Array Memory Work

Up

a[x-1][y]

Up
a[x-1][y]

Down
a[x+1][y]

Up
a[x-1][y]

Left
a[x][y-1]

Down
a[x+1][y]

Up
a[x-1][y]

Left
a[x][y-1]

Right
a[x][y+1]

Down
a[x+1][y]

Up
a[x-1][y]
x-1 < 0

Left
a[x][y-1]

Right
a[x][y+1]

Down
a[x+1][y]
x+1 >= row

Up

a[x-1][y]

x-1 < 0

Left

a[x][y-1]

y-1 < 0

Right

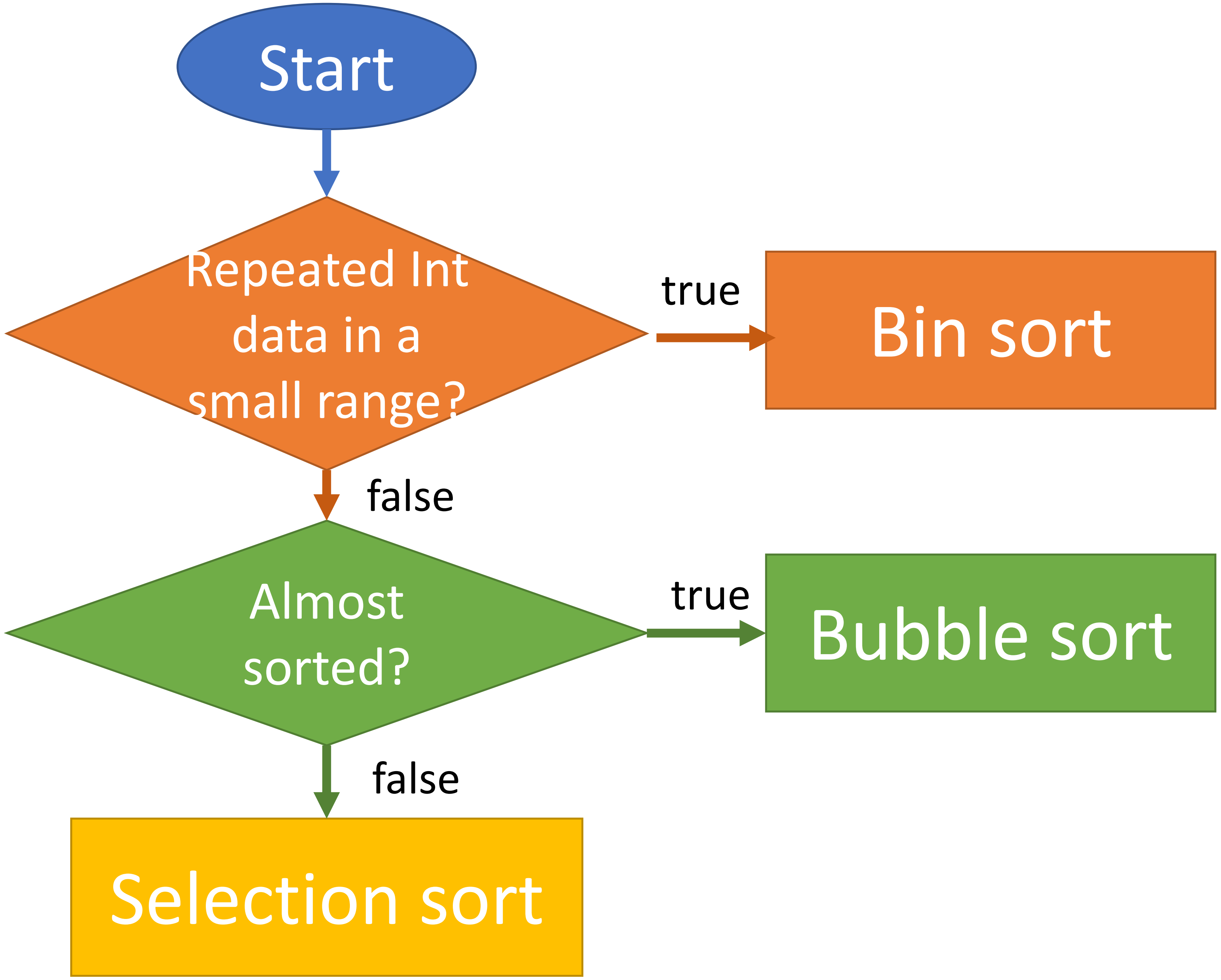
a[x][y+1]

y+1 >= col

Down

a[x+1][y]

x+1 >= row


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

1D Loop

```
for (int i = 0; i < a.length; i++)
```

2D Loops

```
for (int i = 0; i < row; i++)
    for (int j = 0; j < col; j++)
```

Swap

```
int temp = a[1];
a[1] = a[2];
a[2] = temp;
```

CompareTo = Strings

```
if (max < a [i])

if (max.compareTo(a[i]) < 0)
```

Algorithm speeds

(in order from fastest to slowest)

1. O(1), constant time, NO LOOP
2. O(logn), logarithmic time
3. O(n), linear time, ONE LOOP
4. O(n logn)
5. O(n^2), quadratic time, TWO NESTED LOOPS
6. O(n^3), cubic time
7. O(n^4)
8. O(n!), turtles walk faster.

Algorithm speeds

(in order from fastest to slowest)

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n^2)$
6. $O(n^3)$
7. $O(n!)$

The Grade 11 algorithms and their speeds:

Speed	Algorithms
O(1)	Swap, finding the length

The Grade 11 algorithms and their speeds:

Speed	Algorithms
O(n)	print, min, max, sum, average, linear search, Bin sort

The Grade 11 algorithms and their speeds:

Speed	**Algorithms**
$O(n^2)$	Selection sort, Bubblesort

What is the moral of Bentley’s example?

Fast Hardware can not compensate for a slow algorithm.

Print 1D Array

```
int name [] = new int [23];

for(int i=0; i<name.length; i++) {
    System.out.println(name[i]);
}
```

Declare 1D Array

```
int name [] = new int [4];

int name [] = {0, 2, 4, 8};
```

Print 2D Arrays

```
int row = 23;
int col = 34;
int name [] = new int [row][col];

for(int i=0; i<row; i++) {
    for(int j=0; j<col; j++) {
      System.out.print(name[i][j]);
   }
}
```

Print if over 5 (1D array)

```
for (int i = 0 ; i < a.length ; i++) {
    if (a [i] >= 5)
        System.out.println (a [i] + " ");
}
System.out.println ();
```

Print Parallel 1D Arrays

```
int name [] = new int [23];
String name2 [] = new String [23];

for(int i=0; i<name.length; i++) {
    sop(name[i] + " "+name2[i] );
}
```

Max 1D Array

```
char max = a [0];
for (int i = 0 ; i < a.length ; i++) {
   if (max < a [i])
      max = a [i];
}
System.out.println ("Largest: " + max);
```

Sum 1D Array

```
int sum = 0;
for (int i = 0 ; i < a.length ; i++)
    sum += a [i];
System.out.println ("Sum is " + sum);
```

Print Parallel 1D Arrays

```
int name [] = new int [23];
String name2 [] = new String [23];

for(int i=0; i<name.length; i++) {
   sop(name[i] + " "+name2[i] );
}
```

Print 2D Array

```
for(int i=0; i<row; i++) {
    for(int j=0; j<col; j++) {
        System.out.print(a[i][j]+"\t");
    }
    System.out.println();
}
```