Turing Machines and DTIME

Exposition by William Gasarch—U of MD

Computability

The course so far:

Computability

The course so far:

1. We defined regular via DFAs. With this definition we could prove some languages are regular and some are not.

Computability

The course so far:

1. We defined regular via DFAs. With this definition we could prove some languages are regular and some are not.
2. We defined CFL via CFGs. With this definition we could prove some languages are CFL and some are not. (We never proved that some langs are not CFL, but we could have.)

Computability

The course so far:

1. We defined regular via DFAs. With this definition we could prove some languages are regular and some are not.
2. We defined CFL via CFGs. With this definition we could prove some languages are CFL and some are not. (We never proved that some langs are not CFL, but we could have.)

We want to prove that some languages are compuable and some are not computable. We need a model of computation, the Turing Machine. The details of it are complicated, and we mostly won't be using it, but we need to know that there is a rigorous model.

WARNING: The Next Two Slides are Messy

On the next two slides we will

1. Define a Turing Machine.
2. Define a configuration of a Turing Machine and how it operates.

WARNING: The Next Two Slides are Messy

On the next two slides we will

1. Define a Turing Machine.
2. Define a configuration of a Turing Machine and how it operates.

I will use this definition only once in the entire course.

WARNING: The Next Two Slides are Messy

On the next two slides we will

1. Define a Turing Machine.
2. Define a configuration of a Turing Machine and how it operates.

I will use this definition only once in the entire course.
I will use it to prove the most important therem of the course:

(Cook-Levin) SAT is NP-complete.

WARNING: The Next Two Slides are Messy

On the next two slides we will

1. Define a Turing Machine.
2. Define a configuration of a Turing Machine and how it operates.

I will use this definition only once in the entire course.
I will use it to prove the most important therem of the course:

(Cook-Levin)SAT is NP-complete.

The definition is not hard, but it is ugly and somewhat arbitrary.
But it is needed to prove the Cook-Levin Theorem.

Turing Machines Def

Def A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

- Q is a finite set of states. It has the state h.
- Σ is a finite alphabet. It contains the symbol $\#$.
- $\delta : (Q - \{h\}) \times \Sigma \rightarrow Q \times \Sigma \cup \{R, L\}$
- $s \in Q$ is the start state, h is the halt state.

Note There are many variants of Turing Machines- more tapes, more heads. All equivalent.

Turing Machines Conventions

We use the following convention:

1. On input $x \in \Sigma^*$, $x = x_1 \cdots x_n$, the machine starts with tape

$$\# x_1 x_2 \cdots x_n \#\#\#\# \cdots$$

   that is one way infinite.
2. The head is initially looking at the x_n.
3. $\delta(q, \sigma) = (p, \tau)$: state changes $q \rightarrow p$, σ is replaced with τ.
4. $\delta(q, \sigma) = (p, L)$: state changes $q \rightarrow p$, head moves Left. ($\delta(q, \sigma) = (p, R)$ similar).
5. TM is in state h: DONE. Left most square has a 1 (0) then M ACCEPTS (REJECTS) x.

Note We can code TMs into numbers. We say Run $M_x(y)$ which means run the TM coded by x on input y.

How Powerful are Turing Machines?

1. There is a JAVA program for function f iff there is a TM that computes f.
2. Everything computable can be done by a TM.

Other Models of Computation: Computability

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

Other Models of Computation: Computability

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

They ended up all being equivalent.

Other Models of Computation: Computability

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

They ended up all being equivalent.

This is what makes computability theory work! We will almost never look at the details of a Turing Machine. To show a function is computable we just write psuedocode to compute it.

Other Models of Computation: Complexity

1. There is a JAVA program for function f iff there is a TM that computes f.
2. Everything computable can be done by a TM.

Other Models of Computation

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

Other Models of Computation

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

They ended up all being equivalent WITHIN POLY TIME.

Other Models of Computation

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

They ended up all being equivalent WITHIN POLY TIME.

This is what makes complexity theory work! We will almost never look at the details of a Turing Machine. To show a function is computable in Poly Time we just write psuedocode and show that it halts within time poly in the length of the input.

