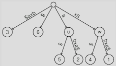
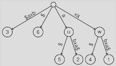


Datenstrukturen und Effiziente Algorithmen

Vorlesung *Datenstrukturen und Effiziente Algorithmen* im WS 18/19

Marc Hellmuth
Institut für Mathematik und Informatik
Universität Greifswald

Marc Hellmuth



- basics
- some simple graph theory
- some types of algorithms
  (which you should know)

Recap: What is an Algorithm?

Etymology

The word "algorithm" has its roots in Latinizing the name of Muhammad ibn Musa al-Khwarizmi ($\sim$ 780-850) in a first step to algorismus, who was a Persian mathematician, astronomer, geographer, and scholar in the House of Wisdom in Baghdad.

He wrote a treatise "on the Hindu-Arabic numeral system" in Arabic language.

Recap: What is an Algorithm?

Etymology

The word "algorithm" has its roots in Latinizing the name of Muhammad ibn Musa al-Khwarizmi ($\sim$ 780-850) in a first step to algorismus, who was a Persian mathematician, astronomer, geographer, and scholar in the House of Wisdom in Baghdad.

He wrote a treatise "on the Hindu-Arabic numeral system" in Arabic language.

Translated into Latin during the 12th century under the title *Algoritmi de numero Indorum. "Algoritmi on the numbers of the Indians"*.

Recap: What is an Algorithm?

Etymology

The word "algorithm" has its roots in Latinizing the name of Muhammad ibn Musa al-Khwarizmi ($\sim$ 780-850) in a first step to algorismus, who was a Persian mathematician, astronomer, geographer, and scholar in the House of Wisdom in Baghdad.

He wrote a treatise "on the Hindu-Arabic numeral system" in Arabic language.

Translated into Latin during the 12th century under the title *Algoritmi de numero Indorum. "Algoritmi on the numbers of the Indians"*.

"Algoritmi" was the translator's Latinization of Al-Khwarizmi's name. In the 15th century, under the influence of the Greek word 'number' (cf. 'arithmetic'), the Latin word was altered to algorithmus, and the corresponding English term 'algorithm' is first attested in the 17th century; the modern sense was introduced in the 19th century.

Recap: What is an Algorithm?

Etymology

The word "algorithm" has its roots in Latinizing the name of Muhammad ibn Musa al-Khwarizmi ($\sim$ 780-850) in a first step to algorismus, who was a Persian mathematician, astronomer, geographer, and scholar in the House of Wisdom in Baghdad.

He wrote a treatise "on the Hindu-Arabic numeral system" in Arabic language.

Translated into Latin during the 12th century under the title *Algoritmi de numero Indorum. "Algoritmi on the numbers of the Indians"*.

"Algoritmi" was the translator's Latinization of Al-Khwarizmi's name. In the 15th century, under the influence of the Greek word 'number' (cf. 'arithmetic'), the Latin word was altered to algorithmus, and the corresponding English term 'algorithm' is first attested in the 17th century; the modern sense was introduced in the 19th century.

First algorithm (computer program) by Ada Lovelace (1843) for the for the Analytical Engine (by Charles Babbage) to compute Bernoulli numbers.

Recap: What is an Algorithm?

Informal

Every well-defined computable procedure which

- has as input (a finite set of) some values
- applies a sequence of operations on the values
- has as output (a finite set of) some values

Recap: What is an Algorithm?

(More) Formal

Via Turing-machines (Alan Turing)

TM mathematically models a machine that mechanically operates on a tape.

On this tape are symbols, which the machine can read and write, one at a time, using a tape head.

Operation is fully determined by a finite set of elementary instructions such as *"in state 42, if the symbol seen is 0, write a 1; if the symbol seen is 1, change into state 17; in state 17, if the symbol seen is 0, write a 1 and change to state 6;*

Recap: What is an Algorithm?

(More) Formal

Via Turing-machines (Alan Turing)

TM mathematically models a machine that mechanically operates on a tape.

On this tape are symbols, which the machine can read and write, one at a time, using a tape head.

Operation is fully determined by a finite set of elementary instructions such as *"in state 42, if the symbol seen is 0, write a 1; if the symbol seen is 1, change into state 17; in state 17, if the symbol seen is 0, write a 1 and change to state 6;*

A computational rule for solving a problem is called Algorithm if and only if there exists an equivalent Turing machine to this computation rule and stops for each input that has a solution.

Recap: What is an Algorithm?

(More) Formal

Via Turing-machines (Alan Turing)

TM mathematically models a machine that mechanically operates on a tape.

On this tape are symbols, which the machine can read and write, one at a time, using a tape head.

Operation is fully determined by a finite set of elementary instructions such as *"in state 42, if the symbol seen is 0, write a 1; if the symbol seen is 1, change into state 17; in state 17, if the symbol seen is 0, write a 1 and change to state 6;*

A computational rule for solving a problem is called Algorithm if and only if there exists an equivalent Turing machine to this computation rule and stops for each input that has a solution.

