

# Randomized Algorithms

*Spezialvorlesung*  
*Wintersemester 05/06*

René Beier

Stefan Canzar

Stefan Funke

# Randomized Quicksort

**Input:** a set  $S$  of  $n$  different numbers

**Output:** the numbers of  $S$  in increasing order

**Algorithm:** randQS( $S$ )

1. Choose  $y$  from  $S$  uniformly at random
2.  $S_1 := \{x \in S \mid x < y\}; S_2 := \{x \in S \mid x > y\}$
3. If  $S_1 \neq \emptyset$  then randQS( $S_1$ )
4. output  $y$
5. If  $S_2 \neq \emptyset$  then randQS( $S_2$ )

# Example

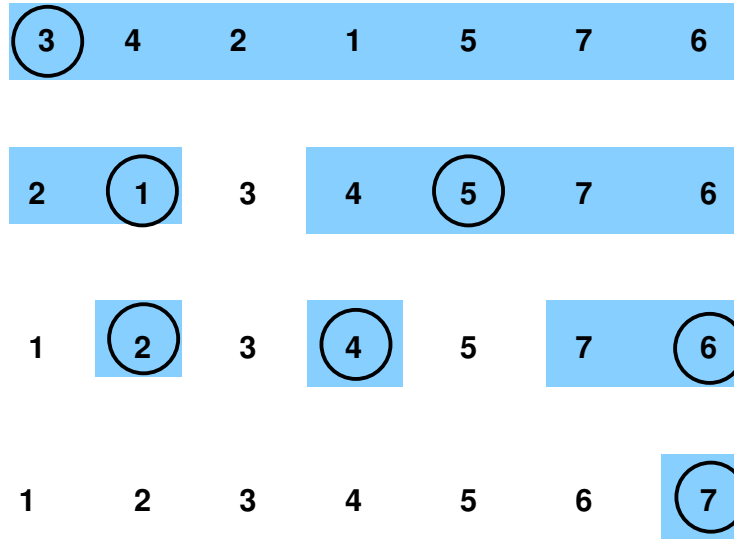
## Quicksort

- Algorithm
- Example
- The worst case
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

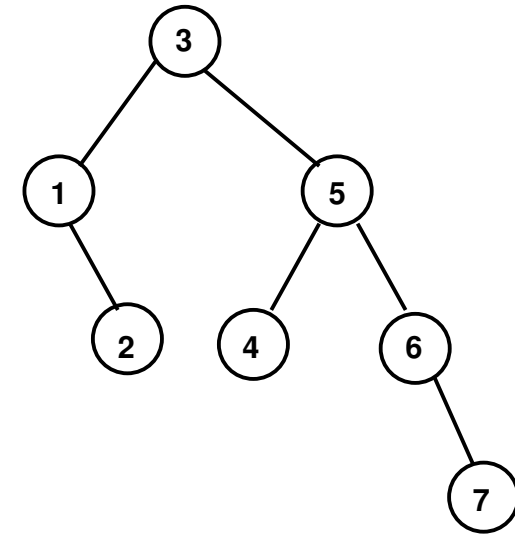
## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

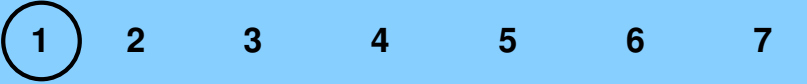




execution



recursion tree T



# The worst case

execution	#comp.	probability
	6	1/7
	5	1/6
	4	1/5
	1	1/2
	0	1

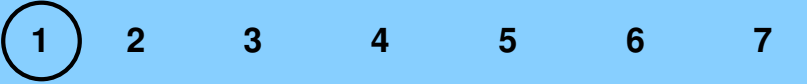




## Quicksort

- Algorithm
- Example
- **The worst case**
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

# The worst case

execution	#comp.	probability
	6	1/7
	5	1/6
	4	1/5
	1	1/2
	0	1

#comparisons:  $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \Theta(n^2)$

