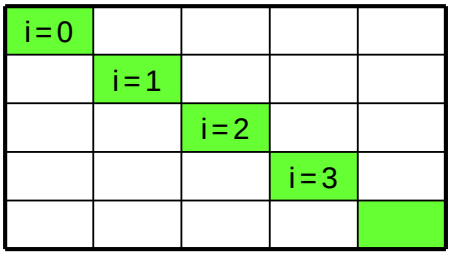
選擇排序法 (selection sort)

在右面找最小的
互相換轉

data[i]	i=0	i=1	i=2	i=3	i=4	i=5	i=6
start	90	20	40	10	80	70	50
0	10	20	40	90	80	70	50
1		20	40	90	80	70	50
2			40	90	80	70	50
3				50	80	70	90
4					70	80	90
5						80	90




```
void selectionSort (){
        int i, j, min, temp;

        for (i=0; i<max-1; i++){
                min = ____;        // 當目前a[i]是最小的
                // 在a[i]右邊選個最小的 a[min]
                for (j=_____; j<max; j++){
                        if (a[____]<a[____]) min=____;
                }

                temp     = a[____];
                a[____] = a[____];
                a[____] = temp;
        }
}
```

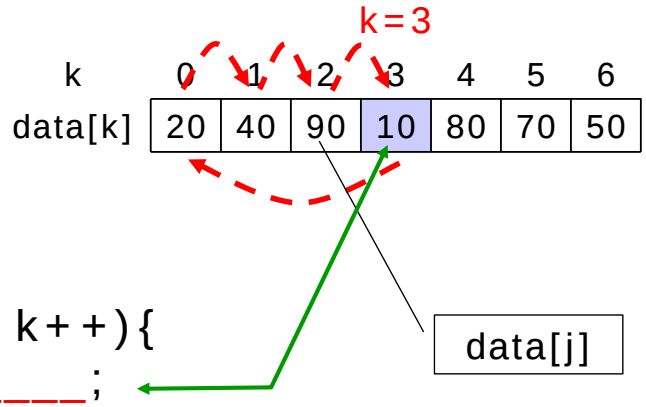
if (min != i) ...

插入排序法 (insertion sort)

把目前data[k]
移到左面最合適的地方

data[i]	0	1	2	3	4	5	6
start	90	20	40	10	80	70	50
k=1	20	90	40				
k=2	20	40	90	10			
k=3	10	20	40	90	80		
k=4	10	20	40	80	90	70	
k=5	10	20	40	70	80	90	50
k=6	10	20	40	50	70	80	90

插入排序法 (insertion sort)

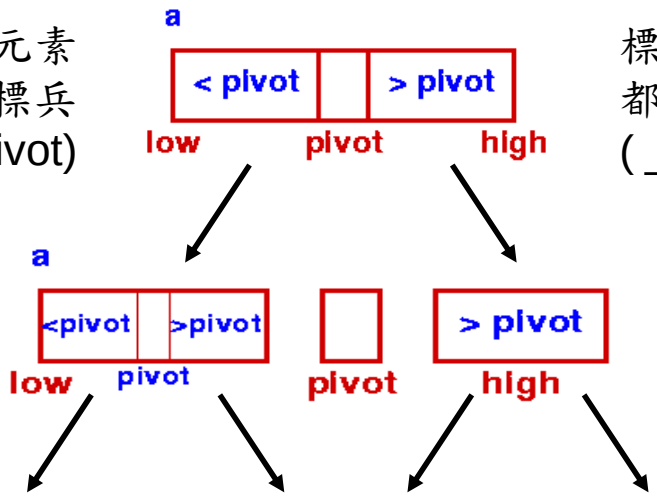



```
void insertionSort (i){
        int k, j, itemk;
        for (k=1;  k<max;  k++){
                itemk = ________;

                j = k-1;

                data[3]=data[2]; j--;
                data[2]=data[1]; j--;
                data[1]=data[0]; j--;

                data[0] = itemk;
        }
}
```

快速排序法 (quick sort)