Necessary Conditions:

- The procedure must be clearly describable in a finite text (finite).
- Every step of the procedure must be executable (executability).
- The method needs only finite amount of memory at any given time (dynamic finiteness).
- The procedure may only need a finite number of steps (termination).

Recap: What is a Data Structure ?

Data Structure

is a data organization, management and storage format that enables efficient access and modification

Recap: What is a Data Structure ?

Data Structure

is a data organization, management and storage format that enables efficient access and modification

Efficiency

= "Speed" and "economical usage of resources".

Recap: What is a Data Structure ?

Data Structure

is a data organization, management and storage format that enables efficient access and modification

Efficiency

= "Speed" and "economical usage of resources".

Example Blackboard:

- Merge-sort VS Insertion-sort

Recap: What is a Data Structure ?

Data Structure

is a data organization, management and storage format that enables efficient access and modification

Efficiency

= "Speed" and "economical usage of resources".

Example Blackboard:

- Merge-sort VS Insertion-sort

Reminder: O-/Θ-/Ω-Notation

For positive functions f and g, we define

- $g(n) \in O(f(n)) :\Leftrightarrow \exists c > 0, n_0 > 0 : \forall n > n_0 : g(n) \leq cf(n)$
- $g(n) \in \Omega(f(n)) :\Leftrightarrow \exists c > 0, n_0 > 0 : \forall n > n_0 : g(n) \geq cf(n)$
- $g(n) \in \Theta(f(n)) :\Leftrightarrow g(n) \in O(f(n))$ and $g(n) \in \Omega(f(n))$.

The notation $g(n) = O(f(n))$ is also very commonly used.

Graph Terminology

directed graph	gerichteter Graph	(V, E), $E \subseteq V \times V$
undirected graph	ungerichteter Graph	(V, E), either $E \subseteq \binom{V}{2}$ or $E \subseteq V \times V$ with $(u, v) \in E \iff (v, u) \in E$
vertex, node	Knoten	$v \in V$
edge	Kante	$(u, v) \in E$ (directed) $\{u, v\} \in E$ (undirected)
degree	Grad	$\deg(v) = \lvert\{u \in V \mid, \{u, v\} \in E\}\rvert$
in-degree	Eingrad	$\text{indeg}(v) = \lvert\{u \in V \mid, (u, v) \in E\}\rvert$
out-degree	Ausgrad	$\text{outdeg}(v) = \lvert\{u \in V \mid, (v, u) \in E\}\rvert$
adjacent	benachbart	two vertices can be adjacent
incident	inzident	an edge can be incident to a vertex
path	Weg	$u \rightsquigarrow v$, $(u = u_0, u_1, \ldots, u_k = v)$ and (u_i, u_{i+1}) are edges $0 \leq i \leq k - 1$
length of path	Länge eines Weges	k
edge on path	Kante auf Weg	edges (u_i, u_{i+1}) are on path
simple path	Pfad	$i \neq j \Rightarrow u_i \neq u_j$
self-loop	Selbstschleife	$(u, u) \in E$
cycle	Kreis	$u \rightsquigarrow u$
DAG, directed acyclic graph	gerichteter azyklischer Graph	gerichteter Graph ohne Kreise

Graph Terminology - Blackboard

- isomorphism
- (induced) subgraph ((induzierter) Teilgraph)
- complete graph (vollständiger Graph)
- complement (Komplement)
- Tree (Baum)
- connected component (Zusammenhangskomponente)
- Forest (Wald)
- Spanningtree (Spannbaum)

Results (Undirected Graphs) - Blackboard/Exercise

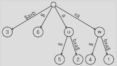
Lemma 1.1

The following statements are equivalent

- The graph $T = (V, E)$ is a tree.
- For all $u, v \in V$ there is exactly one simple path $u \rightsquigarrow v$.
- T is connected and $|E| = |V| - 1$.

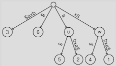
Corollary 1.1

If $G = (V, E)$ is connected, then $|E| \geq |V| - 1$ and G has a spanningtree $T \subseteq G$.