We will need this later in the course when we study P and NP.

Decidable Sets

Def A set A is DECIDABLE if there is a Turing Machine M such that

$$x \in A \rightarrow M(x) = Y$$

$$x \notin A \rightarrow M(x) = N$$

Time Classes

Def Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\le O(T(|x|))$.

Time Classes

Def Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\le O(T(|x|))$.
What do you think of this definition? Discuss.

Time Classes

Def Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\le O(T(|x|))$.
What do you think of this definition? Discuss.

Its Terrible!

Time Classes

Def Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\leq O(T(|x|))$.
What do you think of this definition? Discuss.

Its Terrible!
The definition depends on the details of the type of Turing Machine. 1-tape? 2-tapes? This should not be what we care about.

Time Classes

Def Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\le O(T(|x|))$.
What do you think of this definition? Discuss.

Its Terrible!
The definition depends on the details of the type of Turing Machine. 1-tape? 2-tapes? This should not be what we care about.

So what to do?

- Prove theorems about $\mathrm{DTIME}(T(n))$ where the model does not matter. (Time hierarchy theorem)).
- Define time classes that are model-independent (P, NP stuff)

Concrete Time Hierarchy Thm

Exposition by William Gasarch—U of MD

Concrete Time Hierarchy Thm

Def Let $A \subseteq \{0,1\}^*$. $A \in \mathrm{DTIME}(n^3)$ is there is a Java Program J such that the following hold.

1. If $x \in A$ then $J(x)$ outputs YES.
2. If $x \notin A$ then $J(x)$ outputs NO.
3. The number of steps $J(x)$ takes is $\le |x|^3$.

Concrete Time Hierarchy Thm

Def Let $A \subseteq \{0,1\}^*$. $A \in \mathrm{DTIME}(n^3)$ is there is a Java Program J such that the following hold.

1. If $x \in A$ then $J(x)$ outputs YES.
2. If $x \notin A$ then $J(x)$ outputs NO.
3. The number of steps $J(x)$ takes is $\le |x|^3$.

We will prove the following:

Concrete Time Hierarchy Thm

Def Let $A \subseteq \{0,1\}^*$. $A \in \mathrm{DTIME}(n^3)$ is there is a Java Program J such that the following hold.

1. If $x \in A$ then $J(x)$ outputs YES.
2. If $x \notin A$ then $J(x)$ outputs NO.
3. The number of steps $J(x)$ takes is $\leq |x|^3$.

We will prove the following:

Thm There exists a set of strings A such that

Concrete Time Hierarchy Thm

Def Let $A \subseteq \{0,1\}^*$. $A \in \mathrm{DTIME}(n^3)$ is there is a Java Program J such that the following hold.

1. If $x \in A$ then $J(x)$ outputs YES.
2. If $x \notin A$ then $J(x)$ outputs NO.
3. The number of steps $J(x)$ takes is $\le |x|^3$.

We will prove the following:

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.

Concrete Time Hierarchy Thm

Def Let $A \subseteq \{0,1\}^*$. $A \in \mathrm{DTIME}(n^3)$ is there is a Java Program J such that the following hold.

1. If $x \in A$ then $J(x)$ outputs YES.
2. If $x \notin A$ then $J(x)$ outputs NO.
3. The number of steps $J(x)$ takes is $\le |x|^3$.

We will prove the following:

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

ASCII Table

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Coding Symbols Into 7-bit Strings

The ASCII table maps symbols into decimal numbers between 0 and 127. We include leading 0's.

Coding Symbols Into 7-bit Strings

The ASCII table maps symbols into decimal numbers between 0 and 127. We include leading 0's.

- Space maps to 032.

Coding Symbols Into 7-bit Strings

The ASCII table maps symbols into decimal numbers between 0 and 127. We include leading 0's.

- Space maps to 032.
- !, ", #, $, %, &, ', (,), *, +, ',' -, ., / map to 033,...,047.

Coding Symbols Into 7-bit Strings

The ASCII table maps symbols into decimal numbers between 0 and 127. We include leading 0's.

