最长递增子序列算法

从一个序列当中找到最长递增子序列的个数 贪心算法 找更有潜力的

2 3 7 6 8 4 9 5

23

237

236

2368

2348

23489

23459

当前和序列的最后一项比较，如果比最后一项大，直接追加到尾部，如果比尾部小，则找到序列当中比他当前大的都替换掉

最长递增子序列的个数没问题，那么最后再把序列搞对了就好

采用二分查找 找到比当前项大的那个人

最后倒序排列即可

时间复杂度 ologn

```
function getSequence(arr) { // 最终的结果是索引
    const len = arr.length;
    const result = [0]; // 索引  递增的序列 用二分查找性能高
    const p = arr.slice(0); // 里面内容无所谓 和 原本的数组相同 用来存放索引
    let start;
    let end;
    let middle;
    for (let i = 0; i < len; i++) { // O(n)
        const arrI = arr[i];
        if (arrI !== 0) { // vue3当中不处理0的情况，0代表新增的节点
            let resultLastIndex = result[result.length - 1];
            // 取到索引对应的值
            if (arr[resultLastIndex] < arrI) {
                p[i] = resultLastIndex; // 标记当前前一个对应的索引
                result.push(i);
                // 当前的值 比上一个人大 ， 直接push ， 并且让这个人得记录他的前一个
                continue
            }
            // 二分查找 找到比当前值大的那一个
            start = 0;
            end = result.length - 1;
```

```
            while (start < end) { // 重合就说明找到了 对应的值  // O(logn)
                middle = ((start + end) / 2) | 0; // 找到中间位置的前一个
                if (arr[result[middle]] < arrI) {
                    start = middle + 1
                } else {
                    end = middle
                } // 找到结果集中，比当前这一项大的数
            }
            // start / end 就是找到的位置
            if (arrI < arr[result[start]]) { // 如果相同 或者 比当前的还大就不换
了
                if (start > 0) { // 才需要替换
                    p[i] = result[start - 1]; // 要将他替换的前一个记住
                }
                result[start] = i;
            }
        }
    }
    let i = result.length // 总长度
    let last = result[i - 1] // 找到了最后一项
    while (i-- > 0) { // 根据前驱节点一个个向前查找
        result[i] = last // 最后一项肯定是正确的
        last = p[last]
    }
    return result;
}
console.log(getSequence([2, 3, 1, 5, 6, 8, 7, 9, 4]))
```