Teoria da Computação

Apresentação da Disciplina

Mirtha Lina Fernández Venero
mirtha.lina@ufabc.edu.br
Sala 529-2, Bloco A

setembro 2017

Agenda

Apresentação

Mirtha Lina Fernández Venero

- PhD. in Software,
  Universitat Politècnica de Catalunya
  http://www.cs.upc.edu/
  Grupo de Lógica e Programação

- **Principais Experiências/Interesses de Pesquisa e Ensino** (porém não limitadas): métodos formais; verificação e dedução automatizada; linguagens de programação; compiladores

Isso tem a ver com a Teoria da Computação?

Teoria da Computação?

- **Teoria?**

Teoria da Computação?

- **Teoria?** (científica): conjunto de conhecimentos que procura explicar, com alto grau de exatidão e generalidade, fenômenos da natureza

Teoria da Computação?

- **Teoria?** (científica): conjunto de conhecimentos que procura explicar, com alto grau de exatidão e generalidade, fenômenos da natureza
- **Computação?**

Teoria da Computação?

- **Teoria?** (científica): conjunto de conhecimentos que procura explicar, com alto grau de exatidão e generalidade, fenômenos da natureza
- **Computação?** processo ou cálculo usado para resolver um **problema** usando um **algoritmo**

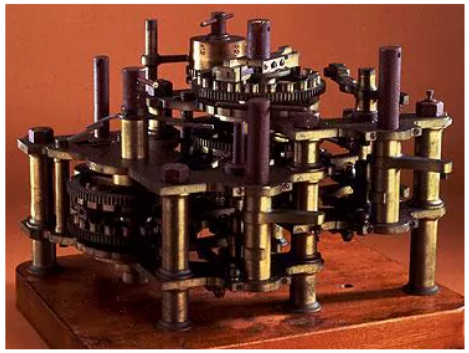
Teoria da Computação?

- **Teoria?** (científica): conjunto de conhecimentos que procura explicar, com alto grau de exatidão e generalidade, fenômenos da natureza
- **Computação?** processo ou cálculo usado para resolver um **problema** usando um **algoritmo**
- **Problema?**
- **Algoritmo?**

Teoria da Computação?

- **Teoria?** (científica): conjunto de conhecimentos que procura explicar, com alto grau de exatidão e generalidade, fenômenos da natureza
- **Computação?** processo ou cálculo usado para resolver um **problema** usando um **algoritmo**
- **Problema?** Conjunto de instâncias e suas soluções
- **Algoritmo?** conjunto finito de passos (bem definidos) que descrevem o processo de cálculo e terminam em tempo finito





Teoria da Computação?

- **Teoria?** (científica): conjunto de conhecimentos que procura explicar, com alto grau de exatidão e generalidade, fenômenos da natureza
- **Computação?** processo ou cálculo usado para resolver um **problema** usando um **algoritmo**
- **Problema?** Conjunto de instâncias e suas soluções
- **Algoritmo?** conjunto finito de passos (bem definidos) que descrevem o processo de cálculo e terminam em tempo finito

O que estuda a Teoria da Computação?

- Como definir de forma geral e com exatidão um problema?
- Como definir de forma geral e com exatidão um algoritmo?
- Todas as funções definem algoritmos? Quais funções/problemas não são computáveis? Quais são os limites da Computação?
- Quais algoritmos são os melhores para resolver um problema? Quais problemas podem ser resolvidos de forma eficiente?

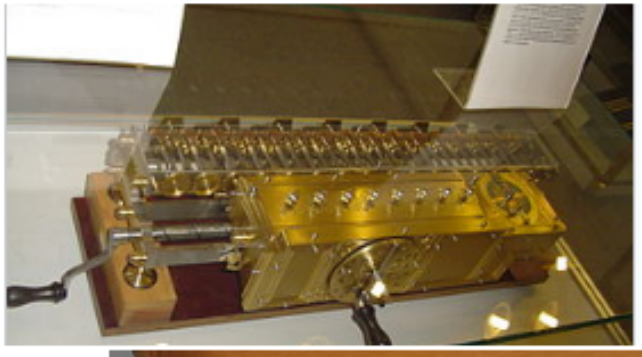
O que estuda a Teoria da Computação?

