Randomized Selection

Andreas Klappenecker



Randomized Selection

Randomized-Select(A,p,r,i) // return the ith smallest elem. of A[p..r] if (p == r) then return A[p];

q := Randomized-Partition(A,p,r); // compute pivot k := q-p+1; // number of elements <= pivot

if (i==k) then return A[q]; // found ith smallest element elseif (i < k) then return Randomized-Select(A,p,q-1,i); else Randomized-Select(A,q+1,r, i-k);

Partition

Randomized-Partition(A,p,r)
i := Random(p,r);
swap(A[i],A[r]);
Partition(A,p,r);

Almost the same as Partition, but now the pivot element is not the rightmost element, but rather an element from A[p..r] that is chosen uniformly at random.

Running Time

The worst case running time of Randomized-Select is (n²)
 The expected running time of Randomized-Select is (n)
 No particular input elicits worst case running time.

Running Time

- Let T(n) denote the random variable describing the running time of Randomized-Select on input of A[p..r].
- Suppose A[p..r] contains n elements. Each element of A[p..r] is equally likely to be the pivot, so A[p..q] has size k with probability 1/n.
- $X_k = I\{\text{the subarray } A[p..q] \text{ has } k \text{ elements}\}$
- \odot E[X_k] = 1/n (assuming elements are distinct)

Running Time

Let's assume that T(n) is monotonically growing.

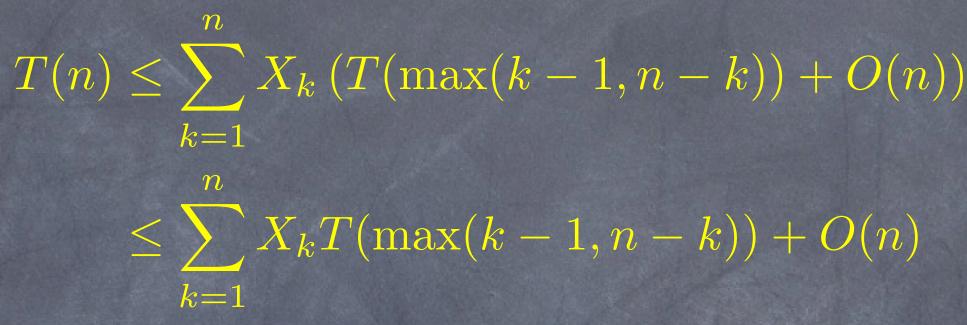
Three choices: (a) find ith smallest element right away, (b) recurse on A[p..q-1], or (c) recurse on A[p+1,r].

• When $X_k = 1$, then

A[p..q-1] has k-1 elements and

A[p+1..r] has n-k elements.

Recurrence



- Assume that we always recurse to larger subarray - O(n) for partitioning - $X_k = 1$ for a single choice, so partition once



Expected Running Time

 $E[T(n)] \le \sum E[X_k T(\max(k-1, n-k))] + O(n)$ k=1 $= \sum E[X_k]E[T(\max(k-1, n-k))] + O(n)$ k=1 $= \sum_{n=1}^{n} \frac{1}{n} E[T(\max(k-1, n-k))] + O(n)$



Expected Running Time

$E[T(n)] \le \sum_{k=\lfloor n/2 \rfloor}^{n} \frac{2}{n} E[T(k)] + O(n)$

One can prove by induction that E[T(n)] = O(n).



