# Randomised Algorithms

Lecture 3: Concentration Inequalities, Application to Quick-Sort, Extensions

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025

UNIVERSITY OF CAMBRIDGE

# Outline

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

# QuickSort

QuickSort (Input $A[1], A[2], \ldots, A[n]$)

Pick an element from the array, the so-called pivot<br>
**If** $n = 0$ or $n = 1$ **then**<br>
    **return** $A$<br>
**else**<br>
    Create two subarrays $A_1$ and $A_2$ (without the pivot) such that:<br>
        $A_1$ contains the elements that are smaller than the pivot<br>
        $A_2$ contains the elements that are greater (or equal) than the pivot<br>
    QuickSort($A_1$)<br>
    QuickSort($A_2$)<br>
    **return** $A$

# QuickSort

QuickSort (Input $A[1], A[2], \ldots, A[n]$)

Pick an element from the array, the so-called pivot

**If** $n = 0$ or $n = 1$ **then**

    **return** $A$

**else**

    Create two subarrays $A_1$ and $A_2$ (without the pivot) such that:

        $A_1$ contains the elements that are smaller than the pivot

        $A_2$ contains the elements that are greater (or equal) than the pivot

    QuickSort($A_1$)

    QuickSort($A_2$)

    **return** $A$

- Example: Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.

# QuickSort

QuickSort (Input $A[1], A[2], \ldots, A[n]$)

Pick an element from the array, the so-called pivot

**If** $n = 0$ or $n = 1$ **then**

    **return** $A$

**else**

    Create two subarrays $A_1$ and $A_2$ (without the pivot) such that:

        $A_1$ contains the elements that are smaller than the pivot

        $A_2$ contains the elements that are greater (or equal) than the pivot

    QuickSort($A_1$)

    QuickSort($A_2$)

    **return** $A$

- Example: Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.
  $\Rightarrow A_1 = (2, 1, 5, 3, 4)$ and $A_2 = (8, 9, 7)$

# QuickSort

QUICKSORT (Input $A[1], A[2], \ldots, A[n]$)
Pick an element from the array, the so-called pivot
**If** $n = 0$ or $n = 1$ **then**
    **return** $A$
**else**
    Create two subarrays $A_1$ and $A_2$ (without the pivot) such that:
        $A_1$ contains the elements that are smaller than the pivot
        $A_2$ contains the elements that are greater (or equal) than the pivot
    QUICKSORT($A_1$)
    QUICKSORT($A_2$)
    **return** $A$

- Example: Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.
  $\Rightarrow A_1 = (2, 1, 5, 3, 4)$ and $A_2 = (8, 9, 7)$
- Worst-Case Complexity (number of comparisons) is $\Theta(n^2)$,

# QuickSort

QuickSort (Input $A[1], A[2], \ldots, A[n]$)

Pick an element from the array, the so-called pivot<br>
**If** $n = 0$ or $n = 1$ **then**<br>
    **return** $A$<br>
**else**<br>
    Create two subarrays $A_1$ and $A_2$ (without the pivot) such that:<br>
        $A_1$ contains the elements that are smaller than the pivot<br>
        $A_2$ contains the elements that are greater (or equal) than the pivot<br>
    QuickSort($A_1$)<br>
    QuickSort($A_2$)<br>
    **return** $A$

- Example: Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.
  $\Rightarrow A_1 = (2, 1, 5, 3, 4)$ and $A_2 = (8, 9, 7)$
- Worst-Case Complexity (number of comparisons) is $\Theta(n^2)$, while Average-Case Complexity is $O(n \log n)$.

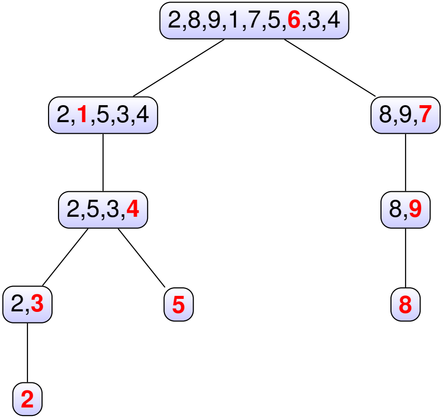
# QuickSort

QuickSort (Input $A[1], A[2], \ldots, A[n]$)
Pick an element from the array, the so-called pivot
**If** $n = 0$ or $n = 1$ **then**
  **return** $A$
**else**
  Create two subarrays $A_1$ and $A_2$ (without the pivot) such that:
    $A_1$ contains the elements that are smaller than the pivot
    $A_2$ contains the elements that are greater (or equal) than the pivot
  QuickSort($A_1$)
  QuickSort($A_2$)
  **return** $A$

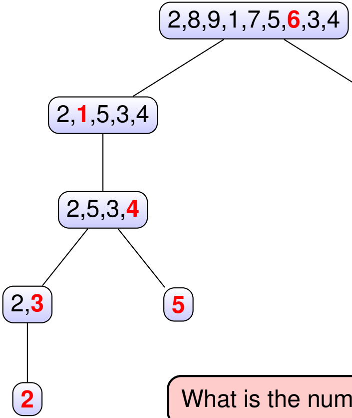
- Example: Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.
  $\Rightarrow A_1 = (2, 1, 5, 3, 4)$ and $A_2 = (8, 9, 7)$
- Worst-Case Complexity (number of comparisons) is $\Theta(n^2)$, while Average-Case Complexity is $O(n \log n)$.

We will now give a proof of this “well-known” result!

# QuickSort: How to Count Comparisons

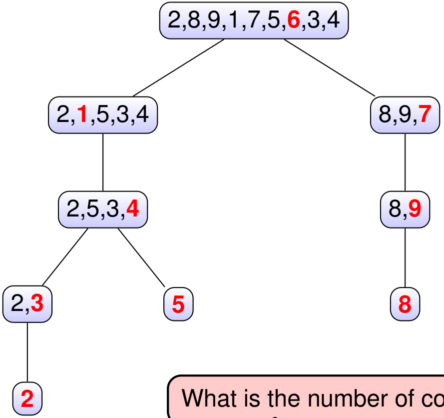


# QuickSort: How to Count Comparisons




What is the number of comparisons?

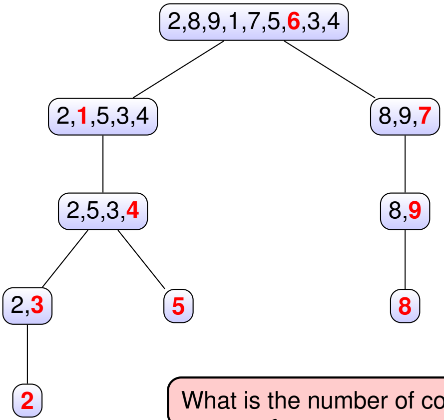
# QuickSort: How to Count Comparisons




What is the number of comparisons?

