Randomized Select and Randomized Quicksort

1 Randomized Select

The algorithm RANDOMIZED-SELECT selects out the k-th order statistics of an arbitrary array.

1.1 Algorithm

The algorithm RANDOMIZED-SELECT works by partitioning the array \( A \) according to RANDOMIZED-PARTITION, and recurses on one of the resulting arrays.

\[
\text{RANDOMIZED-SELECT}(A, p, r, i) \\
1 \quad \text{if } p = r \\
2 \quad \quad \text{then return } A[p] \\
3 \quad q \leftarrow \text{RANDOMIZED-PARTITION}(A, p, r) \\
4 \quad k \leftarrow q - p + 1 \\
5 \quad \text{if } i \leq k \\
6 \quad \quad \text{then return } \text{RANDOMIZED-SELECT}(A, p, q, i) \\
7 \quad \quad \text{else return } \text{RANDOMIZED-SELECT}(A, q + 1, r, i - k)
\]

\[
\text{RANDOMIZED-PARTITION}(A, p, r) \\
1 \quad i \leftarrow \text{RANDOM}(p, r) \\
2 \quad \text{exchange } A[p] \leftrightarrow A[i] \\
3 \quad \text{return } \text{PARTITION}(A, p, r)
\]

Both of the algorithms above are as in CLRS.

1.2 Analysis of Running Time

Let \( T(n) \) be the expected running time Randomized Select. We would like to write out a recursion for it.

Let \( E_i \) denote the event that the random partition divides the array into two arrays of size \( i \) and \( n - i \). Then we see that

\[
T(n) \leq n + \sum_{i=0}^{n-1} Pr(E_i) \left( \max(T(i), T(n - i)) \right),
\]

(1)
Recitation 4: Randomized Select and Randomized Quicksort

where by taking the max we assume that we are recursing on the larger subarray (hence we have the less than or equal sign).

For simplicity, let us assume that \( n \) is even. Note that \( \max (T(i), T(n - i)) \) is always the same as \( \max (T(n - i), T(i)) \). This allows us to extend the chain of inequalities to

\[
T(n) \leq n + 2 \sum_{i=0}^{n/2-1} \Pr(E_i) \left( \max (T(i), T(n - i)) \right).
\]

Also, since the partition element is chosen randomly, it is equally likely to partition the array into sizes 0, 1, \( \cdots \), \( n - 1 \). So \( \Pr(E_i) = \frac{1}{n} \) for all \( i \). This leads us to

\[
T(n) \leq n + 2 \sum_{i=0}^{n/2-1} \left( \max (T(i), T(n - i)) \right).
\]

We will not show, via substitution, that \( T(n) = O(n) \).

**Theorem 1** Let \( T(n) \) denote the expected running time of randomized select. Then \( T(n) = O(n) \).

**Proof.** We will show by the method of substitution. Let’s say that \( T(n) \leq cn \), and check that it works.

We must first check the base case. This is obvious, however, since \( T(n') \) is a constant for some small constant \( n' \).

Now let us check the inductive case. Assume that \( T(k) \leq ck \) for all \( k < n \), and we now want to show that \( T(n) \leq cn \).

\[
T(n) \leq n + 2 \sum_{i=0}^{n/2-1} \left( \max (T(i), T(n - i)) \right) \leq n + 2 \sum_{i=0}^{n/2-1} \left( \max (ci, c(n - i)) \right).
\]

We note that that this is the same as

\[
n + 2 \sum_{i=n/2}^{n-1} ci.
\]

The term \( \frac{2}{n} \sum_{i=n/2}^{n-1} (ci) \) is the same as \( \frac{2c}{n} \sum_{i=n/2}^{n-1} i \). So we get

\[
T(n) \leq n + c \left( \frac{2}{n} \sum_{i=n/2}^{n-1} i \right) \leq n + c (3n/4) = n \left( 1 + \frac{3c}{4} \right).
\]

Hence if we take \( c = 4 \) (which works for the case \( T(1) \leq 4 \) as well) we get

\[
T(n) \leq n \left( 1 + \frac{3 \times 4}{4} \right) = n (1 + 3) = 4n,
\]

as we wanted.
2 Randomized Quicksort

2.1 Algorithm

The algorithm RANDOMIZED-QUICKSORT works by partitioning the array $A$, and recursively sorts both partitions.

 RANDOMIZED-QUICKSORT($A, p, r$)
1  if $p < r$
2    then $q \leftarrow$ RANDOMIZED-PARTITION($A, p, r$)
3    RANDOMIZED-QUICKSORT($A, p, q - 1$)
4    RANDOMIZED-QUICKSORT($A, q + 1, r$)

2.2 Analysis of Running Time

Let $T(n)$ be the expected running time Randomized Quicksort. Let $E_i$ denote the event that the array is partitioned into two arrays of size $i$ and $n - i - 1$. The pivot value is not included in either partition. Then we have

$$T(n) \leq \sum_{i=0}^{n-1} Pr(E_i)(T(i) + T(n - i - 1) + \Theta(n)),$$

(8)

Because $Pr(E_i) = \frac{1}{n}$ for all $i$, we have

$$T(n) \leq \frac{1}{n} \sum_{i=0}^{n-1} (T(i) + T(n - i - 1) + \Theta(n)),$$

(9)

$$T(n) \leq \frac{2}{n} \sum_{i=0}^{n-1} T(i) + \Theta(n),$$

(10)

The same as Randomized select, we use induction to prove that $T(n) = \Theta(n \log n)$. Suppose $T(n) \leq cn \log n$ for some constant $c > 0$. Notice the fact that

$$\sum_{i=0}^{n-1} i \log i \leq \frac{1}{2} n^2 \log n - \frac{1}{8} n^2,$$

(11)

Then for the inductive step, we have

$$T(n) \leq \frac{2}{n} \sum_{i=0}^{n-1} ci \log i + \Theta(n),$$

(12)

$$T(n) \leq \frac{2c}{n} (\frac{1}{2} n^2 \log n - \frac{1}{8} n^2) + \Theta(n),$$

(13)

$$T(n) \leq cn \log n - (\frac{cn}{4} - \Theta(n)),$$

(14)

When $c$ is chosen large enough, $T(n) \leq cn \log n$. 