(Some not necessarily disjoint) Types of Algorithms

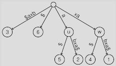
- Iterative Algorithms
- Recursive Algorithms
- Dynamic Algorithms
- Heuristic Algorithms
- Probabilistic Algorithms



Iterative Alg (Exmpl: n-Factorial)

int FAC_ITER(int n)

```
int SOLUTION ← 1
for int i = 2, ..., n do
    SOLUTION ← SOLUTION*i
return SOLUTION
```



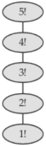
Recursive Alg (Exmpl: n-Factorial)

Tab. 2.3 Rekursive Methodenaufrufe und Rückwärts-Einsetzen

fakultaet(5)	Methodenaufruf
5*fakultaet(4)	1. rekursiver Aufruf
5*4*fakultaet(3)	2. rekursiver Aufruf
5*4*3*fakultaet(2)	3. rekursiver Aufruf
5*4*3*2*fakultaet(1)	4. rekursiver Aufruf
5*4*3*2*1	Abbruchbedingung, keine weiteren Aufrufe.
5*4*3*2	Einsetzen des Ergebnisses aus 3. Aufruf
5*4*6	Einsetzen des Ergebnisses aus 2. Aufruf
5*24	Einsetzen des Ergebnisses aus 1. Aufruf
120	Resultat

int FAC_REC(int n)

```
if n = 1 then
    return 1
return n∗FAC_REC(n − 1)
```



bb. 2.8 Lineare Rekursion

Algorithmen kompakt und verständlich, Rimscha, Springer, 2017

Recursive Alg (Exmpl: Tower of Hanoi)

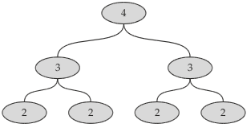
HANOI_REC(string SOURCE, string TARGET, string BUFFER, int *n*)

```
if n = 1 then
    printout "Move topmost disc from" SOURCE "to" TARGET
else
    HANOI_REC(SOURCE, BUFFER, TARGET, n − 1)
    HANOI_REC(SOURCE, TARGET, BUFFER, 1)
    HANOI_REC(BUFFER, TARGET, SOURCE, n − 1)
```

Abb. 2.10 Ablauf des Türme-von-Hanoi-Spiels bei einer Höhe von 4

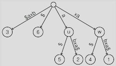
```
hanoi("Links", "Mitte", "Rechts", 4);
Oberste Scheibe von Links nach Rechts versetzen.
Oberste Scheibe von Links nach Mitte versetzen.
Oberste Scheibe von Rechts nach Mitte versetzen.
Oberste Scheibe von Links nach Rechts versetzen.
Oberste Scheibe von Mitte nach Links versetzen.
Oberste Scheibe von Mitte nach Rechts versetzen.
Oberste Scheibe von Links nach Rechts versetzen.
Oberste Scheibe von Links nach Mitte versetzen.
Oberste Scheibe von Rechts nach Mitte versetzen.
Oberste Scheibe von Rechts nach Links versetzen.
Oberste Scheibe von Mitte nach Links versetzen.
Oberste Scheibe von Rechts nach Mitte versetzen.
Oberste Scheibe von Links nach Rechts versetzen.
Oberste Scheibe von Links nach Mitte versetzen.
Oberste Scheibe von Rechts nach Mitte versetzen.
```

Abb. 2.11 Baumrekursion



[2] Es ist jeweils n angegeben, die uninteressanten Methodenaufrufe mit Parameter n = 1 sind hier aus Gründen der Übersichtlichkeit nicht dargestellt.

Algorithmen kompakt und verständlich, Rimscha, Springer, 2017



Dynamic Alg (Exmpl: Fibonacci number)

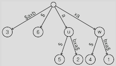
init $fib(0) = 0$, $fib(1) = 1$, $fib(i) = -1$

FIB_DYN-REC(array *fib*, in *n*)

```
if fib(n) ≥ 0 then
      return fib(n)
fib(n) = FIB_DYN-REC(fib,n-2) + FIB_DYN-REC(fib,n-1);
return fib(n)
```

FIB_DYN-ITER(array *fib*, in *n*)

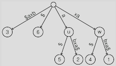
```
for i = 2, ..., n do
      fib(i) = fib(i − 2) + fib(i − 1)
return fib(n)
```



Heuristic Alg (Exmpl: Coin Change)

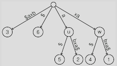
GREEDY_COINCHANGE(int *change*, int $m_1, \ldots,$ int m_k (denomination))

$NumberCoins \leftarrow 0$
$U \leftarrow change$
for t=k,...,1 **do**
    $x_t \leftarrow \lfloor U/m_t \rfloor$
    $U \leftarrow U - x_t m_t$
    $NumberCoins \leftarrow NumberCoins + x_t$
return $x_1, \ldots, x_k, NumberCoins$



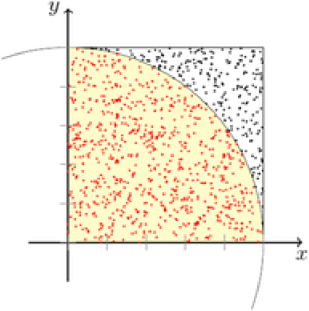
Probablistic Alg (Blackboard)

- Las Vegas (always produces the correct result or it informs about the failure)
- Monte Carlo (may be incorrect with a certain (typically small) probability)



Probablistic Alg (Blackboard)

- Las Vegas (always produces the correct result or it informs about the failure)
- Monte Carlo (may be incorrect with a certain (typically small) probability)



source: wikipedia