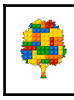
Section H
Unit #6
TSP
Lex Order 2



Content

Section H Unit #6 Lexicographic Order part 2

Introduction to Unit #6 (TSP) Lexicographic Order

We go back to our travelling salesperson sketch which I hope you have saved. But you will also be copying blocks of code from the lexicographical order sketch I hope that you have also saved. If not you can just go back and type them out again.

Sketch H6.1 next order

We create an empty array to keep track of the order of the cities and we will call it 'order' surprisingly enough. Also we will create a new function called **nextOrder()**, we copy and paste (or you could type it in) a block of code from **Lexicographic Order**, although we will change the names from **vals** to **order**.

sketch.js

```
let cities = []
let totalCities = 5
let recordDistance
let bestEver
let order = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < totalCities; i++)
  {
    let v = createVector(random(width), random(height))
    cities[i] = v
    order[i] = i
  }
  let d = calcDistance(cities)
  recordDistance = d
  bestEver = cities.slice()
}

function draw()
{
  background(220)
  for (let i = 0; i < cities.length; i++)
```

```
  {
    stroke(0)
    fill(0)
    circle(cities[i].x, cities[i].y, 5)
  }
  beginShape()
  for (let i = 0; i < cities.length; i++)
  {
    stroke(0)
    noFill()
    strokeWeight(1)
    vertex(cities[i].x, cities[i].y)
  }
  endShape()
  beginShape()
  for (let i = 0; i < cities.length; i++)
  {
    stroke(200, 0, 0)
    noFill()
    strokeWeight(3)
    vertex(bestEver[i].x, bestEver[i].y)
  }
  endShape()
  let i = floor(random(cities.length))
  let j = floor(random(cities.length))
  swap(cities, i, j)
  let d = calcDistance(cities)
  if (d < recordDistance)
  {
    recordDistance = d
    bestEver = cities.slice()
  }
}
```

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


function calcDistance(points)
{
  let sum = 0
  for (let i = 0; i < points.length - 1; i++)
  {
    let d = dist(points[i].x, points[i].y, points[i + 1].x,
points[i + 1].y)
    sum += d
  }
  return sum
}


function nextOrder()
{
  let largestI = -1
  for (let i = 0; i < order.length - 1; i++)
  {
    if (order[i] < order[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
```

```
  }
  let largestJ = -1
  for (let j = 0; j < order.length; j++)
  {
    if (order[largestI] < order[j])
    {
      largestJ = j
    }
  }
  swap(order, largestI, largestJ)
  let endArray = order.splice(largestI + 1)
  endArray.reverse()
  order = order.concat(endArray)
}
```

Sketch H6.2 adding visual information

We are going to change the shape of the canvas and add in a few other bits and pieces from the lexicographical order sketch. It is just a few lines so no need to copy and paste. We draw the cities in the top half of the window and some text in the bottom half. Also we now call the nextOrder() function.

Remember to change vals to order. You can play with the text appearance.

sketch.js

```
let cities = []
let totalCities = 5
let recordDistance
let bestEver
let order = []

function setup()
{
  createCanvas(400, 800)
  for (let i = 0; i < totalCities; i++)
  {
    let v = createVector(random(width), random(height / 2))
    cities[i] = v
    order[i] = i
  }
  let d = calcDistance(cities)
  recordDistance = d
  bestEver = cities.slice()
}

function draw()
{
  background(220)
```

```
  for (let i = 0; i < cities.length; i++)
  {
    stroke(0)
    fill(0)
    circle(cities[i].x, cities[i].y, 5)
  }
  beginShape()
  for (let i = 0; i < cities.length; i++)
  {
    stroke(0)
    noFill()
    strokeWeight(1)
    vertex(cities[i].x, cities[i].y)
  }
  endShape()
  beginShape()
  for (let i = 0; i < cities.length; i++)
  {
    stroke(200, 0, 0)
    noFill()
    strokeWeight(3)
    vertex(bestEver[i].x, bestEver[i].y)
  }
  endShape()
  let i = floor(random(cities.length))
  let j = floor(random(cities.length))
  swap(cities, i, j)
  let d = calcDistance(cities)
  if (d < recordDistance)
  {
    recordDistance = d
    bestEver = cities.slice()
  }
```

```
  textSize(64)
  let s = ''
  for (let i = 0; i < order.length; i++)
  {
    s += order[i]
  }
  text(s, 20, 3*height/4)
  nextOrder()
}

