二元搜尋
binary search

資訊之芽/語法班
Gin

討論狀況

- 我們討論的狀況都是假設陣列中的元素已經排序好了，從小排到大 or 從大排到小，並且假設所有元素皆不重複。
- Ex.　1 5 7 228 319 689 921

線性搜尋(linear search)

- 從頭掃到尾，但速度很慢，如果要找的元素在最後一個，要跑 n 次！
- 沒有善加利用到已經排序的性質！這不用排序也能做！

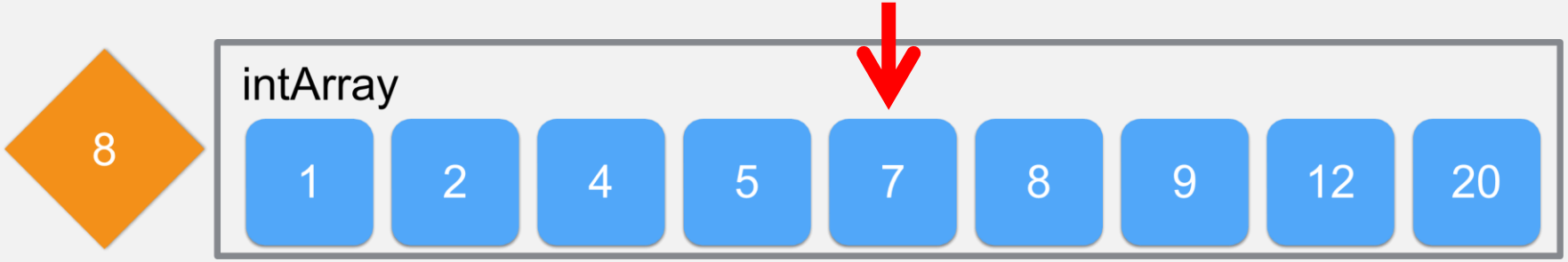
```
for(int i = 0; i < length; i++)
  if(array[i] == key)
    return i;
```

二分搜尋(binary search)

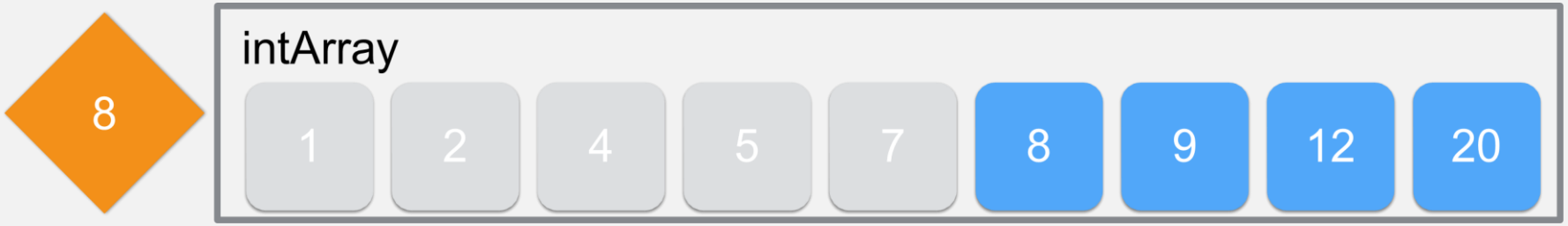
- 又稱二元搜尋
- 作法之前作業有提過，就是把它切半搜，再切半搜，有點像在玩終極密碼!?
- 使用這個方法搜尋，陣列一定要經過排序！

二分搜尋(binary search)

