

Week 4

Test 1: Next Fri 1:30 - 2:20

(Followed by Lecture 2:30 - 4)

Test will cover: Weeks 1-3

Time complexity, nondet time

P , NP , $coNP$, EXP , PH , NP -completeness

Deterministic Time hierarchy thm (statement + proof)

Space complexity: L , NL , $PSPACE$, $NPSPACE$

$NPSPACE(s) \subseteq SPACE(s^2)$ Savich's thm + proof

$SPACE$ Hierarchy Thm (just statement)

Relationships between space/time classes

→ Add'l Practice Problems for Test 1 posted

Today - Space complexity cont'd.

Last Class: $NSPACE(S) \subseteq SPACE(S^2)$

Today: More surprises/collapses

Space Surprises

1. Savitch '70 : $NSPACE[s] \subseteq SPACE[s^2]$ ✓
 2. Barrington '86 :
Any size s formula can be evaluated
with only $O(1)$ bits of space! *
 3. Immerman - Stelepcsenyi '87 : $NSPACE(s) = coNSPACE[s]$
($\therefore NL = coNL$)
 4. Reingold '05 : $USTCONN$ (undirected reachability) in $logSPACE$
 5. Williams '25 : $TIME(T) \subseteq SPACE(\sqrt{T \log T})$ *
- } we will probably skip

* : TODAY

Theorem [R. Williams '25]

$$\text{TIME}(T) \subseteq \text{SPACE}(\sqrt{T \log T})$$

$\Uparrow$

Implies $\text{SPACE}(n)$ not contained in $\text{TIME}(n^{1.99})$

Note: similarly if $\text{TIME}(T) \subseteq \text{SPACE}(T^\epsilon) \forall \epsilon > 0$
Then $\text{PSPACE} \neq P$!

Proof heavily relies on:

Cook-Mertze : Low space alg for tree evaluation

which relies on ideas from:

Barrington's Thm, Cleve-BenOR } "Catalytic"
Computing

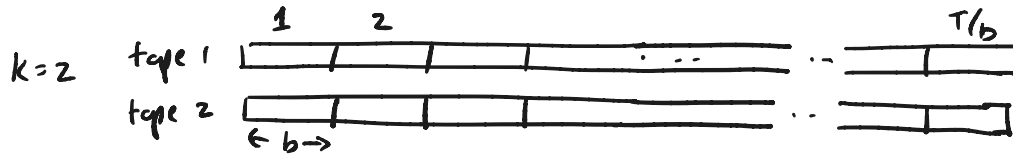
Proof

Let M be a k -tape time T TM.

Define config graph $g_{M,x}$:

* Note: this is not the same config graph we defined last class for Savitch's Thm

- We divide time steps into T/b intervals, each of length b
- Partition each tape into blocks each of length b

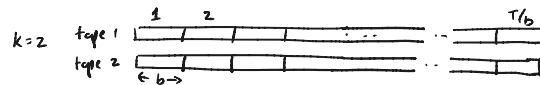


$$V = \{v_0, v_1, \dots, v_{T/b}\}$$

v_t : represents t^{th} time interval of computation of $M(x)$

Define config graph $g_{M,x}$:

- We divide time steps into T/b intervals, each of length b
- Partition each tape into blocks each of length b



$$V = \{v_0, v_1, \dots, v_{T/b}\}$$

v_t : represents t^{th} time interval of computation of $M(x)$

content(v_t): ① $\forall$ tapes j : ① Names of all tape- j blocks visited in interval t
** at most 2 per tape **

Call them $B_{j,1}(t), B_{j,2}(t)$

② Contents of $B_{j,1}(t), B_{j,2}(t)$ at end of interval t

② Position of all heads at end of interval t

③ State of $M(x)$ at end of interval t

Define config graph $g_{M,x}$:

- We divide time steps into T/b intervals, each of length b
- Partition each tape into blocks each of length b



$$V = \{v_0, v_1, \dots, v_{T/b}\}$$

v_t : represents t^{th} time interval of computation of $M(x)$

content(v_t): ① $\forall$ tapes j : ① Names of all tape j blocks visited in interval t
 at most 2 per tape

Call them $B_{j,1}(t), B_{j,2}(t)$

② Contents of $B_{j,1}(t), B_{j,2}(t)$ at end of interval t

space
 $O(k \cdot b) = O(b)$

② Position of all heads at end of interval t

③ state of $M(x)$ at end of interval t

v_0 : start config

content(v_0): initial config

} x is hardwired
 so no space

Define config graph $g_{M,x}$:

