

Back to Randomized Quicksort

Procedure $Q_S(S)$;

Input: A set S .

Output: The set S in sorted order.

1. If $|S| \leq 1$ then return S , else
- 2.(a) Choose a random element y uniformly from S .
(b) Compare all elements of S to y . Let

$$S_1 = \{x \in S - \{y\} \mid x \leq y\}$$

$$S_2 = \{x \in S - \{y\} \mid x > y\}.$$

(Elements in S_1 and S_2 are in the same order as in S .)

- (c) Return the list:

$$Q_S(S_1), y, Q_S(S_2).$$

Analysis

Let T = number of comparisons in a run of QuickSort.

Theorem 1.

$$E[T] = O(n \log n).$$

Proof. Let s_1, \dots, s_n be the elements of S in sorted order.

For $i = 1, \dots, n$, and $j > i$, define 0-1 random variable $X_{i,j}$, s.t.

$X_{i,j} = 1$ iff s_i is compared to s_j in the run of the algorithm.

The number of comparisons in running the algorithm is

$$T = \sum_{i=1}^n \sum_{j>i} X_{i,j}.$$

We are interested in $E[T]$.

What is the probability that $X_{i,j} = 1$?

s_i is compared to s_j iff either s_i or s_j is chosen as a “split item” before any of the $j - i - 1$ elements between s_i and s_j are chosen.

Elements are chosen uniformly at random \rightarrow elements in the set $[s_i, s_{i+1}, \dots, s_j]$ are chosen uniformly at random.

$$Pr(X_{i,j} = 1) = \frac{2}{j - i + 1}.$$

$$E[X_{i,j}] = \frac{2}{j - i + 1}.$$

$$E[T] = E\left[\sum_{i=1}^n \sum_{j>i} X_{i,j}\right] =$$

$$\sum_{i=1}^n \sum_{j>i} E[X_{i,j}] = \sum_{i=1}^n \sum_{j>i} \frac{2}{j - i + 1} \leq$$

$$2n \sum_{k=1}^n \frac{1}{k} \leq 2nH_n = 2n \log n + O(n). \quad \square$$

Probabilistic Analysis of QuickSort

Theorem 2. *The expected run time of (deterministic) Quicksort on a random input, uniformly chosen from all possible permutation of S is $O(n \log n)$.*

Proof.

Set $X_{i,j}$ as before.

If all permutations have equal probability, all permutations of S_i, \dots, S_j have equal probability, thus

$$Pr(X_{i,j}) = \frac{2}{j - i + 1}.$$

$$E\left[\sum_{i=1}^n \sum_{j>i} X_{i,j}\right] = O(n \log n).$$

□

Randomized Algorithms:

- Analysis is true for **any** input.
- The sample space is the space of random choices made by the algorithm.
- Repeated runs are independent.

Probabilistic Analysis:

- The sample space is the space of all possible inputs.
- If the algorithm is **deterministic** repeated runs give the same output.

Randomized Algorithm classification

A **Monte Carlo Algorithm** is a randomized algorithm that may produce an incorrect solution.

For decision problems: A **one-side error** Monte Carlo algorithm errs only on one possible output, otherwise it is a **two-side error** algorithm.

A **Las Vegas** algorithm is a randomized algorithm that **always** produces the correct output.

In both types of algorithms the run-time is a random variable.

Selection

Input: A set S of n distinct elements, and an integer $1 \leq i \leq n$.

Output: The i -th smallest element in S .

Random-Select(S, i) ($1 \leq i \leq |S|$).

1. If $|S| = 1$ then return S .
2. Choose a random element y uniformly from S
3. Compare all elements of S to y . Let

$$S_1 = \{x \in S \mid x < y\}, \quad S_2 = \{x \in S \mid x > y\}.$$

4. If $|S_1| = i - 1$ then return y
 elseif $|S_1| \geq i$ then return **Random-Select**(S_1, i)
 else return **Random-Select**($S_2, i - |S_1| - 1$);

Correctness and Worst-case runtime

Theorem 3. *The algorithm returns a singleton with the correct value.*

Proof.

By induction on the depth of the recursion.

In each call to $\text{Random-Select}(S', i')$, $i' \leq |S'|$ and the i' largest element in S' is the i largest element in S .