- Como definir de forma geral e com exatidão um problema?
  Linguagens Formais
- Como definir de forma geral e com exatidão um algoritmo?
  Modelos formais de Cômputo
- Todas as funções definem algoritmos? Quais funções/problemas não são computáveis? Quais são os limites da Computação?
  Computabilidade
- Quais algoritmos são os melhores para resolver um problema? Quais problemas podem ser resolvidos de forma eficiente?
  Algoritmos e Complexidade Computacional

Agenda

Precursores da Teoria da Computação

- Calculadora mecânica de **Blaise Pascal** em 1652
- Calculadora mecânica de **Gottfried Leibniz** em 1694
- Arithmometer de **Thomas de Colmar** em 1851

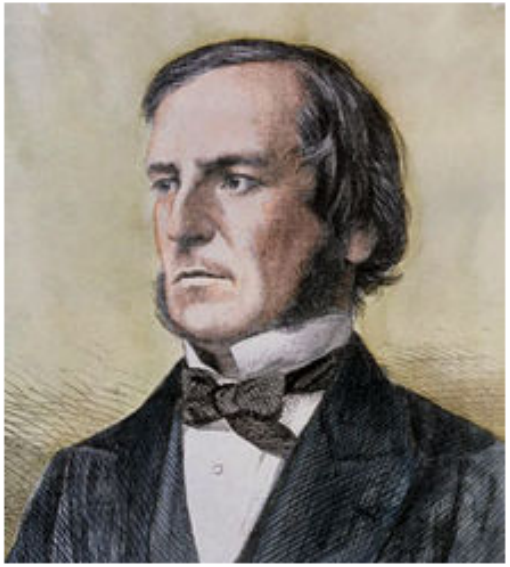


Precursores da Teoria da Computação

- **Charles Babbage** (1791-1871): primeiro computador automático (The Difference Engine-1822) e o primeiro computador programável (The Analytical Engine-1837)
- **Ada Lovelace** (1815-1852): primeira programadora da máquina de Babbage (também escreveu o manual)

Precursores da Teoria da Computação - século XIX

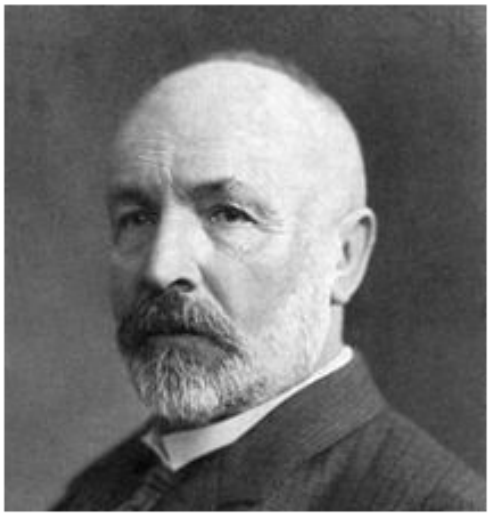
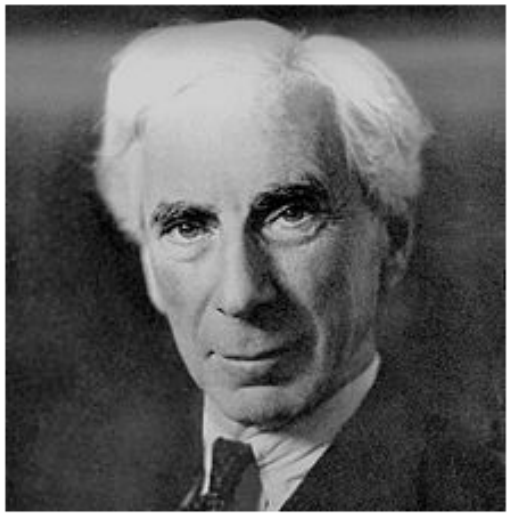
- **George Boole** (1815-1864): álgebra/lógica booleana (*The Mathematical Analysis of Logic*-1847; *An Investigation of the Laws of Thought*-1854)
- **Augustus De Morgan** (1806-1871): leis da lógica (*Formal Logic*-1847) e a álgebra relacional (*Syllabus of a Proposed System of Logic*-1860)