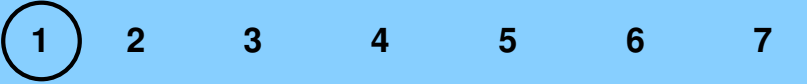




## Quicksort

- Algorithm
- Example
- **The worst case**
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

# The worst case

execution	#comp.	probability
	6	1/7
	5	1/6
	4	1/5
	1	1/2
	0	1

$$\text{\#comparisons: } \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \Theta(n^2)$$

$$\text{prob: } \prod_{i=1}^n \frac{1}{i} = \frac{1}{n!}$$

## Quicksort

- Algorithm
- Example
- **The worst case**
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

# Very Basic Probability Theory

## Quicksort

- Algorithm
- Example
- The worst case
- **Very Basic Probability Theory**
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

**Probability Space:** A (finite) sample space  $\Omega$  and a probability measure  $Pr$

**Sample Space  $\Omega$ :** A set of elementary events

**Probability measure  $Pr$ :** A function  $Pr : \mathcal{P}(\Omega) \rightarrow \mathbb{R}$  with

- For each  $\mathcal{E} \subseteq \Omega, 0 \leq Pr[\mathcal{E}] \leq 1$
- $Pr[\Omega] = 1$
- For each  $\mathcal{E} \subseteq \Omega, Pr[\mathcal{E}] = \sum_{\omega \in \mathcal{E}} Pr[\{\omega\}]$

**Random Variable  $X$ :** a function  $X : \Omega \rightarrow \mathbb{R}$ .

**Expectation** of a random variable:  $E[X] = \sum_{x \in \mathbb{N}} x \cdot Pr[X = x]$

**Fact:**  $E[X + Y] = E[X] + E[Y]$ .

# Analyzing Quick-sort

## Quicksort

- Algorithm
- Example
- The worst case
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

- We want to count the number of comparisons we perform.
- Let  $s_i$  be the  $i$ th smallest element of  $S$ .
- We define random variables  $X_{ij}$  as

$$X_{ij} = \begin{cases} 1 & \text{if we compare } s_i \text{ and } s_j \\ 0 & \text{otherwise} \end{cases}$$

- The expected running time (#comparisons) is

$$E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n E[X_{ij}]$$

- Thus we have to compute  $E[X_{ij}]$ .

