Announcements:

- Don’t forget: ps3 due on Friday

- Project presentations start next week!

See schedule, and check when yours is!

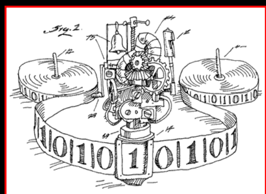
- Course Evaluations

How Fine-Grained Improvements Can Imply Circuit Complexity Lower Bounds

Ryan Williams

Circuit Complexity:
A Crash Course

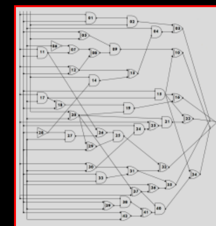
Algorithms



Can take in **arbitrarily long inputs** and still solve the problem

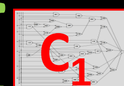
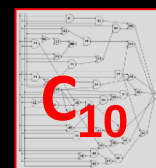
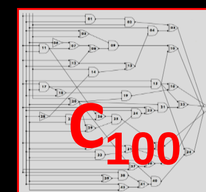
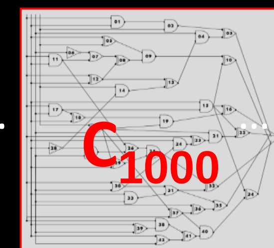
$f : \{0,1\}^* \to \{0,1\}$

Circuits



Can only take in fixed-length inputs

$g : \{0,1\}^n \to \{0,1\}$

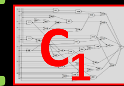
Circuit Family = {  C_1, …  C_{10}, …  C_{100}, …  C_{1000}, … }

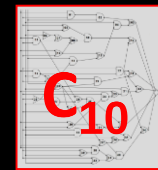
For each n, have a circuit C_n to be run on all inputs of length n

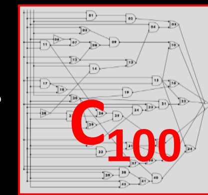
> Circuit model has "programs with ***infinite-length descriptions***"

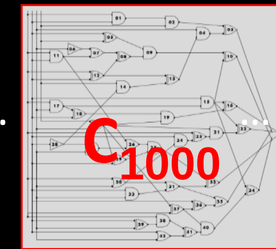
P/poly = { $f : \{0,1\}^* \to \{0,1\}$ computable by a **circuit family** $\{C_n\}$
where for every n, the ***size*** **of C_n is at most poly(n)** }

Each circuit is "small" relative to its number of inputs

Circuit Family = { 

, … 

, … 

, … 

, … }

P/poly = { $f : \{0,1\}^* \to \{0,1\}$ computable by a **circuit family** **{C_n}** where for every n, the ***size* of C_n is at most poly(n)** }

Conjecture: NP $\not\subset$ P/poly

Why study this model?

Proving limitations on what circuit families can compute is a step towards a ***non-asymptotic complexity theory*:**

Concrete limitations on computing within the known universe
"Any computer solving most instances of this 10^4-bit problem needs at least 10^{125} bits to be described"

Universe stores < 10^{125} bits **[Bekenstein '70s]** **[Meyer-Stockmeyer '70s]**

Functions with High Circuit Complexity

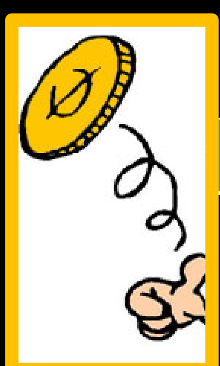
"Most" functions require *huge* circuits!

Theorem [Shannon '49, Lupanov '58]

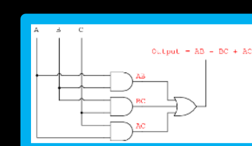
With high probability,
a randomly chosen function $\boldsymbol{f} : \{0,1\}^n \to \{0,1\}$

does not have circuits of size less than $\mathbf{2^n/n}$

(and: every $\boldsymbol{f}$ has a circuit of size about $\mathbf{2^n/n}$)



0	1	0	1	1	0	1	0	0	1



Which "natural" functions exhibit this exponential behavior?