- We divide time steps into T/b intervals, each of length b
- Partition each tape into blocks each of length b



$$V = \{v_0, v_1, \dots, v_{T/b}\}$$

Edges: $v_{t'} \rightarrow v_t$ if we need info stored in $v_{t'}$ to determine contents(v_t)

v_j : Let $B_{j,1}(t)$, $B_{j,2}(t)$ be the tape- j blocks touched in interval t

Let $t_{j,1}$ be last previous time interval that visited block $B_{j,1}(t)$

" $t_{j,2}$ be last " " " " " $B_{j,2}(t)$

Then $(v_{t_{j,1}} \rightarrow v_t)$ and $(v_{t_{j,2}} \rightarrow v_t)$ } $\leq 2K+1$ edges into v_t
 Also $v_{t-1} \rightarrow v_t$ } $= O(1)$ indegree

Define config graph $g_{M,x}$:

- We divide time steps into T/b intervals, each of length b
- Partition each tape into blocks each of length b



$$V = \{v_0, v_1, \dots, v_{T/b}\}$$

Edges: $v_{t'} \rightarrow v_t$ if we need info stored in $v_{t'}$ to determine contents(v_t)

V_j : Let $B_{j,1}(t)$, $B_{j,2}(t)$ be the tape- j blocks touched in interval t

Let $t_{j,1}$ be last previous interval that visited block $B_{j,1}(t)$

" $t_{j,2}$ be last " " " " " $B_{j,2}(t)$

Then $(v_{t_{j,1}} \rightarrow v_t)$ and $(v_{t_{j,2}} \rightarrow v_t)$

Also $v_{t-1} \rightarrow v_t$

← while $v_{t_{j,1}}$, $v_{t_{j,2}}$ store contents of blocks $B_{j,1}(t)$, $B_{j,2}(t)$ they don't contain head / state info for start of interval t

Define config graph $g_{M,x}$:

- We divide time steps into T/b intervals, each of length b
- Partition each tape into blocks each of length b



$$V = \{v_0, v_1, \dots, v_{T/b}\}$$

Edges: $v_{t'} \rightarrow v_t$ if we need info stored in $v_{t'}$ to determine contents(v_t)

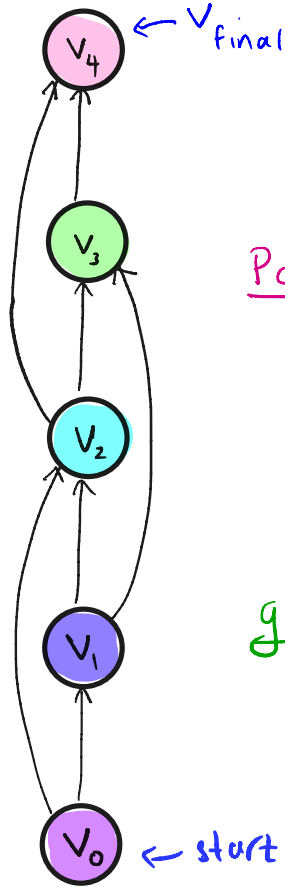
V_j : Let $B_{j,1}(t)$, $B_{j,2}(t)$ be the tape- j blocks touched in interval t

Let $t_{j,1}$ be last previous interval that visited block $B_{j,1}(t)$

" $t_{j,2}$ be last " " " " " $B_{j,2}(t)$

Then $(v_{t_{j,1}} \rightarrow v_t)$ and $(v_{t_{j,2}} \rightarrow v_t)$ } $\leq 2K+1$ edges into v_t
 Also $v_{t-1} \rightarrow v_t$ } $= O(1)$ indegree

$g_{M,x}$:



Parameters : Set $b = \sqrt{T}$

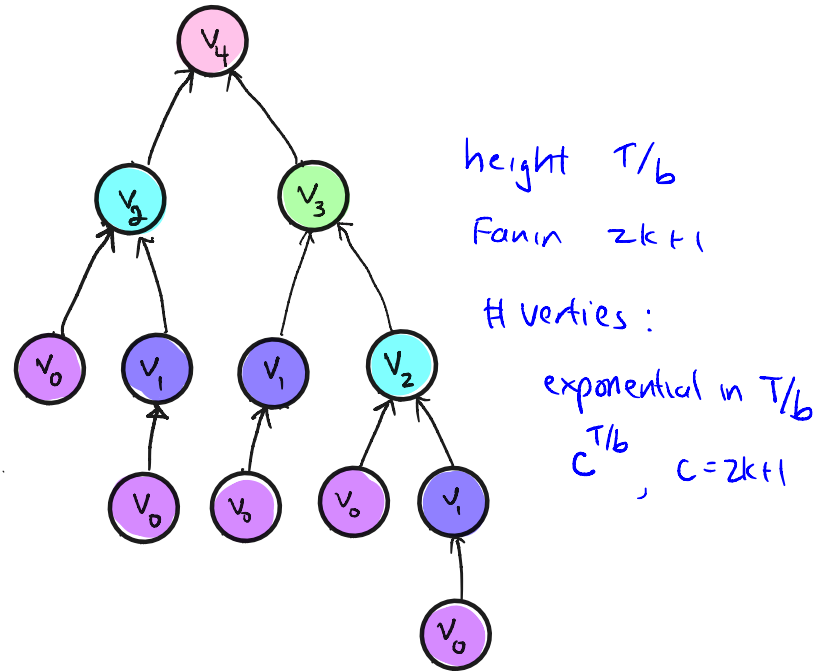
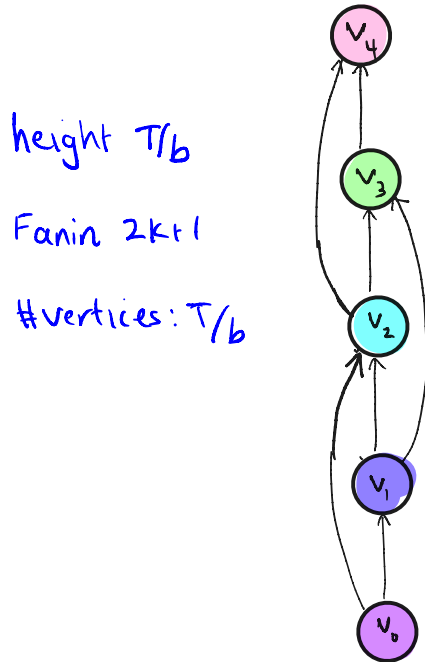
So $g_{M,x}$ is a constant-indegree line graph with $T/b = \sqrt{T}$ vertices, and each vertex stores $O(b) = O(\sqrt{T})$ bits of info

Goal

Assume for now $g_{M,x}$ is given (on input tape)
Want to determine contents of final interval v_{final}
in space $\sim O(\sqrt{T})$ (in particular:
final state of M at v_{final})

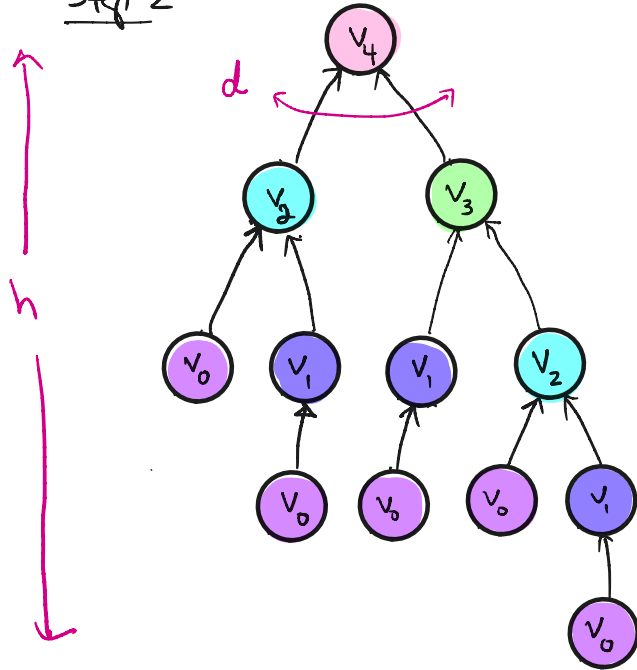
Idea I Follow optimal Tree Pebbling Strategy

Step 1: "Blow up" $G_{M,x}$ into a tree:



Idea I Follow optimal Tree Pebbling Strategy

Step 2



Follow Minimal Pebbling strategy for a tree.

