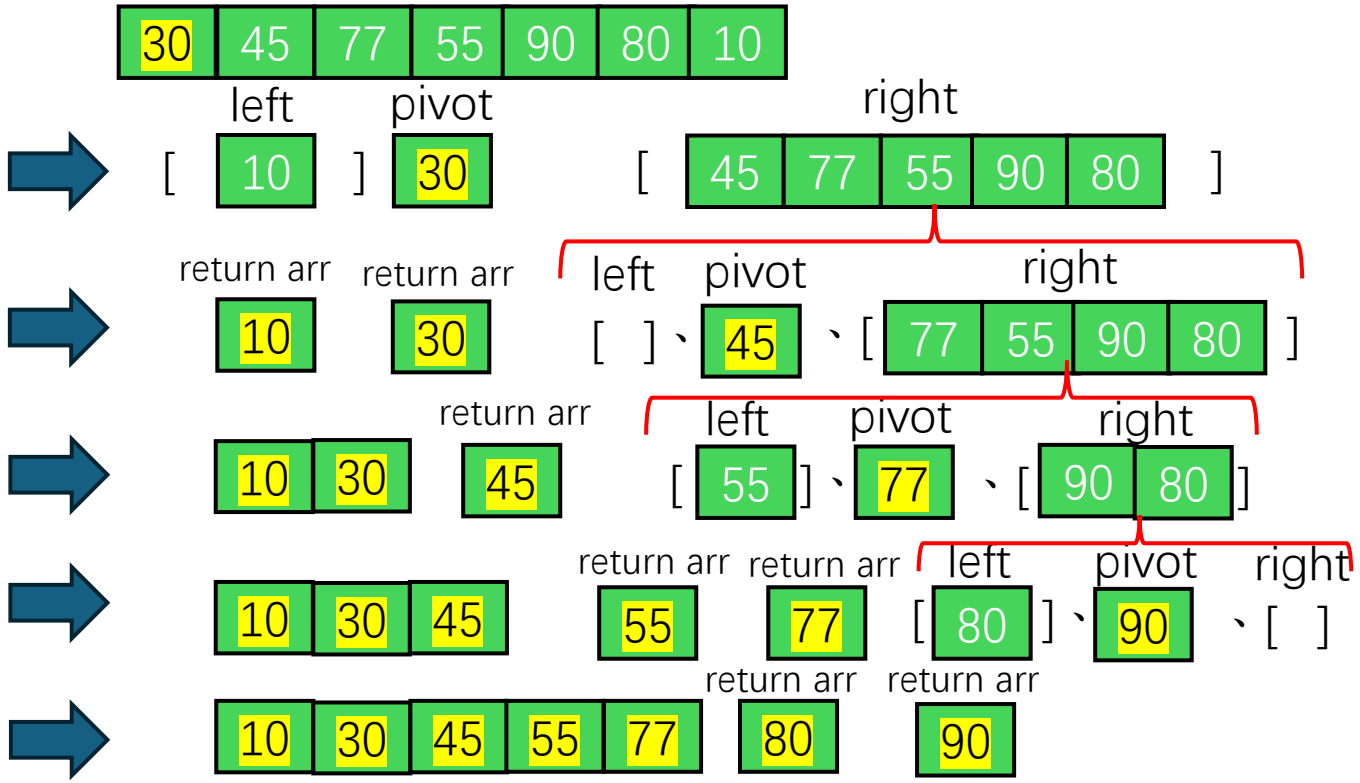
Javascript 快速排序法+二分搜尋法

```
JS quicksort.js ×
JS quicksort.js > prompt.get() callback
1  var prompt = require('prompt');
2
3  prompt.start();
4
5  let data = [30,45,77,55,90,80,10];
6
7  function quicksort(arr) {
8    if (arr.length <= 1) {
9      return arr;
10   }
11   let left = [];
12   let right = [];
13   let pivot = arr[0];
14   for (i = 1; i < arr.length; i++) {
15     let num = arr[i];
16     if (num < pivot) {
17       left.push(num);
18     } else {
19       right.push(num);
20     }
21   }
22   return [...quicksort(left), pivot, ...quicksort(right)];
23 }
24 console.log(quicksort(data));
25
26 function binarysearch(arr,goal) {
27   let head = 0;
28   let end = arr.length - 1;
29   let mid;
30   while (head <= end) {
31     mid = ((head + end) / 2) | 0;
32     if (goal < arr[mid]) {
33       end = mid - 1;
34     } else if (goal > arr[mid]) {
35       head = mid + 1;
36     } else {
37       return ("搜尋選項在第" + (mid + 1) + "項");
38     }
39   }
40   return ("無搜尋資料");
41 }
42
43 prompt.get(['number'],function(err,result) {
44     console.log(binarysearch(quicksort(data),result.number));
45 });
```