When $|S'| = 1$, it includes the i largest element in S . \square

Run-time

Theorem 4. *The worst-case run-time of the algorithm is $O(n^2)$.*

Proof. In the worst case the size of the set that includes the i -th largest element decreases by one in each iteration. \square

Expected run-time

Theorem 5. *The expected run-time of the algorithm is $O(n)$.*

Proof. Without loss of generality we can assume that in each iteration the i -th largest element is in the larger of the two sets S_1 and S_2 .

Let r.v. $T(n)$ = denote the run-time on a set of n elements.

For each $k = 1, 2, \dots, n$, define indicator r.v.s X_k denoting the event that the set S_1 has exactly $k - 1$ elements.

$$E[X_k] = 1/n$$

$$\begin{aligned}
T(n) &\leq \sum_{k=1}^n X_k (T(\text{Max}[k-1, n-k]) + \alpha n) \\
&= \sum_{k=1}^n X_k T(\text{Max}[k-1, n-k]) + \alpha n \\
E[T(n)] &\leq E \left[\sum_{k=1}^n X_k T(\text{Max}[k-1, n-k]) + \alpha n \right] \\
&= \sum_{k=1}^n E[X_k T(\text{Max}[k-1, n-k])] + \alpha n \\
&= \sum_{k=1}^n E[X_k] E[T(\text{Max}[k-1, n-k])] + \alpha n \\
&= \frac{1}{n} \sum_{k=1}^n E[T(\text{Max}[k-1, n-k])] + \alpha n \\
&\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \alpha n
\end{aligned}$$

We show that $E[T(n)] \leq cn$ for some constant $c > 0$, by induction on n .

$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \alpha n \\ &= \frac{2c}{n} \left(\sum_k^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k \right) + \alpha n \\ &\leq \frac{3}{4}cn + c/2 + \alpha n \\ &\leq cn \end{aligned}$$

for a suitably large c .

Theorem 6. *If X and Y are independent random variable*

$$E[XY] = E[X] \cdot E[Y],$$

Proof.

$$E[XY] = \sum_i \sum_j i \cdot j \Pr((X = i) \cap (Y = j)) =$$

$$\sum_i \sum_j ij \Pr(X = i) \cdot \Pr(Y = j) =$$

$$\left(\sum_i i \Pr(X = i) \right) \left(\sum_j j \Pr(Y = j) \right).$$

□

□

Linear Time Deterministic Selection Algorithm

Theorem 7. *There is a deterministic algorithm that finds the i -th smallest element in an unsorted array of n elements in $O(n)$ time.*

Select (S, i) - Selects the i -th smallest element in the set S .

1. $n = |S|$. If $n = 1$ then return S .
2. Partition S into $\lfloor \frac{n}{5} \rfloor$ groups of 5 elements each, and a leftover group of up to 4 elements.
3. Find the median of each of the groups, let R be the set of these $\lfloor \frac{n}{5} \rfloor$ values.
4. $y = \text{Select}(R, \lceil \frac{|R|}{2} \rceil)$;
5. Compare all elements of S to y . Let
$$S_1 = \{x \in S \mid x < y\}, \quad S_2 = \{x \in S \mid x > y\}.$$
6. If $|S_1| = i - 1$ then return y
 elseif $|S_1| \geq i$ then return $\text{Select}(S_1, i)$
 else return $\text{Select}(S_2, i - |S_1| - 1)$;

Correctness and Runtime

Theorem 8. *The algorithm returns the correct value.*

Proof. By induction on the depth of the recursion. \square

Run-time

Theorem 9. *The run-time of the algorithm is $O(n)$.*

Proof.

How many elements in S are larger than y , the “median of medians” value computed in step 4 of the algorithm?

Excluding the leftover group, and the group that includes y , in at least half of the remaining groups, there are at least three elements that are $> y$. Thus, at least

$$3\left(\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil - 2\right) \geq \frac{3n}{10} - 6$$

in S are greater than y .

Similarly, at least $\frac{3n}{10} - 6$ elements in S are $\leq y$.

Thus, select is called in step 6 with at most $\frac{7n}{10} + 6$ elements.

$T(n)$ = run-time on sets of size n .

$$T(n) \leq T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10} + 6) + \alpha n.$$

We show that $T(n) \leq cn$ for some constant $c > 0$, by induction on n .

$$\begin{aligned} T(n) &\leq c(n/5 + 1) + c(7n/10 + 6) + \alpha n \\ &\leq 9cn/10 + 7c + \alpha n \\ &\leq cn \end{aligned}$$

for $n > 70$ and sufficiently large c . \square