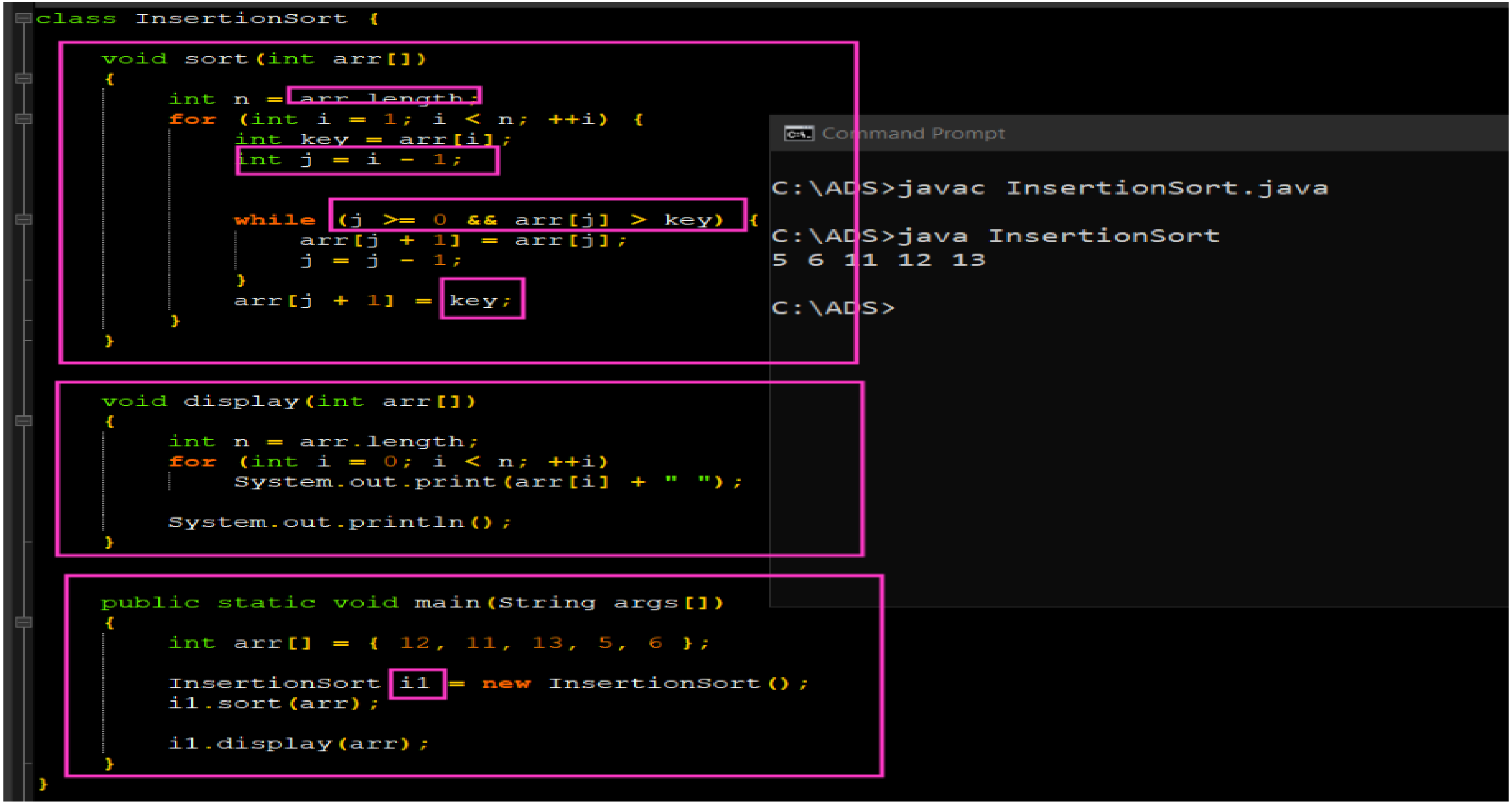
```java
class SelectionSort
{
    void sort(int arr[])
    {
        int n = arr.length;

        for (int i = 0; i < n-1; i++)
        {
            int min = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[min])
                    min = j;

            int temp = arr[min];
            arr[min] = arr[i];
            arr[i] = temp;
        }
    }

    void display(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i]+" ");
        System.out.println();
    }

    public static void main(String args[])
    {
        SelectionSort s1 = new SelectionSort();
        int arr[] = {64,25,12,22,11};
        s1.sort(arr);
        System.out.println("Sorted array");
        s1.display(arr);
    }
}
```



```
class SelectionSort
{
    void sort(int arr[])
    {
        int n = arr.length;

        for (int i = 0; i < n-1; i++)
        {
            int min = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[min])
                    min = j;

            int temp = arr[min];
            arr[min] = arr[i];
            arr[i] = temp;
        }
    }

    void display(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i]+" ");
        System.out.println();
    }

    public static void main(String args[])
    {
        SelectionSort s1 = new SelectionSort();
        int arr[] = {64,25,12,22,11};
        s1.sort(arr);
        System.out.println("Sorted array");
        s1.display(arr);
    }
}
```