Note that the number of comparison by QUICKSORT is equivalent to the sum of the depths of all nodes in the tree (why?).

# QuickSort: How to Count Comparisons




What is the number of comparisons?

Note that the number of comparison by QUICKSORT is equivalent to the sum of the depths of all nodes in the tree (why?). In this case:

$$0 + 1 + 1 + 2 + 2 + 3 + 3 + 3 + 4 = 19.$$

# Randomised QuickSort: Analysis (1/4)

How to pick a good pivot? We don't, **just pick one at random.**

# Randomised QuickSort: Analysis (1/4)

How to pick a good pivot? We don't, **just pick one at random.**

This should be your standard answer in this course ☺

# Randomised QuickSort: Analysis (1/4)

How to pick a good pivot? We don't, **just pick one at random.**

This should be your standard answer in this course ☺

Let us analyse QUICKSORT with random pivots.

# Randomised QuickSort: Analysis (1/4)

How to pick a good pivot? We don't, **just pick one at random.**

This should be your standard answer in this course ☺

Let us analyse QuickSort with random pivots.

1. Assume $A$ consists of $n$ different numbers, w.l.o.g., $\{1, 2, \ldots, n\}$

# Randomised QuickSort: Analysis (1/4)

How to pick a good pivot? We don't, **just pick one at random.**

This should be your standard answer in this course ☺

Let us analyse QUICKSORT with random pivots.

1. Assume $A$ consists of $n$ different numbers, w.l.o.g., $\{1, 2, \ldots, n\}$
2. Let $H_i$ be the deepest level where element $i$ appears in the tree.
   Then the number of comparison is $H = \sum_{i=1}^{n} H_i$

# Randomised QuickSort: Analysis (1/4)

How to pick a good pivot? We don't, **just pick one at random.**

This should be your standard answer in this course ☺

Let us analyse QuickSort with random pivots.

1. Assume $A$ consists of $n$ different numbers, w.l.o.g., $\{1, 2, \ldots, n\}$
2. Let $H_i$ be the deepest level where element $i$ appears in the tree. Then the number of comparison is $H = \sum_{i=1}^{n} H_i$
3. We will prove that there exists $C > 0$ such that

$$\mathbf{P}[H \leq Cn \log n] \geq 1 - n^{-1}.$$

# Randomised QuickSort: Analysis (1/4)

How to pick a good pivot? We don't, **just pick one at random.**

This should be your standard answer in this course ☺

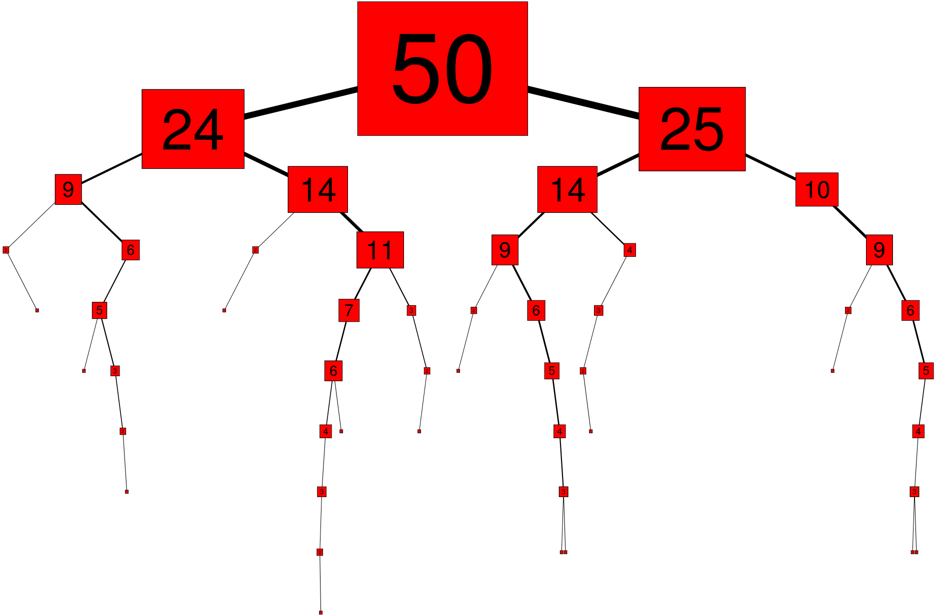
Let us analyse QUICKSORT with random pivots.

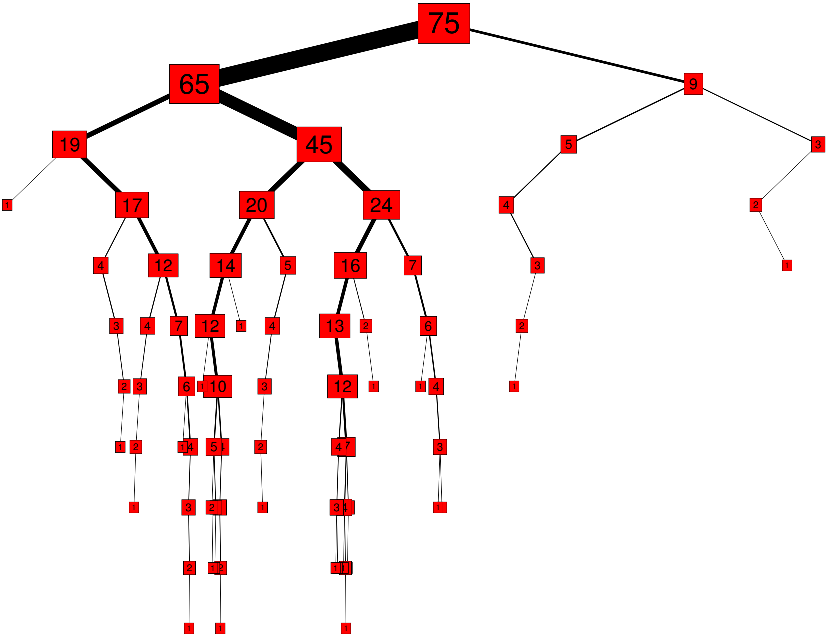
1. Assume $A$ consists of $n$ different numbers, w.l.o.g., $\{1, 2, \ldots, n\}$
2. Let $H_i$ be the deepest level where element $i$ appears in the tree.
   Then the number of comparison is $H = \sum_{i=1}^{n} H_i$
3. We will prove that there exists $C > 0$ such that

$$\mathbf{P}[H \leq Cn \log n] \geq 1 - n^{-1}.$$

4. Actually, we will prove sth slightly stronger:

$$\mathbf{P}\left[\bigcap_{i=1}^{n} \{H_i \leq C \log n\}\right] \geq 1 - n^{-1}.$$


