Instead of always using A[r] as the pivot, using random sampling, we will use a randomly chosen element from subarray A[p..r].
Because the pivot element is randomly chosen, we expect the split of the input array to be reasonably well balanced on average.
The worst case running time for randomized Quick-sort is $\Theta(n \log n)$.

```
RANDOMIZED-PARTITION (A, p, r)
1.  i ← RANDOM(p, r)
2.  exchang A[i] ↔ A[r]
3.  return PARTITION(A, p, r)

PARTITION(A, p, r)
1.  x ← A[r]
2   i ← p-1
3   for j ← p to r-1
4   {if x ≥ A[j]
5     {i = i+1
6       exchange A[i] ↔ A[j]
7     }}
8   exchang A[i+1] ↔ A[r]
9   return i+1
RANDOMIZED-QUICKSORT(A, p, r)
1  if (p < r)
2  {  q ← RANDOMIZED-PARTITION(A, p, r)
3     RANDOMIZED-QUICKSORT(A, p, q-1)
4     RANDOMIZED-QUICKSORT(A, q+1, r)
5  }
```