Precursores da Teoria da Computação - XIX, XX

Alguns dos matemáticos que contribuíram na fundação da ToC

- **Georg Cantor** (1845-1918): teoria de conjuntos, bijeções entre conjuntos, diagonalização
- **Bertrand Russell** (1872-1970): teoria de conjuntos, paradoxo Russell
- **David Hilbert** (1862-1943): lógica matemática, teoria de provas, famosa lista de 23 problemas abertos em 1902


Precursores da Teoria da Computação - XX

Kurt Gödel (1906-1978): provou a completude da lógica de predicados (1929), os teoremas da não completude e definiu formalmente a classe das funções recursivas

GIT2 (1931): *todo sistema axiomático consistente que contem a aritmética elementar é incompleto**

- Sistema consistente: no existe sentencia tal que ela **e** sua negação podem ser provadas
- Sistema completo: toda sentencia válida pode ser provada

Exemplo de ditado popular: Não existe produto bom, bonito e barato. As três características são "inconsistentes"; escolhendo duas o produto está "incompleto".

* i.e. existem sentencias verdadeiras que não podem ser provadas. Ligado ao 2do problema de Hilbert.

Agenda

Fundação da ToC - Entscheidungsproblem

- Proposto por Hilbert em 1928: achar um algoritmo que dada uma sentencia da lógica de primeira ordem responda sim ou não dependendo se a sentencia é universalmente válida ou não.
- Pelo teorema da completude de Gödel (1929), a lógica de primeira ordem é completa.

Como definir formalmente um algoritmo?

Existe algoritmo para o Entscheidungsproblem?

Fundação da ToC - Entscheidungsproblem

- Proposto por Hilbert em 1928: achar um algoritmo que dada uma sentencia da lógica de primeira ordem responda sim ou não dependendo se a sentencia é universalmente válida ou não.
- Pelo teorema da completude de Gödel (1929), a lógica de primeira ordem é completa.

Como definir formalmente um algoritmo?

Existe algoritmo para o Entscheidungsproblem?

Resposta: Não - Alonzo Church e Alan Turing

Fundadores da ToC - Alonzo Church (1903-1995)

- criador do λ-cálculo em 1930 e da versão não tipada (e consistente) em 1936 que constitui um modelo universal de computação (Turing-complete)
- provou que o problema de Entscheidungs não é decidível
- orientador de grandes da ToC como John B. Rosser, Stephen C. Kleene, Michael O. Rabin, Dana Scott, Alan Turing

Fundadores da ToC - Alan Turing (1912-1954)

- criador da primeira definição formal de algoritmo, a máquina de Turing (1936)
- provou que o problema de Entscheidungs não é decidível, mostrando que o problema da parada (halting problem) não é
- precursor da inteligência artificial

Fundadores da ToC - Tese de Church-Turing

- mostraram (junto com Kleene e Rosser) que as funções recursivas de Gödel, o λ-cálculo e a máquina de Turing são modelos de cômputo equivalentes e coincidem com a noção informal de função efetivamente calculável

Áreas da Teoria da Computação

- **Computabilidade**: foca na existência ou não dum algoritmo para resolver problemas
  - **Linguagens Formais**: as instâncias dos problemas podem ser representadas como cadeias sobre um alfabeto. Os problemas de decisão podem ser representados como linguagens sobre um alfabeto. Outros problemas (e.g. busca e otimização) não são computacionalmente mais difíceis que os problemas de decisão
  - **Modelos formais de Cômputo**
    - Computação tradicional: máquinas de Turing, λ-cálculo, funções recursivas, gramáticas, sistemas de reescrita/Post/L, máquinas de acesso aleatório, autômatos, etc
    - Computação bioinspirada
    - Computação quântica
  - **Decidabilidade**: quais problemas podem ser resolvidos usando um modelo de cômputo?