50
24
25
9
14
14
10
6
11
9
4
9
5
7
3
6
3
6
3
6
3
6
5
5
4
4
4
3
3
3

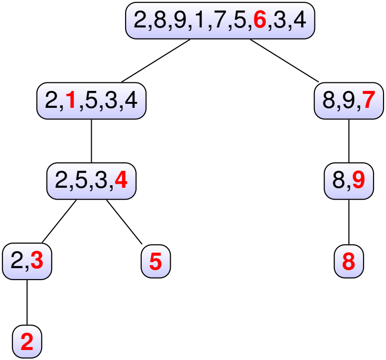

75
65
9
19
45
5
3
1
17
20
24
4
2
4
12
14
5
16
7
3
1
3
4
7
12
1
4
13
2
6
2
2
3
6
1
10
3
12
1
1
4
1
1
1
2
1
4
5
4
2
4
7
3
1
3
2
1
3
1
1
2
1
2
1
1
1
1
1
1

# Randomised QuickSort: Analysis (2/4)

- Let $P$ be a path from the root to the deepest level of some element

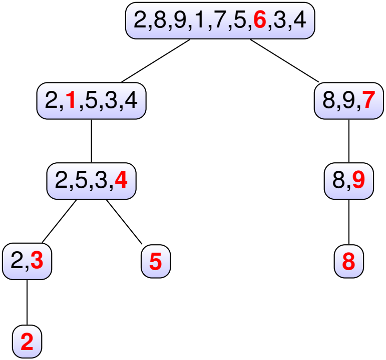
# Randomised QuickSort: Analysis (2/4)

- Let $P$ be a path from the root to the deepest level of some element



# Randomised QuickSort: Analysis (2/4)

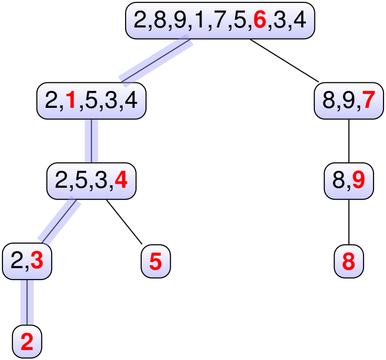
- Let $P$ be a path from the root to the deepest level of some element




- Element 2: $(2, 8, 9, 1, 7, 5, \mathbf{6}, 3, 4) \to (2, \mathbf{1}, 5, 3, 4) \to (2, 5, 3, \mathbf{4}) \to (2, \mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

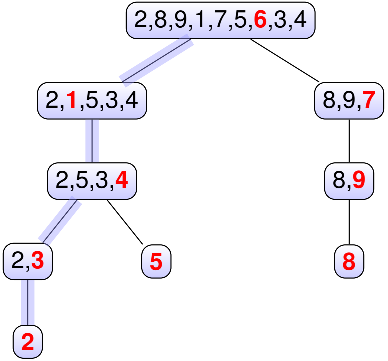
- Let $P$ be a path from the root to the deepest level of some element




- Element 2: $(2, 8, 9, 1, 7, 5, \mathbf{6}, 3, 4) \to (2, \mathbf{1}, 5, 3, 4) \to (2, 5, 3, \mathbf{4}) \to (2, \mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

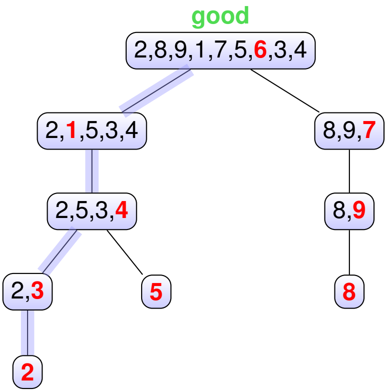
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**




- Element 2: $(2,8,9,1,7,5,\mathbf{6},3,4) \to (2,\mathbf{1},5,3,4) \to (2,5,3,\mathbf{4}) \to (2,\mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

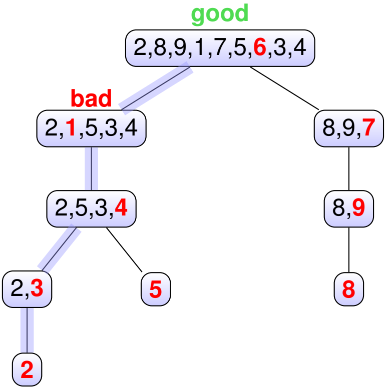
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**




- Element 2: $(2, 8, 9, 1, 7, 5, \mathbf{6}, 3, 4) \to (2, \mathbf{1}, 5, 3, 4) \to (2, 5, 3, \mathbf{4}) \to (2, \mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

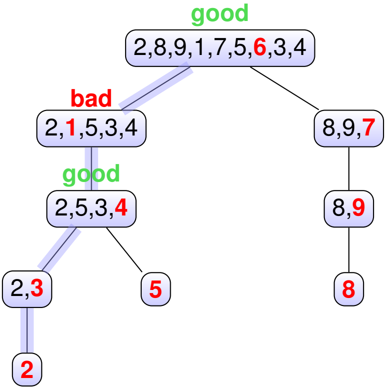
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**




- Element 2: $(2,8,9,1,7,5,\mathbf{6},3,4) \to (2,\mathbf{1},5,3,4) \to (2,5,3,\mathbf{4}) \to (2,\mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

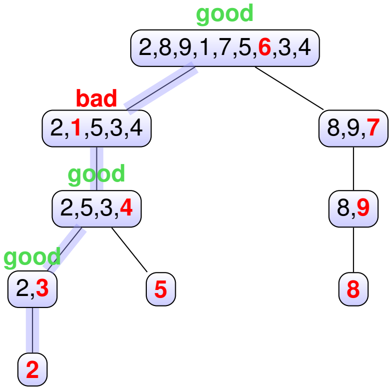
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**




- Element 2: $(2,8,9,1,7,5,\mathbf{6},3,4) \to (2,\mathbf{1},5,3,4) \to (2,5,3,\mathbf{4}) \to (2,\mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