Circuits and Derandomization

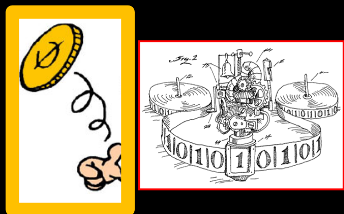
Thm [Nisan-Wigderson, Impagliazzo-Wigderson 90s]

If there is a $f : \{0,1\}^* \to \{0,1\}$
computable in $2^{O(n)}$ **time**
that does not have circuits of size at most $2^{n/100}$

Then Randomized Time $\equiv$ Deterministic Time

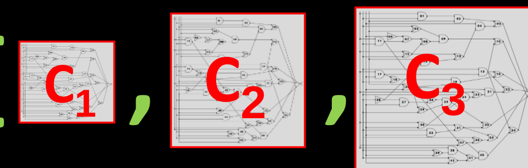
Rough intuition:
If f "looks random" to all circuits,
then f can be used to replace true randomness
in any computation!

Algorithms vs Circuit Families



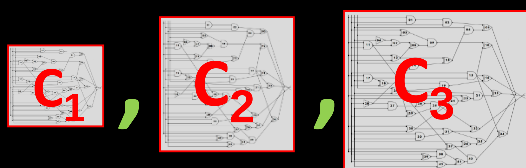
T(n) *time*
on inputs of length n

BPP is in P/poly



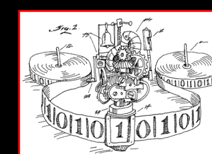
$\{C_1, C_2, C_3, \dots\}$

C_n has ≈ T(n) *size*



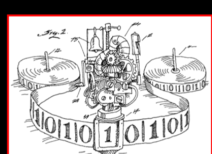
$\{C_1, C_2, C_3, \dots\}$

There is a family where every C_n has ≈ *size n*



No algorithm at all!

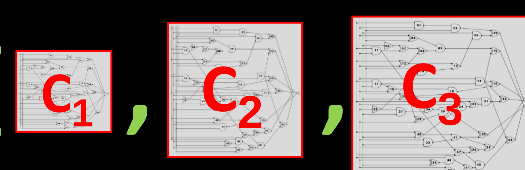
Some undecidable problems are in P/poly



EXPONENTIAL TIME
(2^n *steps*)

????

EXP is in P/poly is open!



$\{C_1, C_2, C_3, \dots\}$

every C_n has ≈ n^2 *size !!*

Conjecture: NP $\not\subset$ P/poly

Here endeth the Crash Course

Two Difficult Areas of Research

Fine-Grained Improvements for Solving NP Problems

Given: **Verifier V(x, y)** which reads w(|x|) bits of witness y, runs in t(|x|) time.

Find: **Deterministic or Randomized Algorithm** which:

1. Runs in ***less than*** **$2^{w(|x|)}$ t(|x|)** time
2. Given any input x, finds a witness y such that V(x,y) accepts (or conclude none)

Circuit Complexity (Non-Uniform Algorithms)

Given: Any **NP** problem Π (or **NEXP** problem!)

Find: Sequence of algorithms **$\{A_n\}$** such that for some k:

1. $|A_n| \leq n^k + k$
2. On all inputs x of length **n**, **$A_n(x)$** correctly solves Π on x in **$O(n^k)$** time.

Prove that no such sequences of algorithms exist for Π

One Seems Easier Than The Other...

Fine-Grained Improvements for Solving NP Problems

Given: **Verifier V(x, y)** which reads w(|x|) bits of witness y, runs in t(|x|) time.

Find: **Deterministic or Randomized Algorithm** which:

1. Runs in ***less than*** **$2^{w(|x|)}$ t(|x|)** time
2. Given any input x, finds a witness y such that V(x,y) accepts (or conclude none)

Circuit Complexity (Non-Uniform Algorithms)

Given: Any **NP** problem Π
(or **NEXP** problem!)

