算法入门（上）

算法比你想象中简单

版权声明

联系方式

如果你想要购买本课程
请微信联系 xiedaimala02 或 xiedaimala03

如果你发现有人盗用本课程
请微信联系 xiedaimala02 或 xiedaimala03

小试牛刀

如何找出两个数中较小的那一个

必备知识

- 数据结构
  - ✓ 用数组 [a, b] 表示两个数字
  - ✓ 你能想到这一点，就说明你在使用数据结构
- 编程知识
  - ✓ 问号冒号表达式 ?:

两个数找出较小的那个

- 代码

```
let minOf2 = (numbers) => {
  if(numbers[0] < numbers[1]){
    return numbers[0]
  }else{
    return numbers[1]
  }
}
```

- 优化代码

```
let minOf2 = numbers =>
  numbers[0] < numbers[1]
    ? numbers[0] : numbers[1]
```

两个数找出较小的那个

- 优化代码

```
let minOf2 = numbers =>
  numbers[0] < numbers[1]
    ? numbers[0] : numbers[1]
```

- 再优化代码

```
let minOf2 = ([a, b]) => a < b ? a : b
```

这种写法叫做析构赋值，之后的课程会反复使用

- 调用

- ✓ `minOf2([1,2]) // 1` 这是小白调用法
- ✓ `minOf2.call(null, [1,2]) //` 这是高手调用法

现成 API

- JS 内置了 Math.min
  - Math.min(1,2) // 1
  - Math.min.call(null, 1,2)
  - Math.min.apply(null, [1,2])
- 关于 Math
  - 看起来 Math 像 Object 一样是构造函数
  - 实际上 Math 只是一个普通对象
  - 这是唯一的特例：首字母大写是构造函数

举一反三

两个数变成三个数

三个数找出最小的那个

- 代码

```
let minOf3 = ([a,b,c]) => {
  return minOf2([minOf2([a,b]), c])
}
```

或者

```
let minOf3 = ([a,b,c]) => {
  return minOf2([a, minOf2([b,c])])
}
```

- 推理

```
let minOf4 = ([a,b,c,d]) => {
  return minOf2([a, minOf3([b,c,d])])
}
```

任意长度数组求最小值，都可以通过 minOf2 实现

推广

任意长度数组求最小值

找出最小的那个

- 代码

```
let min = (numbers) => {
  return min(
    [numbers[0], min(numbers.slice(1))]
  )
}
```

这个代码会死循环不停调用自己，需要添加一个中止条件

找出最小的那个（续）

- 代码

```
let min = (numbers) => {
  if(numbers.length > 2){
    return min(
      [numbers[0], min(numbers.slice(1))]
    )
  }else{
    return Math.min.apply(null, numbers)
  }
}
```

这就是递归

递归

- 特点
  - ✓ 函数不停调用自己，每次调用的参数略有不同
  - ✓ 当满足某个简单条件时，则实现一个简单的调用
  - ✓ 最终算出结果
- 理解
  - ✓ 可以用代入法快速理解递归
  - ✓ 可以用调用栈快速理解递归

升级

将正整数数组从小到达排序

排序算法

- 思路
  - ✓ 用递归实现
  - ✓ 用循环实现

递归思路

选择排序

小试牛刀

将长度为 2 的数组排序

长度为 2 的数组排序

- 代码

```
let sort2 = ([a,b]) => {
  if(a < b){
    return [a,b]
  }else{
    return [b,a]
  }
}
```

- 优化代码

```
let sort2 = ([a,b]) =>
  a < b ? [a,b] : [b,a]
```

长度为 3 的数组排序

- 代码

```
let sort3 = ([a,b,c]) => {
  return [min([a,b,c]), sort2([???])]
}
```

我们发现无法将最小值从数组里删掉

- 改进代码

```
let sort3 = (numbers) => {
  let index = minIndex(numbers)
  let min = numbers[index]
  numbers.splice(index, 1)
  // 从 numbers 里删掉 min
  return [min].concat(sort2(numbers))
}
```

长度为 4 的数组排序

- 代码

```
let sort4 = (numbers) => {
  let index = minIndex(numbers)
  let min = numbers[index]
  numbers.splice(index, 1)
  return [min].concat(sort3(numbers))
}
```

推广

任意长度的数组排序

任意长度的数组排序

- 代码

```
let sort = (numbers) => {
  let index = minIndex(numbers)
  let min = numbers[index]
  numbers.splice(index, 1)
  return [min].concat(sort(numbers))
}
```

死循环……

minIndex

- 代码

```
let minIndex = (numbers) =>
  numbers.indexOf(min(numbers))
```

这是一个取巧的办法，以后会教更好的

任意长度的数组排序(续)

- 代码

```
let sort = (numbers) => {
  if(numbers.length > 2){
    let index = minIndex(numbers)
    let min = numbers[index]
    numbers.splice(index, 1)
    return [min].concat(sort(numbers))
  }else{
    return numbers[0]<numbers[1] ? numbers :
           numbers.reverse()
  }
}
```

用代入法看看 sort([12,5,8,7,9]) 的过程

代码会错

用 console.log 调试

总结

- 求最小值
  - 2 个数
  - 3 个数
  - N 个数
- 排序
  - 2 个数
  - 3 个数
  - N 个数
- 用到的东西
  - 数组（数据结构）
  - 递归