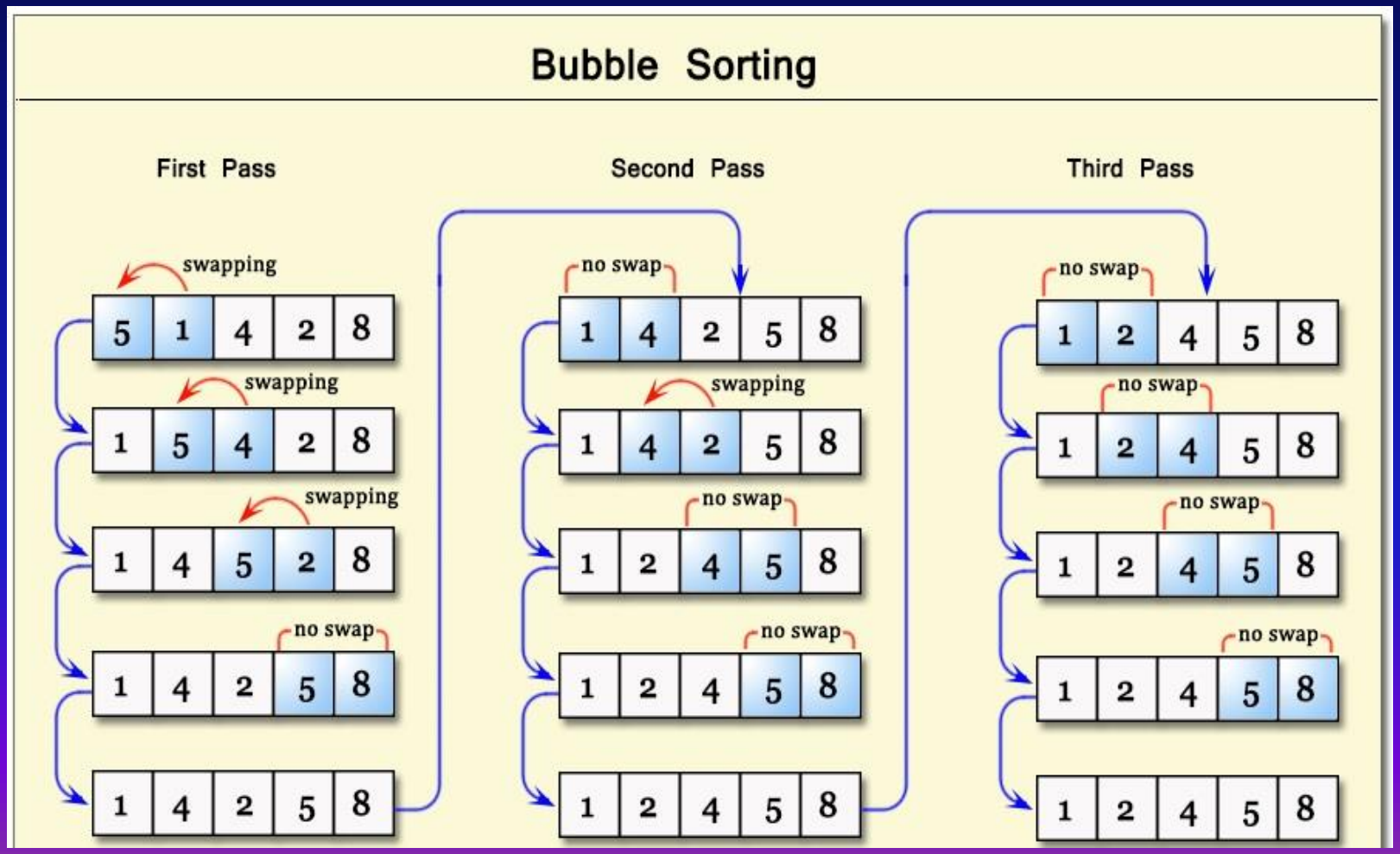
Bubble Sort


Bubble Sorting
First Pass
Second Pass
Third Pass
swapping
no swap
5 1 4 2 8
1 5 4 2 8
1 4 5 2 8
1 4 2 5 8
1 4 2 5 8
1 4 2 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8

