LECT 03

Note Title

<u>TODAY</u>

- Non-Uniform Computation
  ($P/_{poly}$, Circuits, Branching Programs, Formulae)
- $P/_{poly} \equiv$ Poly-size Circuits.
- Some easy counting based bounds
- Neciporuk's lower bound on formula size.

<u>Notation</u> for $L \subseteq \{0,1\}^*$

let $L_n = L \cap \{0,1\}^n$

<u>Non-Uniform</u> $\equiv$ Not uniform w.r.t. input length

i.e. diff. alg. A_n for L_n (for n)

Various Models

Advice Turing Machines: Consider two input turing machine M with advice string sequence $\bar{a} = (a_1, a_2, a_3, \ldots a_n, \ldots\ldots)$

Language accepted by M with advice sequence $\bar{a}$ is

$$L(M, \bar{a}) = \{x \mid M(x, a_{|x|}) \text{ accepts}\}$$

Resources: In addition to the usual ones, count $\ell(n) = |a_n|$.

if $M, \bar{a}$ runs in time $t(n)$ on first input of length n then $L(M, \bar{a}) \in \mathrm{DTIME}(t(n))/\ell(n)$

$$P/_{poly} = \bigcup_{\text{poly } p} DTIME(p(n)) \Big/ p(n)$$

Important class.

Similarly can talk about $NP/_{poly}$, $L/_{poly}$ ....

Circuits (over basis of gates $g_1 \ldots g_k$)

(example binary AND, OR
+ unary NOT)

Circuit : DAG with

n sources : input gates : in-degree = 0 :

$$\underbrace{\text{labelled } x_1 \ldots x_n}_{\text{distinct}}$$

remaining vertices : computing gates :

$\underbrace{\text{labelled with basis functions}}_{\text{not - distinct}}$

indegree = in-deg of function

outdegree = arbitrary

Some $\underline{\underline{m}}$ of vertices also designated output gates

---

Computes function mapping $(x_1 \ldots x_n) \to (y_1 \ldots y_m)$ by letting <u>input vertices</u> be labelled with input values ; <u>computing gates</u> be labelled with value of function at input nodes