Áreas da Teoria da Computação

- **Computabilidade**: foca na existência ou não dum algoritmo para resolver problemas usando um modelo de cômputo
- **Algoritmos**: foca em resolver um problema decidível
  - dependendo do problema: busca, ordenação, grafos, otimização combinatória, etc
  - dependendo da técnica: divisão e conquista, gulosos, programação dinâmica, busca exaustiva/heurística, algoritmos probabilísticos ou de aproximação, etc

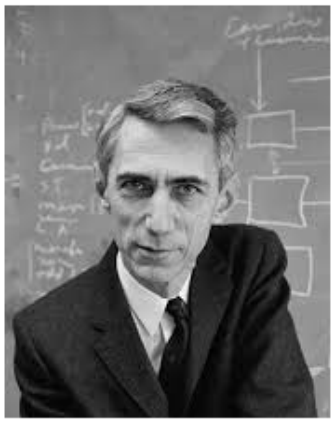
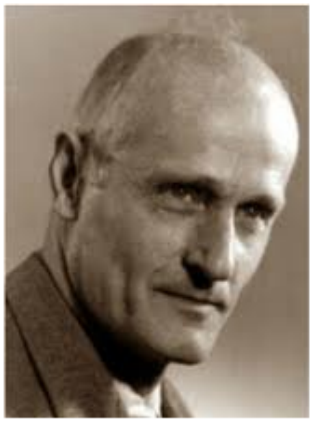
Áreas da Teoria da Computação

- **Computabilidade**: foca na existência ou não dum algoritmo para resolver problemas usando um modelo de cômputo
- **Algoritmos**: foca em resolver um problema decidível
  - dependendo do problema: busca, ordenação, grafos, otimização combinatória, etc
  - dependendo da técnica: divisão e conquista, gulosos, programação dinâmica, busca exaustiva/heurística, algoritmos probabilísticos ou de aproximação, etc
- **Complexidade Computacional**: foca na solução eficiente de problemas decidíveis; classifica os problemas e.g. dependendo da quantidade de recursos que são necessários para sua solução algorítmica
  - complexidade tempo
  - complexidade de espaço
  - Complexity Zoo, Complexity Zoology Inclusion Diagram

Contribuições à Teoria da Computação - XX

- **Emil Leon Post** (1897-1954): sistema e problema de correspondência de Post
- **John von Neumann** (1903-1957): arquitetura von Neumann de computadores, teoria de conjuntos
- **Claude Shannon** (1916-2001): circuitos digitais, teoria da informação
- **Stephen Kleene** (1909-1994): funções recursivas e ERs
- **Noam Chomsky** (1928-): linguagens formais, gramáticas




Contribuições à ToC - Prêmio Turing

"Prêmio Nobel da Computação", concedido todo ano desde 1966 pela ACM a pessoas com grandes contribuições à computação

- 1966 - **Alan J. Perlis**: linguagens de programação -LPs- e compiladores
- 1972 - **Edsger Dijkstra**: LPs (ALGOL) e algoritmos
- 1974 - **Donald Knuth**: LPs, algoritmos, "The Art of Computer Programming"
- 1976 - **Michael Rabin + Dana Scott**: autômatos
- 1977 - **John Backus**: especificação LPs (FORTRAN)
- 1978 - **Robert Floyd**: parsing, semantics, verificação, síntese
- 1980 - **C. A. R. Hoare**: LPs
- 1982 - **Stephen Cook**: complexidade
- 1984 - **Niklaus Wirth**: LPs (MODULA, PASCAL)
- 1985 - **Richard M. Karp**: algoritmos, complexidade
- 1986 - **John Hopcroft + Robert Tarjan**: algoritmos
- 1987 - **John Cocke**: compiladores
- 1991 - **Robin Milner**: LPs (ML), concorrência, semantics

