

```
QuickSort(from,to):  
  if to>from  
    i:=Partition(from,to);  
    QuickSort(from,i-1);  
    QuickSort(i+1,to);
```

RandomizedPartition(from,to):

```
// post: pivot is at its correct position A[ret]
```

```
//      from<=j<=ret => A[j]<=pivot
```

```
//      ret<j<=to => A[j]>pivot
```

```
*** change here ***
```

```
swap(A[to],A[random(from,to)]); // pick a random index as a pivot
```

```
*****
```

```
pivot:=A[to];
```

```
i:=from-1;
```

```
for j:=from to to
```

```
    // inv: from<=k<=i => A[k]<=pivot
```

```
    //      i<k<j => A[k]>pivot
```

```
    if A[j]<=pivot
```

```
        i:=i+1
```

```
        swap(A[j],A[i])
```

```
return i;
```

```
clear array count[0..k];

for i:=1 to n
    count[A[i].key]++

pos[0]:=1;
for i:=1 to k
    pos[i]:=pos[i-1]+count[i-1];
// now pos[i] is the first position where
// integer i will come in the sorted array B

for i:=1 to n
    B[pos[A[i].key]]:=A[i];
    pos[A[i].key]++;
```

```
for i:=1 to d
  use counting sort to sort A[1..n] by
  the d-th least significant digit (i.e. k=10)
  // inv: array is sorted by last i digits
```