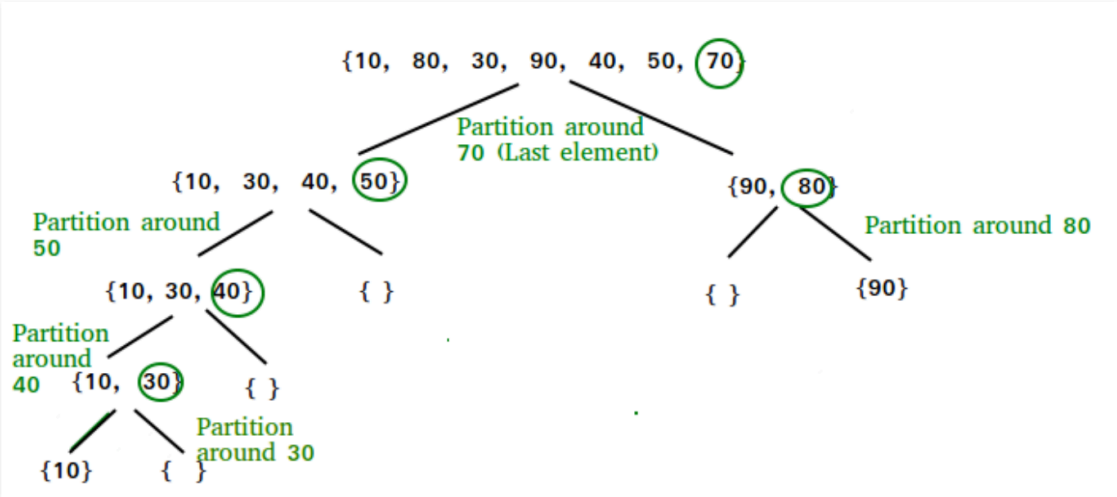
一開始在上課時，理解Quick Sort不是太難，跟Insertion Sort比起來感覺Quick Sort會比較好進行。於是我先參考了老師Github 的Quick Sort解答，老師建立了兩個Class分別為 ListNode及Solution，但Solution的部分我比較難理解，接著老師也有在課堂上給我們看了Recursive Pseudocode的影片，覺得有些理解，所以也上網參考了別人的做法。這裏我直接用partition（）的方法用來根據指定的分隔符將字符串進行分割。接著看這張圖，基本上程式碼原理是跟的這張圖的意思來進行。



Partition（arr,low,high），使用array、low左邊、high右邊。以最右邊的為pivot，那Arr[i]就是比pivot小的最右邊的數，所以i+1跟high交換值，只是要把中間那個數放到他應該要放的位子，舉例來說：假設現在數列是 1 8 2 6 4 7 5這樣，那排完應該是1 2 4 6 8 7 5，但是這樣還沒排完，因為pivot要放中間，所以才會換1 2 4 5 8 7 6。

```
def partition(arr,low,high):  #partition() 方法用來根據指定的分隔符將字符串進行分割。
    i = ( low-1 )         # index of smaller element，將位置往左邊，把i設成最左邊-1
    pivot = arr[high]     # pivot，選一個中心點，右邊的

    for j in range(low , high): #for 與 in 連用， in 後面接多個元素的物件。

        if   arr[j] < pivot: # If current element is smaller than the pivot

            # increment index of smaller element
            i = i+1 #原本左邊的要+1
            arr[i],arr[j] = arr[j],arr[i] #把i與j得值交換

    arr[i+1],arr[high] = arr[high],arr[i+1]
    return ( i+1 )

def quickSort(arr,low,high):
    if low < high:

        pi = partition(arr,low,high)

        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)

# Driver code to test above
arr = [10, 7, 8, 9, 1, 5]
n = len(arr)
quickSort(arr,0,n-1)
print ("Sorted array is:")
for i in range(n):
    print ("%d" %arr[i]),
```

接著Pi 就是中間pivot的位置，所以回傳他，因為pi是在正確的位子。舉一個例：數列是 1 8 4 6 2 7 5，第一次partition後會變成 1 4 2 5 8 7 6。除了5 (pivot) 的位子是正確的之外，其他的不能保證是正確的，所以一個 1 4 2 要做一次 partition及8 7 6 要做一次partition。程式碼：quicksort(arr,low,pi-1)及quicksort(arr,pi+1,high)。