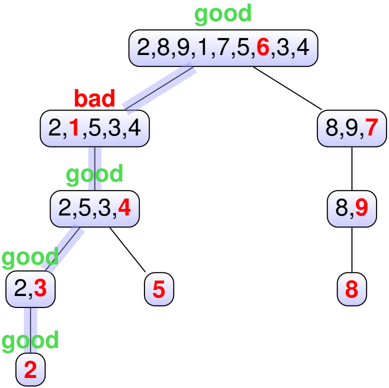
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**




- Element 2: $(2,8,9,1,7,5,6,3,4) \to (2,1,5,3,4) \to (2,5,3,4) \to (2,3) \to (2)$

# Randomised QuickSort: Analysis (2/4)

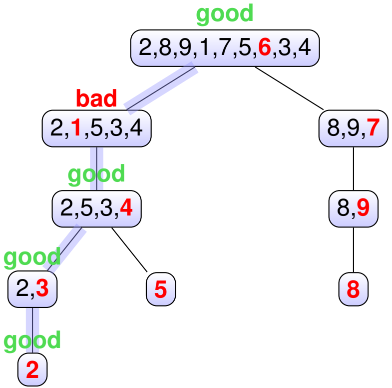
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**




- Element 2: $(2,8,9,1,7,5,\mathbf{6},3,4) \to (2,\mathbf{1},5,3,4) \to (2,5,3,\mathbf{4}) \to (2,\mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

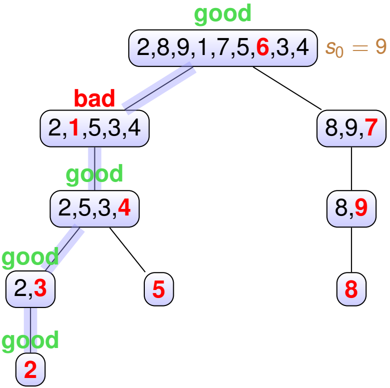
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**
- Further let $s_t$ be the size of the array at level $t$ in $P$.




- Element 2: $(2,8,9,1,7,5,6,3,4) \rightarrow (2,1,5,3,4) \rightarrow (2,5,3,4) \rightarrow (2,3) \rightarrow (2)$

# Randomised QuickSort: Analysis (2/4)

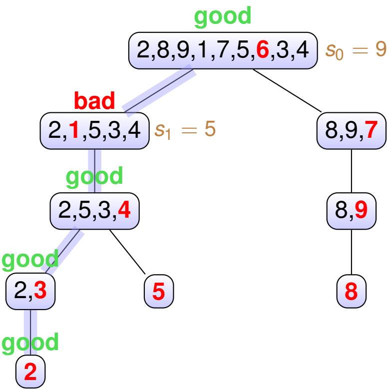
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**
- Further let $s_t$ be the size of the array at level $t$ in $P$.




- Element 2: $(2,8,9,1,7,5,\mathbf{6},3,4) \to (2,\mathbf{1},5,3,4) \to (2,5,3,\mathbf{4}) \to (2,\mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

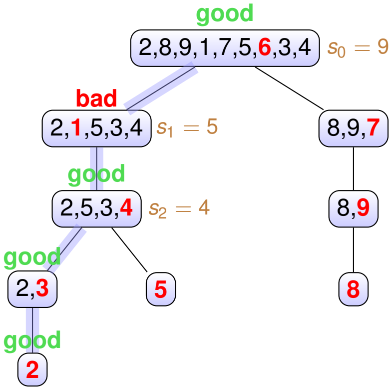
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**
- Further let $s_t$ be the size of the array at level $t$ in $P$.




- Element 2: $(2, 8, 9, 1, 7, 5, \mathbf{6}, 3, 4) \to (2, \mathbf{1}, 5, 3, 4) \to (2, 5, 3, \mathbf{4}) \to (2, \mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

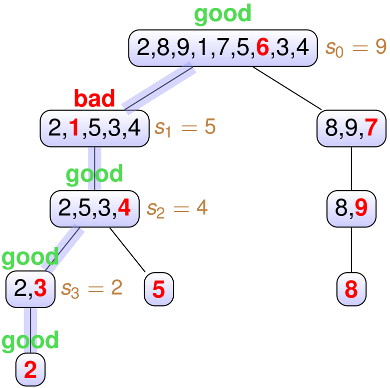
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**
- Further let $s_t$ be the size of the array at level $t$ in $P$.




- Element 2: $(2, 8, 9, 1, 7, 5, \mathbf{6}, 3, 4) \to (2, \mathbf{1}, 5, 3, 4) \to (2, 5, 3, \mathbf{4}) \to (2, \mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

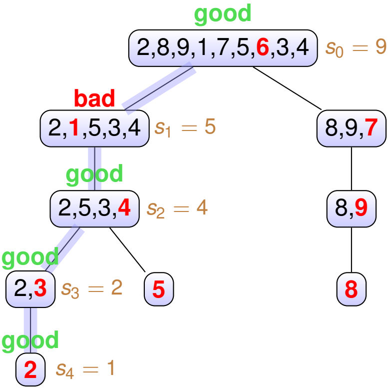
- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**
- Further let $s_t$ be the size of the array at level $t$ in $P$.




- Element 2: $(2,8,9,1,7,5,\mathbf{6},3,4) \to (2,\mathbf{1},5,3,4) \to (2,5,3,\mathbf{4}) \to (2,\mathbf{3}) \to (\mathbf{2})$

# Randomised QuickSort: Analysis (2/4)

- Let $P$ be a path from the root to the deepest level of some element
  - A node in $P$ is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
  - otherwise, the node is **bad**
- Further let $s_t$ be the size of the array at level $t$ in $P$.




- Element 2: $(2, 8, 9, 1, 7, 5, \mathbf{6}, 3, 4) \rightarrow (2, \mathbf{1}, 5, 3, 4) \rightarrow (2, 5, 3, \mathbf{4}) \rightarrow (2, \mathbf{3}) \rightarrow (\mathbf{2})$

## Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k + 1$, the condition $s_k > 1$ is necessary

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path $P$ possibly run until we have $s_k = 1$?

## Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k+1$, the condition $s_k > 1$ is necessary

> How far could such a path $P$ possibly run until we have $s_k = 1$?

- We start with $s_0 = n$

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k+1$, the condition $s_k > 1$ is necessary

How far could such a path $P$ possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k+1$, the condition $s_k > 1$ is necessary

How far could such a path $P$ possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- **Second Case**, **bad** node: $s_{k+1} \leq s_k$.

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k + 1$, the condition $s_k > 1$ is necessary

> How far could such a path $P$ possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- **Second Case**, **bad** node: $s_{k+1} \leq s_k$.

$\Rightarrow$ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path $P$.

