A Slick Analysis of QuickSort

Let $Q(A)$ be number of comparisons done on input array $A$:

- $R_{ij}$: event that rank $i$ element is compared with rank $j$ element, for $1 \leq i < j \leq n$.
- $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise 0.
- $Q(A) = \sum_{1 \leq i < j \leq n} X_{ij}$.

By linearity of expectation,

$$E[Q(A)] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{1 \leq i < j \leq n} \Pr[R_{ij}].$$

**Question:** What is $\Pr[R_{ij}]$?

With ranks: $7 5 9 1 3 4 8 6$ and $6 4 8 1 2 3 7 5$.

As such, probability of comparing 5 to 8 is $\Pr[R_{4,7}]$.

- If pivot too small (say 3 [rank 2]). Partition and call recursively:
  - Decision if to compare 5 to 8 moved to subproblem.

- If pivot too large (say 9 [rank 8]):
  - Decision if to compare 5 to 8 moved to subproblem.
A Slick Analysis of QuickSort

Question: What is \( \Pr[R_{i,j}] \)?

Choosing a pivot using priorities

- Assign every element in array is a random priority (say in \([0, 1]\)).
- \( \text{pivot} \) = the element with lowest priority in subproblem.

\[ \iff R_{i,j} \text{ happens if either } i \text{ or } j \text{ have lowest priority out of elements in rank } i \ldots j. \]

- There are \( k = j - i + 1 \) relevant elements.

- \( \Pr[R_{i,j}] = \frac{2}{k} = \frac{2}{j - i + 1}. \)
A Slick Analysis of QuickSort

Lemma

\[ \Pr[R_{ij}] = \frac{2}{j-i+1}. \]

Proof.

1. Let \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \) be sort of \( A \).
2. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \).
3. Observation: \( a_i \) is compared with \( a_j \) \( \iff \) either \( a_i \) or \( a_j \) is chosen as a pivot from \( S \) at separation.
4. Observation: Given: Pivot chosen from \( S \).
   The probability that it is \( a_i \) or \( a_j \) is exactly \( \frac{2}{|S|} = \frac{2}{(j-i+1)} \) since the pivot is chosen uniformly at random from the array.

Yet another analysis of QuickSort

You should never trust a man who has only one way to spell a word

Part II

Quick sort with high probability

Consider element \( e \) in the array.

\( S_1, S_2, \ldots, S_k \): subproblems \( e \) participates in during QuickSort execution:

Definition

\( e \) is lucky in the \( j \)th iteration if \( |S_j| \leq \frac{3}{4} |S_{j-1}| \).

Key observation: The event that \( e \) is lucky in \( j \)th iteration...

... is independent of the event that \( e \) is lucky in \( k \)th iteration, (If \( j \neq k \))

\( X_j = 1 \iff e \) is lucky in the \( j \)th iteration.
Yet another analysis of QuickSort

Continued...

Claim

\[ \Pr[X_j = 1] = 1/2. \]

Proof.

1. \( X_j \) determined by \( j \) recursive subproblem.
2. Subproblem has \( n_{j-1} = |X_{j-1}| \) elements.
3. \( j \)th pivot rank \( \in [n_{j-1}/4, (3/4)n_{j-1}] \implies e \) lucky in \( j \)th iter.
4. \( \Pr[e \text{ is lucky}] \geq |[n_{j-1}/4, (3/4)n_{j-1}]|/n_{j-1} = 1/2. \]

Observation

If \( X_1 + X_2 + \ldots + X_k = \lceil \log_{4/3} n \rceil \) then \( e \) subproblem is of size one. Done!

Observation

Probability \( e \) participates in \( \geq k = 40 \lceil \log_{4/3} n \rceil \) subproblems. Is equal to

\[
\Pr[X_1 + X_2 + \ldots + X_k \leq \lceil \log_{4/3} n \rceil] \\
\leq \Pr[X_1 + X_2 + \ldots + X_k \leq k/4] \\
\leq 2 \cdot 0.68^{k/4} \leq 1/n^5.
\]

Conclusion

QuickSort takes \( O(n \log n) \) time with high probability.