Time Complexity:O(n^2)

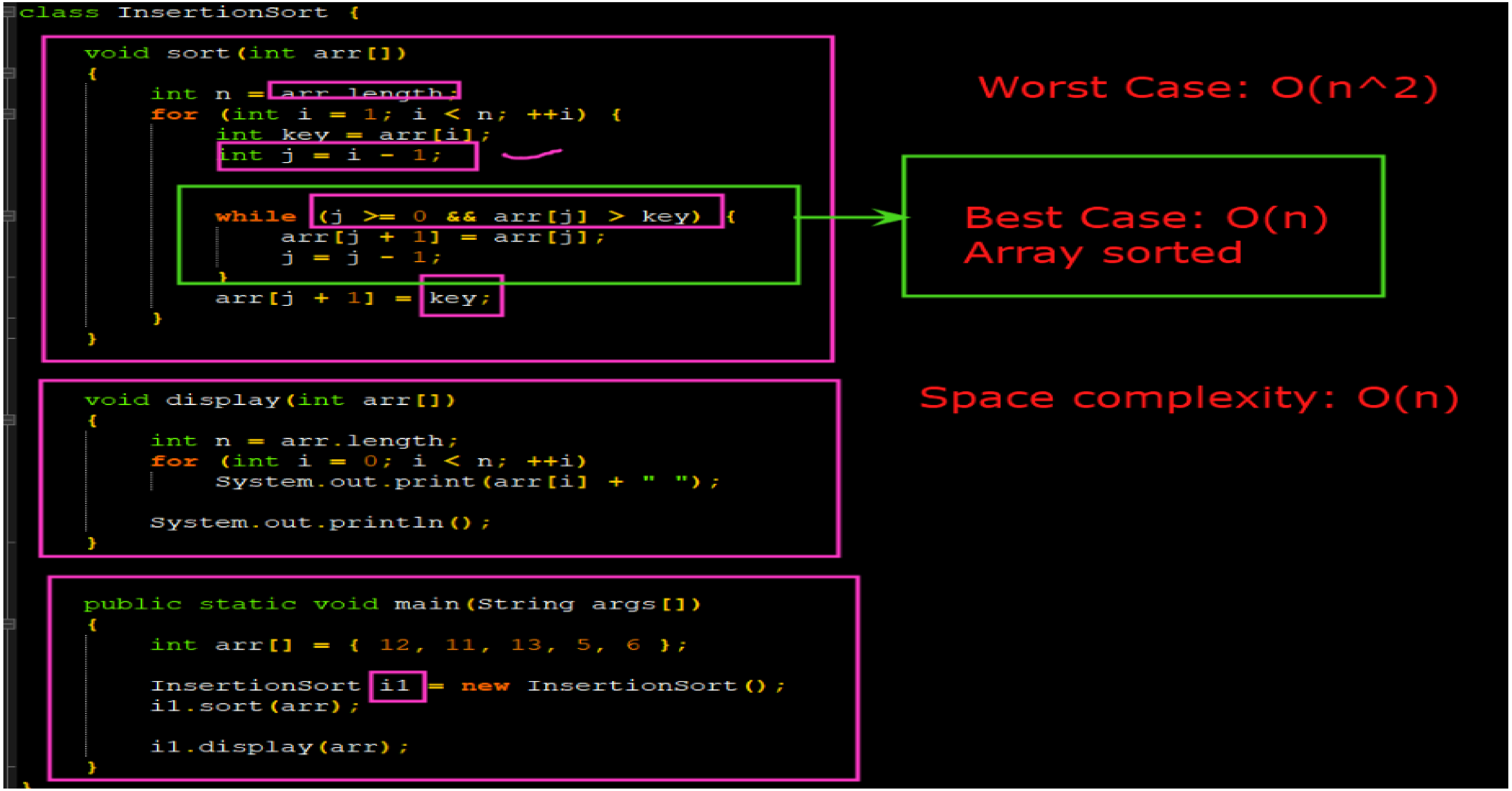
Space Complexity:O(n)



```
class InsertionSort {
    void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;

            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    void display(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");

        System.out.println();
    }

    public static void main(String args[])
    {
        int arr[] = { 12, 11, 13, 5, 6 };

        InsertionSort i1 = new InsertionSort();
        i1.sort(arr);

        i1.display(arr);
    }
}
```

