CPSC 536F Jan 13, 2022

Goal next 1 to 2 classes:

(1) describe current open problems in Boolean formula size complexity

(2) introduce some sample formula problems that use

- probabilistic method
- spectral methods to count # trees with n leaves

Last time: we defined a formula,

e.g.

$$S = \neg\left((\neg x_1 \wedge x_2) \vee (\neg x_1) \right)$$

on:

here $n = 2, 3, \ldots$

variables: $x_1, \ldots, x_n$

allowed $\left\{ \begin{matrix} \text{operations} \\ \underline{\text{gates}} \end{matrix} \right\}$:

$\neg$ (neg)

$\wedge$ (AND, conjunction)

$\vee$ (OR)

A formula with only $\neg, \wedge, \vee$ gates has a "normal form" where we push the $\neg$ to the leaves de Morgan laws:

$$\neg(p \wedge q) = (\neg p) \vee (\neg q)$$

$$\neg(p \vee q) = (\neg p) \wedge (\neg q)$$

$$S = \neg\big(((\neg x_1) \wedge (x_2)) \vee (\neg x_1)\big)$$

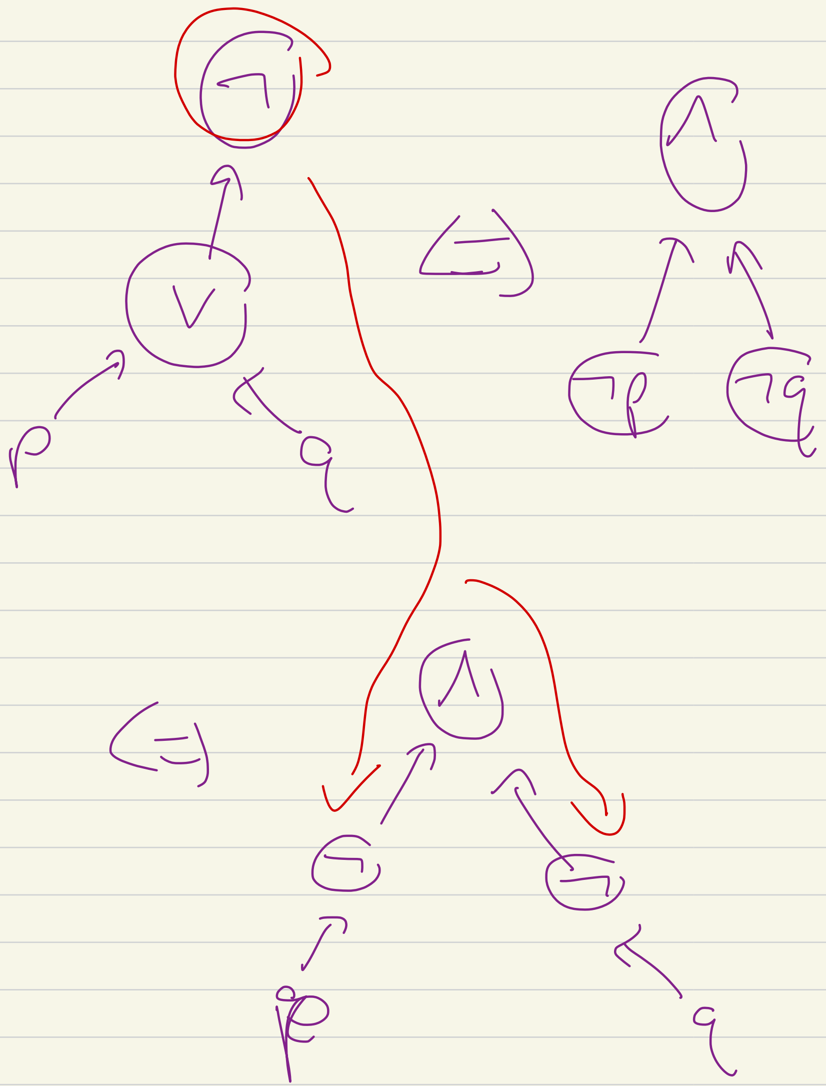
$\longleftrightarrow$ tree

root

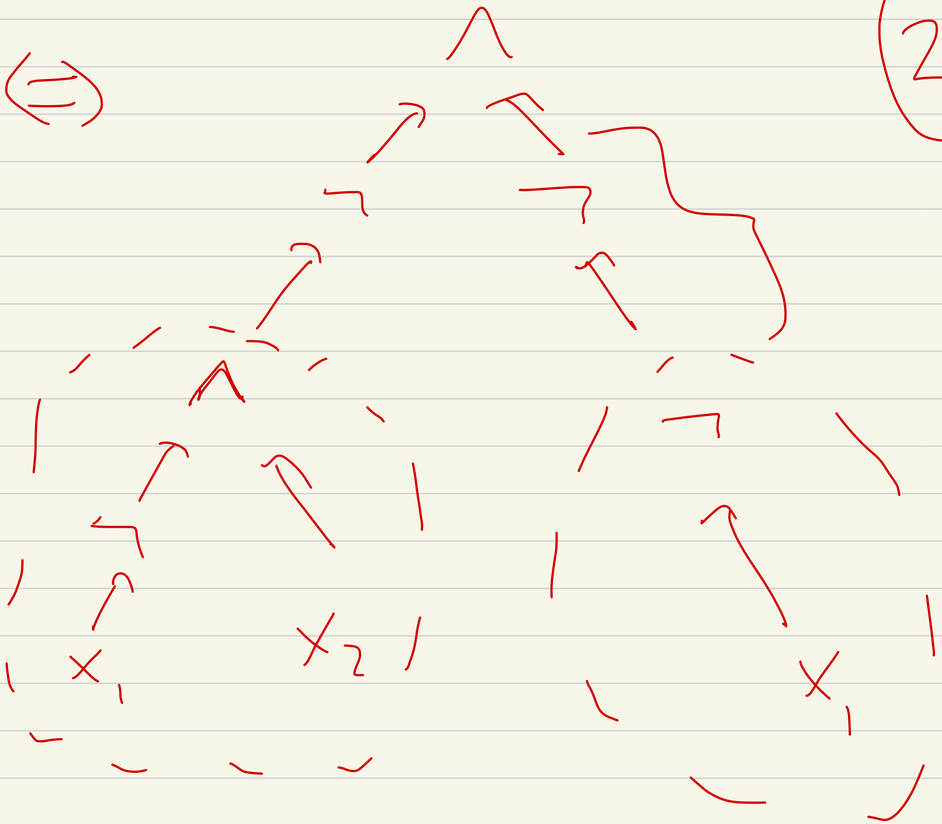
Original Tree

leaves →

So