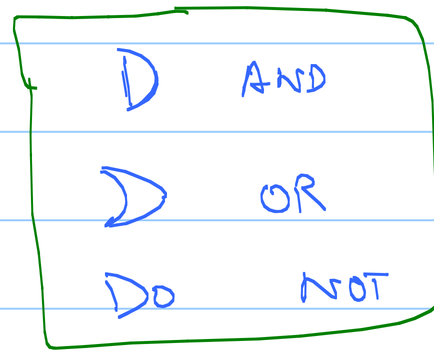
---

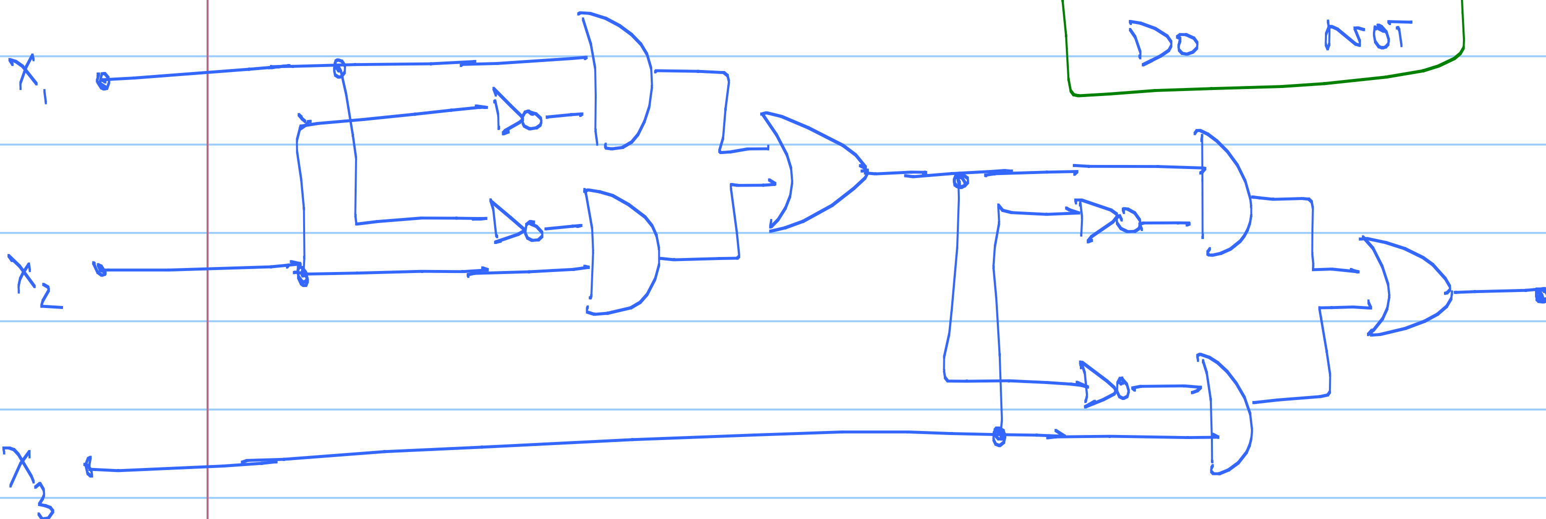
Circuit C with n inputs

1 output

decides L_n if $\forall x \; L(x) = C(x)$.

Example





Measures: Principal

Size = # edges in DAG

Circuit Complexity of L = Size of C_n as function of n

Depth = longest path in DAG.

Easy to show "EXERCISES"

1. Circuit complexity changes only by constant factor as basis changes from one finite (complete) set to another.

2. $L \in DTIME(t(n)) / (\ell(n))$

$\Rightarrow L$ has circuit complexity $O((t(n) + \ell(n))^2)$

(Turing Machine tableau, YADA YADA YADA)

3. L has circuit complexity $s(n)$

$\Rightarrow L \in DTIME(s(n)^2) / s(n)$ (ditto)

Conclusion: $P/_{poly} \equiv$ Poly-size circuits.

Circuit complexity of function $f \equiv$ Non-uniform time

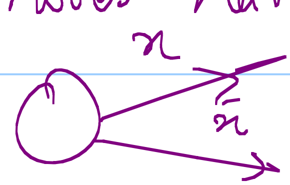
What about Non-uniform Space?

Need another model

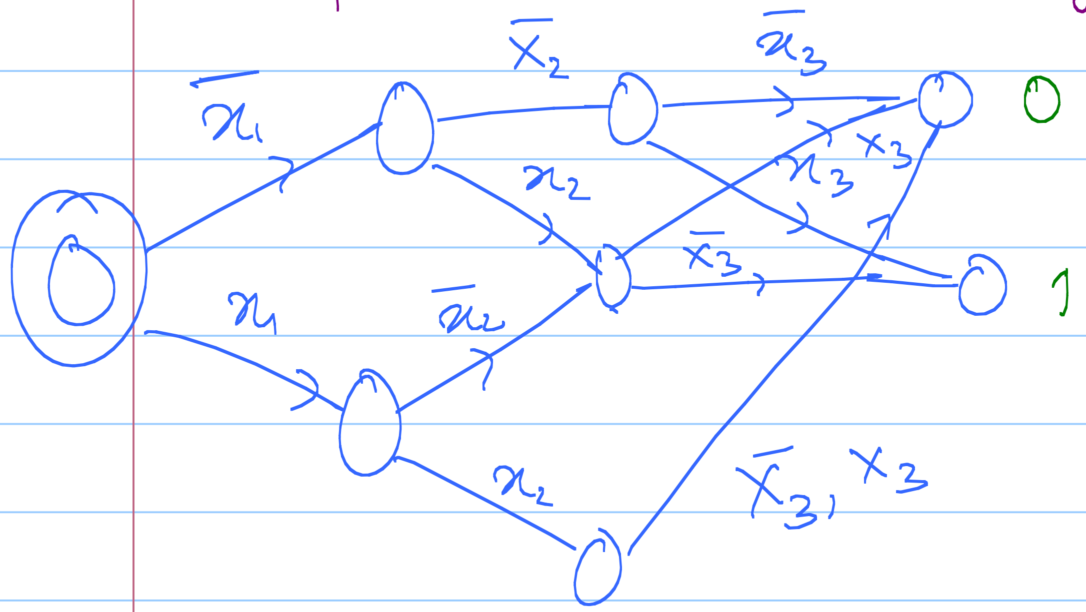
BRANCHING PROGRAMS (only defining for decision problems)

BP also given by DAG

- One source = start node
- Two sinks labelled 0 & 1;
- Remaining nodes have outdegree 2 labelled by complementary literals.