- Space maps to 032.
- !, ", #, $, %, &, ', (,), *, +, ',' -, ., / map to 033,. . .,047.
- 0,. . . ,9 code to 048,. . ., 057.

Coding Symbols Into 7-bit Strings

The ASCII table maps symbols into decimal numbers between 0 and 127. We include leading 0's.

- Space maps to 032.
- !, ", #, $, %, &, ', (,), *, +, ',' -, ., / map to 033,...,047.
- 0,...,9 code to 048,..., 057.
- :, ;, $<$, $=$, $>$, ?, code to 058 to 064.

Coding Symbols Into 7-bit Strings

The ASCII table maps symbols into decimal numbers between 0 and 127. We include leading 0’s.

- Space maps to 032.
- !, ”, #, $, %, &, ’, (,), *, +, ‘,’ -, ., / map to 033,. . .,047.
- 0,. . . ,9 code to 048,. . ., 057.
- :, ;, <, =, >, ?, code to 058 to 064.
- A,. . . ,Z code to 065,. . .,090.

Coding Symbols Into 7-bit Strings

The ASCII table maps symbols into decimal numbers between 0 and 127. We include leading 0's.

- Space maps to 032.
- !, ", #, $, %, &, ', (,), *, +, ',' -, ., / map to 033,...,047.
- 0,...,9 code to 048,..., 057.
- :, ;, <, =, >, ?, code to 058 to 064.
- A,...,Z code to 065,...,090.
- a,...,z code to 097,...,122.

Coding Symbols Into 7-bit Strings

The ASCII table maps symbols into decimal numbers between 0 and 127. We include leading 0's.

- Space maps to 032.
- !, ”, #, $, %, &, ’, (,), *, +, ‘,’ -, ., / map to 033,...,047.
- 0,...,9 code to 048,..., 057.
- :, ;, <, =, >, ?, code to 058 to 064.
- A,...,Z code to 065,...,090.
- a,...,z code to 097,...,122.
- I won't bother with the rest. See table.

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Each symbol maps to 3-digits. Concatenate them.

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Each symbol maps to 3-digits. Concatenate them.

$x = x + 12$

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Each symbol maps to 3-digits. Concatenate them.

$x = x + 12$
x maps to 120.

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Each symbol maps to 3-digits. Concatenate them.

$x = x + 12$
x maps to 120.
$=$ maps to 061

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Each symbol maps to 3-digits. Concatenate them.

$x = x + 12$
x maps to 120.
$=$ maps to 061
$+$ maps to 043

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Each symbol maps to 3-digits. Concatenate them.

$x = x + 12$

x maps to 120.

$=$ maps to 061

$+$ maps to 043

1 maps to 049

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Each symbol maps to 3-digits. Concatenate them.

$x = x + 12$
x maps to 120.
$=$ maps to 061
$+$ maps to 043
1 maps to 049
2 maps to 050

Mapping Java Program to $\mathbb{N}$

Let J be a Java Program. It is a sequence of symbols.

Each symbol maps to 3-digits. Concatenate them.

$x = x + 12$
x maps to 120.
$=$ maps to 061
$+$ maps to 043
1 maps to 049
2 maps to 050

So this piece of code maps to 120,061,120,043,049,050

Mapping $\mathbb{N}$ to Java Programs

We assume that, given a sequence of symbols, can tell if it's a Java Program.

Mapping $\mathbb{N}$ to Java Programs

We assume that, given a sequence of symbols, can tell if it's a Java Program.

We want to map $\mathbb{N}$ to Java Programs.

Mapping $\mathbb{N}$ to Java Programs

We assume that, given a sequence of symbols, can tell if it's a Java Program.

We want to map $\mathbb{N}$ to Java Programs.

Let $\downarrow$ be the program that, on any input, halts and outputs YES.

Mapping $\mathbb{N}$ to Java Programs

We assume that, given a sequence of symbols, can tell if it's a Java Program.

We want to map $\mathbb{N}$ to Java Programs.
Let $\downarrow$ be the program that, on any input, halts and outputs YES.

1. Input(i).

Mapping $\mathbb{N}$ to Java Programs

We assume that, given a sequence of symbols, can tell if it's a Java Program.