Command Prompt

```
C:\ADS>javac InsertionSort.java

C:\ADS>java InsertionSort
5 6 11 12 13

C:\ADS>
```

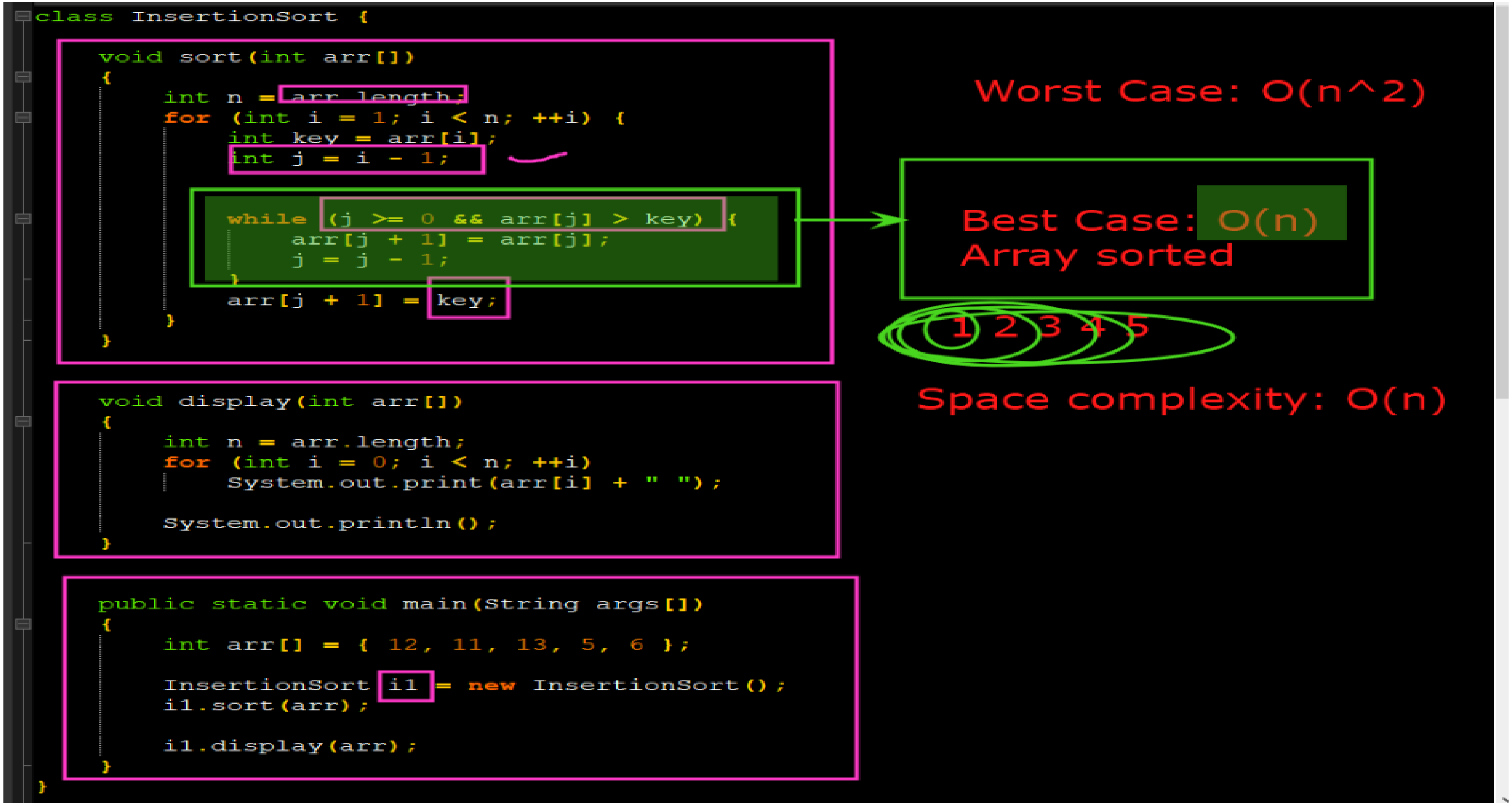


```java
class InsertionSort {
    void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;

            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    void display(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");

        System.out.println();
    }

    public static void main(String args[])
    {
        int arr[] = { 12, 11, 13, 5, 6 };

        InsertionSort i1 = new InsertionSort();
        i1.sort(arr);

        i1.display(arr);
    }
}
```

