Algorithm Theory

03 - Randomization

Design Principle for algorithms

Dr. Alexander Souza

Winter term 11/12

Randomization

allowed to flip coins and decide on the outcomes of the coin flips

- Types of randomized algorithms
- Randomized Quicksort LV
- Randomized primality test MC
- Cryptography RSA

Two types
Monte Carlo mostly correct
Las Vegas allways correct

1. Types of randomized algorithms

- **Las Vegas algorithms**

  always correct; expected running time

  Example: randomized Quicksort

  Worst-case: $O(n^2)$ running time

  average-case: $O(n \log n)$ expected running time

- **Monte Carlo algorithms** (mostly correct):

  probably correct; guaranteed running time

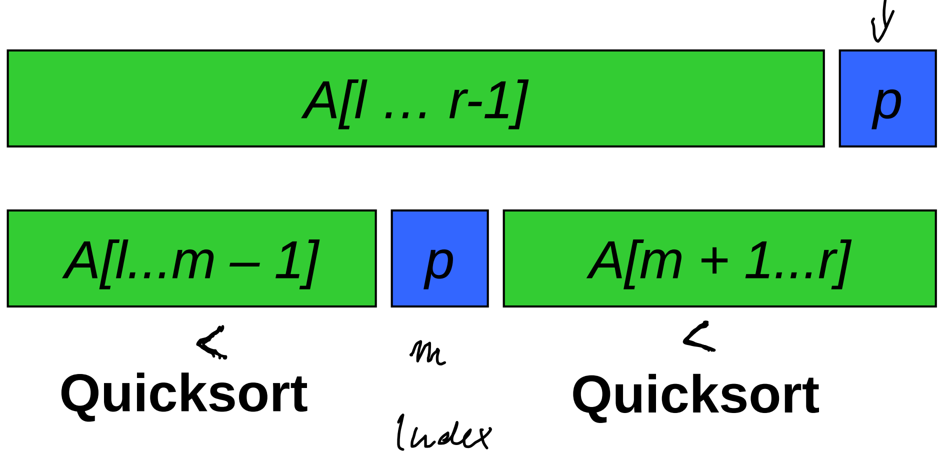
  Example: randomized primality test

  "n composite": n is a composite number

  "probably prime": n could be composite or prime

2. Quicksort

Unsorted range *A*[*l*,*r*] in an array *A*:   pivot element



Quicksort

Algorithm: Quicksort

Input: unsorted range [*l*, *r*] in array *A*

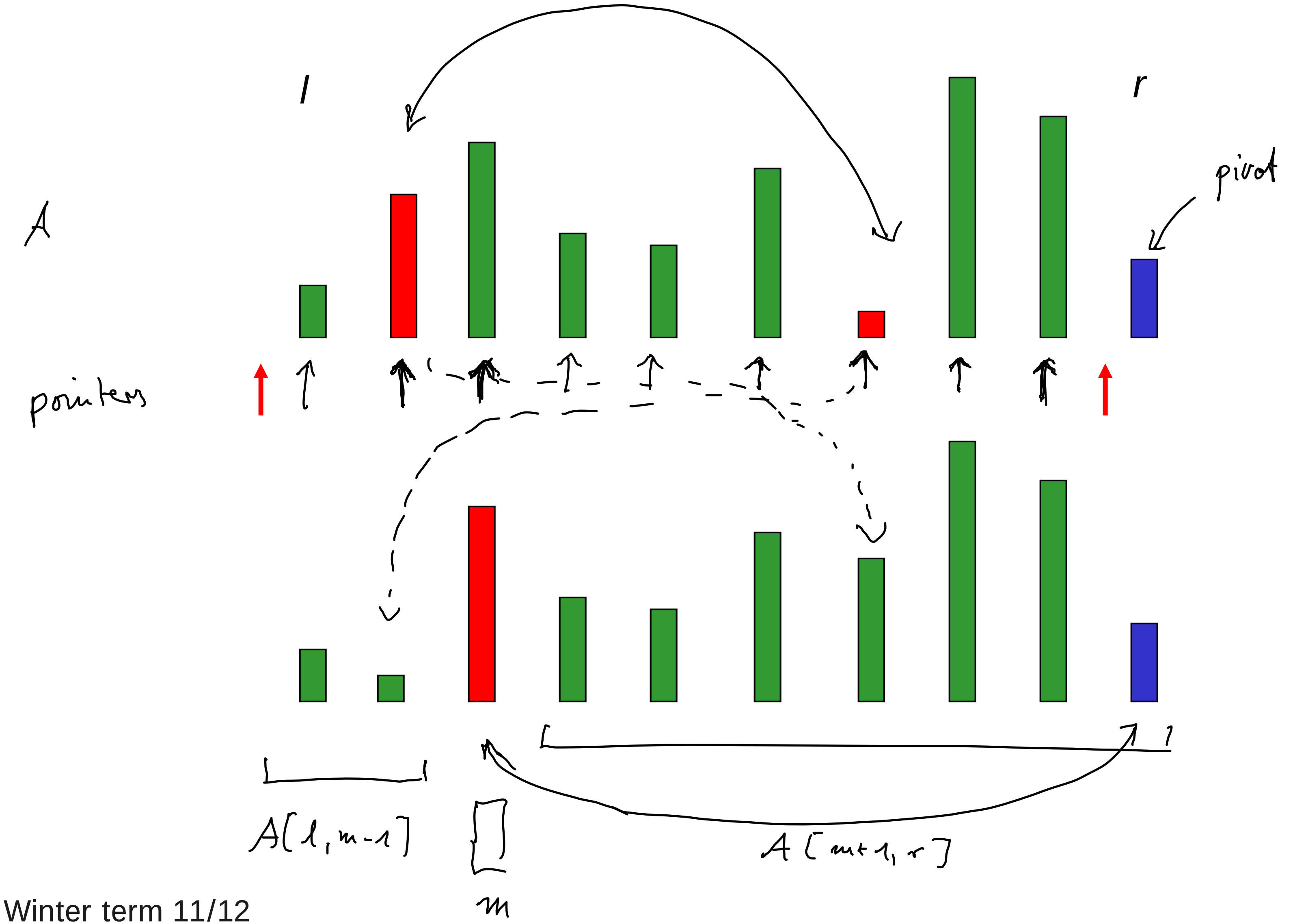
Output: sorted range [*l*, *r*] in array *A*

```
1  if  r > l
2      then choose pivot element p = A[r]
3           m = divide(A, l , r)
            /* divide A according to p:
            A[l],....,A[m – 1] ≤ p ≤ A[m + 1],...,A[r]
            */
4           Quicksort(A, l , m - 1)
5           Quicksort (A, m + 1, r)
```