We want to map $\mathbb{N}$ to Java Programs.
Let $\downarrow$ be the program that, on any input, halts and outputs YES.

1. Input(i).
2. If numb of digits $\not\equiv 0 \pmod 3$, add 0's to left until is.

Mapping $\mathbb{N}$ to Java Programs

We assume that, given a sequence of symbols, can tell if it's a Java Program.

We want to map $\mathbb{N}$ to Java Programs.
Let $\downarrow$ be the program that, on any input, halts and outputs YES.

1. Input(i).
2. If numb of digits $\not\equiv 0 \pmod 3$, add 0's to left until is.
3. i now maps so a sequence of symbols J.

Mapping $\mathbb{N}$ to Java Programs

We assume that, given a sequence of symbols, can tell if it's a Java Program.

We want to map $\mathbb{N}$ to Java Programs.
Let $\downarrow$ be the program that, on any input, halts and outputs YES.

1. Input(i).
2. If numb of digits $\not\equiv 0 \pmod 3$, add 0's to left until is.
3. i now maps so a sequence of symbols J.
4. If J IS NOT a valid Java Program then map i to $\downarrow$.

Mapping $\mathbb{N}$ to Java Programs

We assume that, given a sequence of symbols, can tell if it's a Java Program.

We want to map $\mathbb{N}$ to Java Programs.
Let $\downarrow$ be the program that, on any input, halts and outputs YES.

1. Input(i).
2. If numb of digits $\not\equiv 0 \pmod 3$, add 0's to left until is.
3. i now maps so a sequence of symbols J.
4. If J IS NOT a valid Java Program then map i to $\downarrow$.
5. If J IS a valid Java Program then map i to J.

The Sequences of All Java Programs

Let J_i be the Java program that i maps to. So

The Sequences of All Java Programs

Let J_i be the Java program that i maps to. So

$$J_1, J_2, \ldots, \ldots \text{ is the list of all Java Programs.}$$

The Sequences of All n^3-time Java Programs

We only want to look at programs that take $\leq n^3$ times.

The Sequences of All n^3-time Java Programs

We only want to look at programs that take $\leq n^3$ times.
Let J'_i be the program that does the following:

The Sequences of All n^3-time Java Programs

We only want to look at programs that take $\leq n^3$ times.

Let J'_i be the program that does the following:

1. Input(x). $|x| = n$.

The Sequences of All n^3-time Java Programs

We only want to look at programs that take $\leq n^3$ times.
Let J_i' be the program that does the following:

1. Input(x). $|x| = n$.
2. Run $J_i(x)$ but keep track of the number of steps.

The Sequences of All n^3-time Java Programs

We only want to look at programs that take $\le n^3$ times.
Let J'_i be the program that does the following:

1. Input(x). $|x| = n$.
2. Run $J_i(x)$ but keep track of the number of steps.
3. If the program has taken $\ge n^3$ steps and has not halted yet then output NO and halt.

The Sequences of All n^3-time Java Programs

We only want to look at programs that take $\leq n^3$ times.
Let J'_i be the program that does the following:

1. Input(x). $|x| = n$.
2. Run $J_i(x)$ but keep track of the number of steps.
3. If the program has taken $\geq n^3$ steps and has not halted yet then output NO and halt.

$$J'_1, J'_2, \ldots, \ldots \text{ is the list of all } n^3\text{-time Java Programs .}$$

Upshot If $A \in \mathrm{DTIME}(n^3)$ then there exists i such that J'_i recognizes A.

The Time Hierarchy Thm

Thm There exists a set of strings A such that

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

Proof Let A be decided by the following program

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

Proof Let A be decided by the following program

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

Proof Let A be decided by the following program

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J_n'(0^n)$.

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

Proof Let A be decided by the following program

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J_n'(0^n)$.
3. If result is YES then output NO and stop.

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

Proof Let A be decided by the following program

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J'_n(0^n)$.
3. If result is YES then output NO and stop.
4. If result is NO then output YES and stop.

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

Proof Let A be decided by the following program

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J'_n(0^n)$.
3. If result is YES then output NO and stop.
4. If result is NO then output YES and stop.