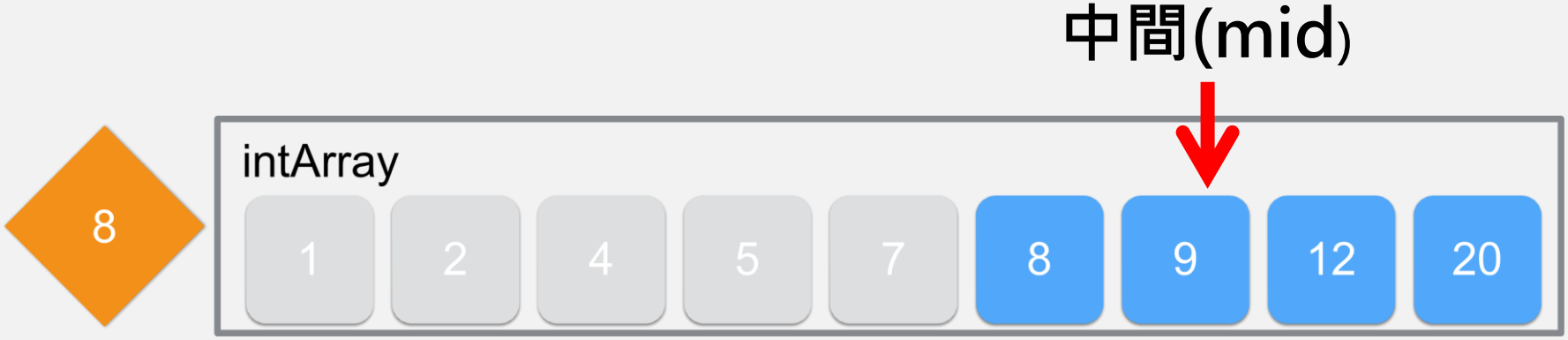
中間(mid)



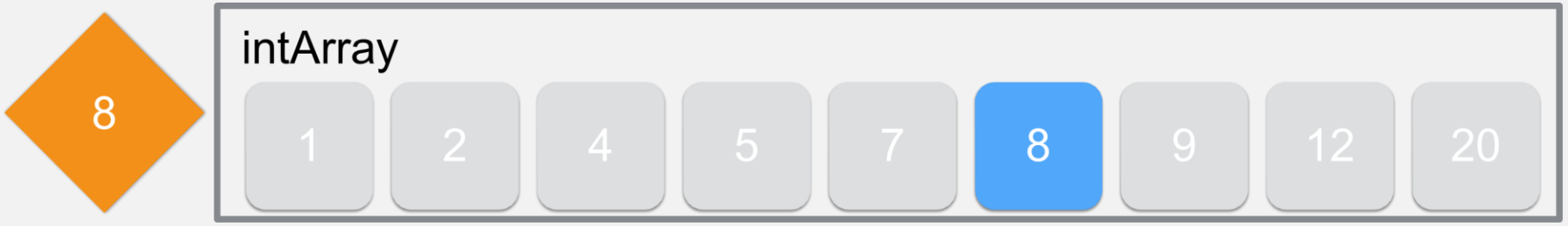
7 < 8 所以會在 mid 的右半邊！



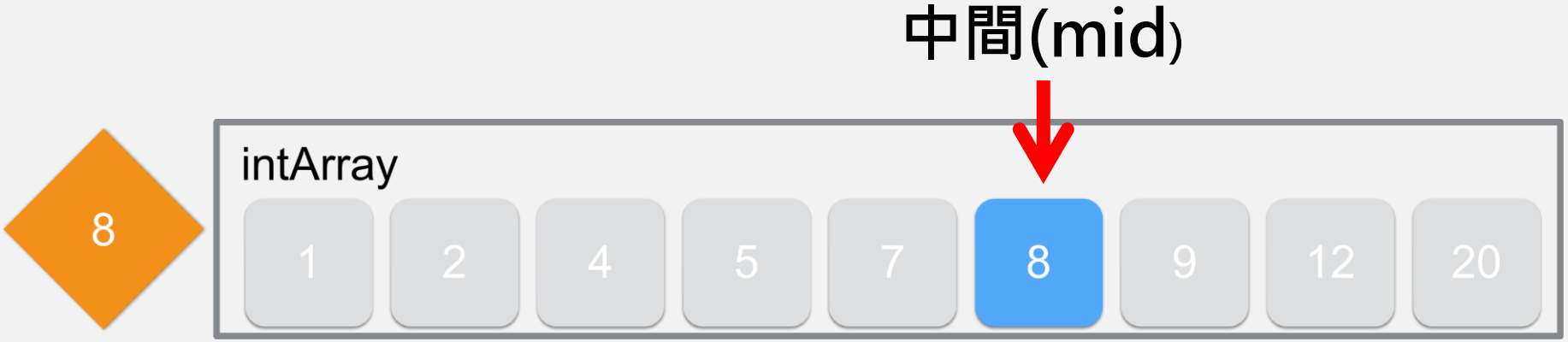
二分搜尋(binary search)



