Foundations of Computer Science
Lecture 28

Efficiency: The Class \textbf{P}, \textbf{NP} and \textbf{NP}-Completeness

Running Time
Efficiently Solvable Problems
Boundary Between Efficient and Inefficient
Computer: Universal Turing Machine $U_{TM}$

Program and Input: $\langle M \rangle \#w$. $U_{TM}$ simulates $M$ on $w$.

No Program Verifier, no Ultimate-Debugger, no PCP-Solver. 😞

No means No.
The Path Forward: Focus on Decidable Problems
The Path Forward: Focus on Decidable Problems
The Path Forward: Focus on Decidable Problems

Decider
\[ U_{TM} = \text{computer} \]
\[ TM = \text{Algorithm} \]

CFG Parsing
DFA RegExp

FOCS

Theory of Computing

Discrete Math
The Path Forward: Focus on Decidable Problems

- Decider
  - $U_{TM} = \text{computer}$
  - $TM = \text{Algorithm}$

- Theory of Computing
  - CFG Parsing
  - DFA RegExp

- FOCS
  - Proof, logic
    - INDUCTION

- Discrete Math
  - Recursion
  - Struct. Induction
  - Sums, Asymptotics
  - Number theory
  - Graphs
  - Counting
  - Probability
The Path Forward: Focus on Decidable Problems

FOCS

Decider
$U_{TM} = \text{computer}$
$TM = \text{Algorithm}$

CFG
Parsing

DFA
RegExp

Graph theory
Linear Algebra
Probability Theory
Multivariate Calc.

Proof, logic
INDUCTION

Recursion
Struct. Induction

Sums, Asymptotics

Number theory

Graphs

Counting

Probability

Creator: Malik Magdon-Ismail
Efficiency: The Class P, NPand NP-Completeness: 3 / 10
The Path Forward: Focus on Decidable Problems

Decider
$U_{tm} = \text{computer}$
$TM = \text{Algorithm}$

CFG Parsing
DFA RegExp

Proof, logic
INDUCTION
Recursion
Struct. Induction
Sums, Asymptotics
Number theory
Graphs
Counting
Probability

Graph theory
Linear Algebra
Probability Theory
Multivariate Calc.

FOCS

Theory of Computing

Efficiency: The Class $P$, $NP$ and $NP$-Completeness: 3/10

Creator: Malik Magdon-Ismail

Today →
The Path Forward: Focus on Decidable Problems

Computability & Complexity

FOCS

Theory of Computing

Decider
- $U_{tm} = \text{computer}$
- $TM = \text{Algorithm}$

CFG Parsing

DFA RegExp

Graph theory
- Linear Algebra
- Probability Theory
- Multivariate Calc.

Induction

Recursion
- Struct. Induction

Sums, Asymptotics

Number theory

Graphs

Counting

Probability

Efficiency: The Class P, NP and NP-Completeness: 3 / 10

Creator: Malik Magdon-Ismail
The Path Forward: Focus on Decidable Problems
The Path Forward: Focus on Decidable Problems

Decider

\( U_{tm} = \text{computer} \)

\( TM = \text{Algorithm} \)

\( \text{CFG} \)

\( \text{Parsing} \)

\( \text{DFA} \)

\( \text{RegExp} \)

\( \text{Graph theory} \)

\( \text{Linear Algebra} \)

\( \text{Probability Theory} \)

\( \text{Multivariate Calc.} \)

\( \text{Proof, logic} \)

\( \text{INDUCTION} \)

\( \text{Recursion} \)

\( \text{Struct. Induction} \)

\( \text{Sums, Asymptotics} \)

\( \text{Number theory} \)

\( \text{Graphs} \)

\( \text{Counting} \)

\( \text{Probability} \)

\( \text{FAST (P)} \)

\( \text{Polynomial} \)

\( \text{FAST (NP)} \)

\( \text{Unbounded Parallelism} \)

\( \text{SLOW Exponential} \)

\( \text{Boolean Circuits} \)

\( \text{Chapters 28 & 29} \)

\( \text{E} \)

\( \text{UTM} = \text{computer} \)

\( \text{TM} = \text{Algorithm} \)

\( \text{CFG Parsing} \)

\( \text{DF A RegExp} \)

\( \text{Proof, logic} \)

\( \text{INDUCTION} \)

\( \text{Recursion} \)

\( \text{Struct. Induction} \)

\( \text{Sums, Asymptotics} \)

\( \text{Number theory} \)

\( \text{Graphs} \)

\( \text{Counting} \)

\( \text{Probability} \)

\( \text{FAST (P)} \)

\( \text{Polynomial} \)

\( \text{FAST (NP)} \)

\( \text{Unbounded Parallelism} \)

\( \text{SLOW Exponential} \)

\( \text{Boolean Circuits} \)

\( \text{Chapters 28 & 29} \)

\( \text{P = NP?} \)

\( \text{Computability & Complexity} \)

\( \text{Algorithms} \& \text{DS} \)

\(- \text{Approximation} \)

\(- \text{Randomized} \)

\(- \text{Distributed} \)

\( \text{Cryptography} \)

\( \text{Data} \)