$$E[X_{ij}] = 0 \cdot Pr[X_{ij} = 0] + 1 \cdot Pr[X_{ij} = 1] = Pr[X_{ij} = 1]$$

# Recursion Tree

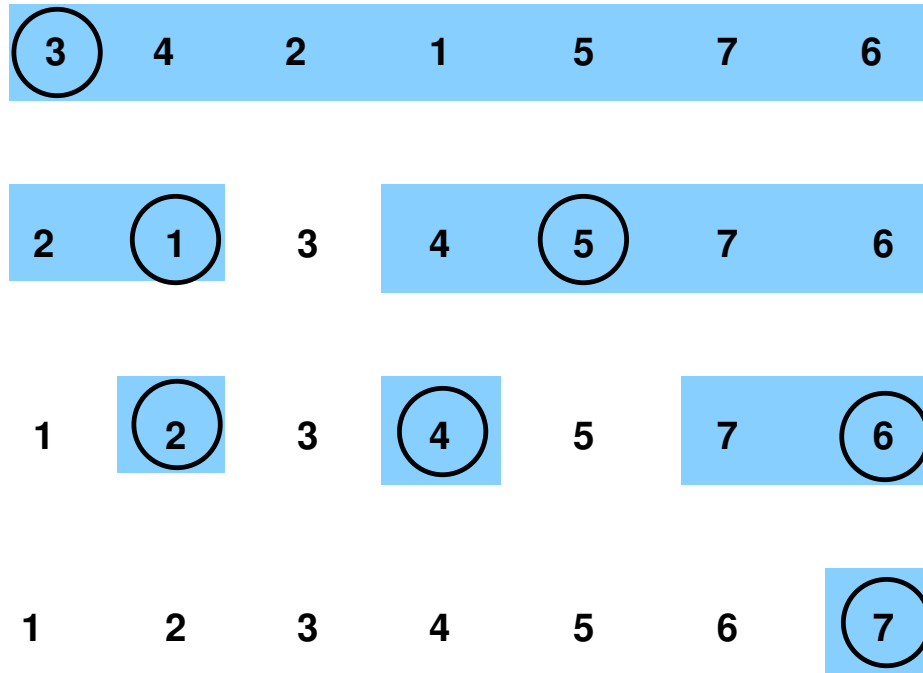
## Quicksort

- Algorithm
- Example
- The worst case
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

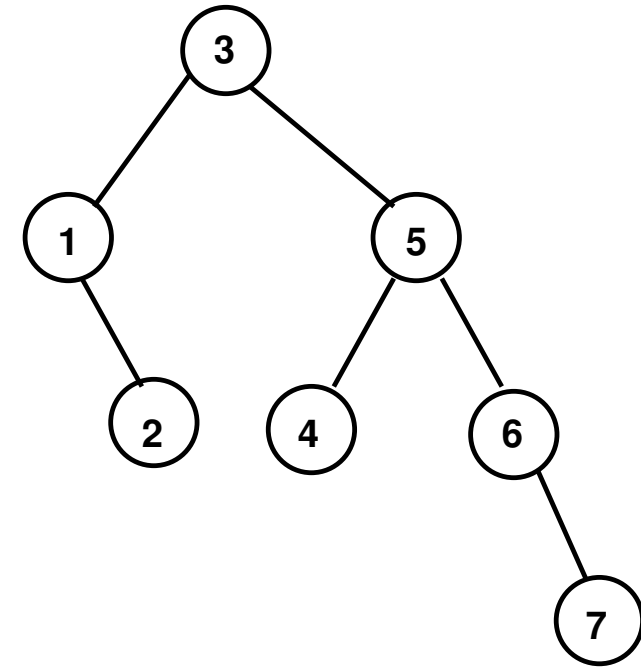
## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

execution



recursion tree T



$\pi = (3, 1, 5, 2, 4, 6, 7)$ : level order traversal

**Lemma:** There is a comparison between  $s_i$  and  $s_j$  ( $i < j$ ), iff  $s_i$  or  $s_j$  occurs earlier in  $\pi$  than any other element in  $S_{ij} = \{s_i, s_{i+1}, \dots, s_j\}$ .

## Quicksort

- Algorithm
- Example
- The worst case
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- **Lemma**
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

## Quicksort

- Algorithm
- Example
- The worst case
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- **Lemma**
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

**Lemma:** There is a comparison between  $s_i$  and  $s_j$  ( $i < j$ ), iff  $s_i$  or  $s_j$  occurs earlier in  $\pi$  than any other element in  $S_{ij} = \{s_i, s_{i+1}, \dots, s_j\}$ .

$s^*$ : element from  $S_{ij}$  that occurs first in  $\pi$ .

Observe: No pivot preceding  $s^*$  in  $\pi$  splits  $S_{i,j}$   
 $\Rightarrow S_{ij}$  contained in subtree rooted at  $s^*$ .

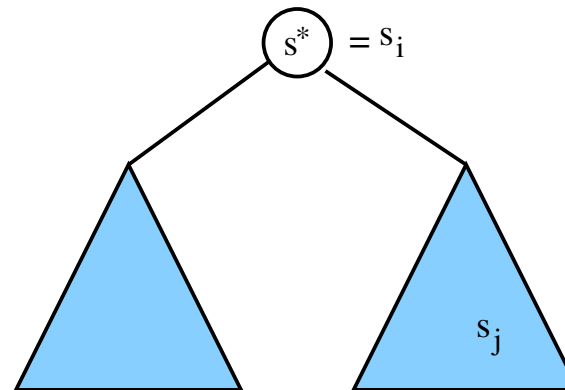
# Lemma

**Lemma:** There is a comparison between  $s_i$  and  $s_j$  ( $i < j$ ), iff  $s_i$  or  $s_j$  occurs earlier in  $\pi$  than any other element in  $S_{ij} = \{s_i, s_{i+1}, \dots, s_j\}$ .