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k+1$, the condition $s_k > 1$ is necessary

> How far could such a path $P$ possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- **Second Case**, **bad** node: $s_{k+1} \leq s_k$.

> This even holds always, i.e., deterministically!

$\Rightarrow$ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path $P$.

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k+1$, the condition $s_k > 1$ is necessary

> How far could such a path $P$ possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- **Second Case**, **bad** node: $s_{k+1} \leq s_k$.

> This even holds always, i.e., deterministically!

$\Rightarrow$ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path $P$.

- Assume $|P| \geq C \log n$ for $C := 24$

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k+1$, the condition $s_k > 1$ is necessary

> How far could such a path $P$ possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- **Second Case**, **bad** node: $s_{k+1} \leq s_k$.

> This even holds always, i.e., deterministically!

$\Rightarrow$ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path $P$.

- Assume $|P| \geq C \log n$ for $C := 24$

  $\Rightarrow$ number of **bad** nodes in the first $24 \log n$ levels is more than $21 \log n$.

# Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ and construct the path $P = P(i)$ one level by one
- For $P$ to proceed from level $k$ to $k+1$, the condition $s_k > 1$ is necessary

How far could such a path $P$ possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- **Second Case**, **bad** node: $s_{k+1} \leq s_k$.

This even holds always, i.e., deterministically!

$\Rightarrow$ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path $P$.

- Assume $|P| \geq C \log n$ for $C := 24$

  $\Rightarrow$ number of **bad** nodes in the first $24 \log n$ levels is more than $21 \log n$.

Let us now upper bound the probability that this "bad event" happens!

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:

# Randomised QuickSort: Analysis (4/4)

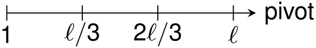
- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}\left[ X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \right] \leq \frac{2}{3}$

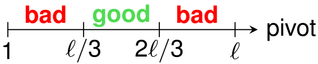
# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$



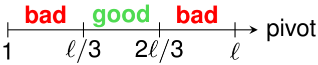
# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$



# Randomised QuickSort: Analysis (4/4)

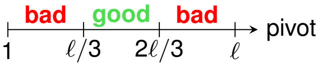
- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$




**Question:** Edge Case: What if the path $P$ does not reach level $j$?

## Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$

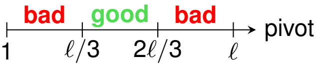



**Question:** Edge Case: What if the path $P$ does not reach level $j$?

**Answer:** We can then simply define $X_j$ as 0.

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}\left[ X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \right] \leq \frac{2}{3}$



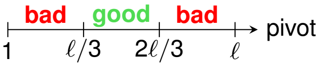

- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

**Question:** Edge Case: What if the path $P$ does not reach level $j$?

**Answer:** We can then simply define $X_j$ as 0.

# Randomised QuickSort: Analysis (4/4)

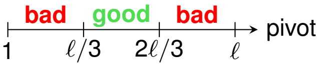
- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$




- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}\left[ X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \right] \leq \frac{2}{3}$

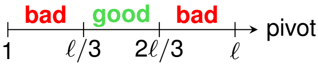



- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

> We can now apply a **Chernoff Bound**! (We use the “nice” version.)

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.



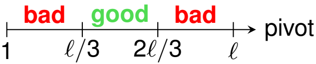

- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

> We can now apply a **Chernoff Bound**! (We use the “nice” version.)

- We have $\mathbf{E}[\, X \,] \leq (2/3) \cdot 24 \log n = 16 \log n$

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$



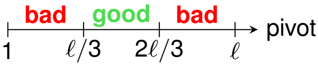

- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

> We can now apply a **Chernoff Bound**! (We use the "nice" version.)

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the "nicer" Chernoff Bounds

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)



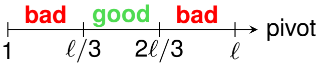

We can now apply a **Chernoff Bound**! (We use the "nice" version.)

- We have $\mathbf{E}[\, X \,] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the "nicer" Chernoff Bounds

$$\mathbf{P}[\, X \geq \mathbf{E}[\, X \,] + t \,] \leq e^{-2t^2/n}$$

## Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.




- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

> We can now apply a **Chernoff Bound**! (We use the "nice" version.)

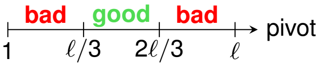
- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the "nicer" Chernoff Bounds

> $\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$

$$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n]$$

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.




- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

> We can now apply a **Chernoff Bound**! (We use the “nice” version.)

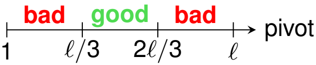
- We have $\mathbf{E}[\, X \,] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the “nicer” Chernoff Bounds

$$\mathbf{P}[\, X \geq \mathbf{E}[\, X \,] + t \,] \leq e^{-2t^2/n}$$

$$\mathbf{P}[\, X > 21 \log n \,] \leq \mathbf{P}[\, X > \mathbf{E}[\, X \,] + 5 \log n \,] \leq e^{-2(5 \log n)^2/(24 \log n)}$$

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$




- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

We can now apply a **Chernoff Bound**! (We use the "nice" version.)

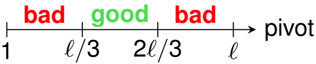
- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the "nicer" Chernoff Bounds

$$\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$$

$$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2/(24 \log n)} \leq n^{-2}.$$

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.




- $\mathbf{P}[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

We can now apply a **Chernoff Bound**! (We use the “nice” version.)

- We have $\mathbf{E}[\, X \,] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the “nicer” Chernoff Bounds

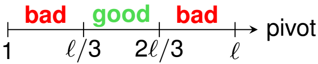
$$\mathbf{P}[\, X \geq \mathbf{E}[\, X \,] + t \,] \leq e^{-2t^2/n}$$

$$\mathbf{P}[\, X > 21 \log n \,] \leq \mathbf{P}[\, X > \mathbf{E}[\, X \,] + 5 \log n \,] \leq e^{-2(5 \log n)^2/(24 \log n)} \leq n^{-2}.$$

- Hence $P$ has more than $24 \log n$ nodes with probability at most $n^{-2}$.

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.




- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

We can now apply a **Chernoff Bound**! (We use the "nice" version.)

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the "nicer" Chernoff Bounds

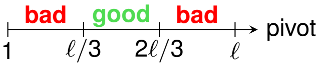
$$\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$$

$$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2/(24 \log n)} \leq n^{-2}.$$

- Hence $P$ has more than $24 \log n$ nodes with probability at most $n^{-2}$.
- As there are in total $n$ paths, by the union bound, the probability that at least one of them has more than $24 \log n$ nodes is at most $n^{-1}$.