\(- \text{ML/AI/DM/NLP} \)

\(- \text{Vision} \)

\(- \text{Graphics} \)

\(- \text{Comp. Finance} \)

\( \text{Networks} \)

\(- \text{Computers} \)

\(- \text{Social} \)

\(- \text{Data} \text{ (e.g. www)} \)

\( \text{Robotics} \)

\( \text{Security} \)

\( \text{Programming Languages} \)

\(- \text{Compilers} \)

\(- \text{Distributed} \)

\( \text{Program Analysis} \)

\(- \text{Testing} \)

\(- \text{Verification} \)

\( \text{DB Systems} \)

\( \text{Parallel computing} \)

\( \text{Operating systems} \)

\( \text{Architecture} \)

\( \text{Cre ator: Malik Magdon-Ismail} \)

\( \text{Efficiency: The Class P, NP and NP-Completeness: 3 / 10} \)
Time complexity: Asymptotic worst-case analysis.

The class P: Efficiently solvable problems.

Polynomial on one architecture means polynomial on pretty much any architecture.
Running Time
Time Complexity
Decidable But Non-Polynomial
...the high technology so celebrated today is essentially a mathematical technology.

“To err is human, but to really foul things up you need a computer.” – Paul Ehrlich
the high technology so celebrated today is essentially a mathematical technology.

“To err is human, but to really foul things up you need a computer.” – Paul Ehrlich

- **Mariner rocket explodes (1962).** Formula into code bug resulted in no smoothing of deviations.
  - Luckily Stanislav “funny feeling in my gut” Petrov thought: “surely they’d use more missiles?”
- **Therac 25 (1985).** Concurrent programming bug killed patients through massive 100× radiation overdose.
- **AT&T Lines Go Dead (1990).** 75 million calls dropped (one line of buggy code in software upgrade).
- **Pentium floating point long-division bug (1993).** Cost: $475 million – flawed division table.
- **Ariane rocket explosion (1996).** Cost: $500 million – overflow in 64-bit to 16-bit conversion.
- **Y2K (1999).** Cost: $500 billion spent because year was stored as 2 digits to save space.
- **Mars Climate Orbiter Crash (1998).** Cost: $125 million lost due to metric to imperial units bug.
- **Tesla Self-Driving Car (2016). 1 dead.** Auto-pilot didn’t “see” tractor-trailer.
- **Financial Disasters:** London Stock Exchange down due to single server bug (2009; billions of pounds of trading); Knight Capital computer glitch trigers stock sale (2012; 500 million lost and Knight’s value drops by 75%).
- **Airline Disasters:**
  - AirFrance 447 2009, **228 dead**: pitot-tube failure feeds inconsistent data to programs which then panic pilot.
  - Spanair 5022, 2008, **154 dead**: malware virus.
  - AdamAir 574, 2007, **102 dead**: navigation system errors (and pilot errors).
  - KoreanAir 801, 1997, **228 dead**: ground proximity warning system bug.
  - AeroPerú 603, 1996, **70 dead**: altimeter failures.
  - Scottish RAF Chinook, 1994, **29 dead**: faulty test program.
  - AirFrance 296, 1988, **3 dead**: altimeter bug.
  - IranAir 655, 1988, **290 dead**: shot down by US Aegis combat system (misidentified as attacking military plane).
  - KoreanAir 007, 1983, **269 dead**: autopilot took plane into Soviet airspace where it got shot down.
  - Boeing 737 Max, 2018,2019, **346 dead**: attack sensor + algorithm errors.
...the high technology so celebrated today is essentially a mathematical technology.

“To err is human, but to really foul things up you need a computer.” – Paul Ehrlich

- Mariner rocket explodes (1962). Formula into code bug resulted in no smoothing of deviations.
  - Luckily Stanislav “funny feeling in my gut” Petrov thought: “surely they’d use more missiles?”
- Therac 25 (1985). Concurrent programming bug killed patients through massive 100× radiation overdose.
- AT&T Lines Go Dead (1990). 75 million calls dropped (one line of buggy code in software upgrade).
- Y2K (1999). Cost: $500 billion spent because year was stored as 2 digits to save space.
- Financial Disasters: London Stock Exchange down due to single server bug (2009; billions of pounds of trading); Knight Capital computer glitch trigers stock sale (2012; 500 million lost and Knight’s value drops by 75%).
- Airline Disasters:
  - AirFrance 447 2009, 228 dead: pitot-tube failure feeds inconsistent data to programs which then panic pilot.
  - AdamAir 574, 2007, 102 dead: navigation system errors (and pilot errors).
  - Scottish RAF Chinook, 1994, 29 dead: faulty test program
  - AirFrance 296, 1988, 3 dead: altimeter bug.
  - IranAir 655, 1988, 290 dead: shot down by US Aegis combat system (misidentified as attacking military plane).
  - KoreanAir 007, 1983, 269 dead: autopilot took plane into Soviet airspace where it got shot down.
  - Boeing 737 Max, 2018,2019, 346 dead: attack sensor + algorithm errors.
- Software errors cost the U.S. $60 billion annually in rework, lost productivity and actual damages.

Put effort to make sure your program works fully correctly all the time.