Removing the Min

10 min

Min-heaps would be useless if we couldn't retrieve the minimum value. We've gone through a lot of work to maintain that value because we're going to need it!

Our goal is to efficiently remove the minimum element from the heap. You'll recall that we always locate the minimum element at

Preview: Docs Loading link description

index

1 (a placeholder element occupies index 0).

Our internal heap mirrors a

Preview: Docs Loading link description

binary

tree. There's a delicate balance of parent and child relationships we would ruin by directly removing the minimum.

```
console.log(minHeap.heap)
// [null, 10, 21, 13, 61, 22, 23, 99]
minHeap.popMin()
// 10
// [null, ???, 21, 13, 61, 22, 23, 99]
```

We need to remove an element that has no children, in this case, the last element. If we swap the minimum with the last element, we can easily remove the minimum from the end of the list.

[None, (10), 21, 13, 61, 22, 23, {99}]

minHeap.popMin()

SWAP minimum with last

[None, {99}, 21, 13, 61, 22, 23, (10)]

remove minimum

[None, 99, 21, 13, 61, 22, 23]

10

Terrific! We removed the minimum element with minimal disruption. Unfortunately, our heap is out of shape again with 99 sitting where the minimum element should be. We will solve this in exercises to come…

Instructions

1. Checkpoint 1 Passed

1.

To retrieve the minimum value of our heap, we need to define a class method.

- Define .popMin() below the constructor within our MinHeap class.
- Within .popMin(), check if our heap is empty. If it is, return null.

Hint

Use this.size to determine if our heap is empty.

2. Checkpoint 2 Passed

2.

Next, we want to:

- exchange the last element of the heap with the minimum element at index 1 using .swap()
- remove the last element from the heap, and save it in a const min variable
- decrement the heap size.

Hint

To swap two elements with .swap() do the following:

```
const a = 5, b = 7;
this.swap(a, b);
console.log(a, b); // should return 7 5
```

To remove the last element from an array we use the .pop() method and assign it to a const variable as follows:

```
const myArray = [ 6, 3, 4 ];
const popped = myArray.pop(); // return 4
```

3. Checkpoint 3 Passed

3.

Display:

- a message to show that the first element ${this.heap[1]} is swapped with the last element, ${this.heap[this.size]} (do this before the actual swap)

- a message that shows that the minimum element has been removed followed by the content of the heap; use the stringRemove at the beginning of the message.

Hint

The message to show swapping between the first and last elements of the heap may look like this:

console.log(`\n.. Swap ${this.heap[1]} with last element ${this.heap[this.size]}`);

The message to show that the minimum element has been removed followed by the content of the heap may look like this:

console.log(`.. Removed ${min} from heap`, this.heap);

4. Checkpoint 4 Passed

4.

Lastly, return the min variable in .popMin().

5. Checkpoint 5 Passed

5.

Open **script.js** and run the test code.

MinHeap.js

```
class MinHeap {
 constructor() {
  this.heap = [ null ];
  this.size = 0;
 }

 popMin() {
  if (this.size === 0) {
   return null;
  }
  else {
   console.log(`\n.. Swap ${this.heap[1]} with last element ${this.heap[this.size]}`);
```

```
    this.swap(1, this.size);
    const min = this.heap.pop();
    this.size--;
    console.log(`.. Removed ${min} from heap`);
    console.log('..',this.heap);
    return min;
  }
}

add(value) {
  console.log(`.. adding ${value}`);
  this.heap.push(value);
  this.size++;
  this.bubbleUp();
  console.log(`added ${value} to heap`, this.heap);
}

bubbleUp() {
  let current = this.size;
  while (current > 1 && this.heap[getParent(current)] > this.heap[current]) {
    console.log('..', this.heap);
    console.log(`.. swap index ${current} with ${getParent(current)}`);
    this.swap(current, getParent(current));
    current = getParent(current);
  }
}

swap(a, b) {
  [this.heap[a], this.heap[b]] = [this.heap[b], this.heap[a]];
```

```
  }

}

const getParent = current => Math.floor((current / 2));
const getLeft = current => current * 2;
const getRight = current => current * 2 + 1;

module.exports = MinHeap;
```

script.js

```
// import MinHeap class
const MinHeap = require('./MinHeap');

// instantiate a MinHeap class
const minHeap = new MinHeap();

// helper function to return a random integer
function randomize() { return Math.floor(Math.random() * 40); }

// populate minHeap with random numbers
for (let i=0; i < 6; i++) {
  minHeap.add(randomize());
}

// display the bubbled up numbers in the heap
console.log('Bubbled Up', minHeap.heap);
```

```
// remove the minimum value from heap
minHeap.popMin();
```

>> Output

```
.. adding 39
added 39 to heap [ null, 39 ]
.. adding 22
.. [ null, 39, 22 ]
.. swap index 2 with 1
added 22 to heap [ null, 22, 39 ]
.. adding 9
.. [ null, 22, 39, 9 ]
.. swap index 3 with 1
added 9 to heap [ null, 9, 39, 22 ]
.. adding 27
.. [ null, 9, 39, 22, 27 ]
.. swap index 4 with 2
added 27 to heap [ null, 9, 27, 22, 39 ]
.. adding 38
added 38 to heap [ null, 9, 27, 22, 39, 38 ]
.. adding 16
.. [ null, 9, 27, 22, 39, 38, 16 ]
.. swap index 6 with 3
added 16 to heap [ null, 9, 27, 16, 39, 38, 22 ]
Bubbled Up [ null, 9, 27, 16, 39, 38, 22 ]

.. Swap 9 with last element 22
.. Removed 9 from heap
.. [ null, 22, 27, 16, 39, 38 ]
```