1) This is a program decides A by definition.

The Time Hierarchy Thm

Thm There exists a set of strings A such that

1. There is a Java Program J that, on input x, will output YES if $x \in A$, and will output NO if $x \notin A$.
2. $A \notin \mathrm{DTIME}(n^3)$.

Proof Let A be decided by the following program

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J_n'(0^n)$.
3. If result is YES then output NO and stop.
4. If result is NO then output YES and stop.

1) This is a program decides A by definition.
2) Proof that $A \notin \mathrm{DTIME}(n^3)$ on next slide.

$A \notin \mathrm{DTIME}(n^3)$

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J'_n(0^n)$.
3. If result is YES then output NO and stop.
4. If result is NO then output YES and stop.

Let $A(0^n)$ be YES if $0^n \in A$ and NO if $0^n \notin A$.

$A \notin \mathrm{DTIME}(n^3)$

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J'_n(0^n)$.
3. If result is YES then output NO and stop.
4. If result is NO then output YES and stop.

Let $A(0^n)$ be YES if $0^n \in A$ and NO if $0^n \notin A$.

J'_1 cannot recognize A: $J'_1(0^1)$ and $A(0^1)$ DIFFER.

$A \notin \mathrm{DTIME}(n^3)$

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J'_n(0^n)$.
3. If result is YES then output NO and stop.
4. If result is NO then output YES and stop.

Let $A(0^n)$ be YES if $0^n \in A$ and NO if $0^n \notin A$.

J'_1 cannot recognize A: $J'_1(0^1)$ and $A(0^1)$ DIFFER.

J'_2 cannot recognize A: $J'_2(0^2)$ and $A(0^2)$ DIFFER.

$\vdots$

J'_n cannot recognize A: $J'_n(0^n)$ and $A(0^n)$ DIFFER.

$A \notin \mathrm{DTIME}(n^3)$

1. Input(x). If $x \notin 0^*$ output NO and stop. Otherwise $x = 0^n$.
2. Run $J'_n(0^n)$.
3. If result is YES then output NO and stop.
4. If result is NO then output YES and stop.

Let $A(0^n)$ be YES if $0^n \in A$ and NO if $0^n \notin A$.

J'_1 cannot recognize A: $J'_1(0^1)$ and $A(0^1)$ DIFFER.

J'_2 cannot recognize A: $J'_2(0^2)$ and $A(0^2)$ DIFFER.

$\vdots$

J'_n cannot recognize A: $J'_n(0^n)$ and $A(0^n)$ DIFFER.

So NO J'_n recognizes A. Hence $A \notin \mathrm{DTIME}(n^3)$.

How Complicated is A? Do I Care?

We have

How Complicated is A? Do I Care?

We have

1. A is decidable.

How Complicated is A? Do I Care?

We have

1. A is decidable.
2. A is NOT in $\mathrm{DTIME}(n^3)$ for Java.

How Complicated is A? Do I Care?

We have

1. A is decidable.
2. A is NOT in $\mathrm{DTIME}(n^3)$ for Java.
3. If you prefer Python or C or MATLAB or ... you can do this same proof and get A is NOT in $\mathrm{DTIME}(n^3)$ on those devices.

How Complicated is A? Do I Care?

We have

1. A is decidable.
2. A is NOT in $\mathrm{DTIME}(n^3)$ for Java.
3. If you prefer Python or C or MATLAB or ... you can do this same proof and get A is NOT in $\mathrm{DTIME}(n^3)$ on those devices.
4. So how complicated is A?

How Complicated is A? Do I Care?

We have

1. A is decidable.
2. A is NOT in $\mathrm{DTIME}(n^3)$ for Java.
3. If you prefer Python or C or MATLAB or ... you can do this same proof and get A is NOT in $\mathrm{DTIME}(n^3)$ on those devices.
4. So how complicated is A?

1. If use 2-tape Turing Machines then A is in $\mathrm{DTIME}(n^3 \log n)$.

How Complicated is A? Do I Care?

