Last update: April 16, 2025 Translated From: e-maxx.ru

Kth order statistic in $O(N)$

Given an array A of size N and a number K. The problem 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 n, unsigned k)
{
    using std::swap;
    for (unsigned l=1, r=n; ; )
    {
        if (r <= l+1)
        {
            // the current part size is either 1 or 2, so it is easy to find
the answer
            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 among a[l], a[l + 1], a[r]
        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, which must contain the required
element
        if (j >= k)
            r = j-1;
        if (j <= k)
            l = i;
    }
}
```

Notes

- The randomized algorithm above is named quickselect. You should do random shuffle on A before calling it or use a random element as a barrier for it to run properly. There are also deterministic algorithms that solve the specified problem in linear time, such as median of medians.
- std::nth_element solves this in C++ but gcc's implementation runs in worst case $O(n \log n)$ time.
- Finding K smallest elements can be reduced to finding K-th element with a linear overhead, as they're exactly the elements that are smaller than K-th.

Practice Problems

- Leetcode: Kth Largest Element in an Array
- CODECHEF: Median

Contributors:

ErzhanDS (73.68%) adamant-pwn (11.84%) jakobkogler (5.26%) likecs (2.63%) shark_s (2.63%) mhayter (1.32%) random1001guy (1.32%) dufferzafar (1.32%)