Bin Sort

A quirky sorting algorithm

Declare the bin array.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
0	0	0	0

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
0	0	0	0

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

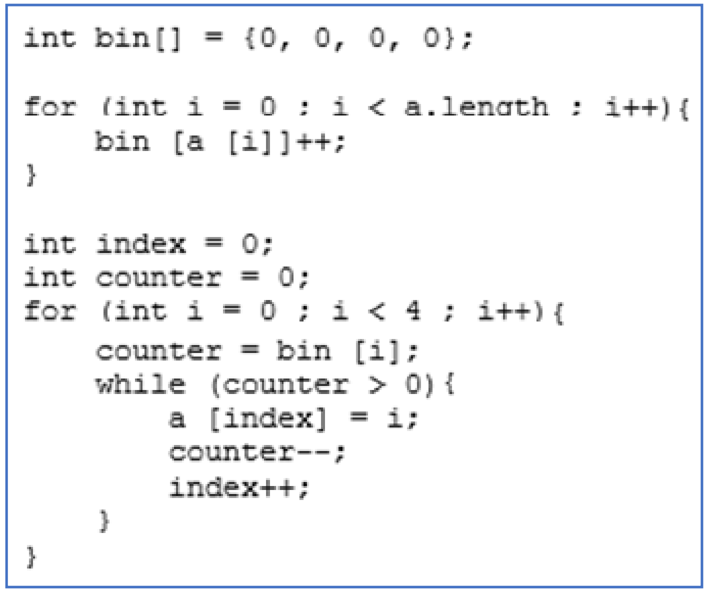
The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
0	1	0	0

Go through the array and count the items.



```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
0	2	0	0

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

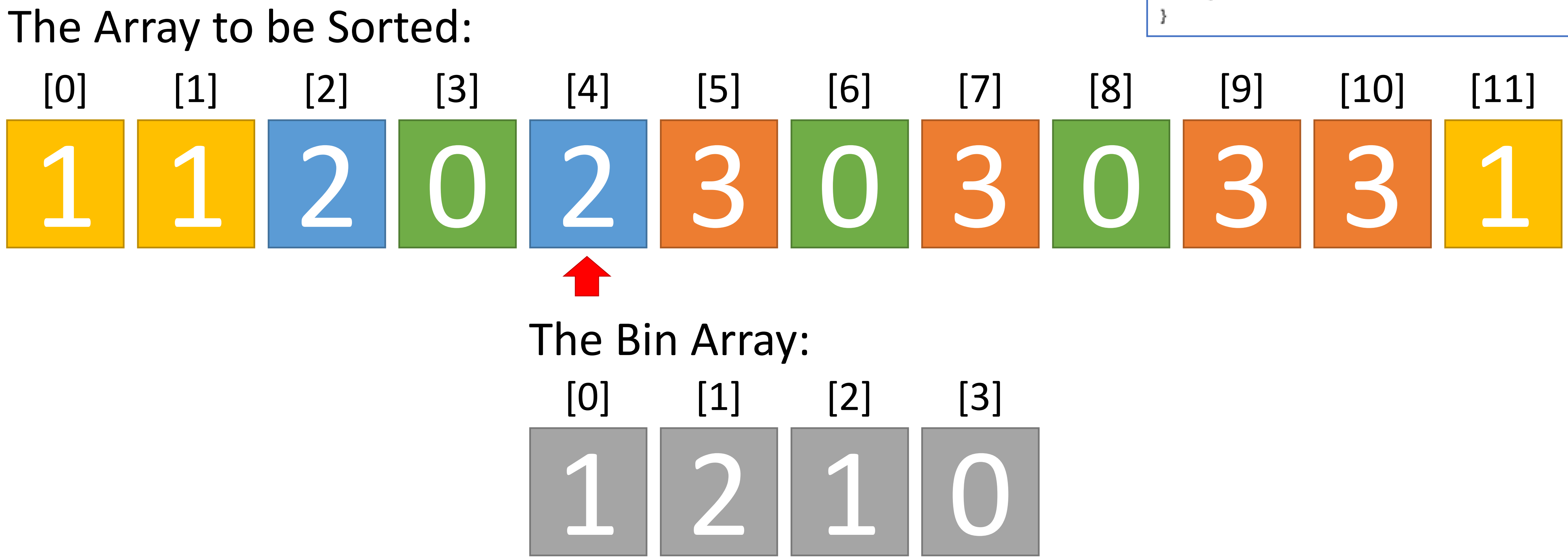
[0]	[1]	[2]	[3]
0	2	1	0

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```



Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
1	2	2	0

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
1	2	2	1

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
2	2	2	1

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
2	2	2	2

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
3	2	2	2

Go through the array and count the items.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
3	2	2	3

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

Go through the array and count the items.

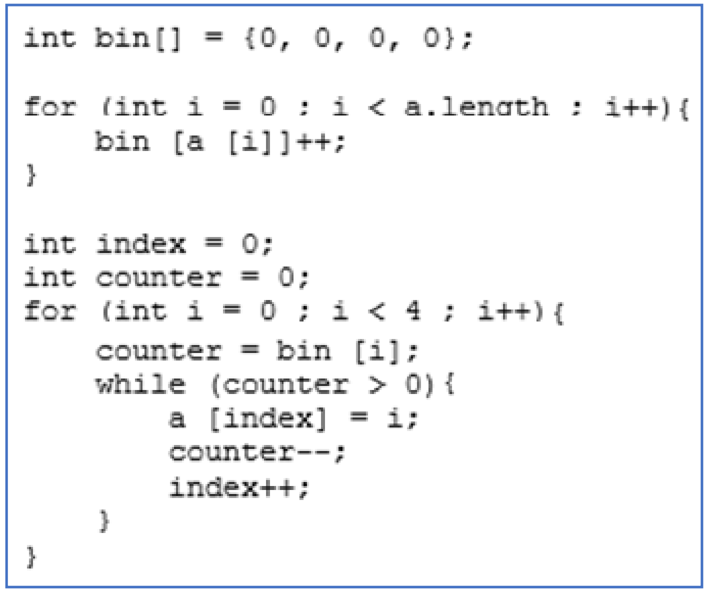
The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
3	2	2	4

Erase the old array and print out the correct number of each item.



