Theoretical computer science: Turing machines

Matthias Springer

Hasso-Plattner-Institut

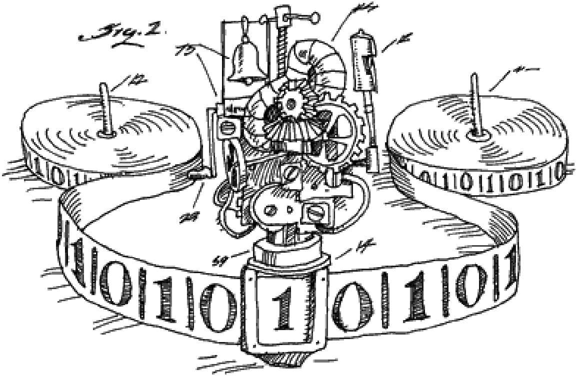
January 5, 2011

Overview

1. Turing machines
2. Universal Turing machines
3. Church-Turing thesis
4. Sorting: Insertion sort
5. Graph algorithm: Traveling salesman problem
6. Graph algorithm: Hamiltonian cycle
7. Cook's theorem
8. $\mathcal{P}$ versus $\mathcal{NP}$ problem

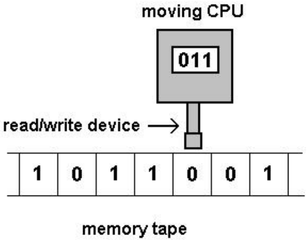
Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Turing machines [6]



Mathematical definition [4]

- Set of tape symbols Γ
- Set of states Q
- Initial state q_0
- Transition function:
  $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$



Turing machines as computation automatons

- **Input:** content of tape
- **Program:** transition function
- **Output:** content of tape after computation
- **Computation**
  1. Read tape symbol
  2. Evaluate transition function
  3. Write new tape symbol, enter new state, move reading head
  4. Go to step 1

Turing machine program for multiplying unary numbers [1]

Transition function: (q, s, q', s', d)
$(q_0, 1, q_1, 1, L), (q_1, b, q_2, *, R), (q_2, b, q_3, b, L), (q_2, *, q_2, *, R)$
$(q_2, 1, q_2, 1, R), (q_2, X, q_2, X, R), (q_2, A, q_2, A, R), (q_3, 1, q_4, b, K)$
$(q_3, X, q_4, X, L), (q_4, 1, q_4, 1, L), (q_4, X, q_5, X, L), (q_5, *, q_8, *, R)$
$(q_5, 1, q_6, A, L), (q_5, A, q_5, A, L), (q_6, b, q_7, 1, R), (q_6, *, q_6, *, L)$
and so on...

Example: 1111 x 111 $\Rightarrow$ 111111111111

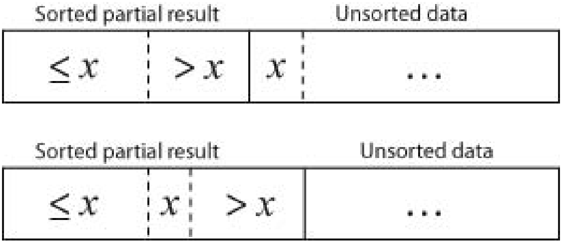
Universal Turing machines

- **Problem:** Looking for a machine which can run any program.
- **Universal Turing machine**
  - *Input:* transition function a Turing machine, input data (properly encoded) on the tape
  - *Program:* simulation of the Turing machine (quite complex, but possible!)
  - *Output:* part of the content of the tape

Church-Turing thesis

- **Problem:** We're using computers, not Turing machines!
- Effectively calculable functions = Turing calculable functions
- Turing calculable functions = calculable functions (by machines)
- Calculation complexity matches among computations models
- *Examples:* Equivalence of Turing machine, RAM, λ-calculus, most programming languages

Sorting: Insertion sort



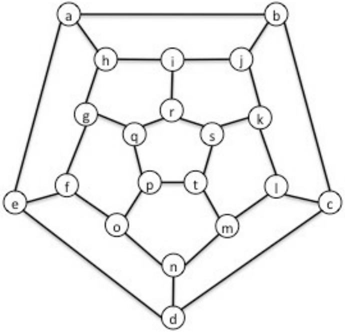
- Time necessary to insert one element: $\mathcal{O}(n)$
- Time necessary to insert n elements: $\mathcal{O}(n^2)$
  $\Rightarrow$ SORTING $\in \mathcal{P}$