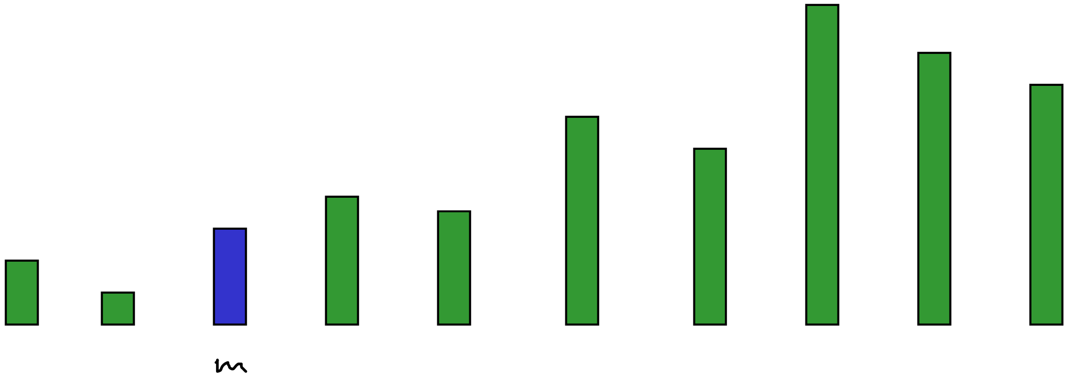
$a \in A[l, m-1]$ $a < p$

$a \in A[m+1, r]$ $p < a$

Partitioning step



Partitioning step

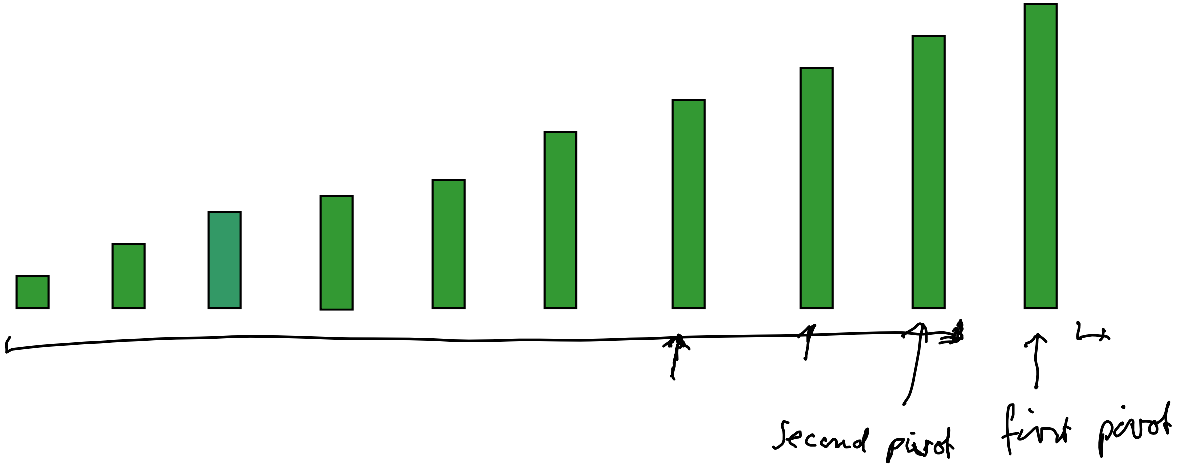


divide(A, l , r):

• returns the index of the pivot element in A
• running time $O(r - l)$

comparisons

Worst-case input



n elements*:*

Running time: $\underline{(n-1)} + \underline{(n-2)} + \ldots + \underline{2} + \underline{1} = n(n-1)/2 = \Theta(n^2)$

3. Randomized Quicksort

Random choices are allowed: Pivot element chosen uniformly at random

Algorithm: Quicksort

Input: unsorted range [*l*, *r*] in array *A*
Output: sorted range [*l*, *r*] in array *A*

uniformly

1 **if** $r > l$
2 **then** choose pivot element $p = A[i]$ in the range [*l*, *r*] at random
3 swap $A[i]$ and $A[r]$ A[i]
4 m = divide(A, l, r)
/* divide *A* according to *p*:
$A[l],....,A[m-1] \leq p \leq A[m+1],...,A[r]$
*/
5 Quicksort(A, l, $m - 1$)
6 Quicksort(A, $m + 1$, r)

Identical to det. version of Q.S.

Analysis 1

$T(n)$ = expected number of comparisons of randomized Quicksort on an array of size n

n elements; let S_i be the i-th smallest element

With probability $1/n$, S_1 is the pivot element:
subproblems of sizes 0 and $n-1$

$\emptyset$ | S_1 | $n-1$ |

$$\frac{1}{n} \cdot \left(T(0) + T(n-1) + n-1 \right)$$

divide

•
•
•

With probability $1/n$, S_k is the pivot element:
subproblems of sizes $k-1$ and $n-k$

| $k-1$ | S_k | $n-k$ |

$$\frac{1}{n} \cdot \left(T(k-1) + T(n-k) + n-1 \right)$$

•
•
•

With probability $1/n$, S_n is the pivot element:
subproblems of sizes $n-1$ and 0

| $n-1$ | S_n | $\emptyset$

$$\frac{1}{n} \cdot \left(T(n-1) + T(0) + n-1 \right)$$

$$T(n) = \sum_{k=0}^{n-1} \frac{1}{n} \cdot \left(T(k) + T(n-1-k) \right) + n - 1$$