$s^*$ : element from  $S_{ij}$  that occurs first in  $\pi$ .

Observe: No pivot preceding  $s^*$  in  $\pi$  splits  $S_{i,j}$   
 $\Rightarrow S_{ij}$  contained in subtree rooted at  $s^*$ .

Case I:  $s^* \in \{s_i, s_j\}$ :



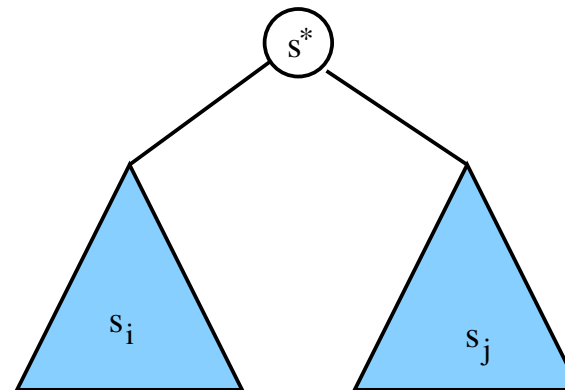
# Lemma

**Lemma:** There is a comparison between  $s_i$  and  $s_j$  ( $i < j$ ), iff  $s_i$  or  $s_j$  occurs earlier in  $\pi$  than any other element in  $S_{ij} = \{s_i, s_{i+1}, \dots, s_j\}$ .

$s^*$ : element from  $S_{ij}$  that occurs first in  $\pi$ .

Observe: No pivot preceding  $s^*$  in  $\pi$  splits  $S_{i,j}$   
 $\Rightarrow S_{ij}$  contained in subtree rooted at  $s^*$ .

Case II:  $s^* \notin \{s_i, s_j\}$ :



## Quicksort

- Algorithm
- Example
- The worst case
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

### Lemma:

$$\Pr [s^* \in \{s_i, s_j\}] = \frac{2}{j - i + 1}$$

Each element in  $S_{i,j}$  equally likely to be the first element in  $\pi$ .

## Quicksort

- Algorithm
- Example
- The worst case
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma
- Finalizing the Analysis

## Verifying Matrix Multiplication

## Las Vegas - Monte Carlo

**Lemma:**  $\Pr[X_{ij} = 1] = 2/(j - i + 1)$

$$\Rightarrow \mathbf{E}[X_{ij}] = 0 \cdot \Pr[X_{ij} = 0] + 1 \cdot \Pr[X_{ij} = 1] = 2/(j - i + 1)$$

- Algorithm
- Example
- The worst case
- Very Basic Probability Theory
- Analyzing Quick-sort
- Recursion Tree
- Lemma
- Lemma

**Lemma:**  $\Pr[X_{ij} = 1] = 2/(j - i + 1)$

$$\Rightarrow \mathbf{E}[X_{ij}] = 0 \cdot \Pr[X_{ij} = 0] + 1 \cdot \Pr[X_{ij} = 1] = 2/(j - i + 1)$$

**Theorem:** The expected number of comparisons performed is

$$\begin{aligned} \sum_{i=1}^{n-1} \sum_{j=i+1}^n \mathbf{E}[X_{ij}] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j - i + 1} \\ &= \sum_{i=2}^n \sum_{k=2}^i \frac{2}{k} \\ &= 2 \sum_{i=2}^n (H_i - 1) \\ &\leq 2nH_n \approx 2n \log n \end{aligned}$$

Verify  $A \cdot B = C$  for given matrices  $A, B$  and  $C$ .