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.
- $\mathbf{P}\left[ X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \right] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)




We can now apply a **Chernoff Bound**! (We use the “nice” version.)

- We have $\mathbf{E}\left[ X \right] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the “nicer” Chernoff Bounds

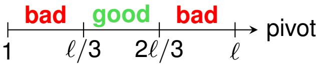
$\mathbf{P}\left[ X \geq \mathbf{E}\left[ X \right] + t \right] \leq e^{-2t^2/n}$

$$\mathbf{P}\left[ X > 21 \log n \right] \leq \mathbf{P}\left[ X > \mathbf{E}\left[ X \right] + 5 \log n \right] \leq e^{-2(5 \log n)^2/(24 \log n)} \leq n^{-2}.$$

- Hence $P$ has more than $24 \log n$ nodes with probability at most $n^{-2}$.
- As there are in total $n$ paths, by the union bound, the probability that at least one of them has more than $24 \log n$ nodes is at most $n^{-1}$.
- This implies $\mathbf{P}\left[ \bigcap_{i=1}^{n} \{ H_i \leq 24 \log n \} \right] \geq 1 - n^{-1}$, as needed.

# Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of $P$ to the deepest level of element $i$.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, define an indicator variable $X_j$:
  - $X_j = 1$ if the node at level $j$ is **bad**,
  - $X_j = 0$ if the node at level $j$ is **good**.




- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

We can now apply a **Chernoff Bound**! (We use the "nice" version.)

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the "nicer" Chernoff Bounds

$\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$

$$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2/(24 \log n)} \leq n^{-2}.$$

- Hence $P$ has more than $24 \log n$ nodes with probability at most $n^{-2}$.
- As there are in total $n$ paths, by the union bound, the probability that at least one of them has more than $24 \log n$ nodes is at most $n^{-1}$.
- This implies $\mathbf{P}\left[\bigcap_{i=1}^{n} \{H_i \leq 24 \log n\}\right] \geq 1 - n^{-1}$, as needed. □

# Randomised QuickSort: Final Remarks

- Well-known: any comparison-based sorting algorithm needs $\Omega(n \log n)$
- A classical result: expected number of comparison of randomised QUICKSORT is $2n \log n + O(n)$ (see, e.g., book by Mitzenmacher & Upfal)

# Randomised QuickSort: Final Remarks

- Well-known: any comparison-based sorting algorithm needs $\Omega(n \log n)$
- A classical result: expected number of comparison of randomised QuickSort is $2n \log n + O(n)$ (see, e.g., book by Mitzenmacher & Upfal)

**Exercise:** *[Ex 2-3.6]* Our upper bound of $O(n \log n)$ whp also immediately implies a $O(n \log n)$ bound on the expected number of comparisons!

# Randomised QuickSort: Final Remarks

- Well-known: any comparison-based sorting algorithm needs $\Omega(n \log n)$
- A classical result: expected number of comparison of randomised QUICKSORT is $2n \log n + O(n)$ (see, e.g., book by Mitzenmacher & Upfal)

**Exercise:** *[Ex 2-3.6]* Our upper bound of $O(n \log n)$ whp also immediately implies a $O(n \log n)$ bound on the expected number of comparisons!

- It is possible to deterministically find the best pivot element that divides the array into two subarrays of the same size.
- The latter requires to compute the median of the array in linear time, which is not easy...
- The presented randomised algorithm for QUICKSORT is much easier to implement!

# Outline

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

# Hoeffding's Extension

- Besides **sums of independent Bernoulli** random variables, sums of independent and bounded random variables are very frequent in applications.

## Hoeffding's Extension

- Besides **sums of independent Bernoulli** random variables, sums of independent and bounded random variables are very frequent in applications.
- Unfortunately the distribution of the $X_i$ may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.

# Hoeffding's Extension

- Besides **sums of independent Bernoulli** random variables, sums of independent and bounded random variables are very frequent in applications.
- Unfortunately the distribution of the $X_i$ may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.
- Hoeffding's Lemma helps us here:

# Hoeffding's Extension

- Besides **sums of independent Bernoulli** random variables, sums of independent and bounded random variables are very frequent in applications.
- Unfortunately the distribution of the $X_i$ may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.
- Hoeffding's Lemma helps us here:

**Hoeffding's Extension Lemma**

Let $X$ be a random variable with mean 0 such that $a \leq X \leq b$. Then for all $\lambda \in \mathbb{R}$,

$$\mathbf{E}\left[e^{\lambda X}\right] \leq \exp\left(\frac{(b-a)^2\lambda^2}{8}\right)$$

# Hoeffding's Extension

- Besides **sums of independent Bernoulli** random variables, sums of independent and bounded random variables are very frequent in applications.
- Unfortunately the distribution of the $X_i$ may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.
- Hoeffding's Lemma helps us here:

You can always consider $X' = X - \mathbf{E}[X]$

**Hoeffding's Extension Lemma**

Let $X$ be a random variable with mean 0 such that $a \leq X \leq b$. Then for all $\lambda \in \mathbb{R}$,

$$\mathbf{E}\left[e^{\lambda X}\right] \leq \exp\left(\frac{(b-a)^2\lambda^2}{8}\right)$$

## Hoeffding's Extension

- Besides **sums of independent Bernoulli** random variables, sums of independent and bounded random variables are very frequent in applications.
- Unfortunately the distribution of the $X_i$ may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.
- Hoeffding's Lemma helps us here:

You can always consider $X' = X - \mathbf{E}[X]$

**Hoeffding's Extension Lemma**

Let $X$ be a random variable with mean 0 such that $a \leq X \leq b$. Then for all $\lambda \in \mathbb{R}$,

$$\mathbf{E}\left[e^{\lambda X}\right] \leq \exp\left(\frac{(b-a)^2\lambda^2}{8}\right)$$

We omit the proof of this lemma!