Worst Case: O(n^2)

Best Case: O(n)
Array sorted

Space complexity: O(n)



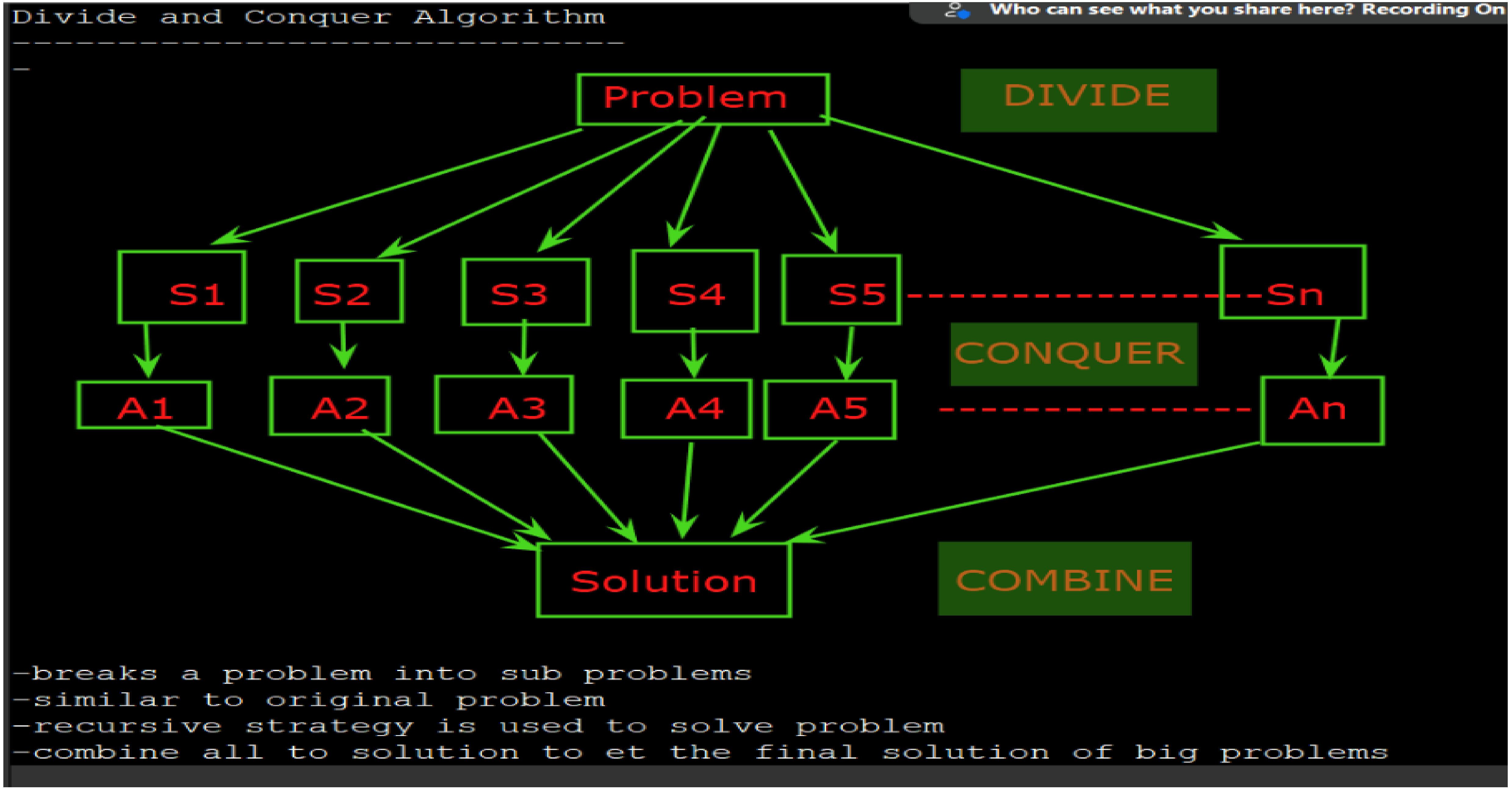
```
class InsertionSort {
    void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;

            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    void display(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");

        System.out.println();
    }

    public static void main(String args[])
    {
        int arr[] = { 12, 11, 13, 5, 6 };

        InsertionSort i1 = new InsertionSort();
        i1.sort(arr);

        i1.display(arr);
    }
}
```