Randomized Quick Selection

Input Unsorted array \( A \) of \( n \) integers

Goal Find the \( j \)th smallest number in \( A \) (rank \( j \) number)

Randomized Quick Selection

1. Pick a pivot element \( \text{uniformly at random} \) from the array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
3. Return pivot if rank of pivot is \( j \).
4. Otherwise recurse on one of the arrays depending on \( j \) and their sizes.

Algorithm for Randomized Selection

Assume for simplicity that \( A \) has distinct elements.

\[ \text{QuickSelect}(A, j): \]

Pick pivot \( x \) uniformly at random from \( A \)
Partition \( A \) into \( A_{\text{less}} \), \( x \), and \( A_{\text{greater}} \) using \( x \) as pivot
if \( |A_{\text{less}}| = j - 1 \) then
    return \( x \)
if \( |A_{\text{less}}| \geq j \) then
    return \( \text{QuickSelect}(A_{\text{less}}, j) \)
else
    return \( \text{QuickSelect}(A_{\text{greater}}, j - |A_{\text{less}}| - 1) \)
QuickSelect analysis

- $S_1, S_2, \ldots, S_k$ be the subproblems considered by the algorithm. Here $|S_1| = n$.
- $S_i$ would be **successful** if $|S_i| \leq (3/4)|S_{i-1}|$
- $Y_1$ = number of recursive calls till first successful iteration. Clearly, total work till this happens is $O(Y_1 n)$.
- $n_i$ = size of the subproblem immediately after the $(i - 1)$th successful iteration.
- $Y_i$ = number of recursive calls after the $(i - 1)$th successful call, till the $i$th successful iteration.
- Running time is $O(\sum_i n_i Y_i)$.

**Example**

<table>
<thead>
<tr>
<th>Inst'</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
<th>$S_7$</th>
<th>$S_8$</th>
<th>$S_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>S_i</td>
<td>$</td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Succ'</td>
<td>$Y_1$ = 2</td>
<td>$Y_2$ = 4</td>
<td>$Y_3$ = 2</td>
<td>$Y_4$ = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n_i$ =</td>
<td>$n_1$ = 100</td>
<td>$n_2$ = 60</td>
<td>$n_3$ = 25</td>
<td>$n_4$ = 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- All the subproblems after $(i - 1)$th successful iteration till $i$th successful iteration have size $\leq n_i$.
- Total work: $O(\sum_i n_i Y_i)$.

QuickSelect analysis

- Total work: $O(\sum_i n_i Y_i)$.
- $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1} n$.
- $Y_i$ is a random variable with geometric distribution
  - Probability of $Y_i = k$ is $1/2^i$.
- $E[Y_i] = 2$.
- As such, expected work is proportional to

$$E\left[\sum_i n_i Y_i\right] = \sum_i E[n_i Y_i] \leq \sum_i E\left[(3/4)^{i-1} n Y_i\right]$$

$$= n \sum_i (3/4)^{i-1} E[Y_i] = n \sum_{i=1} E[Y_i] \leq n \sum (3/4)^{i-1} 2 \leq 8n.$$
QuickSelect analysis via recurrence

Given array $A$ of size $n$ let $Q(A)$ be number of comparisons of randomized selection on $A$ for selecting rank $j$ element.

Note that $Q(A)$ is a random variable

Let $A^i_{\text{less}}$ and $A^i_{\text{greater}}$ be the left and right arrays obtained if pivot is rank $i$ element of $A$.

Algorithm recurses on $A^i_{\text{less}}$ if $j < i$ and recurses on $A^i_{\text{greater}}$ if $j > i$ and terminates if $j = i$.

$$Q(A) = n + \sum_{i=1}^{j-1} \Pr[\text{pivot has rank } i] Q(A^i_{\text{greater}})$$
$$+ \sum_{i=j+1}^{n} \Pr[\text{pivot has rank } i] Q(A^i_{\text{less}})$$

Analyzing the Recurrence

As in QuickSort we obtain the following recurrence where $T(n)$ is the worst-case expected time.

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

Theorem

$T(n) = O(n)$.

Proof.

(Guess and) Verify by induction (see next slide).
Comments on analyzing the recurrence

- Algebra looks messy but intuition suggest that the median is the hardest case and hence can plug $j = n/2$ to simplify without calculus.
- Analyzing recurrences comes with practice and after a while one can see things more intuitively.

**John Von Neumann:**

*Young man, in mathematics you don’t understand things. You just get used to them.*