# Hoeffding Bounds

**Hoeffding's Inequality**

Let $X_1, \ldots, X_n$ be independent random variables with mean $\mu_i$ such that $a_i \leq X_i \leq b_i$. Let $X = X_1 + \ldots + X_n$, and let $\mu = \mathbf{E}[X] = \sum_{i=1}^n \mu_i$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

# Hoeffding Bounds

**Hoeffding's Inequality**

Let $X_1, \ldots, X_n$ be independent random variables with mean $\mu_i$ such that $a_i \leq X_i \leq b_i$. Let $X = X_1 + \ldots + X_n$, and let $\mu = \mathbf{E}[X] = \sum_{i=1}^n \mu_i$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Proof Outline (skipped):

- Let $X_i' = X_i - \mu_i$ and $X' = X_1' + \ldots + X_n'$, then $\mathbf{P}[X \geq \mu + t] = \mathbf{P}[X' \geq t]$
- $\mathbf{P}[X' \geq t] \leq e^{-\lambda t} \prod_{i=1}^n \mathbf{E}\left[e^{\lambda X_i'}\right] \leq \exp\left[-\lambda t + \frac{\lambda^2}{8} \sum_{i=1}^n (b_i - a_i)^2\right]$
- Choose $\lambda = \frac{4t}{\sum_{i=1}^n (b_i - a_i)^2}$ to get the result.

This is not "magic" – you just need to optimise $\lambda$!

# Method of Bounded Differences

Framework

Suppose, we have independent random variables $X_1, \ldots, X_n$. We want to study the random variable:

$$f(X_1, \ldots, X_n)$$

# Method of Bounded Differences

**Framework**

Suppose, we have independent random variables $X_1, \ldots, X_n$. We want to study the random variable:

$$f(X_1, \ldots, X_n)$$

Some examples:

1. $X = X_1 + \ldots + X_n$ (our setting earlier)

# Method of Bounded Differences

Framework

Suppose, we have independent random variables $X_1, \ldots, X_n$. We want to study the random variable:

$$f(X_1, \ldots, X_n)$$

Some examples:

1. $X = X_1 + \ldots + X_n$ (our setting earlier)
2. In balls into bins, $X_i$ indicates where ball $i$ is allocated, and $f(X_1, \ldots, X_m)$ is the number of empty bins

# Method of Bounded Differences

**Framework**

Suppose, we have independent random variables $X_1, \ldots, X_n$. We want to study the random variable:

$$f(X_1, \ldots, X_n)$$

Some examples:

1. $X = X_1 + \ldots + X_n$ (our setting earlier)
2. In balls into bins, $X_i$ indicates where ball $i$ is allocated, and $f(X_1, \ldots, X_m)$ is the number of empty bins
3. In a randomly generated graph, $X_i$ indicates if the $i$-th edge is present and $f(X_1, \ldots, X_{\binom{n}{2}})$ represents the number of connected components of $G$

# Method of Bounded Differences

**Framework**

Suppose, we have independent random variables $X_1, \ldots, X_n$. We want to study the random variable:

$$f(X_1, \ldots, X_n)$$

Some examples:

1. $X = X_1 + \ldots + X_n$ (our setting earlier)
2. In balls into bins, $X_i$ indicates where ball $i$ is allocated, and $f(X_1, \ldots, X_m)$ is the number of empty bins
3. In a randomly generated graph, $X_i$ indicates if the $i$-th edge is present and $f(X_1, \ldots, X_{\binom{n}{2}})$ represents the number of connected components of $G$

In all those cases (and more) we can easily prove concentration of $f(X_1, \ldots, X_n)$ around its mean by the so-called **Method of Bounded Differences**.

# Method of Bounded Differences

A function $f$ is called Lipschitz with parameters $\mathbf{c} = (c_1, \ldots, c_n)$ if for all $i = 1, 2, \ldots, n$,

$$|f(x_1, x_2, \ldots, x_{i-1}, \boldsymbol{x_i}, x_{i+1}, \ldots, x_n) - f(x_1, x_2, \ldots, x_{i-1}, \boldsymbol{\widetilde{x}_i}, x_{i+1}, \ldots, x_n)| \leq c_i,$$

where $x_i$ and $\widetilde{x}_i$ are in the domain of the $i$-th coordinate.

# Method of Bounded Differences

A function $f$ is called Lipschitz with parameters $\mathbf{c} = (c_1, \ldots, c_n)$ if for all $i = 1, 2, \ldots, n$,

$$|f(x_1, x_2, \ldots, x_{i-1}, \mathbf{x_i}, x_{i+1}, \ldots, x_n) - f(x_1, x_2, \ldots, x_{i-1}, \widetilde{\mathbf{x_i}}, x_{i+1}, \ldots, x_n)| \leq c_i,$$

where $x_i$ and $\widetilde{x}_i$ are in the domain of the $i$-th coordinate.

**McDiarmid's inequality**

Let $X_1, \ldots, X_n$ be independent random variables. Let $f$ be Lipschitz with parameters $\mathbf{c} = (c_1, \ldots, c_n)$. Let $X = f(X_1, \ldots, X_n)$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

# Method of Bounded Differences

A function $f$ is called Lipschitz with parameters $\mathbf{c} = (c_1, \ldots, c_n)$ if for all $i = 1, 2, \ldots, n$,

$$|f(x_1, x_2, \ldots, x_{i-1}, \mathbf{x_i}, x_{i+1}, \ldots, x_n) - f(x_1, x_2, \ldots, x_{i-1}, \widetilde{\mathbf{x_i}}, x_{i+1}, \ldots, x_n)| \leq c_i,$$

where $x_i$ and $\widetilde{x}_i$ are in the domain of the $i$-th coordinate.

**McDiarmid's inequality**

Let $X_1, \ldots, X_n$ be independent random variables. Let $f$ be Lipschitz with parameters $\mathbf{c} = (c_1, \ldots, c_n)$. Let $X = f(X_1, \ldots, X_n)$. Then for any $t > 0$,

$$\mathbf{P}[\, X \geq \mu + t\,] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right),$$

and

$$\mathbf{P}[\, X \leq \mu - t\,] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

- Notice the similarity with Hoeffding's inequality! *[Exercise 2/3.14]*

# Method of Bounded Differences

A function $f$ is called Lipschitz with parameters $\mathbf{c} = (c_1, \ldots, c_n)$ if for all $i = 1, 2, \ldots, n$,

$$|f(x_1, x_2, \ldots, x_{i-1}, \mathbf{x_i}, x_{i+1}, \ldots, x_n) - f(x_1, x_2, \ldots, x_{i-1}, \widetilde{\mathbf{x_i}}, x_{i+1}, \ldots, x_n)| \leq c_i,$$

where $x_i$ and $\widetilde{x}_i$ are in the domain of the $i$-th coordinate.

**McDiarmid's inequality**

Let $X_1, \ldots, X_n$ be independent random variables. Let $f$ be Lipschitz with parameters $\mathbf{c} = (c_1, \ldots, c_n)$. Let $X = f(X_1, \ldots, X_n)$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

- Notice the similarity with Hoeffding's inequality! *[Exercise 2/3.14]*
- The proof is omitted here (it requires the concept of martingales).

