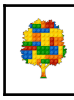
Section H
Unit #5
TSP
Lex Oder 1



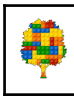
Content

Section H Unit #5 Lexicographical Order part 1

Introduction to Unit #5 (TSP) Lex Order part 1

This puts things in order rather than just have a random order of elements, this algorithm would work for letter, words as well as numbers. We will incorporate this in the next section after we have explored how this works in this section

For this section save the previous code and open a new sketch.

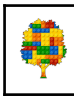


Sketch H5.1 a new sketch

We start a new sketch for a singular purpose with the following bit of code. It will include the swap function from our previous sketch, as it is it is meaningless but it is going to illustrate the principle as we will give it an array to test in a moment.

sketch.js

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}

function swap(a, i, j)
{
  let temp = a[i]
  a[i] = a[j]
  a[j] = temp
}
```



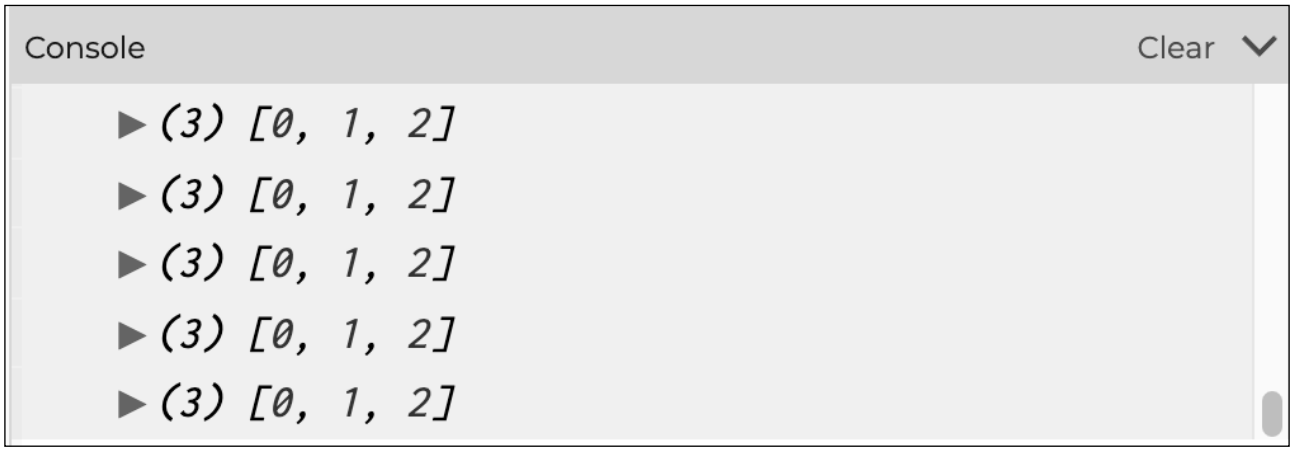
Sketch H5.2 small array

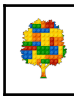
Adding a small array and console.log it.

sketch.js

```
let vals = [0, 1, 2]

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  console.log(vals)
}

function swap(a, i, j)
{
  let temp = a[i]
  a[i] = a[j]
  a[j] = temp
}
```

This is just to check that we get the array





Sketch H5.3 the one before

We look for the one before the largest (largestI variable) element

sketch.js

```
let vals = [0, 1, 2]

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  console.log(vals)
  let largestI = -1
  for (let i = 0; i < vals.length - 1; i++)
  {
    if (vals[i] < vals[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
    console.log('finished')
  }
}

function swap(a, i, j)
```

```
{
  let temp = a[i]
  a[i] = a[j]
  a[j] = temp
}
```

Sketch H5.4 next largest

Find the next largest

sketch.js

```
let vals = [0, 1, 2]

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  console.log(vals)
  let largestI = -1
  for (let i = 0; i < vals.length - 1; i++)
  {
    if (vals[i] < vals[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
    console.log('finished')
  }
  let largestJ = -1
  for (let j = 0; j < vals.length; j++)
  {
```

```
    if (vals[largestI] < vals[j])
    {
      largestJ = j
    }
  }
}

function swap(a, i, j)
{
  let temp = a[i]
  a[i] = a[j]
  a[j] = temp
}
```

Sketch H5.5 do a swap

Now we swap largestI and largestJ using the swap function

sketch.js

```
let vals = [0, 1, 2]

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  console.log(vals)
  let largestI = -1
  for (let i = 0; i < vals.length - 1; i++)
  {
    if (vals[i] < vals[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
    console.log('finished')
  }
  let largestJ = -1
  for (let j = 0; j < vals.length; j++)
  {
```