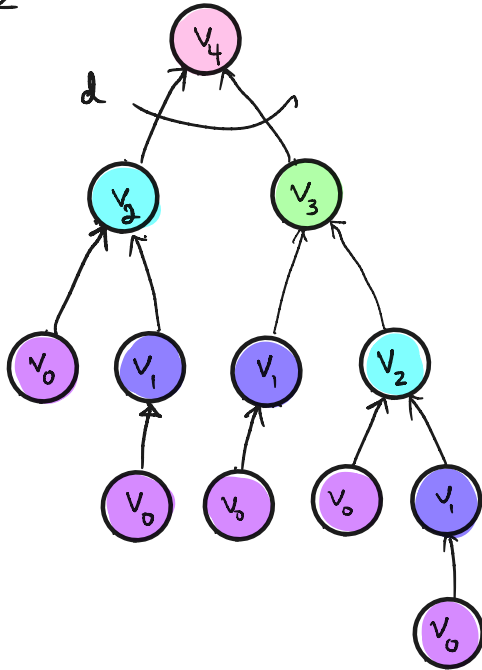
Pebbling rules:

- Can always put pebble on source
- Can always remove a pebble
- If all children of v Pebbled, then can put a pebble on v

Want to pebble sink to minimize
 $\max(\# \text{ pebbles })$ over all steps

Idea I Follow optimal Tree Pebbling Strategy

Step 2



Follow Minimal Pebbling strategy for a tree.

min # pebbles (per step):
 $\sim O(h \cdot d)$

$\therefore$ space is $O(h \cdot d \cdot (bk)) = O(h \cdot d \cdot b)$

space to store
content of one v_i

$h = \frac{T}{b}$ $d = o(1) \Rightarrow$ space $O(T)$ 😞

Can we do better than optimal pebbling strategy?

Circumventing Pebbling Barrier

Mind Blowing Idea: [Cook-Mertze
Also in earlier papers Barrington, ...]

Pebbling always allocates fresh space for each value computed at each level

Instead we can "write over" an old value in a reversible way, so later we can "undo" to recover old value.

Circumventing Pebbling Barrier

Mind Blowing Idea: [Cook-Mertz
Also in earlier papers Barrington, ...]

Instead we can "write over" an old value in a reversible way, so later we can "undo" to recover old value

Theorem [J. Cook, J. Mertz '24]

Given Tree height h , fanin d , b -bits per vertex,

and function $f_v: \{0,1\}^{bd} \rightarrow \{0,1\}^b$

can compute value of root in space $O(h \log(db) + db)$

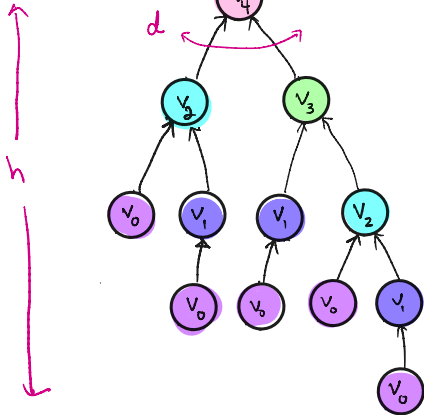
Note: Pebbling strategy gives $O(hdb)$

Theorem [J. Cook, J. Mertz]

Given Tree height h , fanin d , b -bits per vertex,
and functions $f_v: \{0,1\}^{bd} \rightarrow \{0,1\}^b$
can compute value of root in space $O(h \log(db) + db)$

Our situation : $b = \sqrt{T}$
 $h = T/b = \sqrt{T}$
 $d = O(1)$

$$\begin{aligned} \Rightarrow \text{Space } & O(h \log(db) + db) \\ &= O(\sqrt{T} \log b + b) \\ &= O(\sqrt{T} \log \sqrt{T} + \sqrt{T}) \\ &= O(\sqrt{T} \log T) \quad ! \end{aligned}$$



One Issue to Resolve : As in earlier case (Savich Thm), we don't actually have $g_{M,x}$ on input tape.

How to compute $g_{M,x}$ on the fly (still maintaining space $O(T \log T)$)?