Aims

- Give you deeper understanding of core topics
  - Sort algorithms including bubble sort
  - Efficiency of algorithms
  - Computational thinking
- Give you practical ways to teach computing in a fun, thought provoking way
  - away from computers, focus on concepts
- Linked activity sheets and booklets can be downloaded from our website:

Sort Algorithms

- A sort algorithm takes an array of data and puts it into order (either ascending order or descending order)
- eg
  - [5, 7, 2, 99, 4] -> [2, 4, 5, 7, 99]
  - [“cat”, “hat”, “ant”] -> [“ant”, “cat”, “hat”]
- Often used as a way of making things easier to find (eg in a telephone directory)
- There are many sort algorithms some more efficient than others

Towards bubble sort
Compare adjacent entries

We can compare entries at a given position and swap them

IF (array[position] > array [position+1])

THEN

    swap (array, position, position+1)

Towards bubble sort

We can scan down the array doing that on adjacent pairs

```
FOR position = 0 TO 3
{
    IF (array[position] > array [position+1])
    THEN
        swap (array, position, position+1)
}
```

Is that enough to guarantee the array is sorted?

How many times do we do this

- We need to stop just before the end as the end entry has nothing to compare with
  - So for an array of length 5, the last comparison is at position 4, to compare the 4th and 5th entries.
- However positions in arrays in many languages are numbered from 0 not 1
  - So that means it finishes comparing array[3] with array[4]

Towards bubble sort
Multiple passes

We need multiple passes i.e. to do that repeatedly

```
FOR pass = 0 TO 3
{
    FOR position = 0 TO 3
    {
        IF (array[position] > array [position+1])
        THEN
            swap (array, position, position+1)
    }
}
```

How many passes

- How many passes do we need to do to guarantee it is sorted?
- What is the worst situation we could be in?

How many passes

- On the first pass, the biggest value has ended up in the right place
  - We took it with us, where ever it started.
- On the next pass the next biggest is in the right place…and so on
- When the second last one is in the right place there is no where else for the last one to go so it is right too.
- So if there are 10 entries in the array we will need 9 passes or more generally n entries need n-1 passes

A naive version of bubble sort

```
bubblesort (array, n):
    FOR pass = 0 TO (n-2)
    {
        FOR position = 0 TO (n-2)
        {
            IF (array[position] > array [position+1])
            THEN
              swap (array, position, position+1)
         }
    }
```

Can we do better?

Can we do better?

- We have already seen that after the first pass that the biggest value is in the right place
  - So why waste time comparing against something that we know isn't going to move?
- Similarly after 2 passes 2 entries are right (and so on)
  - So on each pass there is one less thing to compare
- We need to stop the inner loop one place earlier on each pass

Can we do better?

FOR position = 0 TO (n-2) …

- On pass 0 we make no change
- On pass 1 we subtract 1 from the stop point
- On pass 2 we subtract 2 from the stop point
- …

We can do this just by subtracting pass

FOR position = 0 TO (n-2) - pass …

A more efficient version of bubble sort

```
bubblesort (array, n):
    FOR pass = 0 TO (n-2)
    {
        FOR position = 0 TO (n-2-pass)
        {
            IF (array[position] > array [position+1])
            THEN
              swap (array, position, position+1)
         }
    }
```

Can we do better?

Can we do better still?

- What happens if the array is already sorted?
- Over and over again we do comparisons, never changing anything

Observation

- If we do a whole pass and nothing changes then it never will - the array is sorted
- Add a flag to detect when this happens and stop

A more efficient version of bubble sort

```
bubblesort (array, n):
        changed := true
        pass := 0
        WHILE (pass <= n-2) AND (changed = true)
          {    changed := false
               FOR position = 0 TO (n-2-pass)
               {
                   IF (array[position] > array [position+1])
                   THEN
                      swap (array, position, position+1)
                      changed := true
                }
               pass := pass + 1
          }
```