We have

1. A is decidable.
2. A is NOT in $\mathrm{DTIME}(n^3)$ for Java.
3. If you prefer Python or C or MATLAB or ... you can do this same proof and get A is NOT in $\mathrm{DTIME}(n^3)$ on those devices.
4. So how complicated is A?

1. If use 2-tape Turing Machines then A is in $\mathrm{DTIME}(n^3 \log n)$.
2. If use 1-tape Turing Machines then A is in $\mathrm{DTIME}(n^4)$.

How Complicated is A? Do I Care?

We have

1. A is decidable.
2. A is NOT in $\mathrm{DTIME}(n^3)$ for Java.
3. If you prefer Python or C or MATLAB or . . . you can do this same proof and get A is NOT in $\mathrm{DTIME}(n^3)$ on those devices.
4. So how complicated is A?

1. If use 2-tape Turing Machines then A is in $\mathrm{DTIME}(n^3 \log n)$.
2. If use 1-tape Turing Machines then A is in $\mathrm{DTIME}(n^4)$.
3. I do not care about models of comp on this level.

Bill and Eric Discuss the Proof

Eric and I had the following conversation before he proofread the slides.

Bill and Eric Discuss the Proof

Eric and I had the following conversation before he proofread the slides.
BILL: Eric, if I told you that there was a set A that was decidable but could not be decided in $O(n^3)$ time, would you care.

Bill and Eric Discuss the Proof

Eric and I had the following conversation before he proofread the slides.

BILL: Eric, if I told you that there was a set A that was decidable but could not be decided in $O(n^3)$ time, would you care.

ERIC: Yes, unless

Bill and Eric Discuss the Proof

Eric and I had the following conversation before he proofread the slides.
BILL: Eric, if I told you that there was a set A that was decidable but could not be decided in $O(n^3)$ time, would you care.

ERIC: Yes, unless

BILL: (cutting him off) Great! Then the class will also find it interesting. Oh, I cut you off, sorry, finish your thought.

Bill and Eric Discuss the Proof

Eric and I had the following conversation before he proofread the slides.

BILL: Eric, if I told you that there was a set A that was decidable but could not be decided in $O(n^3)$ time, would you care.

ERIC: Yes, unless

BILL: (cutting him off) Great! Then the class will also find it interesting. Oh, I cut you off, sorry, finish your thought.

ERIC: I would find the result interesting unless the set A was a contrived set that you constructed for the sole point of being decidable but not in $O(n^3)$ time.

Bill and Eric Discuss the Proof

Eric and I had the following conversation before he proofread the slides.

BILL: Eric, if I told you that there was a set A that was decidable but could not be decided in $O(n^3)$ time, would you care.

ERIC: Yes, unless

BILL: (cutting him off) Great! Then the class will also find it interesting. Oh, I cut you off, sorry, finish your thought.

ERIC: I would find the result interesting unless the set A was a contrived set that you constructed for the sole point of being decidable but not in $O(n^3)$ time.

BILL: You nailed it! The set A is not natural. But even though A is not natural, it is interesting that there is such a set.

Bill and Eric Discuss the Proof

Eric and I had the following conversation before he proofread the slides.
BILL: Eric, if I told you that there was a set A that was decidable but could not be decided in $O(n^3)$ time, would you care.

ERIC: Yes, unless

BILL: (cutting him off) Great! Then the class will also find it interesting. Oh, I cut you off, sorry, finish your thought.

ERIC: I would find the result interesting unless the set A was a contrived set that you constructed for the sole point of being decidable but not in $O(n^3)$ time.

BILL: You nailed it! The set A is not natural. But even though A is not natural, it is interesting that there is such a set.
ERIC: Only an academic.

Why is the Time Hierarchy Thm Interesting?

The set A that I constructed is contrived.

Why is the Time Hierarchy Thm Interesting?

The set A that I constructed is contrived.

Do you find the theorem that there is a decidable A that is not in $\mathrm{DTIME}(n^3)$ interesting? (You can replace n^3 by any computable function.)
Vote YES or NO.

Why is the Time Hierarchy Thm Interesting?

The set A that I constructed is contrived.