```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	1	2	0	2	3	0	3	0	3	3	1

The Bin Array:

[0]	[1]	[2]	[3]
3	3	2	4

Erase the old array and print out the correct number of each item.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]

The Bin Array:

[0]	[1]	[2]	[3]
3	3	2	4

Erase the old array and print out the correct number of each item.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
0	0	0									

The Bin Array:

[0]	[1]	[2]	[3]
3	3	2	4

Erase the old array and print out the correct number of each item.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
0	0	0	1	1	1						

The Bin Array:

[0]	[1]	[2]	[3]
3	3	2	4

Erase the old array and print out the correct number of each item.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
0	0	0	1	1	1	2	2				

The Bin Array:

[0]	[1]	[2]	[3]
3	3	2	4

Erase the old array and print out the correct number of each item.

```
int bin[] = {0, 0, 0, 0};

for (int i = 0 ; i < a.length ; i++){
    bin [a [i]]++;
}

int index = 0;
int counter = 0;
for (int i = 0 ; i < 4 ; i++){
    counter = bin [i];
    while (counter > 0){
        a [index] = i;
        counter--;
        index++;
    }
}
```

The Array to be Sorted:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
0	0	0	1	1	1	2	2	3	3	3	3

The Bin Array:

[0]	[1]	[2]	[3]
3	3	2	4

```
public class binsort
{
    public static void main (String args[])
    {
        new binsort ();
    }


    public binsort ()
    {
        int a[] = {2, 3, 1, 1, 0, 1, 3, 0, 2, 2};
        System.out.println ("The array before: ");
        printarray (a);


        int bin[] = {0, 0, 0, 0};

        for (int i = 0 ; i < a.length ; i++)
        {
            bin [a [i]]++;
        }

        int index = 0;
        int counter = 0;
        for (int i = 0 ; i < 4 ; i++)
        {
            counter = bin [i];
            while (counter > 0)
            {
                a [index] = i;
                counter--;
                index++;
            }
        }
        System.out.println ("The array after: ");
        printarray (a);
    }
```

Bin Sort Algorithm

****Circle and label each of these steps in the adjacent code****

0. Look at the array-to-sort (it is named 'a') What numbers are in it?

_____, _____, _____, _____.

1. Declare your bins in an array. Allocate an element for each number in the array-to-sort.

2. Go through the array-to-sort, counting the how many times each number appears. Store the count in the bins.

bin[]	[0]	[1]	[2]	[3]