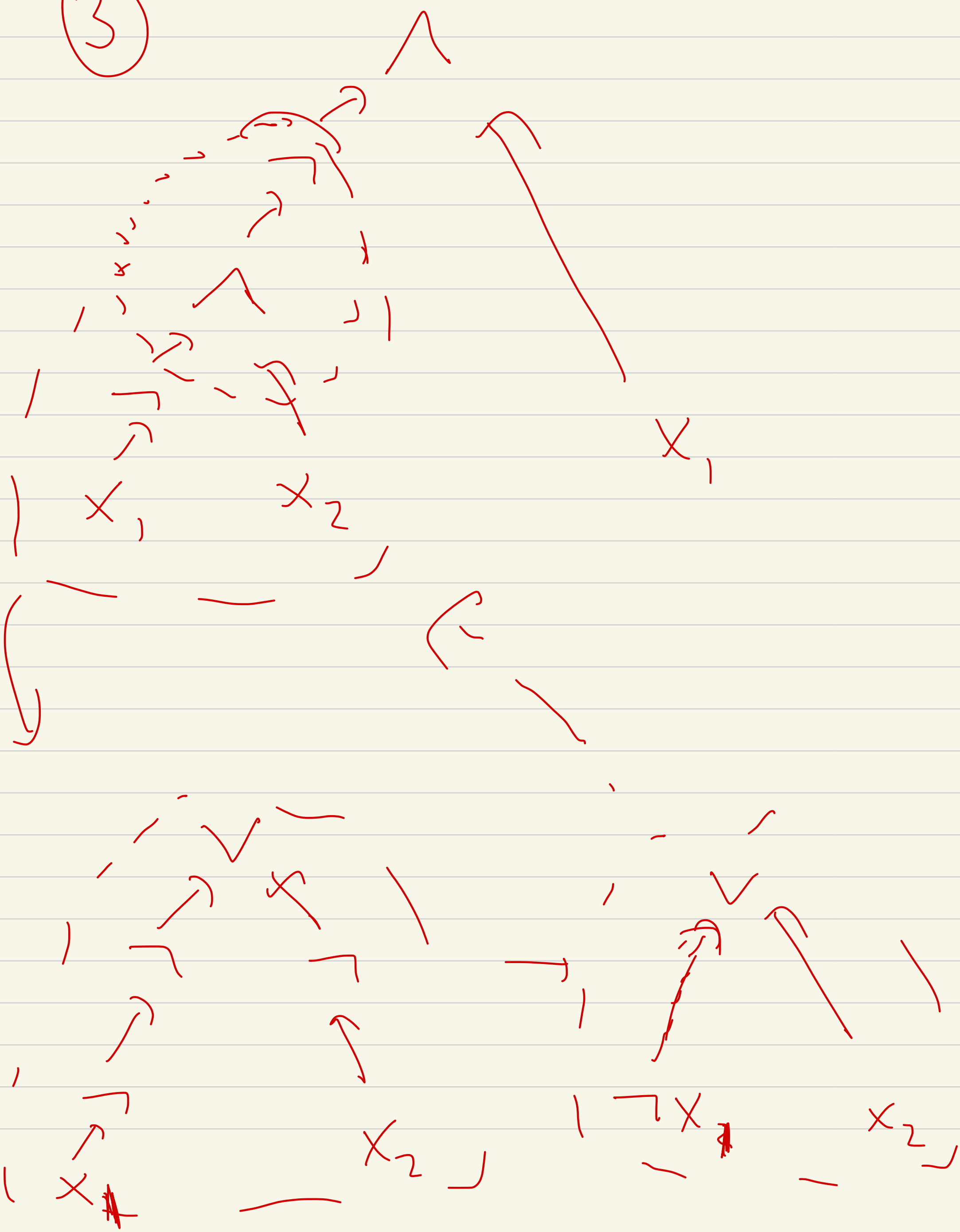


$\Leftarrow$ (2)

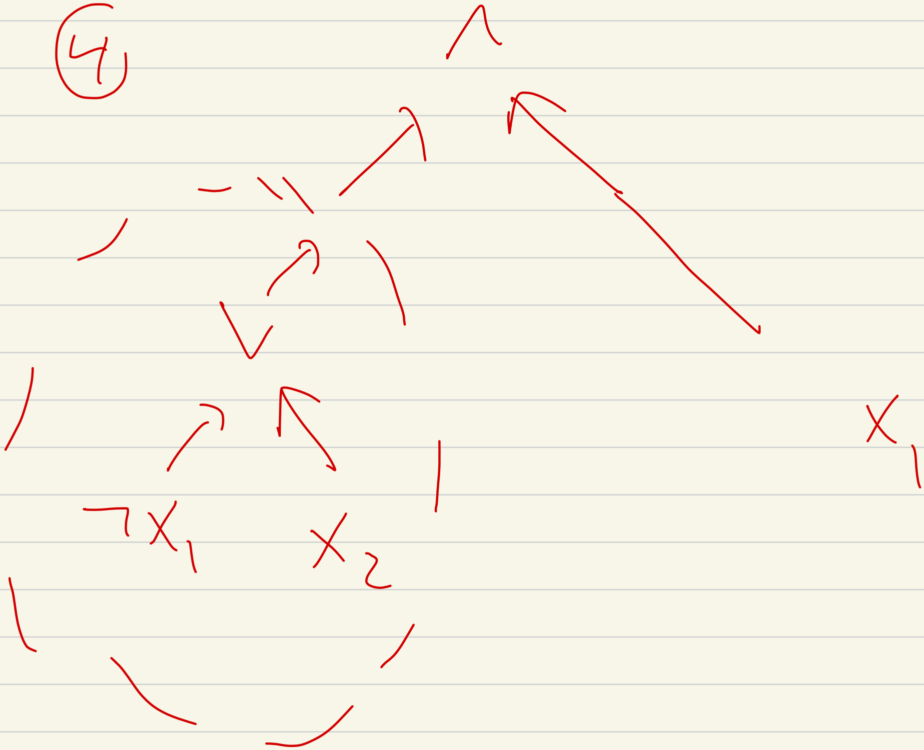


$\neg \neg p \iff p$

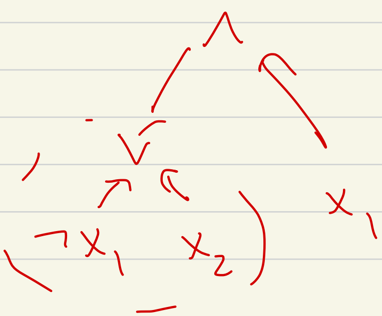
3



4



Hence original formula/tree is equivalent to



$(\neg x_1 \vee x_2) \wedge x_1$

Remark:

$\neg$ really a function on one boolean variable

$$x_1 \mapsto \neg x_1$$

x_1	$\neg x_1$
F	T
T	F

or

x_1	$\neg x_1$
0	1
1	0

often switch

F $\leftrightarrow$ 0

T $\leftrightarrow$ 1

Remark: There are 4 functions on 1 Boolean variable.