這是快速排序法+二分搜尋法的程式

快速排序法的程式是
1.先給予一資料陣列 data
2.在function quicksort中，本程式使用的方式是選定data第一個數字做為基準值，宣告兩個空陣列"左"和"右"
3.將接下來的各個數字小於基準值的放進左陣列，大於的則放進右陣列
4.之後用遞迴的方式使得左陣列、基準值、右陣列個別再跑一次function，前面的if有判斷陣列資料剩下小於等於1個數字時會直接回傳
5.在多次循環下數字就會由小到大排序傳回data中



```javascript
var prompt = require('prompt');

prompt.start();

let data = [30,45,77,55,90,80,10];

function quicksort(arr) {
  if (arr.length <= 1) {
    return arr;
  }
  let left = [];
  let right = [];
  let pivot = arr[0];
  for (i = 1; i < arr.length; i++) {
    let num = arr[i];
    if (num < pivot) {
      left.push(num);
    } else {
      right.push(num);
    }
  }
  return [...quicksort(left), pivot, ...quicksort(right)];
}
console.log(quicksort(data));

function binarysearch(arr,goal) {
  let head = 0;
  let end = arr.length - 1;
  let mid;
  while (head <= end) {
    mid = ((head + end) / 2) | 0;
    if (goal < arr[mid]) {
      end = mid - 1;
    } else if (goal > arr[mid]) {
      head = mid + 1;
    } else {
      return ("搜尋選項在第" + (mid + 1) + "項");
    }
  }
  return ("無搜尋資料");
}

prompt.get(['number'],function(err,result) {
    console.log(binarysearch(quicksort(data),result.number));
});
```