Find: Sequence of algorithms **$\{A_n\}$** such that for some k:

1. **$|A_n| \leq n^k + k$**
2. On all inputs x of length **n**, **$A_n(x)$** correctly solves Π on x in **$O(n^k)$** time.

Prove that no such sequences of algorithms exist for Π

One Seems Easier Than The Other...

Fine-Grained Improvements

- **3-SAT:** $O(1.308^n)$ time
- **k-SAT:** $O(2^{n - n/k})$
- **Hamiltonian Path:** $O(1.66^n)$
- **Vertex Cover:** $O(1.2^n)$

  on degree-3 graphs: $O(1.09^n)$
- **Max-2-SAT:** $O(1.8^n)$
- **3-Coloring:** $O(1.33^n)$
- **k-Coloring:** $O(2^n)$

Circuit Complexity

Given: Any **NP** problem Π
(or **NEXP** problem!)

Find: Sequence of algorithms
$\{A_n\}$ such that for some k:

1. $|A_n| \leq n^k + k$
2. On all inputs x of length **n**, $A_n(x)$ correctly solves Π on x in $O(n^k)$ time.

Prove that no such sequences of algorithms exist for Π

One Seems Easier Than The Other...

Fine-Grained Improvements

- 3-SAT: $O(1.308^n)$ time
- k-SAT: $O(2^{n - n/k})$
- Hamiltonian Path: $O(1.66^n)$
- Vertex Cover: $O(1.2^n)$

  on degree-3 graphs: $O(1.09^n)$
- Max-2-SAT: $O(1.8^n)$
- 3-Coloring: $O(1.33^n)$
- k-Coloring: $O(2^n)$

Circuit Complexity

- For all these algorithms on the LHS, we don't know how to get non-uniform algorithms (circuits) that are any better
- Best lower bound known: There is a function in NP that requires circuits of size $5n + o(n)$
- Cannot yet rule out that **NEXP** is in P/poly...

Faster Algorithms $\Longrightarrow$ Lower Bounds

Faster "Algorithms for Circuits"

[R.W. '10,'11]

***Deterministic* algorithms for:**

- **Circuit SAT** in $O(2^n/n^{10})$ time with n inputs and n^k gates
- **Formula SAT** in $O(2^n/n^{10})$
- $\mathcal{C}$**-SAT** in $O(2^n/n^{10})$
- Given a circuit of n^k size that's either ***UNSAT***, or has $\geq$ ***2^{n-1} SAT assignments***, determine which in $O(2^n/n^{10})$ time

(Easily solved w/ randomness!)

No "Circuits for NEXP"

Would imply:

- **NEXP $\not\subset$ P/poly**
- **NEXP $\not\subset$ Poly-size Formulas**
- **NEXP $\not\subset$ *poly-size* $\mathcal{C}$**

NEXP $\not\subset$ P/poly

Even Faster $\Longrightarrow$ "Easier" Functions

Better "Algorithms for Circuits"
[Murray-W. '18]

Det. algorithm for some $\epsilon > 0$:

- **Circuit SAT** in $O(2^{n-n^{\epsilon}})$ time with n inputs and $2^{n^{\epsilon}}$ gates
- **Formula SAT** in $O(2^{n-n^{\epsilon}})$
- $\boldsymbol{C}$**-SAT** in $O(2^{n-n^{\epsilon}})$
- Given a circuit of $2^{n^{\epsilon}}$ size that's either ***UNSAT***, or has $\geq$ ***2^{n-1} SAT assignments***, determine which in $O(2^{n-n^{\epsilon}})$ time

(Easily solved w/ randomness!)