BP computes function naturally e.g.



Accepts if $x_1 + x_2 + x_3 = 1 \pmod{3}$

BP layered if vertices partitioned into layers $L_1 \dots L_k$;

edges of L_i go to L_i or L_{i+1};

WIDTH(BP) $= \max_i \{|L_i|\}$

Principal measure

$\log(\text{WIDTH}) \approx$ non-uniform space.

$\text{SIZE}(\text{BP}) \triangleq \#$ nodes

$\geq$ WIDTH

Finally

FORMULA $\equiv$ Circuit in which all nodes except input have out-degree 1;

(Why consider : formalization of colloquial term formula)

Some EXERCISES

- FORMULA SIZE $= 2^{O(\text{Formula depth})}$

  (for finite basis) [trivial]

- F-DEPTH(f) in basis B_1

  $= \Theta(\text{F-DEPTH}(f) \text{ in basis } B_2)$

- $\forall \; f : \{0,1\}^n \to \{0,1\}$

  $\text{F-depth}(f) = O(\log(\text{F-SIZE}(f)))$!

  [little harder]

- F-SIZE(f) in basis B_1

  $\leq (\text{F-SIZE}(f) \text{ in basis } B_2)^{O(1)}$

Formulae vs. Branching Programs

- BP-SIZE(f) $\leq O($ F-SIZE(f) in basis (AND, OR, NOT) $)$

(for other bases.. I'm not sure)

Branching Program Size vs. Circuit Size

- Ckt-SIZE(f) $\leq O($ BP-size $)$ [Simple]

To Prove NP $\neq$ P

suffices to give function in NP $-$ $P/_{poly}$

(seems more combinatorial)

To Prove $P \neq L$

suffices to give function in P that does not have poly-size B.P.

Unfortunately all the above open.

Best known ckt-lower bound for function in NP (in basis AND, OR, NOT) is $\sim 4.5n$.

Best known BP-size lower bound ... is $o(n^2)$.

Counting + Elementary Arguments

1. Every function f has F-SIZE in (AND, OR, NOT basis) $O(2^n)$

Proof: $f(x_1 \ldots x_n)$

$$= (x_1 = 0) \wedge f_0(x_2 \ldots x_n)$$
$$\vee (x_1 = 1) \wedge f_1(x_2 \ldots x_n)$$

---

2. # functions with Ckt-Size $s \leq s^{O(s)}$.
(f-size / BP-size)

3. $\exists$ functions with Ckt. Size $\geq_\epsilon \frac{2^n}{n}$

4. $\text{Ckt-Size}(S) \overset{\subset}{\neq} \text{Ckt-size}(S \cdot \log S)$

etc.

NECIPORUK's BP lower bound

Thm: $\exists f$ (el't distinctness) needing $\frac{n^2}{\log^2 n}$ BP size.

Idea: Counting + Restriction.

Proof.

For $S \subseteq [n]$

$\underline{\mathrm{SIZE}_S(BP)}$ = # gates with out-edges labelled by el'ts of S.

If $S_1 \ldots S_k$ = partition of $[n]$

then $\mathrm{SIZE}(BP) \geq \sum_{i=1}^{k} \mathrm{SIZE}_{S_i}(BP)$

Will Create function

$$f(\underbrace{x_1^1 \cdots x_\ell^1}_{S_1}\ \underbrace{x_1^2 \cdots x_\ell^2}_{S_2} \cdots \underbrace{x_1^k \cdots x_\ell^k}_{S_k})$$

s.t. $\forall i$ $\quad \text{BP-SIZE}_{S_i}(f) = \Omega(\ell k - k \log k)$

<u>Theorem follows.</u>

How to lower bound $\text{BP-SIZE}_{S_i}(f)$?