Do you find the theorem that there is a decidable A that is not in $\mathrm{DTIME}(n^3)$ interesting? (You can replace n^3 by any computable function.)
Vote YES or NO.

Reasons I find it intersting.

Why is the Time Hierarchy Thm Interesting?

The set A that I constructed is contrived.

Do you find the theorem that there is a decidable A that is not in $\mathrm{DTIME}(n^3)$ interesting? (You can replace n^3 by any computable function.)
Vote YES or NO.

Reasons I find it intersting.

1. Good to know that there are sets in EXP that are not in P.

Why is the Time Hierarchy Thm Interesting?

The set A that I constructed is contrived.

Do you find the theorem that there is a decidable A that is not in $\mathrm{DTIME}(n^3)$ interesting? (You can replace n^3 by any computable function.)
Vote YES or NO.

Reasons I find it intersting.

1. Good to know that there are sets in EXP that are not in P.
2. The fact that there are unnatural sets in $\mathrm{EXP} - \mathrm{P}$ has been used to show that a natural set is in $\mathrm{EXP} - \mathrm{P}$.

Why is the Time Hierarchy Thm Interesting?

The set A that I constructed is contrived.

Do you find the theorem that there is a decidable A that is not in $\mathrm{DTIME}(n^3)$ interesting? (You can replace n^3 by any computable function.)
Vote YES or NO.

Reasons I find it intersting.

1. Good to know that there are sets in EXP that are not in P.
2. The fact that there are unnatural sets in $\mathrm{EXP} - \mathrm{P}$ has been used to show that a natural set is in $\mathrm{EXP} - \mathrm{P}$.
   The set of winnning board positions in CHESS
   Is in $\mathrm{EXP} - \mathrm{P}$.

General Time Hierarchy Thm

Thm (The Time Hierarchy Thm) For all computable increasing $T(n)$ there exists a decidable set A such that $A \notin \mathrm{DTIME}(T(n))$.
Proof Let $M_1, M_2, \ldots,$ represent all of $\mathrm{DTIME}(T(n))$ (obtain by listing out all Turing Machines and putting a time bound on them). Here is our algorithm for A. It will be a subset of 0^*.

1. Input 0^i.
2. Run $M_i(0^i)$. If the results is 1 then output 0. If the results is 0 then output 1.

For all i, M_i and A DIFFER on 0^i. Hence A is not decided by any M_i. So $A \notin \mathrm{DTIME}(T(n))$.
End of Proof

Full Time Hierarchy Thm (I don't care!)

The Time Hierarchy Thm is usually stated as follows:
Thm (The Time Hierarchy Thm) For all computable increasing $T(n)$ there exists a decidable set A such that
$A \in \mathrm{DTIME}(T(n)\log(T(n)) - \notin \mathrm{DTIME}(T(n))$.

Full Time Hierarchy Thm (I don't care!)

The Time Hierarchy Thm is usually stated as follows:
Thm (The Time Hierarchy Thm) For all computable increasing $T(n)$ there exists a decidable set A such that
$A \in \mathrm{DTIME}(T(n)\log(T(n)) - \notin \mathrm{DTIME}(T(n))$.
The proof I did of our Time Hierarchy Thm can be done more carefully and you will see that $A \in \mathrm{DTIME}(T(n)\log(T(n))$. But this involves specifying the model more carefully.
I DO NOT CARE!

Full Time Hierarchy Thm (I don't care!)

The Time Hierarchy Thm is usually stated as follows:

Thm (The Time Hierarchy Thm) For all computable increasing $T(n)$ there exists a decidable set A such that $A \in \mathrm{DTIME}(T(n)\log(T(n)) - \notin \mathrm{DTIME}(T(n))$.

The proof I did of our Time Hierarchy Thm can be done more carefully and you will see that $A \in \mathrm{DTIME}(T(n)\log(T(n))$. But this involves specifying the model more carefully.

I DO NOT CARE!

But good to know so that you know SOME seperations: $\mathrm{P} \subset \mathrm{EXP}$.

P and EXP

Def

1. $\mathrm{P} = \mathrm{DTIME}(n^{O(1)})$.
2. $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$.