Introduction to Logic and Computer Science
Foundation of Computer Science
Logic in Computer Science

Lecture #13: algorithms

Tamás Mihálydeák, László Aszalós

This work was supported by the construction EFOP-3.4.3-16-2016-00021.
The project was supported by the European Union, co-financed by the European Social Fund.

Problem solving

Problem solving begins with articulating the problem precisely and clearly. From the practical problem, we make a model, that is, we get rid of the less important details, focusing only on the essence. The resulting abstract model can usually be solved by known means.
Successive steps in solving a problem are defined by a specific set of instructions. (See constructing regular expression based automaton of previous lecture)
Many different tasks of the same type can be solved with different starting values.

Concept of algorithm

As there is no commonly accepted algorithm concept, no specific definition is given. However, we can approach it with the following wording:
A clear instruction system that defines the content and order of operations that leads to the desired result from the appropriate initial data.

- An operation is a conversion that uses the current values of the data to produce their new values
  - eg. increasing the value of a variable, sorting data
- The sequence control mechanisms describe:
  - sequence (sequential execution)
  - selection (selection, `if-then-else`)
  - iteration (repetition, loop: `while`, `for`)

Example - Euclidean algorithm

Find the greatest common divisor of two numbers.

- Basic idea: *gcd* does not change if the larger number is replaced by the difference of the two numbers.
- Repeat this step until the two numbers are equal
- This common number is the largest common divisor of the original account.

More formal solution

Let denote m and n the two incoming numbers (the input)

1. Let us divide m by n, let r be the remainder.
2. If $r = 0$, then the algorithm halt, the result is n.
3. Otherwise let $m \leftarrow n$, $n \leftarrow r$, and repeat from step #1

Example - Euclidean algorithm

Formal description in Python

```
def gcd(a, b):
    while b:
        a, b = b, a%b
    return a
```

Properties of an Algorithm

- **Input**: The algorithm must have input values from a specified set.
- **Output**: The algorithm must produce the output values from a specified set of input values. The output values are the solution to a problem.
- **Finiteness**: For any input, the algorithm must terminate after a finite number of steps.
- **Definiteness**: All steps of the algorithm must be precisely defined.
- **Effectiveness**: It must be possible to perform each step of the algorithm correctly and in a finite amount of time. That is, its steps must be basic enough so that, for example, someone using a pencil and a paper could carry them out exactly, and in a finite amount of time. It is not enough that each step is definite (or precisely defined), but it must also be feasible.

History

- 1928, David Hilbert - *there exists and algorithm, to decide that a formula can be derived from axioms?* (effective calculation)
- 1930, Alonzo Church - lambda calculi: formal model of calculation with function application and abstraction
- 1936, Stephen C. Kleene, recursive functions
- 1936, Alan Turing - Turing machine (tapes, head to read/write), theoretical questions
- 1951, A. A. Markov - string replacing

Markov algorithm

- Formal system, the input and the output is text, we replace strings as long as possible.
- For the sake of clear implementation, substitutions are sequenced and the first possible substitution is made.
- If the same substitution is possible in several parts of the text, only the first occurrence is replaced.
- For the sake of universality, we will introduce closing substitutions and once applied, the implementation will stop.

Markov algorithm

Definition

- The $M = \langle \Sigma, H, H' \rangle$ triple is Markov normal algorithm, where
  - Σ is an alphabet,
  - $H \subset \Sigma^\star \times \Sigma^\star$ is an ordered set of substitution rules, and
  - $H' \subseteq H$ is the set of closing substitutions.
- Based on M $y \in \Sigma^\star$ is derivable in one step from $x \in \Sigma^\star$, if there exist u, v, w and $z \in \Sigma^\star$, such that $x = uvw$, $y = uzw$, $(v, z) \in H$, and (v, z) is the first substitution rule where v is a substring of x i.e.
  if $x = svt$ where $s, t \in \Sigma^\star$, then u is a substring of s.
  In notation $x \Rightarrow y$.
- The derivation in one step is called a closing derivation, if the applied rule is a closing substitution, i.e. $(v, z) \in H'$.
- Based on M, y is derivable from x in at least one step ($x \Rightarrow^+ y$), if there exist $n \geq 1$ and $x_0, x_1, \ldots, x_n \in \Sigma^\star$, such that $x = x_0$, $y = x_n$, and $x_i \Rightarrow x_{i+1}$ if $0 \leq i < n$.
- Based on M, y is derivable from x ($x \Rightarrow^\star y$), if $x = y$ or $x \Rightarrow^\star y$.

Markov algorithm

Definition

The Markov algorithm produces the output y for the input x, if

- based on M, y is derivable from x,
- the derivations in one step are not closing derivations, and y does not contain any first string of the rules, or
- the derivations in one step are not closing derivations, except for the last one.

Markov algorithm in Python

```
def replace(text, replacements):
    while True:
        for pat, repl, term in replacements:
            if pat in text:
                text = text.replace(pat, repl, 1)
                if term:
                    return text
                break
        else:
            return text

replace("1011", [("|0", "0||", False),
                 ("1", "0|", False),
                 ("0", "", False)])
```