3. Use the bin array to re-make the array-to-sort. Use the count in each bin to fill the appropriate number of elements.

4. Print the sorted array.

```
public class binsort
{
     public static void main (String args[])
     {
          new binsort ();
     }


     public binsort ()
     {
```
0
```
          int a[] = {2, 3, 1, 1, 0, 1, 3, 0, 2, 2};
          System.out.println ("The array before: ");
          printarray (a);


          int bin[] = {0, 0, 0, 0};

          for (int i = 0 ; i < a.length ; i++)
          {
               bin [a [i]]++;
          }

          int index = 0;
          int counter = 0;
          for (int i = 0 ; i < 4 ; i++)
          {
               counter = bin [i];
               while (counter > 0)
               {
                    a [index] = i;
                    counter--;
                    index++;
               }
          }
          System.out.println ("The array after: ");
          printarray (a);
     }
```

Bin Sort Algorithm

****Circle and label each of these steps in the adjacent code****

0. Look at the array-to-sort (it is named 'a') What numbers are in it?

0, 1, 2, 3.

1. Declare your bins in an array. Allocate an element for each number in the array-to-sort.

2. Go through the array-to-sort, counting the how many times each number appears. Store the count in the bins.

bin[]	[0]	[1]	[2]	[3]

3. Use the bin array to re-make the array-to-sort. Use the count in each bin to fill the appropriate number of elements.

4. Print the sorted array.

```
public class binsort
{
     public static void main (String args[])
     {
          new binsort ();
     }


     public binsort ()
     {
0         int a[] = {2, 3, 1, 1, 0, 1, 3, 0, 2, 2};
          System.out.println ("The array before: ");
          printarray (a);

1         int bin[] = {0, 0, 0, 0};

          for (int i = 0 ; i < a.length ; i++)
          {
               bin [a [i]]++;
          }

          int index = 0;
          int counter = 0;
          for (int i = 0 ; i < 4 ; i++)
          {
               counter = bin [i];
               while (counter > 0)
               {
                    a [index] = i;
                    counter--;
                    index++;
               }
          }
          System.out.println ("The array after: ");
          printarray (a);
     }
```

Bin Sort Algorithm

****Circle and label each of these steps in the adjacent code****

0. Look at the array-to-sort (it is named 'a') What numbers are in it?

0, 1, 2, 3.

1. Declare your bins in an array. Allocate an element for each number in the array-to-sort.

2. Go through the array-to-sort, counting the how many times each number appears. Store the count in the bins.

bin[]	[0]	[1]	[2]	[3]

3. Use the bin array to re-make the array-to-sort. Use the count in each bin to fill the appropriate number of elements.

4. Print the sorted array.

```
    public class binsort
    {
        public static void main (String args[])
        {
            new binsort ();
        }


        public binsort ()
        {
0           int a[] = {2, 3, 1, 1, 0, 1, 3, 0, 2, 2};
            System.out.println ("The array before: ");
            printarray (a);

1           int bin[] = {0, 0, 0, 0};
            for (int i = 0 ; i < a.length ; i++)
            {
2               bin [a [i]]++;
            }

            int index = 0;
            int counter = 0;
            for (int i = 0 ; i < 4 ; i++)
            {
                counter = bin [i];
                while (counter > 0)
                {
                    a [index] = i;
                    counter--;
                    index++;
                }
            }
            System.out.println ("The array after: ");
            printarray (a);
        }
```

Bin Sort Algorithm

****Circle and label each of these steps in the adjacent code****

0. Look at the array-to-sort (it is named 'a') What numbers are in it?

0 , 1 , 2 , 3 .

1. Declare your bins in an array. Allocate an element for each number in the array-to-sort.

2. Go through the array-to-sort, counting the how many times each number appears. Store the count in the bins.

bin[]	[0]	[1]	[2]	[3]
	2	3	3	2

3. Use the bin array to re-make the array-to-sort. Use the count in each bin to fill the appropriate number of elements.

4. Print the sorted array.

```
    public class binsort
    {
        public static void main (String args[])
        {
            new binsort ();
        }


        public binsort ()
        {
0               int a[] = {2, 3, 1, 1, 0, 1, 3, 0, 2, 2};
                System.out.println ("The array before: ");
                printarray (a);


1               int bin[] = {0, 0, 0, 0};
                for (int i = 0 ; i < a.length ; i++)
2               {
                    bin [a [i]]++;
                }
                int index = 0;
                int counter = 0;
3               for (int i = 0 ; i < 4 ; i++)
                {
                    counter = bin [i];
                    while (counter > 0)
                    {
                        a [index] = i;
                        counter--;
                        index++;
                    }
                }
                System.out.println ("The array after: ");
                printarray (a);
        }
```

