K-th order statistic in O(N)

Table of Contents

Given an array **A** of size **N** and a number **K**. The challenge is to find **K**-th largest number in the array, i.e., **K**-th order statistic.

The basic idea - to use the idea of quick sort algorithm. Actually, the algorithm is simple, it is more difficult to prove that it runs in an average of O(N), in contrast to the quick sort.

Implementation (not recursive):

```
template <class T>
T order_statistics (std::vector<T> a, unsigned
{
    using std::swap;
    for (unsigned l=1, r=n; ; )
    {
        if (r <= l+1)
        {
            // the current part size is either
            if (r == l+1 && a[r] < a[l])
```

```
            swap (a[l], a[r]);
        return a[k];
    }

    // ordering a[l], a[l+1], a[r]
    unsigned mid = (l + r) >> 1;
    swap (a[mid], a[l+1]);
    if (a[l] > a[r])
        swap (a[l], a[r]);
    if (a[l+1] > a[r])
        swap (a[l+1], a[r]);
    if (a[l] > a[l+1])
        swap (a[l], a[l+1]);

    // performing division
    // barrier is a[l + 1], i.e. median am
    unsigned
        i = l+1,
        j = r;
    const T
        cur = a[l+1];
    for (;;)
    {
        while (a[++i] < cur) ;
        while (a[--j] > cur) ;
        if (i > j)
            break;
        swap (a[i], a[j]);
    }

    // inserting the barrier
    a[l+1] = a[j];
    a[j] = cur;
```

```
            // we continue to work in that part, w
            if (j >= k)
                r = j-1;
            if (j <= k)
                l = i;
        }
    }
```

To note, in the standard C ++ library, this algorithm has already been implemented - it is called nth_element.

Practice Problems

- CODECHEF: Median