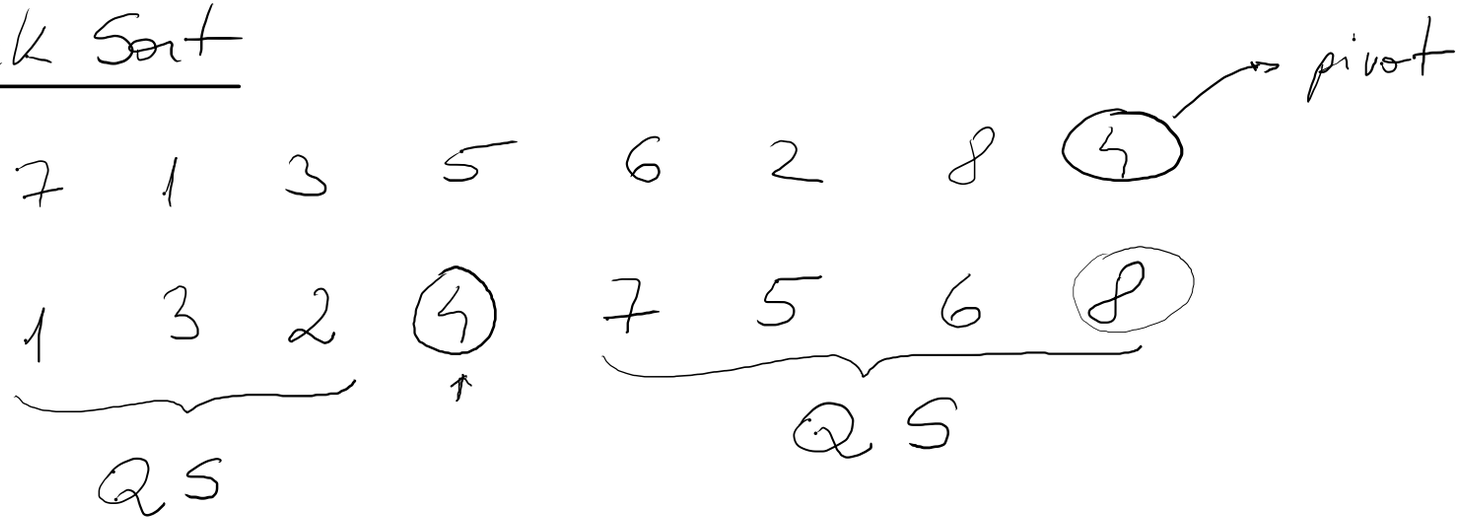
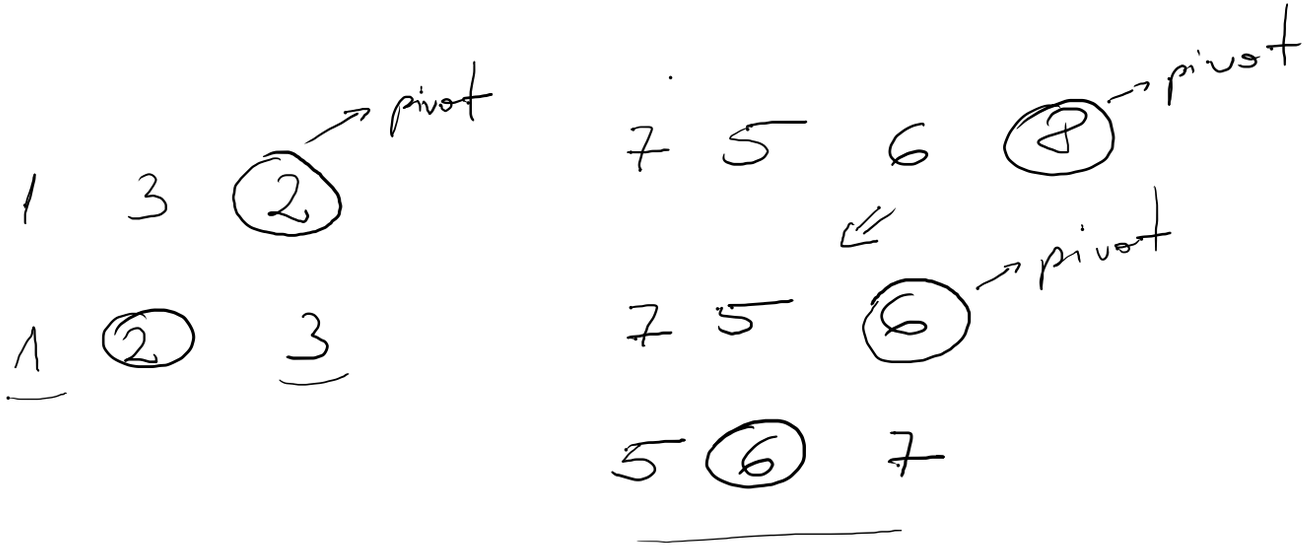
Quick Sort


7 1 3 5 6 2 8 4
pivot
1 3 2 4
7 5 6 8
QS
QS


1 3 2
pivot
1 2 3
7 5 6 8
pivot
7 5 6
pivot
5 6 7

```
void QuickSort(int st, int dr){
    if(st < dr){ → indexul pivotului după rearanjare
        int pindex = partitionare(st, dr);
        QuickSort(st, pindex - 1);
        QuickSort(pindex + 1, dr);
    }
}
```

→ funcția de partitionare va rearanja elementele șirului față de un pivot, arbitrar ales, în așa fel încât valorile mai mici ca el să fie înainte și cele mai mari, după el.

```
int partitionare(int st, int dr) {
    int pivot = a[dr];
    int poz = st;
    for(int i = st; i < dr; ++i)
        if(a[i] < pivot){
            swap(a[i], a[poz]);
            poz++;
        }
    swap(a[dr], a[poz]);
    return poz;
}
```