Bin Sort Algorithm

****Circle and label each of these steps in the adjacent code****

0. Look at the array-to-sort (it is named 'a') What numbers are in it?

<u>0</u>, <u>1</u>, <u>2</u>, <u>3</u>.

1. Declare your bins in an array. Allocate an element for each number in the array-to-sort.

2. Go through the array-to-sort, counting the how many times each number appears. Store the count in the bins.

bin[]	[0]	[1]	[2]	[3]
	2	3	3	2

3. Use the bin array to re-make the array-to-sort. Use the count in each bin to fill the appropriate number of elements.

0	0	1	1	1	2	2	2	3	3

4. Print the sorted array.

```
    public class binsort
    {
        public static void main (String args[])
        {
            new binsort ();
        }


        public binsort ()
        {
0           int a[] = {2, 3, 1, 1, 0, 1, 3, 0, 2, 2};
            System.out.println ("The array before: ");
            printarray (a);

1           int bin[] = {0, 0, 0, 0};

            for (int i = 0 ; i < a.length ; i++)
2           {
                bin [a [i]]++;
            }

            int index = 0;
            int counter = 0;
3           for (int i = 0 ; i < 4 ; i++)
            {
                counter = bin [i];
                while (counter > 0)
                {
                    a [index] = i;
                    counter--;
                    index++;
                }
            }
            System.out.println ("The array after: ");
4           printarray (a);
        }
```

Bin Sort Algorithm

****Circle and label each of these steps in the adjacent code****

0. Look at the array-to-sort (it is named 'a') What numbers are in it?

   0, 1, 2, 3.

1. Declare your bins in an array. Allocate an element for each number in the array-to-sort.

2. Go through the array-to-sort, counting the how many times each number appears. Store the count in the bins.

bin[]	[0]	[1]	[2]	[3]
	2	3	3	2

3. Use the bin array to re-make the array-to-sort. Use the count in each bin to fill the appropriate number of elements.

0	0	1	1	1	2	2	2	3	3

4. Print the sorted array.

Bin Sort Characteristics

- Bin sort cheats! It is a specialized sort made for a very specific situation. It gains it's speed by being specialized.
- Bin sort also requires extra memory for the bins.
- Bin sort also doesn't preserve the data and swap it around. It erases it and starts over.
- Bin sort speed = ____________
- Bin sort only works for ___________ ________ type data that falls in a _________ range.
- It cannot be used with __________, __________, or ______________ data.

Bin Sort Characteristics

- Bin sort cheats! It is a specialized sort made for a very specific situation. It gains it's speed by being specialized.
- Bin sort also requires extra memory for the bins.
- Bin sort also doesn't preserve the data and swap it around. It erases it and starts over.
- Bin sort speed = O(n)
- Bin sort only works for ___________ _______ type data that falls in a _________ range.
- It cannot be used with __________, __________, or ______________ data.

Bin Sort Characteristics

- Bin sort cheats! It is a specialized sort made for a very specific situation. It gains it's speed by being specialized.
- Bin sort also requires extra memory for the bins.
- Bin sort also doesn't preserve the data and swap it around. It erases it and starts over.
- Bin sort speed = __O(n)____
- Bin sort only works for _positive_ _int____ type data that falls in a _small__ range.
- It cannot be used with __________, __________, or _____________ data.

Bin Sort Characteristics

- Bin sort cheats! It is a specialized sort made for a very specific situation. It gains it's speed by being specialized.
- Bin sort also requires extra memory for the bins.
- Bin sort also doesn't preserve the data and swap it around. It erases it and starts over.
- Bin sort speed = O(n)
- Bin sort only works for positive int type data that falls in a small range.
- It cannot be used with double, char, or String data.