Fix finite set, S,

\# functions from $S \to \{0,1\}$

is $2^{|S|}$ $\quad |S| =$ size of S

also written $\#S$

A Boolean function on n variables

(sometimes an n-variate Boolean function) is just a function

$f = f(x_1, \ldots, x_n)$,

$\{F, T\}^n \to \{F, T\}$

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

You can specify any Boolean function

on n variables by its

"truth table" e.g.

$n=2$

X_1	X_2	$X_1 \wedge X_2$ AND	$X_1 \vee X_2$ OR	$X_1 \oplus X_2$ XOR	$\neg(X_1 \wedge X_2)$ NAND
F	F	F	F	F	T
T	F	F	T	T	T
F	T	F	T	T	T
T	T	T	T	F	F

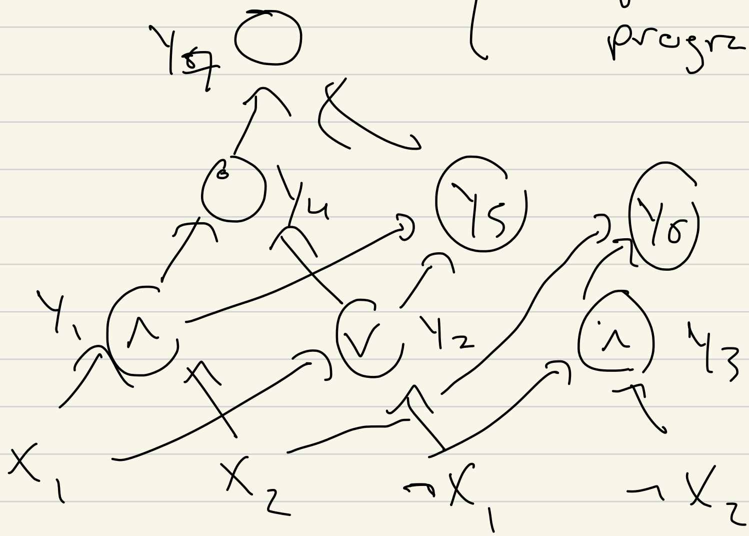
4 possible settings $X_1, X_2 \to \{F, T\}$

total # functions $2^{(2^n)}$ = for $n=2$ $2^4 = 16$

A more general notion of "an algorithm" is a $\left\{ \begin{array}{l} \text{circuit} \\ \text{straight line program} \end{array} \right\}$

directed graph

outdegree of a gate can be >1



$$y_1 = x_1 \wedge x_2 \qquad ---$$

$$y_2 = x_1 \vee \neg x_1$$

$$y_3 = \neg x_1 \wedge \neg x_2$$

Called $\left\{\begin{array}{l}\text{"sequential" or} \\ \text{"straight line"}\end{array}\right\}$ program

Start with $\underbrace{x_1, x_2, \ldots x_n}_{\text{variables}}$

add $\neg x_1, \ldots, \neg x_n$

AND, OR

$y_1 = x_{i_1} \circ x_{i_2}$

$\vdots$

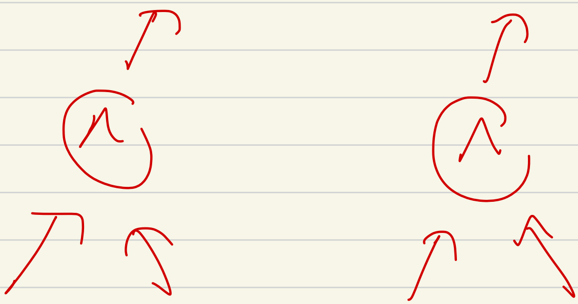
$y_i = $ some literal or y_j $j<i$ $\left\{\begin{array}{l}\text{AND} \\ \text{OR}\end{array}\right\}$ some literal or $y_k, k<i$

Formula: a tree, where

variables: $x_1, \dots, x_n$

literal: $x_1, \dots, x_n$, $\neg x_1, \neg x_2, \dots, \neg x_n$

formula $\leftrightarrow$ tree, interior nodes are gates, each either



outdegree is 1

<u>Circuit</u>: Same, but outdegree arbitrary

A practical question before 1970's (before P vs NP) was given a task, e.g.