Analysis 1

Expected running time:

$$T(n) = \frac{1}{n}\sum_{k=1}^{n}\left(T(\overset{0,1,\dots,n-1}{k-1}) + T(\overset{n-1,n-2,\dots,0}{n-k})\right) + n - 1$$

$$= \frac{2}{n}\sum_{k=0}^{n-1} T(k) + n - 1$$

$$= O(n \log n)$$

Analysis 1

$T(n)$ = exp. # comparisons size n

$$T(0) = 0$$

$$T(n) = \frac{2}{n} \cdot \sum_{k=0}^{n-1} T(k) + n - 1 \overset{!}{=} 2(n+1) H_n - 4n \quad (*)$$

Recall: $H_n = \sum_{k=1}^{n} \frac{1}{k}$ n-th Harmonic number, $H_n = \ln n + O(1)$

Identity: $\sum_{k=0}^{n-1} (k+1) H_k = \frac{n}{2} \cdot \left(n \cdot H_n - \frac{n}{2} - \frac{3}{2} + H_n \right)$

Base Case $n=1$: $T(1) = \frac{2}{1} \cdot 0 + 1 - 1 = 0 = 2 \cdot (1+1) \cdot 1 - 4$

Inductive Case $n-1 \curvearrowright n$:

$$T(n) = n - 1 + \frac{2}{n} \cdot \sum_{k=0}^{n-1} T(k) = n - 1 + \frac{2}{n} \cdot \sum_{k=0}^{n-1} \left(2(k+1) H_k - 4k \right)$$

$$= n - 1 + 2 \cdot \left(n H_n - \frac{n}{2} - \frac{3}{2} + H_n \right) - 4(n-1)$$

