CS 473: Fundamental Algorithms, Spring 2013

Randomized Algorithms: QuickSort and QuickSelect

Lecture 14

March 8, 2013

Part I

Slick analysis of QuickSort

A Slick Analysis of **QuickSort**

Let $\mathbf{Q(A)}$ be number of comparisons done on input array $\mathbf{A}$:

1. For $1 \leq i < j < n$ let R_{ij} be the event that rank i element is compared with rank j element.
2. 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.

$$Q(A) = \sum_{1 \leq i < j \leq n} X_{ij}$$

and hence by linearity of expectation,

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

A Slick Analysis of **QuickSort**

$\boxed{R_{ij} = \text{rank } i \text{ element is compared with rank } j \text{ element.}}$

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

7	5	9	1	3	4	8	6

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

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

1. If pivot too small (say **3** [rank 2]). Partition and call recursively:

   7 5 9 1 3 4 8 6 $\Longrightarrow$ 1 | 3 | 7 5 9 4 8 6

   Decision if to compare **5** to **8** is moved to subproblem.
2. If pivot too large (say **9** [rank 8]):

   7 5 9 1 3 4 8 6 $\Longrightarrow$ 7 5 1 3 4 8 6 | 9

   Decision if to compare **5** to **8** moved to subproblem.

A Slick Analysis of **QuickSort**

Question: What is $\mathbf{Pr}[\mathbf{R}_{i,j}]$?

7	5	9	1	3	4	8	6
6	4	8	1	2	3	7	5

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

1. If pivot is **5** (rank 4). Bingo!

   7 5 9 1 3 4 8 6 $\Longrightarrow$ 1 3 4 | 5 | 7 9 8 6

2. If pivot is **8** (rank 7). Bingo!

   7 5 9 1 3 4 8 6 $\Longrightarrow$ 7 5 1 3 4 6 | 8 | 9

3. If pivot in between the two numbers (say **6** [rank 5]):

   7 5 9 1 3 4 8 6 $\Longrightarrow$ 5 1 3 4 | 6 | 7 8 9

5 and **8** will never be compared to each other.

A Slick Analysis of **QuickSort**

Question: What is $\mathbf{Pr}[\mathbf{R}_{i,j}]$?

Conclusion:

$\mathbf{R}_{i,j}$ happens if and only if:

> **i**th or **j**th ranked element is the first pivot out of **i**th to **j**th ranked elements.

How to analyze this?

Thinking acrobatics!

1. Assign every element in the array a random priority (say in $[0, 1]$).
2. Choose pivot to be the element with lowest priority in subproblem.
3. Equivalent to picking pivot uniformly at random (as **QuickSort** do).

A Slick Analysis of **QuickSort**

Question: What is $\mathbf{Pr}[\mathbf{R}_{i,j}]$?

How to analyze this?

Thinking acrobatics!

1. Assign every element in the array a random priority (say in $[0, 1]$).
2. Choose pivot to be the element with lowest priority in subproblem.

$\Longrightarrow$ $\mathbf{R}_{i,j}$ happens if either **i** or **j** have lowest priority out of elements rank **i** to **j**,

There are $\mathbf{k} = \mathbf{j} - \mathbf{i} + 1$ relevant elements.

$$\mathbf{Pr}\Big[\mathbf{R}_{i,j}\Big] = \frac{2}{k} = \frac{2}{j - i + 1}.$$

A Slick Analysis of **QuickSort**

Question: What is $\mathbf{Pr}[\mathbf{R}_{ij}]$?

Lemma

$\mathbf{Pr}\Big[\mathbf{R}_{ij}\Big] = \frac{2}{j-i+1}$.

Proof.

Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be elements of **A** in sorted order. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

Observation: If pivot is chosen outside **S** then all of **S** either in left array or right array.

Observation: a_i and a_j separated when a pivot is chosen from **S** for the first time. Once separated no comparison.

Observation: a_i is compared with a_j if and only if either a_i or a_j is chosen as a pivot from **S** at separation... $\square$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

Proof.

Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be sort of A. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

Observation: a_i is compared with a_j if and only if either a_i or a_j is chosen as a pivot from S at separation.

Observation: Given that pivot is chosen from S the probability that it is a_i or a_j is exactly $2/|S| = 2/(j-i+1)$ since the pivot is chosen uniformly at random from the array. □

A Slick Analysis of **QuickSort**

Continued...

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

Lemma

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

$$\begin{aligned}
\mathbf{E}\left[Q(A)\right] &= \sum_{1 \le i < j \le n} \mathbf{Pr}\left[R_{ij}\right] = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&\le 2 \sum_{i=1}^{n-1} (H_{n-i+1} - 1) \le 2 \sum_{1 \le i < n} H_n
\end{aligned}$$

Yet another analysis of QuickSort

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

Consider element e in the array.

Consider the subproblems it participates in during **QuickSort** execution:

$S_1, S_2, \ldots, S_k$.

Definition

e is lucky in the jth iteration if $|S_j| \le (3/4)\,|S_{j-1}|$.

Key observation

The event e is lucky in jth iteration
is independent of
the event that e is lucky in kth iteration,
(If $j \ne k$)

$X_j = 1$ **iff** e is lucky in the jth iteration.

Yet another analysis of QuickSort

Continued...

Claim

$\mathbf{Pr}[X_j = 1] = 1/2$.

Proof.