8 < 9 所以會在mid 的左半邊！



二分搜尋(binary search)



8 == 8，找到8了，回傳它在陣列中的位置 (索引值)！

return 5 ;

影片範例：**http://ppt.cc/gh4kf**

二分搜尋(binary search)

Index : (0 + 9) / 2 = 4

9

9 在右半邊！
low = mid + 1

Index : 0

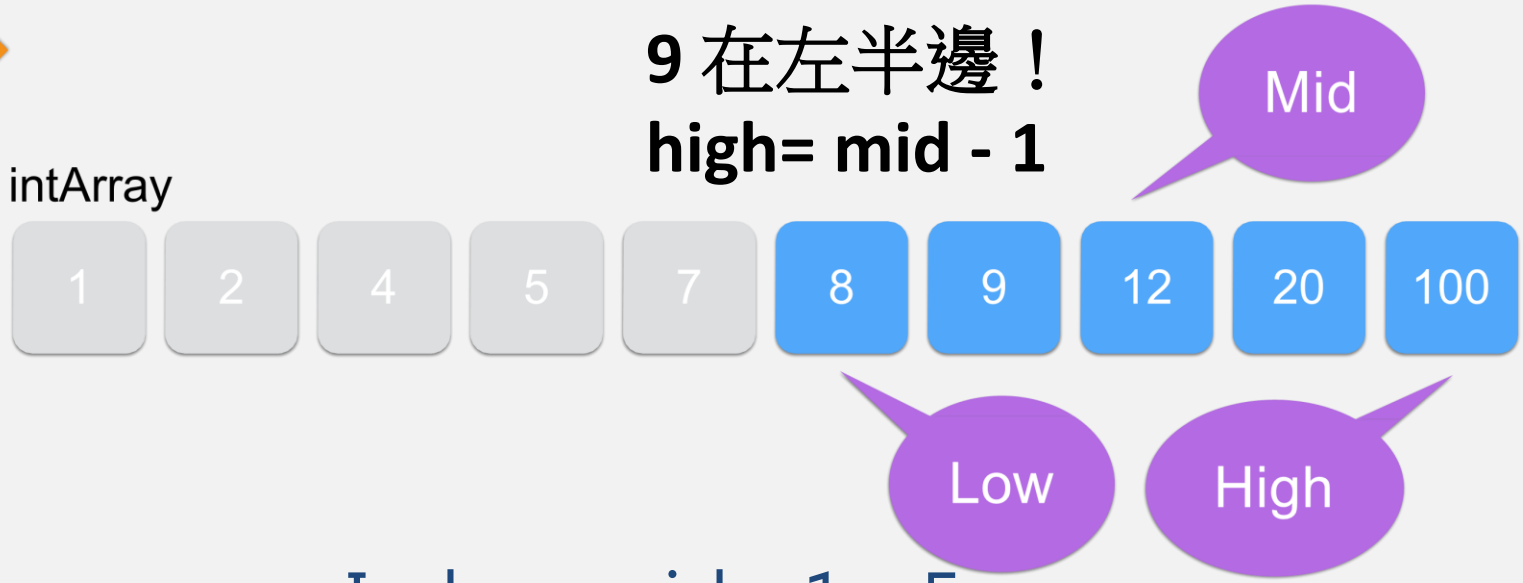
Index : 9

二分搜尋(binary search)