Worst Case: O(n^2)

Best Case: O(n)
Array sorted

1 2 3 4 5

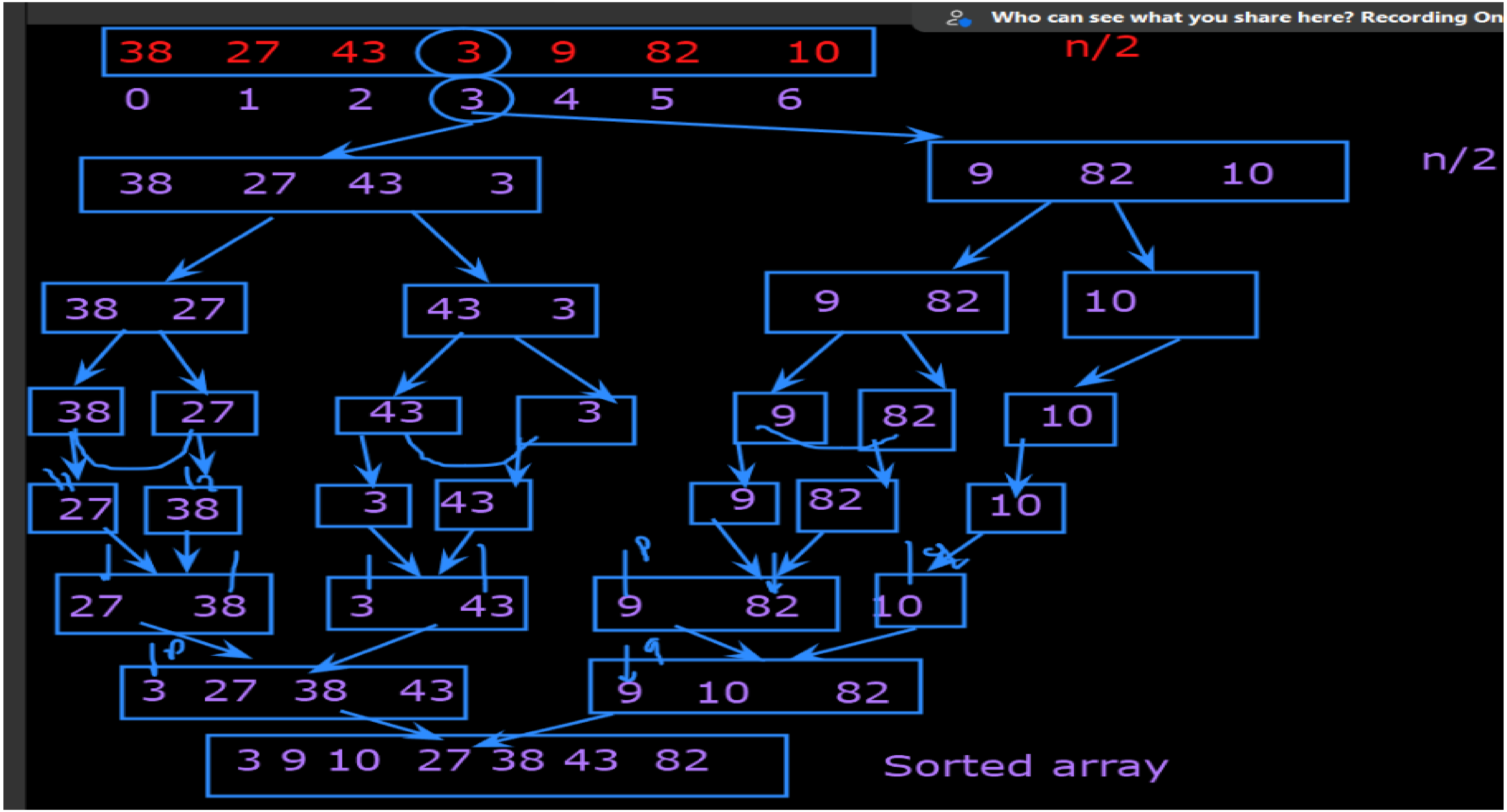
Space complexity: O(n)



Divide and Conquer Algorithm

-

-breaks a problem into sub problems
-similar to original problem
-recursive strategy is used to solve problem
-combine all to solution to et the final solution of big problems


Who can see what you share here? Recording On
38 27 43 3 9 82 10
0 1 2 3 4 5 6
n/2
38 27 43 3
9 82 10
n/2
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27
38
3
43
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82
Sorted array