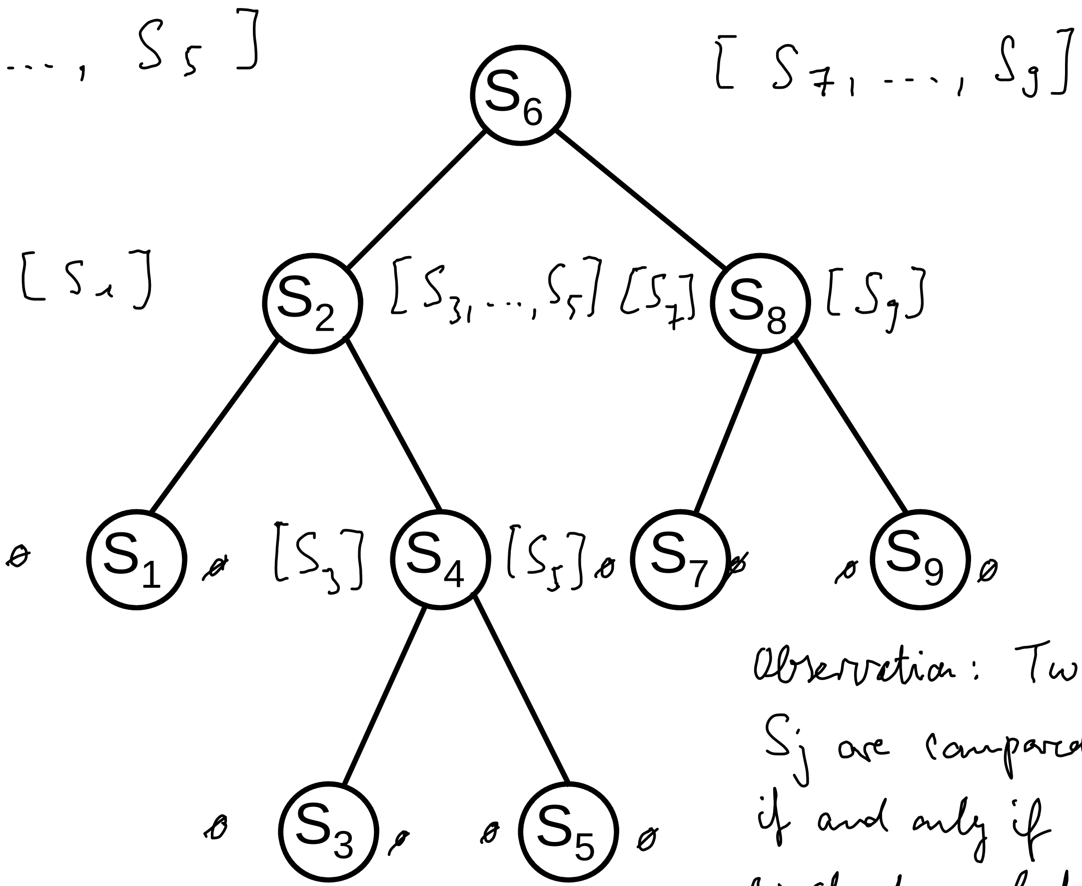
$$= 2(n+1) H_n - n - 3 - 3(n-1)$$

$$= 2(n+1) H_n - 4n$$

$$= O(n \log n)$$

Analysis 2: Representation of QS as a tree

S_i : i-th smallest element



$\pi = S_6\ S_2\ S_8\ S_1\ S_4\ S_7\ S_9\ S_3\ S_5$

Observation: Two elements S_i and S_j are compared by Quicksort if and only if there is an ancestor relation between S_i and S_j.

Analysis 2

Expected number of comparisons:

Random variables

$$X_{ij} = \begin{cases} 1 & \text{if } S_i \text{ is compared to } S_j \\ 0 & \text{otherwise} \end{cases}$$

indicator variable

linearity of expectation

$$E\left[\underbrace{\sum_{i=1}^{n} \sum_{j>i} X_{ij}}\right] = \sum_{i=1}^{n} \sum_{j>i} E[X_{ij}] = \sum_{i=1}^{n} \sum_{j>i} p_{ij}$$