二分搜尋法的程式是

1.輸入一個陣列與要在陣列中尋找的數字

2.數字用prompt javascript的寫法使得能輸入任意數字

3.在 function binarysearch 中，先宣告head=0、end=陣列長度-1和mid

4.本程式使用位元運算子的or方法配上0，使得(head+end)/2在存在小數點時，能捨去小數點後的數字變為整數

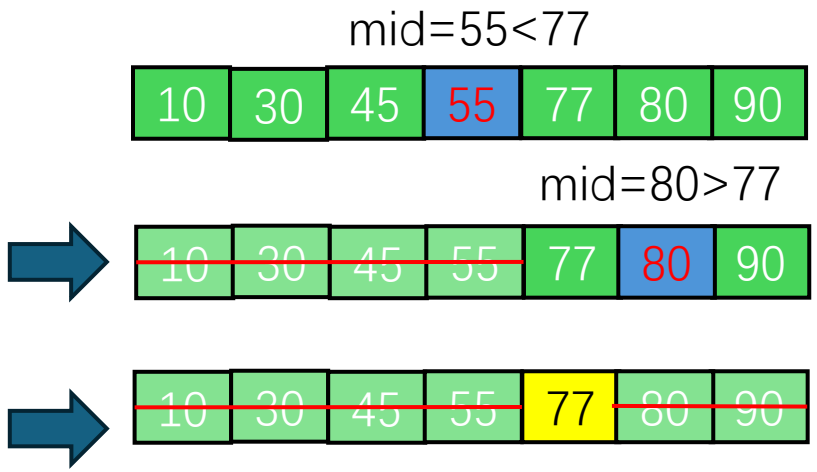
5.當mid位置的數字大於尋求的數字時，會使搜尋範圍從尾部往前縮至一半

6.反之，當mid位置的數字小於尋求的數字時，會使搜尋範圍從部頭往前縮至一半

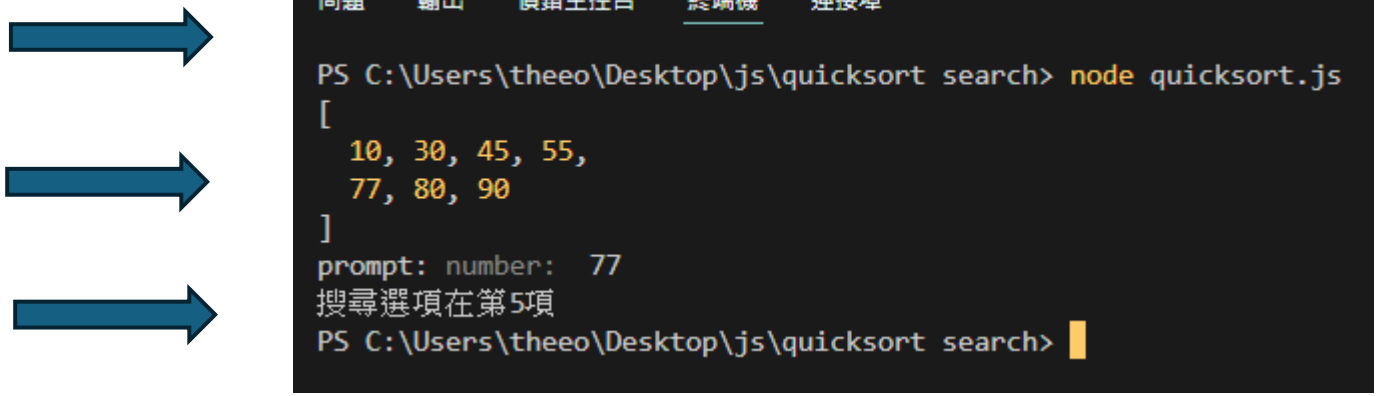
7.反覆進行直到搜尋範圍縮至1個數字時，搜尋資料就在mid+1項的位置

8.若無此資料，最後就會回"無搜尋資料"

以10,30,45,55,80,77,90 搜尋77為例



這是執行結果


問題 輸出 偵錯主控台 終端機 連接埠
PS C:\Users\theeo\Desktop\js\quicksort search> node quicksort.js
[
10, 30, 45, 55,
77, 80, 90
]
prompt: number: 77
搜尋選項在第5項
PS C:\Users\theeo\Desktop\js\quicksort search>

完整程式列表

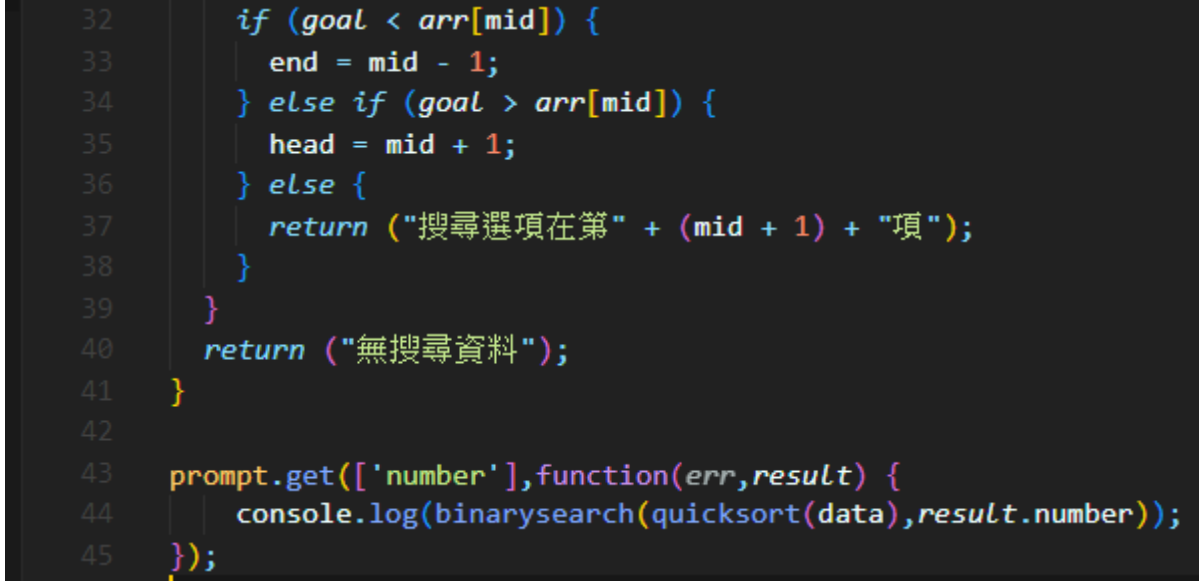
```
var prompt = require('prompt');

prompt.start();

let data = [30,45,77,55,90,80,10];

function quicksort(arr) {
  if (arr.length <= 1) {
    return arr;
  }
  let left = [];
  let right = [];
  let pivot = arr[0];
  for (i = 1; i < arr.length; i++) {
    let num = arr[i];
    if (num < pivot) {
      left.push(num);
    } else {
      right.push(num);
    }
  }
  return [...quicksort(left), pivot, ...quicksort(right)];
}
console.log(quicksort(data));

function binarysearch(arr,goal) {
  let head = 0;
  let end = arr.length - 1;
  let mid;
  while (head <= end) {
    mid = ((head + end) / 2) | 0;
```



```
    if (goal < arr[mid]) {
      end = mid - 1;
    } else if (goal > arr[mid]) {
      head = mid + 1;
    } else {
      return ("搜尋選項在第" + (mid + 1) + "項");
    }
  }
  return ("無搜尋資料");
}

prompt.get(['number'],function(err,result) {
    console.log(binarysearch(quicksort(data),result.number));
});
```