http://amturing.acm.org/, https://en.wikipedia.org/wiki/Turing_Award

Contribuições à ToC - Prêmio Turing

- 1993 - **Juris Hartmanis + Richard E. Stearns**: complexidade
- 1995 - **Manuel Blum**: complexidade, criptografia, verificação
- 1996 - **Amir Pnueli**: verificação
- 2000 - **Andrew Chi-Chih Yao**: complexidade, criptografia
- 2001 - **Ole-Johan Dahl and Kristen Nygaard**: LPs OO (Simula)
- 2003 - **Alan Kay**: LPs OO (Smalltalk)
- 2005 - **Peter Naur**: LP (ALGOL), compiladores
- 2006 - **Frances E. Allen**: compiladores
- 2007 - **Edmund M. Clarke, E. Allen Emerson + Joseph Sifakis**: verificação
- 2008 - **Barbara Liskov**: LPs, computação distribuída
- 2010 - **Leslie G. Valiant**: complexidade, parsing
- 2012 - **Silvio Micali + Shafi Goldwasser**: criptografia, verificação
- 2013 - **Leslie Lamport**: computação paralela e distribuída

Curiosidades

- Entre 2007-2013 o valor do prêmio foi de $250,000; desde 2014 é de $1,000,000 financiado por Google Inc. Somente três mulheres (48 homens) e um latino-americano (Manuel Blum) dentre os que têm recebido o prêmio.

Contribuições à ToC - Outros Prêmios - SIGACT

ACM SIGACT (Association for Computing Machinery Special Interest Group on Algorithms and Computation Theory), 1968

- **Gödel Prize** since 1993 for outstanding papers in the area of theoretical computer science
- **Donald E. Knuth Prize** since 1996 for outstanding contributions to the foundations of computer science:

  2013: Gary L. Miller; 2012: Leonid Levin; 2010: David S. Johnson; 2002: Christos Papadimitriou; 2000: Jeffrey D. Ullman; 1997: Leslie G. Valiant; 1996: Andrew C.-C. Yao
- **SIGACT Distinguished Service Prize** since 1997 for substantial contributions to the Theoretical Computer Science:

  2017: Alistair Sinclair; 2008: Richard Karp; 2000: S. Rao Kosaraju; 1997: David S. Johnson

http://www.sigact.org/main.php

Contribuições à ToC - Outros Prêmios - IEEE CS

IEEE Computer Society: fundada em 1946 e focada na aplicação da computação

- **Charles Babbage Award** since 1989 for significant contributions in the field of parallel computation

  2004: Christos Papadimitriou; 2000: Michael O. Rabin; 1997: Frances Allen; 1995: Richard Karp

  `https://www.computer.org/web/awards/charles-babbage`

- **John von Neumann Medal** since 1990 for outstanding achievements in computer-related science and technology

  2016: Christos Papadimitriou; 2011: C. A. R. Hoare; 2010: John Hopcroft and Jeffrey Ullman; 2009: Susan L. Graham; 2008: Leslie Lamport; 2004: Barbara H. Liskov; 2003: Alfred V. Aho; 1995: Donald E. Knuth; 1994: John Cocke

  `http://www.ieee.org/about/awards/medals/vonneumann.html`

Relação da ToC com Outras Áreas da Computação

- **ToC** ∩ **X**: algoritmo, corretude, análise da complexidade
- **ToC** ∩ **Construção de Compiladores**: linguagens formais, gramáticas, autômatos, algoritmos scheduling, grafos, etc
- **ToC** ∩ **Engenharia de Software/Hardware**: lógica, métodos formais, teste/verificação automatizada
- **ToC** ∩ **Inteligência Artificial**: lógica, autômatos, processamento de linguagem natural, sistemas baseados no conhecimento, dedução e prova automatizada de teoremas, teoria de jogos, planning/scheduling
- **ToC** ⊂ **Theoretical Computer Science**: coding theory (data compression, cryptography, error-correction), computational algebra, computational number theory, computational geometry, computational biology, machine learning, ...

  https://en.wikipedia.org/wiki/Theoretical_computer_science

Agenda

Por que estudar **Teoria da Computação**?