1. Trace Bin Sort on the following arrays:

(a) Starting Array:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
1	2	4	0	3	4	3	2	1	2	3	4	0	0	1

Bins:

[0]	[1]	[2]	[3]	[4]

Final Array:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]

1. Trace Bin Sort on the following arrays:

(a) Starting Array:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
1	2	4	0	3	4	3	2	1	2	3	4	0	0	1

Bins:

[0]	[1]	[2]	[3]	[4]
3	3	3	3	3

Final Array:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]

1. Trace Bin Sort on the following arrays:

(a) Starting Array:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
1	2	4	0	3	4	3	2	1	2	3	4	0	0	1

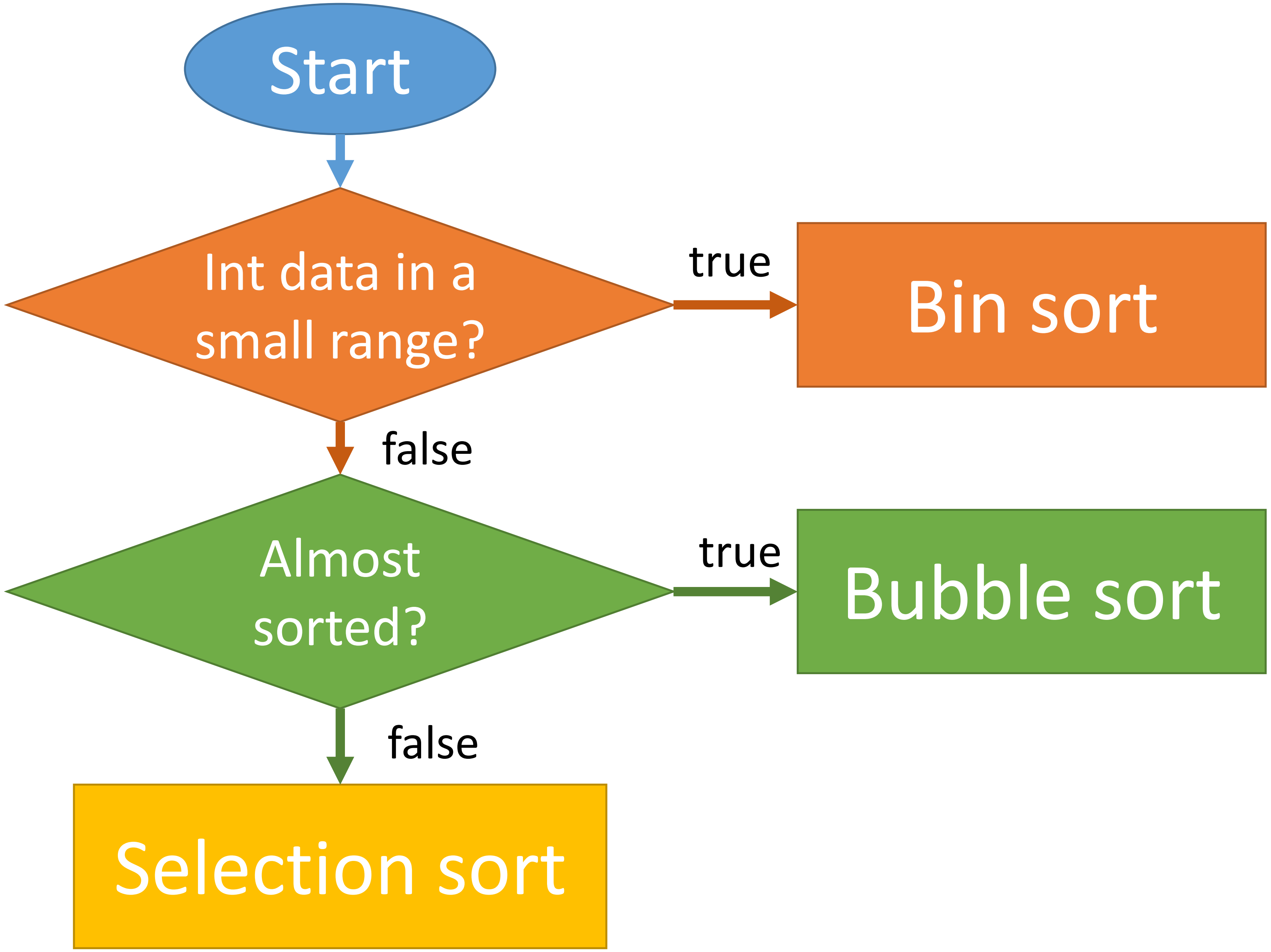
Bins:

[0]	[1]	[2]	[3]	[4]
3	3	3	3	3

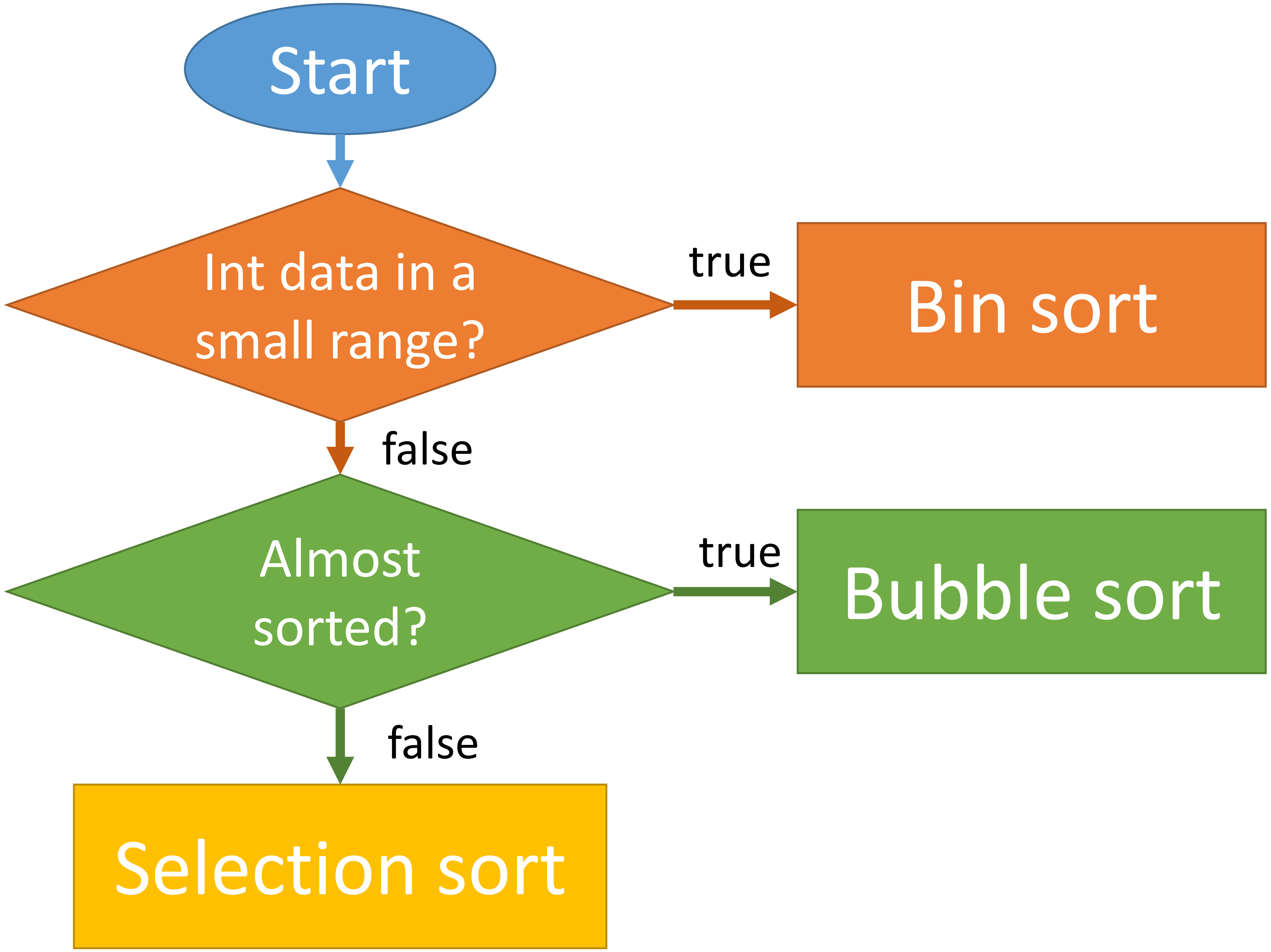
Final Array:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
0	0	0	1	1	1	2	2	2	3	3	3	4	4	4

Which sorting algorithm should you use?



You have an array of doubles in reverse order.


Start
Int data in a small range?
true
Bin sort
false
Almost sorted?
true
Bubble sort
false
Selection sort