expected value

p_{ij} = probability that S_i is compared to S_j

$$E[X_{ij}] = 1 \cdot p_{ij} + 0 \cdot (1 - p_{ij}) = p_{ij}$$

value prob.

Linearity of Expectation

$$\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$$

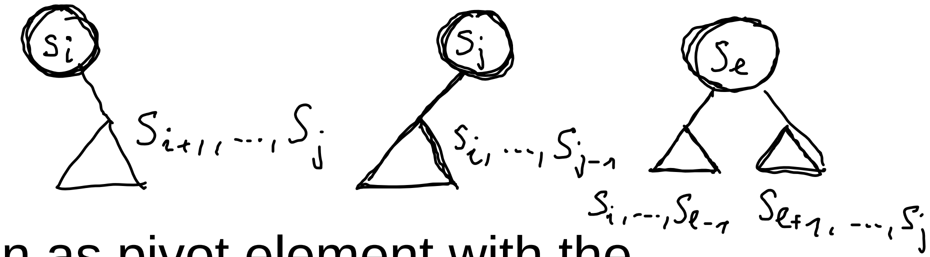
Always true for finite sums,

even if X and Y are dependent

Computing p_{ij}

- S_i is compared to S_j iff S_i or S_j are chosen as pivot element before any S_l , $i<l<j$. $\Longleftrightarrow$ $i < l < j$

  $\{S_i \ \dots \ S_l \ \dots \ S_j\}$

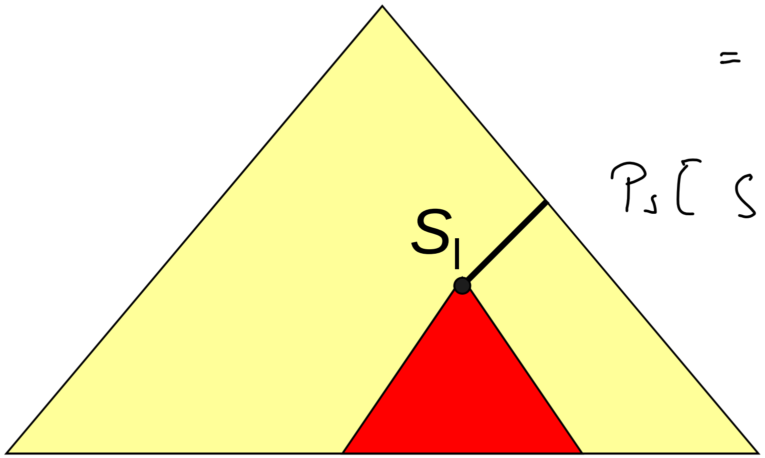


- Any element S_i , … , S_j is chosen as pivot element with the same probability.

