D3: What is a Bisector?

Asked 5 years, 5 months ago Active 5 years, 5 months ago Viewed 22k times

46

18

I'm looking into making charts with D3, and stumbled upon the `d3.bisector` . However, I don't understand what it is or does from the documentation.

Almost all examples that I find in the web use a Date array, similar to the example in the official documentation:

```
var data = [
  {date: new Date(2011,  1, 1), value: 0.5},
  {date: new Date(2011,  2, 1), value: 0.6},
  {date: new Date(2011,  3, 1), value: 0.7},
  {date: new Date(2011,  4, 1), value: 0.8}
];

var bisect = d3.bisector(function(d) { return d.date; }).right;
```

So what does the bisector do, besides picking the date object from the array elements? What does the `*.right` return?

And is it of any use if I have a simple 1-dimensional array, like `var data = [3, 6, 2, 7, 5, 4, 8]` ?

Thanks for enlightening me.

javascript arrays d3.js

asked Nov 12 '14 at 8:44

Terry
11.1k 12 46 77

2 Answers

Active | Oldest | Votes

The underlying idea behind `bisect` is this:

126

Consider the array that you mention - `var data = [3, 6, 2, 7, 5, 4, 8]`

You want to insert a new value let's say `3.5` into `data` array and want to know how would that 'partition' it. In other words, you want to know what would be the index of `3.5` if it were inserted when `data` array is sorted.

```
 var data = [3, 6, 2, 7, 5, 4, 8]

 //Sorted data

[2, 3, 4, 5, 6, 7, 8]

//You want to insert 3.5

The sorted array after insertion of 3.5 should look something like:

[2, 3, 3.5, 4, 5, 6, 7, 8]

So the index of 3.5 in sorted data array is "2".
```

There are situations where you would want to know how the insertion of that element 'bisects' or 'divides' an array. In that case, you would want to first sort that array and do what we call a Binary Search to find out the correct position for insertion of that element.

`bisectLeft` and `bisectRight` take care to clarify the anomaly in a situation where you want to enter an element that already exists in the array. Let's say you want to enter another `3` into the array. There are two situations:

```
3* -> The new element to be entered

[2, 3*, 3, 4, 5, 6, 7, 8] -> entered at "1" (array is still sorted)

[2, 3, 3*, 4, 5, 6, 7, 8] -> entered at "2" (array is still sorted)
```

So depending upon how we take care of this ambiguity, we can enter that element to the 'left' or 'right' of the already existing element. From the docs (Mark the emphasis):

> The returned insertion point i partitions the array into two halves so that all v **<** x for v in array.slice(lo, i) for the left side and all v **>=** x for v in array.slice(i, hi) for the right side.

In `bisectLeft` we get 1 as the suitable index, all the duplicate entries **will** be on the right of that index and the situation is exactly the opposite in `bisecRight` .

Now that you know how `bisectLeft` and `bisectRight` work, the `bisector` just allows us to define a custom `comparator` or `accessor` function to parition the values or make sense of **<** and **>** on objects as well.

So this piece of code:

```
var bisect = d3.bisector(function(d) { return d.date; }).right;

var bisect = d3.bisector(function(a, b) { return a.date - b.date; }).right;
```

Just specifies to use the `bisectRight` option and return a suitable index for the insertion of an element assuming the array is sorted(in ascending order).

So if I were to build up on your example and assuming a `bisector` named `bisect`. And you did:

```
bisect(data, 3); //it would return 2.
```

I hope it clarifies things and gets you started in the right direction.

answered Nov 12 '14 at 9:35

Vivek Pradhan

4,125 3 20 39

9 Brilliant answer! I'm so happy that I finally get it! – Terry Nov 12 '14 at 9:46

From the documentation (that you've linked to):

> Locate the insertion point for x in array to maintain sorted order.

That's what it does. It tells you where a new element should be inserted to still have a sorted array after that. The array can be any kind of structure, which is why there's an accessor function that allows you to "decompose" this structure for the purposes of searching.

The difference between left and right bisects is where the insertion point is (to the left or right of the closest element) -- whether the array is sorted ascending or descending.

One use case for the bisectors is where you want to highlight the closest data point when moving the mouse over a graph, see e.g. this example.

answered Nov 12 '14 at 9:34

Lars Kotthoff

96.5k 13 171 175

Nice brief. And the link is the example that I just want. – Ashi Apr 9 '18 at 11:10