1. Count # distinct functions on S_i obtained by fixing bits of $\{S_j\}_{j \neq i}$.

2. log of above is a l.b. on $\text{BP-SIZE}_{S_i}(f)$

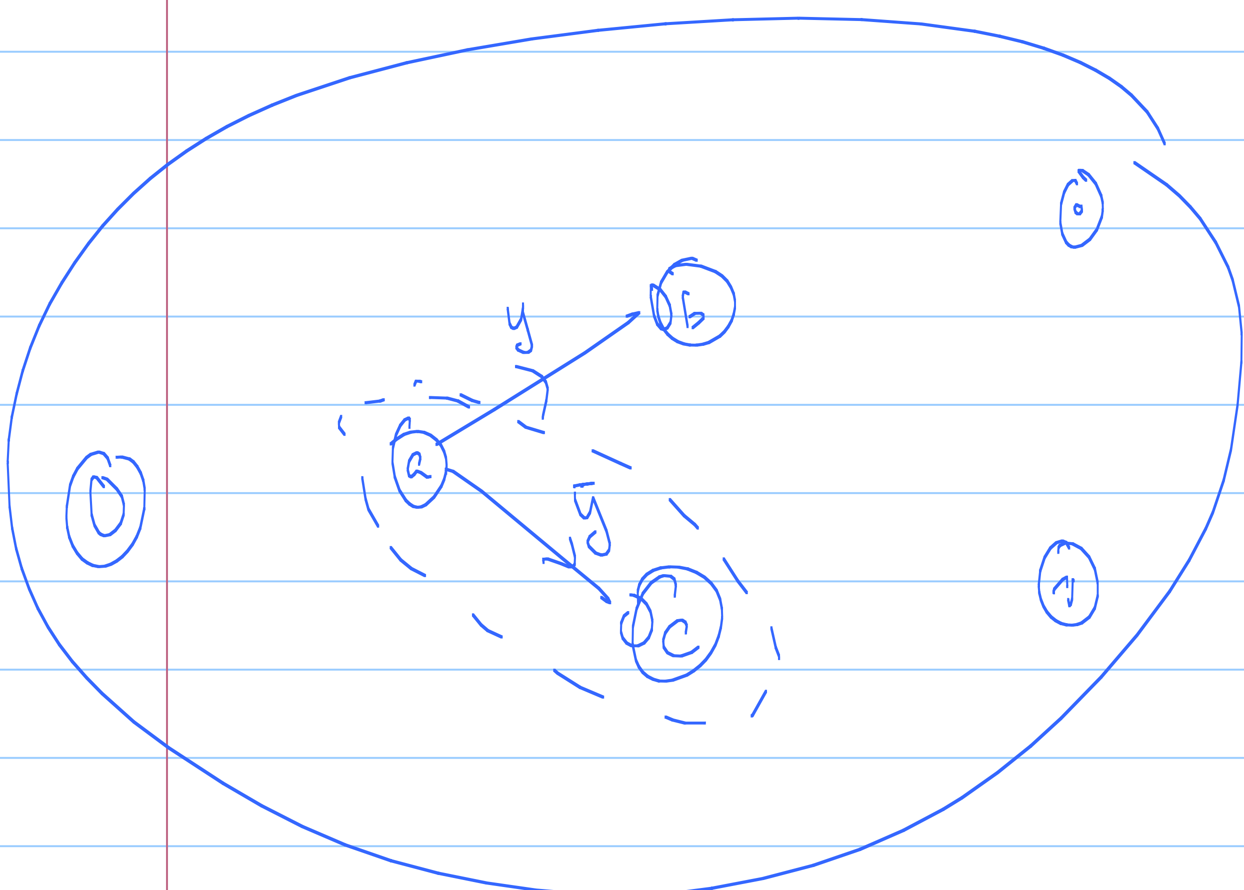
Why? ....

.... Because every such function has

BP of size $\text{BP-SIZE}_{S_i}(f)$

def $\quad g(x_1^1 \ldots x_\ell^1) = f(x_1^1 \ldots x_1^\ell, 0, 1, 1, 0 \ldots)$

then $\quad$ take $BP(f)$



If $\quad y = 0 \quad$ drop y edge & collapse a & c

Repeat ; Only edges remaining are fr S_i

How to create f s.t. $\forall S_i$

functions obtained by restricting other variables

is large ?

$$f(\underbrace{x_1' \ldots x_\ell'}_{w_1}, w_2 \ldots w_k)$$

$= 1$ if $\exists i \neq j$ s.t. $w_i = w_j$

Now by fixing $w_2 \ldots w_k$ distinct in $\{1 .. 2^\ell\}$

get $\binom{2^\ell}{k}$ distinct functions

$\log \binom{2^\ell}{k} \geq \ell k - k \log k$ $\boxtimes$