Randomized Algorithms: QuickSort and QuickSelect

Lecture 14

October 16, 2014

Red, blue, and white Balls

$\mathbf{n}$ balls, $\mathbf{k-2}$ blue balls, and $\mathbf{2}$ red balls.

Game

Pick a ball randomly, and throw it out. Repeat till picking a red or blue balls.

Question: What is the probability that the last ball picked is red?

(A) $\mathbf{1/2}$
(B) $\mathbf{(k-2)/n}$.
(C) $\mathbf{2/n}$.
(D) $\mathbf{2/k}$.
(E) $\mathbf{2/(k-2)}$.

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**

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,

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

A Slick Analysis of **QuickSort**

$\mathbf{R_{ij}}$ = rank **i** element is compared with rank **j** element.

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

7	5	9	1	3	4	8	6

A Slick Analysis of **QuickSort**

$\mathbf{R_{ij}}$ = rank **i** element is compared with rank **j** element.

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

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

A Slick Analysis of **QuickSort**

$\mathbf{R_{ij}}$ = rank **i** element is compared with rank **j** element.

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

	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 $\mathbf{Pr[R_{4,7}]}$.

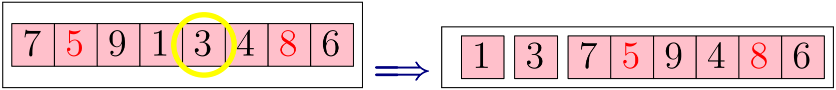
A Slick Analysis of **QuickSort**

$\mathbf{R_{ij}}$ = rank **i** element is compared with rank **j** element.

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

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

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



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

A Slick Analysis of **QuickSort**

$\mathbf{R_{ij}}$ = rank **i** element is compared with rank **j** element.

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

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

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.

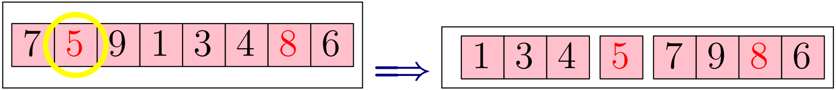
A Slick Analysis of **QuickSort**

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

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 $\mathbf{Pr[R_{4,7}]}$.

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



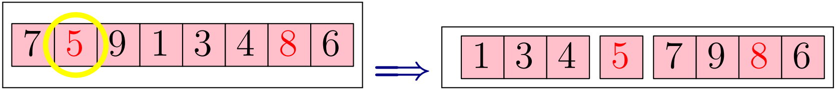
A Slick Analysis of **QuickSort**

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

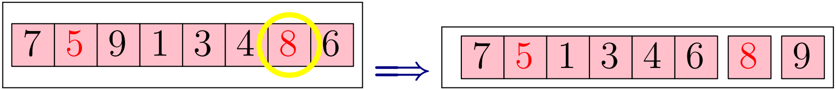
7	5	9	1	3	4	8	6
6	4	8	1	2	3	7	5

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

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



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



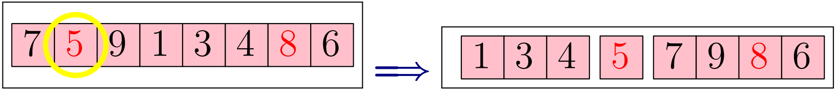
A Slick Analysis of **QuickSort**

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

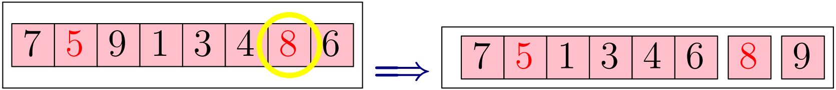
7	5	9	1	3	4	8	6
6	4	8	1	2	3	7	5

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

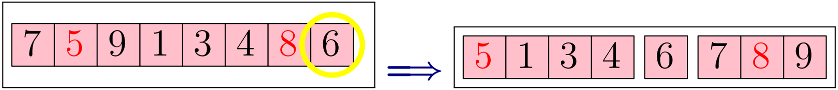
1. If pivot is **5** (rank 4). Bingo!



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



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



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 $\Pr[R_{i,j}]$?

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.

$\implies$ $R_{i,j}$ happens if either i or j have lowest priority out of elements rank i to j,

There are $k = j - i + 1$ relevant elements.

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

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.

$\implies$ $\mathbf{R}_{i,j}$ happens if either $\mathbf{i}$ or $\mathbf{j}$ have lowest priority out of elements rank $\mathbf{i}$ to $\mathbf{j}$,
There are $\mathbf{k = j - i + 1}$ relevant elements.

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

A Slick Analysis of **QuickSort**

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

Lemma

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

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... □

A Slick Analysis of **QuickSort**

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

Lemma

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

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... □

A Slick Analysis of **QuickSort**

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

Lemma

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

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... □

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

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. □

How much is this?

$\mathbf{H_n = \sum_{i=1}^{n} \frac{1}{i}}$ is equal to

(A) $\mathbf{H_n = O(1)}$.

(B) $\mathbf{H_n = O(\log \log n)}$.

(C) $\mathbf{H_n = O(\sqrt{\log n})}$.