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

function calcDistance(points)
{
  let sum = 0
  for (let i = 0; i < points.length - 1; i++)
  {
    let d = dist(points[i].x, points[i].y, points[i + 1].x,
points[i + 1].y)
    sum += d
  }
  return sum
}

function nextOrder()
{
  let largestI = -1
  for (let i = 0; i < order.length - 1; i++)
```

```
  {
    if (order[i] < order[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
  }
  let largestJ = -1
  for (let j = 0; j < order.length; j++)
  {
    if (order[largestI] < order[j])
    {
      largestJ = j
    }
  }
  swap(order, largestI, largestJ)
  let endArray = order.splice(largestI + 1)
  endArray.reverse()
  order = order.concat(endArray)
}
```

Notes

What you get is a sort of count down a reordering of the original array. It doesn't tell you what the oder of the cities are (optimal) it tells you it has gone through every permutation in order.

Challenge

Try 6 and then 7 cities, then try 10.

You should have the cities at the top and the text at the bottom

Sketch H6.3 connecting order array

We want to connect the order array with with what is being drawn. The order array is changing, the array of cities is not changing. We are calculating the distance between cityA and cityB (the next one)

sketch.js

```
let cities = []
let totalCities = 5
let recordDistance
let bestEver
let order = []

function setup()
{
  createCanvas(400, 800)
  for (let i = 0; i < totalCities; i++)
  {
    let v = createVector(random(width), random(height / 2))
    cities[i] = v
    order[i] = i
  }
  let d = calcDistance(cities, order)
  recordDistance = d
  bestEver = cities.slice()
}

function draw()
{
  background(220)
  for (let i = 0; i < cities.length; i++)
  {
    stroke(0)
```

```
    fill(0)
    circle(cities[i].x, cities[i].y, 5)
  }
  beginShape()
  for (let i = 0; i < order.length; i++)
  {
    stroke(0)
    noFill()
    strokeWeight(1)
    let n = order[i]
    vertex(cities[n].x, cities[n].y)
  }
  endShape()
  beginShape()
  for (let i = 0; i < cities.length; i++)
  {
    stroke(200, 0, 0)
    noFill()
    strokeWeight(3)
    vertex(bestEver[i].x, bestEver[i].y)
  }
  endShape()
  let i = floor(random(cities.length))
  let j = floor(random(cities.length))
  swap(cities, i, j)
  let d = calcDistance(cities, order)
  if (d < recordDistance)
  {
    recordDistance = d
    bestEver = cities.slice()
  }
  textSize(64)
  let s = ''
```

```
  for (let i = 0; i < order.length; i++)
  {
    s += order[i]
  }
  text(s, 20, 3*height/4)
  nextOrder()
}

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

function calcDistance(points, order)
{
  let sum = 0
  for (let i = 0; i < order.length - 1; i++)
  {
    let cityAIndex = order[i]
    let cityA = points[cityAIndex]
    let cityBIndex = order[i + 1]
    let cityB = points[cityBIndex]
    let d = dist(cityA.x, cityA.y, cityB.x, cityB.y)
    sum += d
  }
  return sum
}

function nextOrder()
{
  let largestI = -1
```

```
  for (let i = 0; i < order.length - 1; i++)
  {
    if (order[i] < order[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
  }
  let largestJ = -1
  for (let j = 0; j < order.length; j++)
  {
    if (order[largestI] < order[j])
    {
      largestJ = j
    }
  }
  swap(order, largestI, largestJ)
  let endArray = order.splice(largestI + 1)
  endArray.reverse()
  order = order.concat(endArray)
}
```

Sketch H6.4 refactoring

More refactoring, I have highlighted the swap elements we will remove

sketch.js

```
let cities = []
let totalCities = 5
let recordDistance
let bestEver
let order = []

function setup()
{
  createCanvas(400, 800)
  for (let i = 0; i < totalCities; i++)
  {
    let v = createVector(random(width), random(height / 2))
    cities[i] = v
    order[i] = i
  }
  let d = calcDistance(cities, order)
  recordDistance = d
  bestEver = order.slice()
}

function draw()
{
  background(220)
  for (let i = 0; i < cities.length; i++)
  {
    stroke(0)
    fill(0)
```