No "Circuits for Quasi-NP"

Would imply:

- **NTIME[$n^{polylog\, n}$] $\not\subset$ P/poly**
- **NTIME[$n^{polylog\, n}$] $\not\subset$ NC1**
- **NTIME[$n^{polylog\, n}$] $\not\subset$ $\boldsymbol{C}$**

NTIME[$n^{polylog\, n}$] $\not\subset$ P/poly

Even Faster $\Longrightarrow$ "Easier" Functions

Fine-Grained SAT Algorithms

[Murray-W. '18]

Det. algorithm for some $\epsilon > 0$:

- **Circuit SAT** in $O(2^{(1-\epsilon)n})$ time on n inputs and $2^{\epsilon n}$ gates
- **Formula SAT** in $O(2^{(1-\epsilon)n})$
- $\boldsymbol{C}$**-SAT** in $O(2^{(1-\epsilon)n})$
- Given a circuit of $2^{\epsilon n}$ size that's either ***UNSAT***, or has $\geq$ ***2^{n-1} SAT assignments***, determine which in $O(2^{(1-\epsilon)n})$ time

(Easily solved w/ randomness!)

No "Circuits for NP"

Would imply:

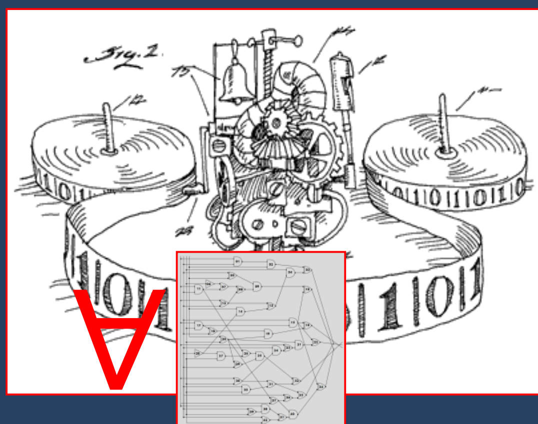
- **NP $\not\subset$ SIZE(n^k) for all k**
- **NP $\not\subset$ Formula-SIZE(n^k)**
- **NP $\not\subset$ C-SIZE(n^k) for all k**

NP $\not\subset$ SIZE(n^k) for all k

Why on Earth would it be true?

SAT? YES/NO

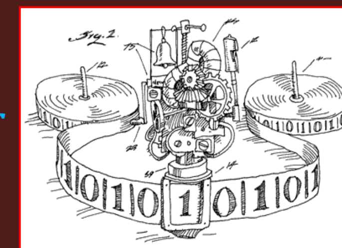
$\exists$



$\forall$

"Non-Trivial" Circuit Analysis Algorithm

$\exists$ "interesting" f



$\forall$

Circuit Lower Bounds

Concrete Lower Bounds From Algs!

Thm [R.W.'11]: NEXP $\not\subset$ ACC^0

Thm [Murray-W'18]: NTIME[$n^{poly(\log n)}$] $\not\subset$ ACC^0

NEXP = NTIME[$2^{n^{O(1)}}$]

ACC^0: ~~polynomial size, constant depth circuits with AND, OR, and MOD[m] gates for some constant m.~~

A simple but Annoying Circuit Class to prove lower bounds for (proposed in 1986)

How It Was Proved

Let $\mathbb{C}$ be a "typical" circuit class (like ACC^0)

Thm A [W'11]:

If for all k, $\mathbb{C}$-SAT on n^k-size is in $O(2^n/n^k)$ time, then NEXP does not have poly-size $\mathbb{C}$-circuits.

Thm B [W'11]:

$\exists\ \varepsilon$, ACC^0-SAT on $2^{n^{\varepsilon}}$ size is in $O(2^{n-n^{\varepsilon}})$ time.

An inefficiency!

Theorem B gives a much stronger algorithm than is needed in Theorem A.

This is exactly the starting point of [Murray-W'18]…

More on Theorem A

Let $\mathbb{C}$ be some circuit class (like ACC^0)

Thm A [W'11]:

If for all k, $\mathbb{C}$-SAT on n^k-size is in $O(2^n/n^k)$ time,
then NEXP does not have poly-size $\mathbb{C}$-circuits.