(D) $\mathbf{H_n = O(\log n)}$.

(E) $\mathbf{H_n = O(\log^2 n)}$.

And how much is this?

$$T_n = \sum_{i=1}^{n-1} \sum_{\Delta=1}^{n-i} \frac{1}{\Delta}$$

is equal to

(A) $T_n = O(n)$.

(B) $T_n = O(n \log n)$.

(C) $T_n = O(n \log^2 n)$.

(D) $T_n = O(n^2)$.

(E) $T_n = O(n^3)$.

A Slick Analysis of **QuickSort**

Continued...

$$\mathbf{E}\Big[\mathbf{Q}(\mathbf{A})\Big] = \sum_{1 \le i < j \le n} \mathbf{E}[\mathbf{X}_{ij}] = \sum_{1 \le i < j \le n} \Pr[\mathbf{R}_{ij}] .$$

Lemma

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

$$\mathbf{E}\Big[\mathbf{Q}(\mathbf{A})\Big] = \sum_{1 \le i < j \le n} \frac{2}{j-i+1}$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$E\left[Q(A)\right] = \sum_{1 \le i < j \le n} \Pr\left[R_{ij}\right] = \sum_{1 \le i < j \le n} \frac{2}{j-i+1}$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$\mathbf{E}\Big[\mathbf{Q(A)}\Big] = \sum_{1 \le i < j \le n} \frac{2}{j - i + 1}$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$\mathbf{E}\Big[Q(A)\Big] = \sum_{1 \le i < j \le n} \frac{2}{j-i+1}$$
$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$E\left[Q(A)\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$E\left[Q(A)\right] = 2\sum_{i=1}^{n-1}\sum_{i<j}^{n}\frac{1}{j-i+1}$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$\mathbf{E}\Big[Q(A)\Big] = 2\sum_{i=1}^{n-1}\sum_{i<j}^{n}\frac{1}{j-i+1}$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$\mathbf{E}\Big[\mathbf{Q(A)}\Big] = 2\sum_{i=1}^{n-1}\sum_{i<j}^{n}\frac{1}{j-i+1} \leq 2\sum_{i=1}^{n-1}\quad\sum_{\Delta=2}^{n-i+1}\frac{1}{\Delta}$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$E\left[Q(A)\right] = 2\sum_{i=1}^{n-1}\sum_{i<j}^{n}\frac{1}{j-i+1} \leq 2\sum_{i=1}^{n-1}\sum_{\Delta=2}^{n-i+1}\frac{1}{\Delta}$$

$$\leq 2\sum_{i=1}^{n-1}(H_{n-i+1}-1) \leq 2\sum_{1\leq i<n} H_n$$

A Slick Analysis of **QuickSort**

Continued...

Lemma

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

$$\mathbf{E}\left[Q(A)\right] = 2\sum_{i=1}^{n-1}\sum_{i<j}^{n}\frac{1}{j-i+1} \leq 2\sum_{i=1}^{n-1}\sum_{\Delta=2}^{n-i+1}\frac{1}{\Delta}$$

$$\leq 2\sum_{i=1}^{n-1}(H_{n-i+1}-1) \leq 2\sum_{1\leq i<n} H_n$$

$$\leq 2nH_n = O(n\log n)$$

Yet another analysis of QuickSort

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

Consider element $\mathbf{e}$ in the array.
Consider the subproblems it participates in during **QuickSort** execution:
$\mathbf{S_1, S_2, \ldots, S_k}$.

Definition

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

Key observation

The event $\mathbf{e}$ is lucky in $\mathbf{j}$th iteration
is independent of
the event that $\mathbf{e}$ is lucky in $\mathbf{k}$th iteration,
(If $\mathbf{j} \neq \mathbf{k}$)

$\mathbf{X_j = 1}$ **iff** $\mathbf{e}$ is lucky in the $\mathbf{j}$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 $n_{j-1} = |S_{j-1}|$ 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 $\geq |[n_{j-1}/4, (3/4)n_{j-1}]| / n_{j-1} = 1/2$. □

Observation

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

Yet another analysis of QuickSort

Continued...

Observation

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

Because...

Theorem

Let $\mathbf{X_n}$ *be the number heads when flipping a coin indepdently* $\mathbf{n}$ *times. Then*

$$\mathbf{Pr}\left[\mathbf{X_n} \leq \frac{\mathbf{n}}{\mathbf{4}}\right] \leq \mathbf{2} \cdot \mathbf{0.68^{n/4}} \textit{ and } \mathbf{Pr}\left[\mathbf{X_n} \geq \frac{\mathbf{3n}}{\mathbf{4}}\right] \leq \mathbf{2} \cdot \mathbf{0.68^{n/4}}$$

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 *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 $\mathbf{A}$ 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* $\mathbf{O(n)}$.

QuickSelect analysis

Analysis via Recurrence

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

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

QuickSelect analysis

Analysis via Recurrence

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

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

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 $T(n) \leq \alpha n$ for some constant $\alpha \geq 1$ to be fixed later.

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

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

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

Analyzing the recurrence

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

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.

Notes

Notes

Notes

Notes