Index : (5 + 9) / 2 = 7

9

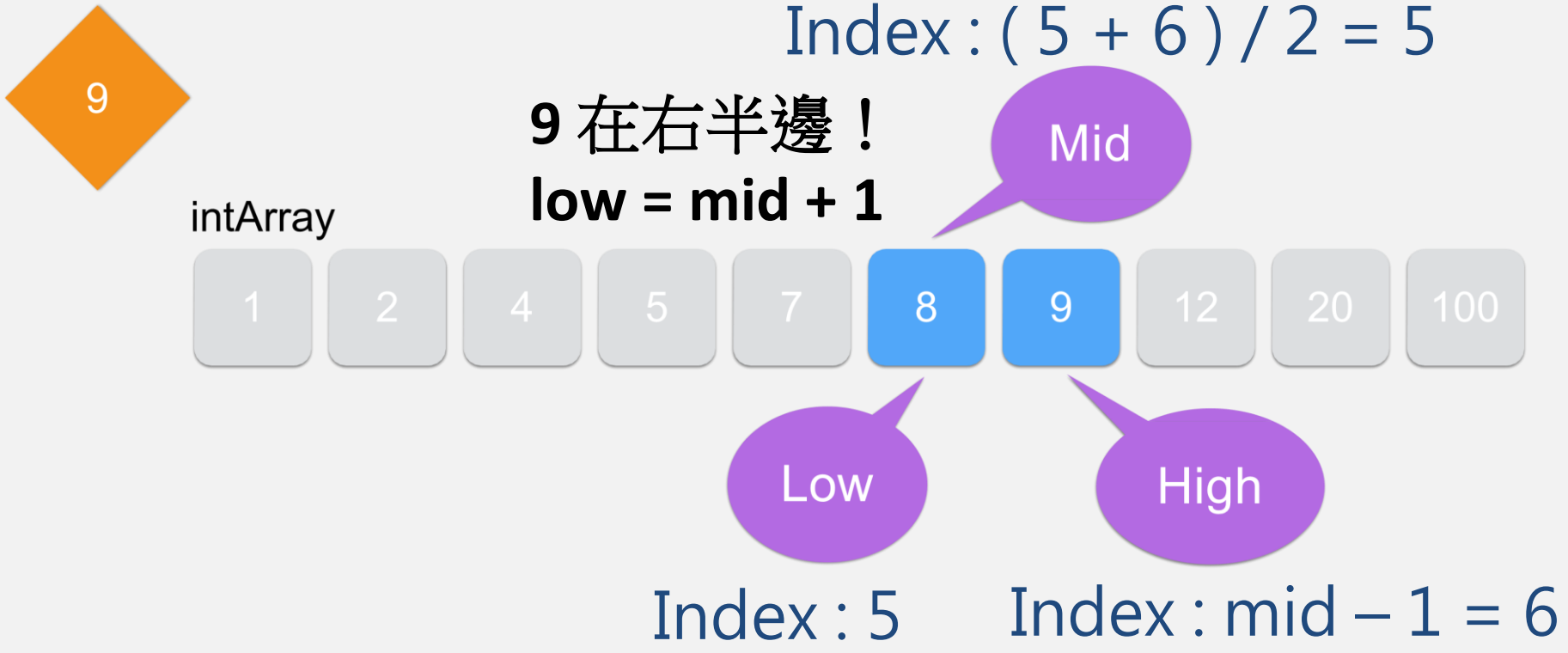
9 在左半邊！
high= mid - 1



Index : mid+ 1= 5

Index : 9

二分搜尋(binary search)



二分搜尋(binary search)

Index : (6 + 6) / 2 = 6

9

array[mid] 是 9！
return mid

Index : mid + 1 = 6

Index : 6

二分搜尋(binary search)

- 這樣即使在最差的情況下，所需要的次數也只要 $\log_2 n$
- 善用排序的優點，加快速度！

```
int binarySearch(int array[], int length, int key)
{
  int low = 0;
  int high = length - 1;
  while(low <= high) {
    int mid = (low + high) / 2

    1. array[mid] = key : 找到了，回傳 mid
    2. array[mid] > key : 在左半邊，把 high 變成 mid - 1
    3. array[mid] < key : 在右半邊，把 low 變成 mid + 1
  }
  如果跑到這裡，也就是while迴圈外，代表沒找到這個數。
}
```