1. X_j determined by j recursive subproblem.
2. Subproblem has $n_{j-1} = |X_{j-1}|$ elements.
3. If jth pivot rank $\in [n_{j-1}/4, (3/4)n_{j-1}]$, then e lucky in jth iter.
4. Prob. e is lucky $\ge |[n_{j-1}/4, (3/4)n_{j-1}]| \,/ n_{j-1} = 1/2$. □

Observation

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

Yet another analysis of QuickSort

Continued...

Observation

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

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

Conclusion

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

Randomized Quick Selection

Input Unsorted array $\mathbf{A}$ of $\mathbf{n}$ integers

Goal Find the $\mathbf{j}$th smallest number in $\mathbf{A}$ (*rank* $\mathbf{j}$ number)

Randomized Quick Selection

1. Pick a pivot element *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 $\mathbf{j}$.
4. Otherwise recurse on one of the arrays depending on $\mathbf{j}$ and their sizes.

Algorithm for Randomized Selection

Assume for simplicity that $\mathbf{A}$ has distinct elements.

```
QuickSelect(A, j):
    Pick pivot x uniformly at random from A
    Partition A into A_less, x, and A_greater using x as pivot
    if (|A_less| = j - 1) then
        return x
    if (|A_less| ≥ j) then
        return QuickSelect(A_less, j)
    else
        return QuickSelect(A_greater, j - |A_less| - 1)
```


QuickSelect analysis

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

QuickSelect analysis

Example

S_i = subarray used in ith recursive call
$|S_i|$ = size of this subarray
Red indicates successful iteration.

Inst'	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
$\lvert S_i \rvert$	100	70	60	50	40	30	25	5	2
Succ'	$Y_1 = 2$		$Y_2 = 4$				$Y_3 = 2$		$Y_4 = 1$
$n_i =$	$n_1 = 100$		$n_2 = 60$				$n_3 = 25$		$n_4 = 2$

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

QuickSelect analysis

Total work: $O(\sum_i n_i Y_i)$.
We have:

1. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n$.
2. Y_i is a random variable with geometric distribution
Probability of $Y_i = k$ is $1/2^i$.
3. $E[Y_i] = 2$.

As such, expected work is proportional to

$$E\left[\sum_i n_i Y_i\right] = \sum_i E\left[n_i Y_i\right] \leq \sum_i E\left[(3/4)^{i-1} n Y_i\right]$$
$$= n \sum_i (3/4)^{i-1} E\left[Y_i\right] = n \sum_{i=1} (3/4)^{i-1} 2 \leq 8n.$$

QuickSelect analysis

Theorem

The expected running time of **QuickSelect** *is* $O(n)$.

QuickSelect analysis

Analysis via Recurrence

1. Given array A of size n let $Q(A)$ be number of comparisons of randomized selection on A for selecting rank j element.
2. Note that $Q(A)$ is a random variable
3. Let A^i_{less} and $A^i_{greater}$ be the left and right arrays obtained if pivot is rank i element of A.
4. Algorithm recurses on A^i_{less} if $j < i$ and recurses on $A^i_{greater}$ if $j > i$ and terminates if $j = i$.

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

Analyzing the Recurrence

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

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

Theorem

$\mathbf{T(n) = O(n)}$.

Proof.

(Guess and) Verify by induction (see next slide). □

Analyzing the recurrence

Theorem

$\mathbf{T(n) = O(n)}$.

Prove by induction that $\mathbf{T(n) \leq \alpha n}$ for some constant $\alpha \geq \mathbf{1}$ to be fixed later.

Base case: $\mathbf{n = 1}$, we have $\mathbf{T(1) = 0}$ since no comparisons needed and hence $\mathbf{T(1) \leq \alpha}$.

Induction step: Assume $\mathbf{T(k) \leq \alpha k}$ for $\mathbf{1 \leq k < n}$ and prove it for $\mathbf{T(n)}$. We have by the recurrence:

$$\begin{aligned} \mathbf{T(n)} &\leq \mathbf{n + \frac{1}{n}\left(\sum_{i=1}^{j-1} T(n-i) + \sum_{i=j}^{n} T(i-1)\right)} \\ &\leq \mathbf{n + \frac{\alpha}{n}\left(\sum_{i=1}^{j-1} (n-i) + \sum_{i=j}^{n} (i-1)\right)} \quad \text{by applying induction} \end{aligned}$$

Analyzing the recurrence

$$\begin{aligned} \mathbf{T(n)} &\leq \mathbf{n + \frac{\alpha}{n}\left(\sum_{i=1}^{j-1} (n-i) + \sum_{i=j}^{n} (i-1)\right)} \\ &\leq \mathbf{n + \frac{\alpha}{n}((j-1)(2n-j)/2 + (n-j+1)(n+j-2)/2)} \\ &\leq \mathbf{n + \frac{\alpha}{2n}(n^2 + 2nj - 2j^2 - 3n + 4j - 2)} \\ & \quad \text{above expression maximized when } \mathbf{j = (n+1)/2}\text{: calculus} \\ &\leq \mathbf{n + \frac{\alpha}{2n}(3n^2/2 - n)} \quad \text{substituting } \mathbf{(n+1)/2} \text{ for } \mathbf{j} \\ &\leq \mathbf{n + 3\alpha n/4} \\ &\leq \mathbf{\alpha n} \quad \text{for any constant } \alpha \geq \mathbf{4} \end{aligned}$$

Comments on analyzing the recurrence

1. Algebra looks messy but intuition suggest that the median is the hardest case and hence can plug $\mathbf{j = n/2}$ to simplify without calculus
2. 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.