```
    if (vals[largestI] < vals[j])
    {
      largestJ = j
    }
  }
  swap(vals, largestI, largestJ)
}

function swap(a, i, j)
{
  let temp = a[i]
  a[i] = a[j]
  a[j] = temp
}
```

Sketch H5.6 reverse

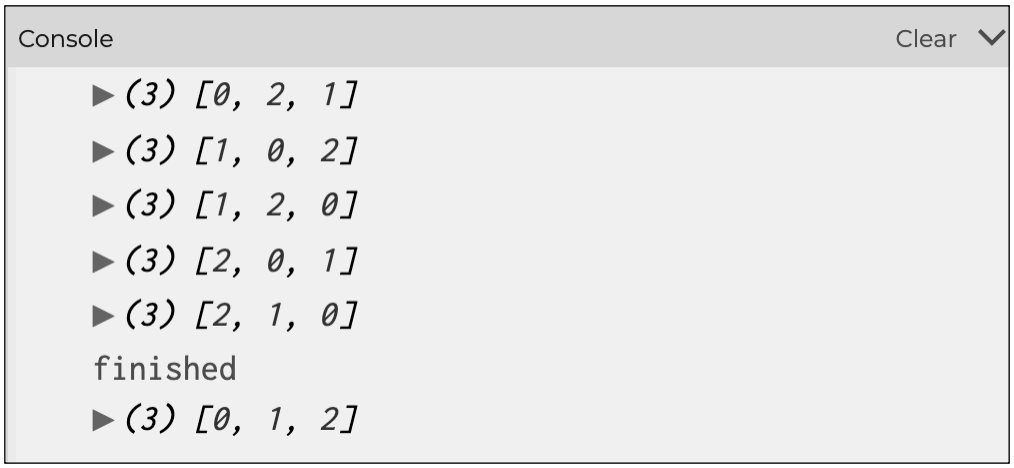
Now we want to reverse the order using the splice() and reverse function. We create an array called endArray[]

sketch.js

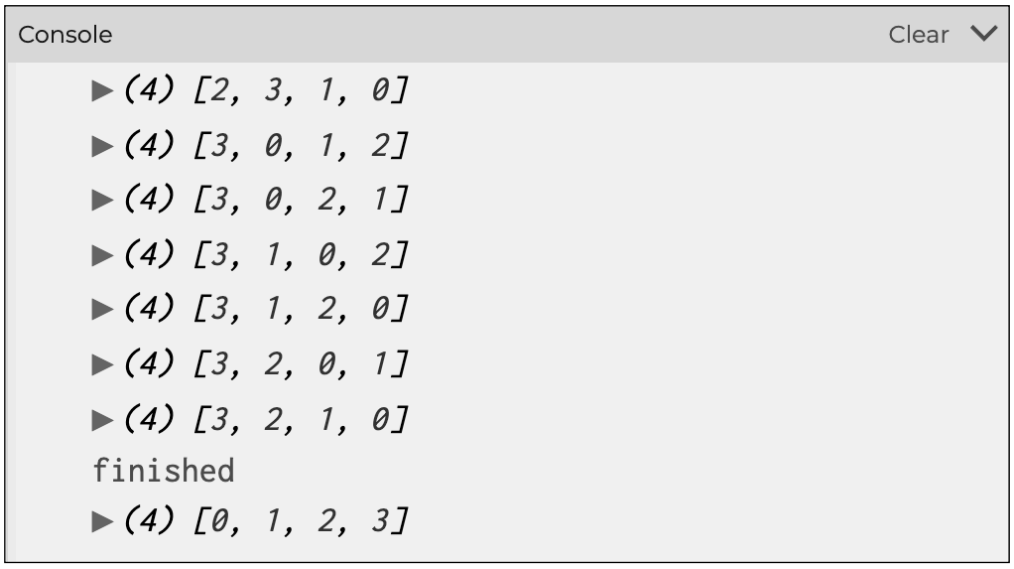
```
let vals = [0, 1, 2]

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  // console.log(vals)
  let largestI = -1
  for (let i = 0; i < vals.length - 1; i++)
  {
    if (vals[i] < vals[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
    console.log('finished')
  }
  let largestJ = -1
  for (let j = 0; j < vals.length; j++)
  {
```

```
    if (vals[largestI] < vals[j])
    {
      largestJ = j
    }
  }
  swap(vals, largestI, largestJ)
  let endArray = vals.splice(largestI + 1)
  endArray.reverse()
  vals = vals.concat(endArray)
  console.log(vals)
}

function swap(a, i, j)
{
  let temp = a[i]
  a[i] = a[j]
  a[j] = temp
}
```

With the three numbers 0, 1, 2 in the array you can see the progression



Starting with let vals = [2, 3, 0, 1] you can see the progression



Sketch H5.7 watch all the permutations

Let's draw it on the canvas and watch it run through all the permutations. It will give you an idea of why it takes such a long time to work through all the permutations in a brute force manner. We put in 10 numbers for it to work through. I recommend that you remove all the console logs. You might want to save this one as well as we will be making reference to it.

sketch.js

```
let vals = [1, 2, 3, 4, 5, 6, 7, 8, 9]

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  let largestI = -1
  for (let i = 0; i < vals.length - 1; i++)
  {
    if (vals[i] < vals[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
  }
  let largestJ = -1
  for (let j = 0; j < vals.length; j++)
```

```
  {
    if (vals[largestI] < vals[j])
    {
      largestJ = j
    }
  }
  swap(vals, largestI, largestJ)
  let endArray = vals.splice(largestI + 1)
  endArray.reverse()
  vals = vals.concat(endArray)
  textSize(64)
  let s = ''
  for (let i = 0; i < vals.length; i++)
  {
    s += vals[i]
  }
  text(s, 20, height/2)
}

function swap(a, i, j)
{
  let temp = a[i]
  a[i] = a[j]
  a[j] = temp
}
```

It will take about 17 hours to work through all the permutations one by one @ 60 frames per second for 10!
So DO NOT sit and wait unless you have nothing better to do