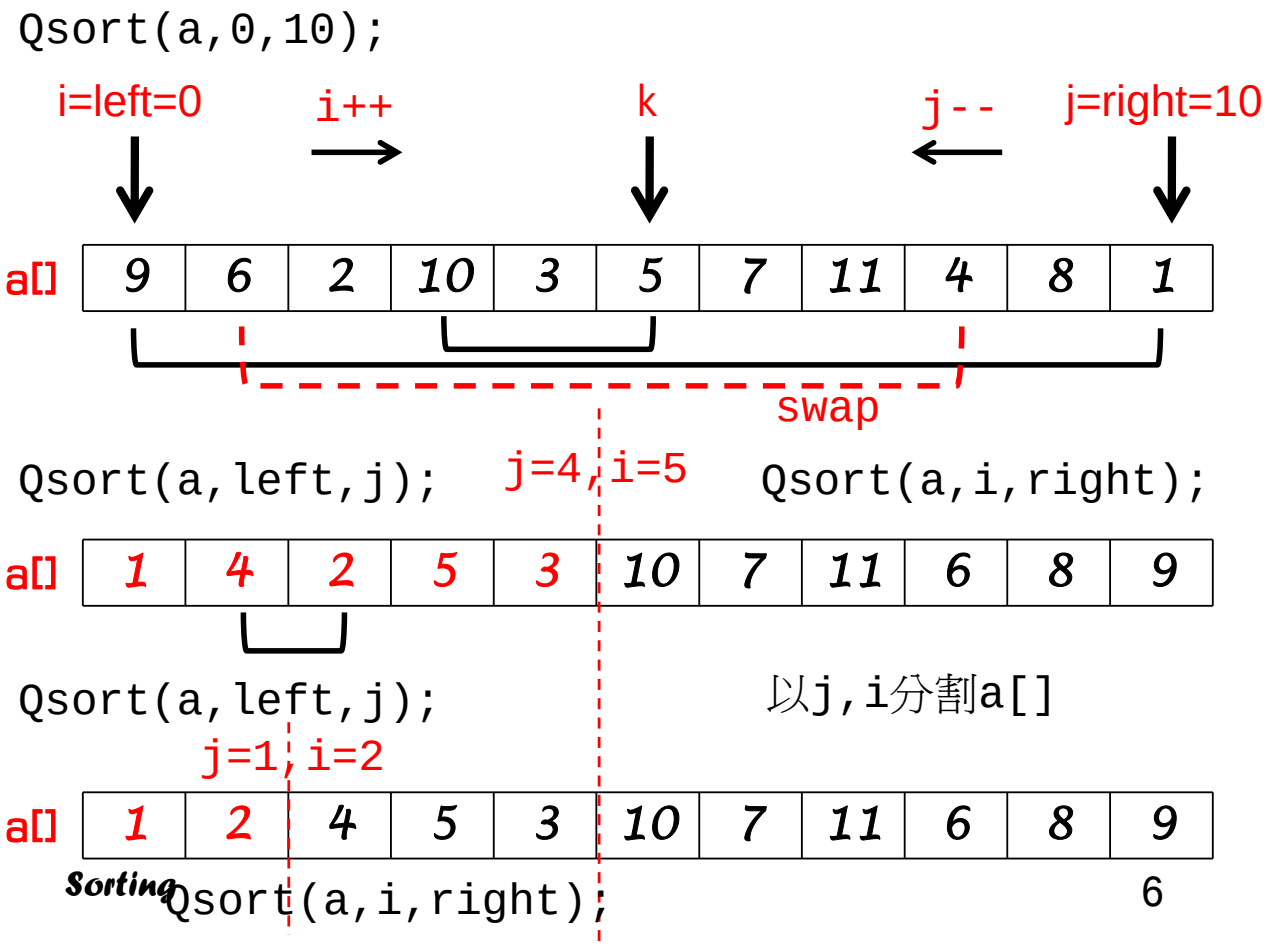
將陣列中間的元素設為標兵 (pivot),
以標兵將陣列a[ ] 分成兩半,

標兵左邊的元素
都小於標兵
(___ pivot)



標兵右邊的元素
都不小於標兵
(___ pivot)

再依相同的方式
將標兵左右兩半排序



```
void Quicksort (int data[],int left,int right) {
        int pivot,tmp, i=left, j=right;
        pivot = data[________________];
        do {
            while(data[__] __ pivot) i++;      // 找i&j
            while(data[__] __ pivot) j--;
            if (i<=j) {
                swap(&data[i],&data[j]);       // 互換
                i++;
                j--;
            }
        } while (i <= j);

        if (left < j)   Quicksort (data,____,_____);
        if (i < right) Quicksort (data,____,_____);
}
```



```
void Bubblesort (int data[], int max) {
        int i,j;
        for (i=0; i<max-1; i++) {
                for (j=0; j<max-i-1; j++)
                        if (data[j] > data[j+1])
                                swap(&data[j],&data[j+1]);
        }
}
```

```
void swap (int __, int __) {
        int t;
        t     = ___;
        ___ = ___;
        ___ = t;
        }
}
```

```
// 互相交換 (錯誤)
void swap (int x, int y) {
        int t;
        t = x;
        x = y;
        y = t;
}
```