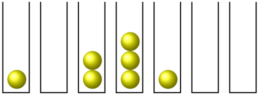
# Outline

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

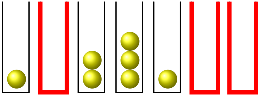
Applications of Method of Bounded Differences

# Application 3: Balls into Bins (again...)



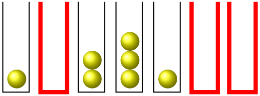
- Consider again $m$ balls assigned uniformly at random into $n$ bins.
- Enumerate the balls from 1 to $m$. Ball $i$ is assigned to a random bin $X_i$

# Application 3: Balls into Bins (again...)



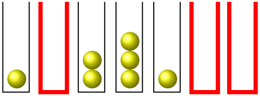
- Consider again $m$ balls assigned uniformly at random into $n$ bins.
- Enumerate the balls from 1 to $m$. Ball $i$ is assigned to a random bin $X_i$
- Let $Z$ be the number of empty bins (after assigning the $m$ balls)
- $Z = Z(X_1, \ldots, X_m)$ and $Z$ is Lipschitz with $\mathbf{c} = (1, \ldots, 1)$

# Application 3: Balls into Bins (again...)



- Consider again $m$ balls assigned uniformly at random into $n$ bins.
- Enumerate the balls from 1 to $m$. Ball $i$ is assigned to a random bin $X_i$
- Let $Z$ be the number of empty bins (after assigning the $m$ balls)
- $Z = Z(X_1, \ldots, X_m)$ and $Z$ is Lipschitz with $\mathbf{c} = (1, \ldots, 1)$
  (If we move one ball to another bin, number of empty bins changes by $\leq 1$.)

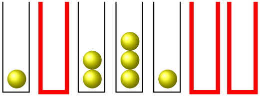
# Application 3: Balls into Bins (again...)



- Consider again $m$ balls assigned uniformly at random into $n$ bins.
- Enumerate the balls from 1 to $m$. Ball $i$ is assigned to a random bin $X_i$
- Let $Z$ be the number of empty bins (after assigning the $m$ balls)
- $Z = Z(X_1, \ldots, X_m)$ and $Z$ is Lipschitz with $\mathbf{c} = (1, \ldots, 1)$
  (If we move one ball to another bin, number of empty bins changes by $\leq 1$.)
- By McDiarmid's inequality, for any $t \geq 0$,

$$\mathbf{P}\left[\,|Z - \mathbf{E}[Z]| > t\,\right] \leq 2 \cdot e^{-2t^2/m}.$$

## Application 3: Balls into Bins (again...)

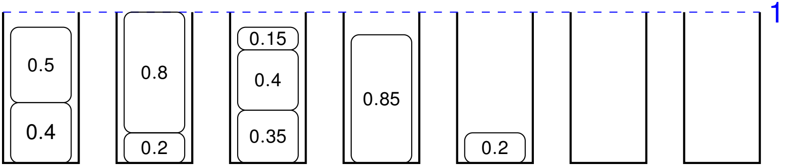


- Consider again $m$ balls assigned uniformly at random into $n$ bins.
- Enumerate the balls from 1 to $m$. Ball $i$ is assigned to a random bin $X_i$
- Let $Z$ be the number of empty bins (after assigning the $m$ balls)
- $Z = Z(X_1, \ldots, X_m)$ and $Z$ is Lipschitz with $\mathbf{c} = (1, \ldots, 1)$
  (If we move one ball to another bin, number of empty bins changes by $\leq 1$.)
- By McDiarmid's inequality, for any $t \geq 0$,

$$\mathbf{P}\left[\,|Z - \mathbf{E}\left[\,Z\,\right]| > t\,\right] \leq 2 \cdot e^{-2t^2/m}.$$

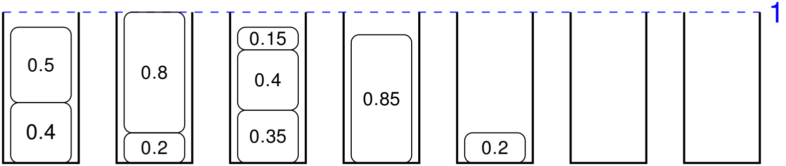
This is a decent bound, but for some values of $m$ it is far from tight and stronger bounds are possible through a refined analysis.

# Application 4: Bin Packing




- We are given $n$ items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the fewest number of unit-capacity bins
- Suppose the item sizes $X_i$ are independent random variables in $[0, 1]$

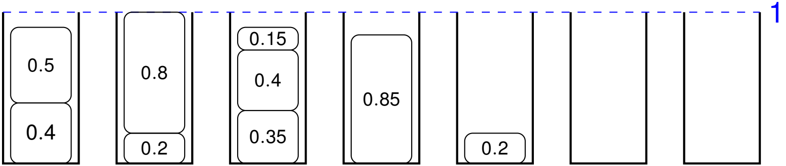
# Application 4: Bin Packing




- We are given $n$ items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the fewest number of unit-capacity bins
- Suppose the item sizes $X_i$ are independent random variables in $[0, 1]$

- Let $B = B(X_1, \ldots, X_n)$ be the optimal number of bins

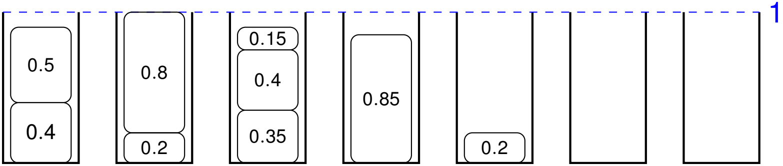
# Application 4: Bin Packing




- We are given $n$ items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the fewest number of unit-capacity bins
- Suppose the item sizes $X_i$ are independent random variables in $[0, 1]$

- Let $B = B(X_1, \ldots, X_n)$ be the optimal number of bins
- The Lipschitz conditions holds with $\boldsymbol{c} = (1, \ldots, 1)$. **Why?**

# Application 4: Bin Packing




- We are given $n$ items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the fewest number of unit-capacity bins
- Suppose the item sizes $X_i$ are independent random variables in $[0, 1]$

- Let $B = B(X_1, \ldots, X_n)$ be the optimal number of bins
- The Lipschitz conditions holds with $\boldsymbol{c} = (1, \ldots, 1)$. **Why?**
- Therefore

$$\mathbf{P}\left[\,|B - \mathbf{E}\left[\,B\,\right]| \geq t\,\right] \leq 2 \cdot e^{-2t^2/n}.$$

This is a typical example where proving concentration is much easier than calculating (or estimating) the expectation!