SELECTING THE kTH SMALLEST ELEMENT

GODFREY MUGANDA

1. The Selection Problem

Consider the problem of finding the kth smallest element of an array of n integers, $a[0..n-1]$. We use a recursive divide and conquer approach, and we consider the more general problem of finding the kth smallest element in a segment $a[L..U]$ determined by two indices, a lower index L and an upper index U.

Given an array $a[L..U]$, we select the element $v = a[L]$ and then move elements around so that $a[L..U]$ becomes partitioned into three ranges by two indices p and q where $L \leq p \leq q \leq U$:

(1) every element of $a[L..p-1]$ is less than v,
(2) every element of $a[p..q]$ is equal to v,
(3) every element of $a[q+1..U]$ is greater than v.

We can do this by writing a method

int [ ] partition (int [ ] a, int L, int U)

that returns and array with two elements, containing p at index 0 and q at index 1.

Note that for any two indices r and s where $r \leq s$, there are

$$s - r + 1$$

elements in the array segment $a[r..s]$.

Given such a partition() method, we can write the method

select(int [ ] a, int L, int U)

to compute the kth smallest element of $a[L..U]$ as follows.

(1) if there are more than k elements in $a[L..p-1]$, that is, if $k \leq p - L$, then recursively return
    select$(a, L, p-1, k)$.
(2) if $p - L < k \leq q - L + 1$, meaning k is greater than the length of $a[L..p-1]$ but is less or equal to the length of $a[L..q]$, then return $a[p]$.
(3) if $k > q - L + 1$, which is the length of $a[L, q]$, then recursively return
    select$(a, q+1, U, k-(q-L+1))$.

The select method can be written as follows.

```
int select(a, L, U)
{
  int [ ] parts = partition(a, L, U);
  int p = parts[0];
  int q = parts[1];
  if (k ≤ p − L) return select(a, L, p − 1, k);
  if (p − L < k and k ≤ q − L + 1) return a[p];
  else
        return select(a, q + 1, U, k − (q − L + 1));
}
```

2. Writing the partition Method

The partition method, as we have said, works as follows.

Given an array $a[L..U]$, we select the element $v = a[L]$ and then move elements around so that $a[L..U]$ becomes partitioned into three ranges by two indices p and q where $L \leq p \leq q \leq U$:

(1) every element of $a[L..p-1]$ is less than v,
(2) every element of $a[p..q]$ is equal to v,
(3) every element of $a[q+1..U]$ is greater than v.

The partition method works in a manner very similar to the partition method used in Quicksort, and will be covered in lecture.