- add 2 m-bit numbers
- multiply " " "

want to compute

$b_1 \; b_2 \; \cdots \; b_m \; b_1' \; \cdots \; b_m'$

$(b_1 \cdots, b_m) + (b_1' \cdots b_m'),$

$(\quad " \quad) \cdot (\quad " \quad)$

etc.

P vs. NP can be stated in terms of circuits and specifically what is the

shortest circuit

" straight line program

min size circuit

to compute certain functions?

We will discuss min formula size needed to compute certain functions, 5 minute break!

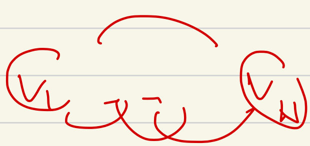
10:17 – 10:22

P vs NP as a problem in circuit complexity

Source:

CPSC 421/501 textbook by M. Sipser, [Sip], Chapter 9

Take 3COLOUR on a graph with N vertizes



each $1 \le i < j \le N$ either

$$\left\{ \begin{array}{l} \text{there is an edge } i - j \quad \text{OR} \\ \text{there isn't} \end{array} \right\}$$

A graph with N vertices vertices is described by $\binom{N}{2} = n$ Boolean variables $x_{12}, x_{13}, \ldots, x_{n-1,n}$

$$v_i \;\underset{T}{\overline{\qquad\qquad}}\; v_j$$

$$x_{ij} = F, T$$

Any prop of a graph on N vertices, becomes a function on $n = \binom{N}{2}$ Boolean variables

So there is a function:

$$\text{3COLOUR}(x_{12}, x_{13}, \ldots, x_{n-1\,n})$$

$$= \begin{cases} T & \text{if graph described by } x_{ij} \\ & \text{has a legal 3-colouring} \\ F & \text{if not} \end{cases}$$

e.g.,

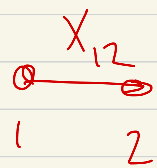
3COLOUR on 10 vertices is really a function

$$\{F,T\}^{\binom{10}{2}} \to \{F,T\}$$

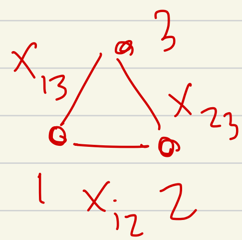
Conj: Consider $L = L(n)$ to be the

smallest size circuit that computes

3COLOUR on $n = \binom{N}{2}$ (for some $N = 1, 2, \ldots$) variables.

3COLOUR(

), $N = 2$, $n = \binom{2}{2} = 1$

3COLOUR(

), $N = 3$

$n = \binom{3}{2}$, vars x_{12}, x_{23}, x_{13}

... $N = 4$, $n = \binom{4}{2} = 6$ vars $x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{34}$

⋮

This gives function

$$3COLOUR : \{F,T\}^{\binom{N}{2}} \to \{F,T\}$$

So

$$3COLOUR : \{F,T\}^{n} \to \{F,T\}$$

for those n of the form $n = \binom{N}{2}$

$n = 1, 3, 6, 10, \ldots$

Conjucture: For any fixed $k = 1, 2, 3, \ldots$, $L(n) \geq n^k$

for n sufficiently large

I.e. $L(n)$ grows faster than any polynomial

It's a bit subtle:

$P = NP$ iff $L(n)$ grows polynomially

AND (you can build poly(n) sized circuits to compute $3COLOUR_n$ in a "uniform way")

$\Longrightarrow$

We will mention a paper of Razborov that proves this for CLIQUE functions but for

Monotone formulas.

We'll mention a paper of

Håstad (Haastad, Hastad)

on the shrinkage exponent

This will give a function —

Andreev's function — that can

computed with poly(n) size circuits

whose min formula size is

$\geq n^3 / (\text{slowly growing function})$ really $n^3/(\log n)^k$

k fixed

Relatively
Easy observations:

<u>Thm 1</u>: For sufficiently large n, the majority of the $2^{(2^n)}$ Boolean functions on n variables require a formula of size $\geq 2^n/3$ to be described.

<u>Thm 2</u>: Any Boolean function

$f: \{0,1\}^n \to \{0,1\}$, $f: \{F,T\}^n \to \{F,T\}$

Can be described by a "deMorgan formula"

(i.e. trees with $x_1, \dots, x_n, \neg x_1, \dots, \neg x_n$ as leaves, and $\wedge, \vee$ as gates)

of size $\leq n 2^n$.

Why is Theorem 2 true?

Idea: use the truth table of the function.

Thm 1 is a more difficult calculation.

Thm 2 can be improved to

$$(2^n n)/2 \ldots$$

=

Class Ends

=