Idea. **Show that if we assume both:**

(1) NEXP has poly-size $\mathbb{C}$-circuits,

and

(2) a faster $\mathbb{C}$-SAT algorithm

Then we can show NTIME$[2^n]$ $\subseteq$ NTIME$[o(2^n)]$

Proof Sketch of Theorem A

Idea. Assume

(1) NEXP has poly-size $\mathbb{C}$-circuits, and

(2) a faster $\mathbb{C}$-SAT algorithm

Show that NTIME$[2^n]$ $\subseteq$ NTIME$[\mathrm{o}(2^n)]$

Take any problem L in **nondeterministic 2^n time**.
Given an input x, we "compute" L on x by:

1. Guessing some witness y of O(2^n) length.
2. Checking y is a witness for x in O(2^n) time.

Proof Sketch of Theorem A

Idea. **Assume**

(1) NEXP has poly-size $\mathbb{C}$-circuits, and

(2) a faster $\mathbb{C}$-SAT algorithm

Show that NTIME[2^n] $\subseteq$ NTIME[o(2^n)]

Take any problem L in **nondeterministic 2^n time**.
Given an input x, we **will** "compute" L on x by:

1. Guessing some witness y of o(2^n) length.
2. Checking y is a witness for x in o(2^n) time.

Proof Sketch of Theorem A

Idea. Assume

(1) NEXP has poly-size $\mathbb{C}$-circuits, and

(2) a faster $\mathbb{C}$-SAT algorithm

Show that NTIME[2^n] $\subseteq$ NTIME[o(2^n)]

Take any problem L in **nondeterministic 2^n time**.
Given an input x, we **will** "compute" L on x by:

1. Guessing some witness y of o(2^n) length.
2. Checking y is a witness for x in o(2^n) time.

Proof Sketch of Theorem A

Idea. Assume

(1) NEXP has poly-size $\mathbb{C}$-circuits, and

(2) a faster $\mathbb{C}$-SAT algorithm

Show that NTIME$[2^n]$ $\subseteq$ NTIME$[$o$(2^n)]$

Take any problem L in **nondeterministic 2^n time**.
Given an input x, we **will** "compute" L on x by:

1. Guessing some witness y of o(2^n) length.
2. Checking y is a witness for x in o(2^n) time.

Guessing Short Witnesses

1. Guess a witness y of o(2^n) length.

Easy Witness Lemma [IKW'02]:
If NEXP has polynomial-size circuits,
then all NEXP problems have "easy witnesses"

Def. An NEXP problem L has easy witnesses if
∀ Verifiers V for L and $\forall x \in L$,
∃ poly($|x|$)-size circuit D_x such that V(x,y) accepts,
where **y = Truth Table of circuit D_x**.

1'. Guess poly($|x|$)-size circuit D_x

Verifying Short Witnesses

2. Check y is a witness for x in $o(2^n)$ time.

Assuming NEXP has polynomial-size circuits,
"easy witnesses" exist for ***every*** verifier V.
We choose a V for an NEXP-complete L so that

Checking a witness for x

$$\equiv$$

Solving a $\mathbb{C}$-UNSAT instance with $poly(|x|)$ size
and $n = |x| + O(\log|x|)$ inputs

Then, $2^n/n^k$ time for $\mathbb{C}$-UNSAT → $o\left(2^{|x|}\right)$ time

Verifying Short Witnesses

2. Check y is a witness for x in $o(2^n)$ time.

Assuming NEXP has polynomial-size circuits,
"easy witnesses" exist for ***every*** verifier V.
We choose a V for an NEXP-complete L so that

Checking a witness for x

$\equiv$

Distinguishing ***unsatisfiable*** circuits from those with ***many*** satisfying assignments

(PCP Theorem!)

Proof Sketch of Theorem A

Idea. Assume

(1) NEXP has poly-size $\mathbb{C}$-circuits, and

(2) a faster $\mathbb{C}$-SAT algorithm

Show that NTIME$[2^n]$ $\subseteq$ NTIME$[\mathrm{o}(2^n)]$

Take any problem L in **nondeterministic 2^n time**.
Given an input x, we **will** "compute" L on x by:

1. **Guessing a circuit D_x of poly(|x|) size**
2. **Checking D_x encodes a witness for x in o(2^n) time**

End