$Pr[\, S_i \text{ is chosen first pivot from } \{S_i, \dots, S_j\}\,]$

$= \dfrac{1}{|\{S_i, \dots, S_j\}|} = \dfrac{1}{j-i+1}$

$Pr[\, S_j \ \dots \,] = \dfrac{1}{j-i+1}$



$\{\dots\ S_i \ \dots \ S_l \ \dots \ S_j \ \dots\}$

$p_{ij} = 2 \,/\, (j-i+1)$

$p_{ij} = Pr[S_i \dots] + Pr[S_j \dots]$

$= \dfrac{2}{j-i+1}$

Analysis 2

Expected number of comparisons:

$$\mathbb{E}\left[\sum_{i=1}^{n}\sum_{j>i} X_{ij}\right] = \sum_{i=1}^{n}\sum_{j>i} p_{ij} = \sum_{i=1}^{n}\sum_{j>i}\frac{2}{j-i+1}$$

$$k = j-i+1$$

$$j = i+1 \Rightarrow k = 2$$

$$j = n \Rightarrow k = n-i+1$$

$$= \sum_{i=1}^{n}\sum_{k=2}^{n-i+1}\frac{2}{k}$$

$$\le 2\sum_{i=1}^{n}\underbrace{\sum_{k=1}^{n}\frac{1}{k}}_{H_n} = 2\cdot\sum_{i=1}^{n} H_n = 2n\cdot H_n$$

$$= O(n \log n)$$

$$\cancel{= 2n\sum_{k=1}^{n}\frac{1}{k}}$$

$$H_n = \sum_{k=1}^{n} 1/k \approx \ln n$$

4. Primality test

Definition:

A natural number $p \geq 2$ is prime iff $a \mid p$ implies that $a = 1$ or $a = p$.

We consider primality tests for numbers $n \geq 2$.

Algorithm: Deterministic primality test (naive approach)

Check all $a = 1, \ldots, \sqrt{n}$

Input: Natural number $n \geq 2$

Output: Answer to the question „Is n prime?“

$n = a \cdot b$

Assume $a > \sqrt{n}$, $b > \sqrt{n}$

$n = a \cdot b > \sqrt{n} \cdot \sqrt{n} = n$ ↯

if n = 2 **then return** true
if n even **then return** false
for i = 1 **to** $\sqrt{n}/2$ **do**
  if $2i + 1$ divides n
  then return false
return true

Need: running time $O(\log^c n)$, c const

polynomial time algorithm? No!

Running time: $\Theta(\sqrt{n})$

Input size: $O(\log n)$

Running time: $\Theta(\sqrt{n}) = \Theta(2^{\log n / 2})$

Primality test

Goal: Monte Carlo alg. with poly. running time

Randomized algorithm

- Polynomial running time.
- If it returns "not prime", then *n* is not prime.
- If it returns "prime", then with probability at most *p, p>0,* *n* is composite.

Want $p \in (0, \frac{1}{2})$

After *k* iterations: with probability p^k , *n* is composite .

Primality test

Fact: For any odd prime number *p:* $2^{p-1} \bmod p = 1$.

Examples: $p = 17$, $2^{16} - 1 = 65535 = 17 * 3855$ $\quad 2^{16} = 17 \cdot 3855 + 1$

$p = 23$, $2^{22} - 1 = 4194303 = 23 * 182361$ $\quad 2^{16} \bmod 17 = 1$

Input : n

Simple primality test:

1 Compute $z = 2^{n-1} \bmod n$

2 **if** $z = 1$

3 **then** "n is possibly prime"

4 **else** "n is composite"

Repeated Squaring : a^n

$$a^n = \left(a^{\frac{n}{2}}\right)^2$$

$$pow(a,n) = \begin{cases} 1 & n = 0 \\ a \cdot pow(a, n-1) & n \text{ odd} \\ \left[pow\left(a, \frac{n}{2}\right)\right]^2 & n \text{ even} \end{cases}$$

Advantage: polynomial running time.

$O(\log^3 n)$