Graphs: Traveling salesman problem (TSP)

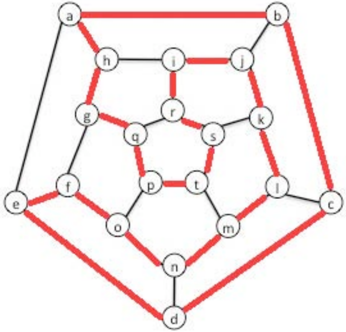
- **Problem:** Given a map of towns, is there a round trip that visits each town at least once and isn't longer than X units?
- At most $n!$ possibilites to check
- Usually need to check all these possible tours
- No better algorithm known so far

Graphs:
Hamiltonian cycle [5]

- **Problem:** Given a graph, is there a round trip that visits each vertex exactly once?
- Quite difficult to decide



Graphs:
Hamiltonian cycle [5]



- **Problem:** Given a graph, is there a round trip that visits each vertex exactly once?
- Quite difficult to decide
- Polynomial reduction to TSP
  - Vertices = Towns
  - Distance between any two vertices is 1
  - Is there round trip which visits each vertex at least once and isn't longer than *n* units?
  - TSP is at least as difficult (hard) as Hamilton

Cook's theorem

- Works only for decision problems (yes/no)
- **Problem:** Is there a most difficult problem X?
- Reduction of every computable problem to X
- *SATISFIABILITY* is such a problem X
  - Given a logical formula, is it satisfiable?
  - *Example:* $(x \vee y \vee z) \wedge (x \wedge \neg x)$
  - Turing machine can be described as a logical formula
  - Evaluation of formula $\equiv$ Output of Turing machine
- There are many such problems X ($\mathcal{NP}$-complete problems)

Cook's theorem: part of a Turing machine formula [1]

- Only one state at a time
  $\forall t, q_1 \in Q, q_2 \in Q, q_1 \neq q_2 : S(t, q_1) \Rightarrow \neg S(t, q_2)$
- Cell can only contain one symbol at a time
  $\forall t, c, s_1 \in \Gamma, s_2 \in \Gamma, s_1 \neq s2 : T(t, c, s_1) \Rightarrow \neg T(t, c, s_2)$
- Initial state is q_0
  $S(0, q_0)$
- ...
- $T(\ldots)$, $S(\ldots)$ are predicates

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

$\mathcal{P}$ versus $\mathcal{NP}$ problem

- Problems in $\mathcal{P}$ can be solved efficiently
- $\mathcal{NP}$-complete problems can't be solved efficiently so far (no algorithm found so far)
- Solve one (1!) such problem efficiently
  $\Rightarrow$ All computable problems can be solved efficiently
- **Assumption:** $\mathcal{P} \neq \mathcal{NP}$
- No proof found so far!
- Prize for valid proof: 1.000.000 USD + Turing Award (most likely)!

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Summary

- **Turing machine:** theoretical computation model used for proofs
- **Chruch-Turing thesis:** *All computers are created equal.*
- **Cook's theorem:** Solve one efficiently, solve them all efficiently!
- Polynomial reduction to proof $\mathcal{NP}$-completeness
- There are problems which most likely can never be solved efficiently! $\mathcal{P} \neq \mathcal{NP}$

Bibligraphy

1 Dewdney, A. K. (2001). *The New Turing Omnibus: 66 Excursions in Computer Science.* New York: Henry Holt.

2 Schoenig, U. (2008). *Theoretische Informatik - kurz gefasst.* 5. Auflage. Heidelberg: Spektrum A kademischer Verlag.

3 Wikipedia (2010, December 9). *Turing machine.* http://en.wikipedia.org/w/index.php?title=Turing_machine&oldid=401534074.

4 http://ideonexus.com/2009/02/05/javascript-turing-machine/

5 http://www.csee.umbc.edu/~chang/cs203.f10/homework.shtml

6 http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/coretechnologies.htm