```
    circle(cities[i].x, cities[i].y, 5)
  }
  beginShape()
  for (let i = 0; i < order.length; i++)
  {
    stroke(0)
    noFill()
    strokeWeight(1)
    let n = order[i]
    vertex(cities[n].x, cities[n].y)
  }
  endShape()
  beginShape()
  for (let i = 0; i < order.length; i++)
  {
    stroke(200, 0, 0)
    noFill()
    strokeWeight(3)
    let n = bestEver[i]
    vertex(cities[n].x, cities[n].y)
  }
  endShape()
  // let i = floor(random(cities.length))
  // let j = floor(random(cities.length))
  // swap(cities, i, j)
  let d = calcDistance(cities, order)
  if (d < recordDistance)
  {
    recordDistance = d
    bestEver = order.slice()
  }
  textSize(64)
  let s = ''
```

```
  for (let i = 0; i < order.length; i++)
  {
    s += order[i]
  }
  text(s, 20, 3*height/4)
  nextOrder()
}

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

function calcDistance(points, order)
{
  let sum = 0
  for (let i = 0; i < order.length - 1; i++)
  {
    let cityAIndex = order[i]
    let cityA = points[cityAIndex]
    let cityBIndex = order[i + 1]
    let cityB = points[cityBIndex]
    let d = dist(cityA.x, cityA.y, cityB.x, cityB.y)
    sum += d
  }
  return sum
}

function nextOrder()
{
  let largestI = -1
```

```
  for (let i = 0; i < order.length - 1; i++)
  {
    if (order[i] < order[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
  }
  let largestJ = -1
  for (let j = 0; j < order.length; j++)
  {
    if (order[largestI] < order[j])
    {
      largestJ = j
    }
  }
  swap(order, largestI, largestJ)
  let endArray = order.splice(largestI + 1)
  endArray.reverse()
  order = order.concat(endArray)
}
```

Sketch H6.5 calculate permutations

We will do a number of things here:
1. calculate the permutations
2. replace the text with this information
3. have a percentage of the permutation completed

To do this we will need a new function to calculate the permutations and make the text smaller. We add a counter to keep track of the number of iterations it has completed. We use a simple algorithm to calculate n! (n factorial), e.g. 5! Is 5*4*3*2*1 = 120

sketch.js

```
let cities = []
let totalCities = 5
let recordDistance
let bestEver
let order = []
let totalPermutations
let count = 1

function setup()
{
  createCanvas(400, 800)
  for (let i = 0; i < totalCities; i++)
  {
    let v = createVector(random(width), random(height / 2))
    cities[i] = v
    order[i] = i
  }
  let d = calcDistance(cities, order)
  recordDistance = d
  bestEver = order.slice()
  totalPermutations = factorial(totalCities)
```

```
}

function draw()
{
  background(220)
  for (let i = 0; i < cities.length; i++)
  {
    stroke(0)
    fill(0)
    circle(cities[i].x, cities[i].y, 5)
  }
  beginShape()
  for (let i = 0; i < order.length; i++)
  {
    stroke(0)
    noFill()
    strokeWeight(1)
    let n = order[i]
    vertex(cities[n].x, cities[n].y)
  }
  endShape()
  beginShape()
  for (let i = 0; i < order.length; i++)
  {
    stroke(200, 0, 0)
    noFill()
    strokeWeight(3)
    let n = bestEver[i]
    vertex(cities[n].x, cities[n].y)
  }
  endShape()
  let d = calcDistance(cities, order)
  if (d < recordDistance)
```

```
  {
    recordDistance = d
    bestEver = order.slice()
  }
  textSize(32)
  noStroke()
  fill(0)
  let percent = 100 * (count / totalPermutations)
  text(nf(percent, 0, 2) + '% completed', 20, 3 * height / 4)
  nextOrder()
}

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

function calcDistance(points, order)
{
  let sum = 0
  for (let i = 0; i < order.length - 1; i++)
  {
    let cityAIndex = order[i]
    let cityA = points[cityAIndex]
    let cityBIndex = order[i + 1]
    let cityB = points[cityBIndex]
    let d = dist(cityA.x, cityA.y, cityB.x, cityB.y)
    sum += d
  }
  return sum
}
```

```
function nextOrder()
{
  count++
  let largestI = -1
  for (let i = 0; i < order.length - 1; i++)
  {
    if (order[i] < order[i + 1])
    {
      largestI = i
    }
  }
  if (largestI == -1)
  {
    noLoop()
  }
  let largestJ = -1
  for (let j = 0; j < order.length; j++)
  {
    if (order[largestI] < order[j])
    {
      largestJ = j
    }
  }
  swap(order, largestI, largestJ)
  let endArray = order.splice(largestI + 1)
  endArray.reverse()
  order = order.concat(endArray)
}

function factorial(n)
{
  if (n == 1)
```

```
  {
    return 1
  }
  else
  {
    return n * factorial(n-1)
  }
}
```