Por que estudar **Teoria da Computação**?

Peter Denning, Is Computer Science Science?
CACM, v. 48, 4, pp. 27 - 31, 2005

Table 2 lists six major categories of computing principles along with examples of important discoveries that are not obvious to amateurs. By exploiting these principles, professionals are able to solve problems that amateurs would find truly baffling.

Area	Problem
Computation	• Unbounded error accumulation on finite machines • Non-computability of some important problems • Intractability of thousands of common problems • Optimal algorithms for some common problems • Production quality compilers

Table 2. Some non-obvious problems solved by computing principles

https://www.cs.mtu.edu/~john/jenning.pdf

Por que estudar **Teoria da Computação**?

Area	Problem
Computation	• Unbounded error accumulation on finite machines • Non-computability of some important problems • Intractability of thousands of common problems • Optimal algorithms for some common problems • Production quality compilers
Communication	• Lossless file compression • Lossy but high-fidelity audio and video compression • Error correction codes for high, bursty noise channels • Secure cryptographic key exchange in open networks
Interaction	• Arbitration problem • Deadlock problem • Timing-dependent (race-conditioned) bug problem • Fast algorithms for predicting throughput and response time • Internet protocols • Cryptographic authentication protocols
Recollection	• Locality • Thrashing • Search • Two-level mapping for access to shared objects
Automation	• Simulations of focused cognitive tasks • Limits on expert systems • Reverse Turing tests
Design	• Objects and information hiding • Levels • Throughput and response time prediction networks of servers

Table 2. Some non-obvious problems solved by computing principles

Objetivos da disciplina

- Apresentar conceitos fundamentais da ToC
- Desenvolver habilidades para lidar com métodos e provas formais
- Identificar e aplicar os conceitos estudados na sua pesquisa

Contribuir a enxergar a computação como algo muito além de programas e linguagens de programação

Para avançar a Ciência da Computação é preciso olhar para tudo aquilo que até hoje não é computável. M

Objetivos da disciplina

- Apresentar conceitos fundamentais da ToC
- Desenvolver habilidades para lidar com métodos e provas formais
- Identificar e aplicar os conceitos estudados na sua pesquisa

Contribuir a enxergar a computação como algo muito além de programas e linguagens de programação

Metodologia

- $\approx$ 50% de teoria
- $\approx$ 50% de exercícios, seminários e projeto

Universidade Federal do ABC

Agenda

Universidade Federal do ABC

Agenda Preliminar (sujeita a alterações)

1	1 - 18/9	Apresentação da Disciplina
	2 - 21/9	Linguagens Formais e Gramáticas
2	3 – 25/9	Autômatos Finitos e Linguagens Regulares I
	4 – 28/9	Exercícios
3	5 – 2/10	Autômatos Finitos e Linguagens Regulares II
	6 – 5/10	Exercícios
4	7 – 9/10	Seminário
	8 – 12/10	Feriado
5	9 – 16/10	Gramáticas Livres de Contexto
	10 – 19/10	Autômatos de Pilha. Linguagens Sensíveis ao Contexto
6	11 – 23/10	Exercícios
	12 – 26/10	Seminário

Universidade Federal do ABC

Agenda Preliminar (sujeita a alterações)

7	13 - 30/10	Máquinas de Turing e Computabilidade I
	14 - 2/11	Feriado
8	15 – 6/11	Máquinas de Turing e Computabilidade II
	16 – 9/11	Prova
9	17 – 13/11	Complexidade Computacional I
	18 – 16/11	Complexidade Computacional II
10	19 – 20/11	Feriado
	20 – 23/11	Exercícios
11	21 – 27/11	Seminário
	22 – 30/11	Projeto
12	23 – 4/12	Apresentação do Projeto
	24 – 7/12	Prova Substituiva/Recuperação