- compute  $A \cdot C$  and compare to  $C \rightarrow$  naive:  $\Theta(n^3)$ , best  $\Theta(n^{2.37})$
- Faster randomized algorithm with small error probability:
  1. Choose  $r = (r_1, \dots, r_n) \in \{0, 1\}^n$  at random.
  2. If  $A(Br) = Cr$  output “TRUE” else output “FALSE”

One-sided error: false positives

Running time:  $O(n^2)$

**Theorem:** If  $AB \neq C$  then  $\Pr[ABr = Cr] \leq 1/2$

Define  $D = AB - C \neq 0$

$$ABr = Cr \Leftrightarrow (AB - C)r = 0 \Leftrightarrow Dr = 0$$

W.l.o.g,  $d_{1,1} \neq 0$

$$\sum_{j=1}^n d_{1,j} r_j = 0$$

$$r_1 = \frac{\sum_{j=2}^n d_{1,j} r_j}{d_{1,1}}$$

- Fix all  $r_j$  but  $r_1$ .
- Equality holds for at most one of the two choices for  $r_1$ .

$$\Rightarrow \mathbf{Pr}[Dr = 0] \leq 1/2$$

Can we decrease the error-probability?

**Idea:** Repeat the test  $k$  times with independent random choices for  $r$

- If some test reports “FALSE” we know for sure  $AB \neq C$
- If  $AB \neq C$  then the probability that all test report “TRUE” is at most  $2^{-k}$
- running time:  $O(kn^2)$

## Las Vegas Algorithm:

- always gives the correct/optimal solution.
- running time is a random variable.
- Example: Quick-sort.

## Monte Carlo Algorithm:

- may produce incorrect solutions.
- has a fixed deterministic running time.
- Example: Verifying Matrix-Multiplication
- For decision problems there are two kinds:
  - one-sided errors:** either *yes* or *no* answers always correct.
  - two-sided errors:** both have non-zero probability to be wrong.

**Lemma:** Let  $X$  be a random variable with non-negative domain.

$$\Pr [X \geq t] \leq \frac{\mathbf{E}[X]}{t}$$

or equivalently

$$\Pr [X \geq t\mathbf{E}[X]] \leq \frac{1}{t}$$

**Proof:** Assume for simplicity that the domain of  $X$  is  $\mathbb{N}_0$

$$\begin{aligned} \mathbf{E}[X] &= \sum_{i \geq 0} i \Pr [X = i] \\ &\geq \sum_{i \geq t} t \Pr [X = i] \\ &= t \Pr [X \geq t] \end{aligned}$$

Given Las Vegas algorithm with expected running time at most  $f(n)$ .

**Idea:** Stop algorithm after  $\alpha f(n)$  time.

- Only error source: Stopping algo before completion.
- Probability:  $\Pr[T > \alpha f(n)] \leq 1/\alpha$

$\Rightarrow$  Monte Carlo algorithm with running time  $\alpha f(n)$  and error rate  $1/\alpha$

Given Monte Carlo algorithm MC with

- deterministic running time at most  $f(n)$  and
- success probability at least  $p(n)$ .

Assume we can verify the correctness of the output in time  $g(n)$ .

**Theorem:** There exists Las Vegas algorithm with expected running time

$$\mathbf{E}[T] \leq \frac{f(n) + g(n)}{p(n)}$$

**Proof:** Repeat calling MC until verifier certifies correctness of output.

- Let  $i$  be the number of iterations of this scheme
- Running time  $T \leq i(f(n) + g(n))$
- $\mathbf{E}[T] \leq (f(n) + g(n))\mathbf{E}[i]$

$$\mathbf{E}[i] = \sum_{k=1}^{\infty} k(1 - p(n))^{k-1} p(n) = \frac{1}{p(n)}$$

as

$$\sum_{k=1}^{\infty} kx^k = \frac{x}{(1-x)^2}, \text{ for } x < 1$$