Relação Nota - Conceito

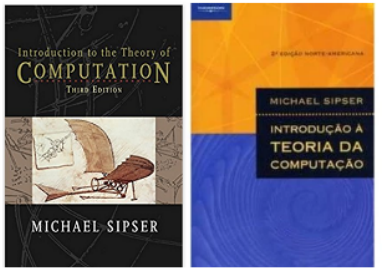
$$\text{Nota} = \text{ExercíciosParticipação} + 0.2 * \text{Seminários} + 0.3 * \text{Prova} + 0.4 * \text{Projeto}$$

- **A** $= [8.5, \infty) \Rightarrow$ excelente participação e compreensão da disciplina
- **B** $= [7.5, 8.5) \Rightarrow$ boa participação e compreensão da disciplina
- **C** $= [6, 7.5) \Rightarrow$ compreensão do conteúdo mais importante da disciplina e capacidade para seguir estudos mais avançados
- **F** $= [0, 6) \Rightarrow$ insuficiente compreensão do conteúdo. A disciplina deve ser cursada novamente.

Universidade Federal do ABC

Agenda

Bibliografia Básica



1. **M. Sipser. Introduction to the Theory of Computation, Cengage Learning 3rd ed, 2012**
2. **J. E. Hopcroft, R. Motwani and J. D. Ullman. Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 2001**
3. H. R. Lewis, C. H. Papadimitriou. Elementos de Teoria da Computação. 2a edição, Bookman Companhia Ed., 2004.
4. P. Linz, An Introduction to Formal Languages and Automata, 6th edition, Jones & Bartlett Learning, 2016
5. A. Maheshwari, M.Smid. Introduction to Theory of Computation, 2014

   http://cg.scs.carleton.ca/~michiel/TheoryOfComputation

Slides em http://professor.ufabc.edu.br/~mirtha.lina/toc.html

Some ToC journals

- Theory of Computing (open access)
- Formal Aspects of Computing
- Journal of the ACM
- SIAM Journal on Computing (SICOMP)
- SIGACT News
- Theoretical Computer Science
- Theory of Computing Systems
- International Journal of Foundations of Computer Science
- Journal of Automata, Languages and Combinatorics
- Acta Informatica
- Fundamenta Informaticae
- ACM Transactions on Computation Theory
- Computational Complexity
- Journal of Complexity
- ACM Transactions on Algorithms

Some ToC conferences

- Annual ACM Symposium on Theory of Computing (STOC)
- Annual IEEE Symposium on Foundations of Computer Science (FOCS)
- ACM/SIAM Symposium on Discrete Algorithms (SODA)
- IEEE Symposium on Logic in Computer Science (LICS)
- Computational Complexity Conference (CCC)
- International Colloquium on Automata, Languages and Programming (ICALP)
- Symposium on Theoretical Aspects of Computer Science (STACS)
- International Symposium on Fundamentals of Computation Theory (FCT)

Some Online Resources

1. **Jeffrey D. Ullman, Automata Theory, Stanford University,** http://automata.lagunita.stanford.edu/
2. Dan Gusfield, Theory of Computation, University of California, Davis (UCDavis), Fall 2011, https://www.youtube.com/playlist?list=PLslgisHe5tBM8UTCt1f66oMkpmjCblzkt
3. Somenath Biswas, Theory of Computation, National Programme on Technology Enhanced Learning (NPTEL), http://nptel.ac.in/courses/106104028/ (Youtube)
4. Gabriel Robins, Theory of Computation, University of Virginia, http://www.cs.virginia.edu/~robins/cs3102/
   Awesome slides, take a look!
5. Theory of Computing website at Cornell University https://www.cs.cornell.edu/research/theory
6. Wikipedia (why not!) for finding links to original sources