Idea: guess $g_{M,x}$ and check

Encoding a "guessed" graph g' :

Edges in g' determined by knowing tape blocks accessed in v_t

$\forall t$ encode name of tape blocks accessed during interval t
 $\log(T/b) \cdot T/b$

Idea: guess $g_{M,x}$ and check

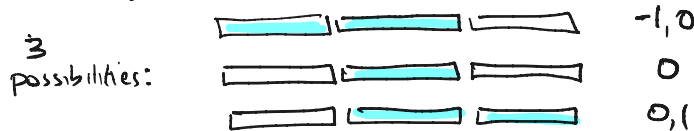
Encoding a "guessed" graph g' :

Edges in g' determined by knowing tape blocks accessed in V_t

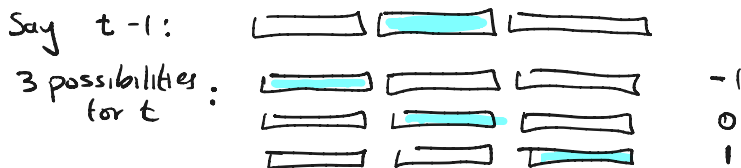
~~$\forall t$ encode name of tape blocks accessed during interval t
 $\log(T/b) \cdot T/b$~~

Better: For each tape j and time interval t use $O(1)$ bits to record:

① Which tape blocks adjacent to one containing head are accessed in interval t



② How tape block containing head changes from interval $t-1 \Rightarrow t$



Idea: guess $g_{M,x}$ and check

① Encoding a "guessed" graph g' :

Edges in g' determined by knowing tape blocks accessed in V_t

~~$\forall t$ encode name of tape blocks accessed during interval t~~
 ~~$\log(T/b) \cdot T/b$~~

Better: For each tape j and time interval t use $O(1)$ bits to record:

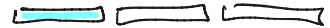
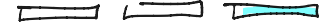
① Which tape blocks adjacent to one containing head are accessed in interval t

3 possibilities:

	-1, 0
	0
	0, 1

② How tape block containing head changes from interval $t-1 \Rightarrow t$

Say $t-1$:

	
3 possibilities for t :	
	-1
	0
	1

② Run CM Alg using guessed g'
if we find a violation (e.g., tape head violates g') FAIL +
try new g'

Starting Intuition behind "Reusing" Space

Start with $R1 : x$ $R2 : y$

Want to swap contents to get : $R1 : y$ $R2 : x$

Naively we need need 3 registers

0. $R1 : x$ $R2 : y$

1. $R0 \leftarrow x$

$R1 \leftarrow y$

Smarter:

0. $R1 : x$ $R2 : y$

$R1 \leftarrow R1 \oplus y = x \oplus y$

$R2 \leftarrow R2 \oplus R1 = y \oplus (x \oplus y) = x$

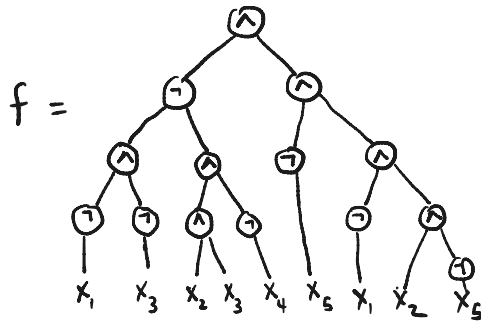
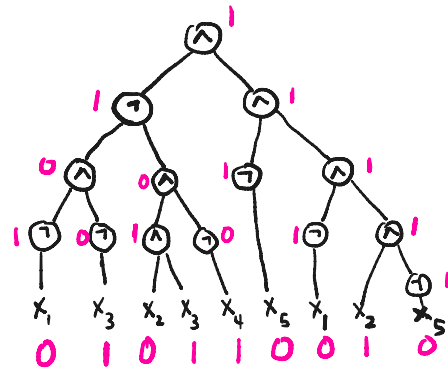
$R1 \leftarrow R1 \oplus R2 = (x \oplus y) \oplus x = y$

Early Famous Example [Barrington]

Formula Evaluation: Input: Boolean formula $f(x_1 \dots x_n)$ over $\wedge, \neg$
and input $\alpha \in \{0, 1\}^n$
Output: $f(\alpha) \in \{0, 1\}$

Example Input f, α

Output : $f(x) = 1$


$$\alpha = 01110$$


Early Famous Example [Barrington]

Formula Evaluation: Input: Boolean formula $f(x_1 \dots x_n)$ over $\wedge, \neg$
and input $\alpha \in \{0,1\}^n$
Output: $f(\alpha) \in \{0,1\}$

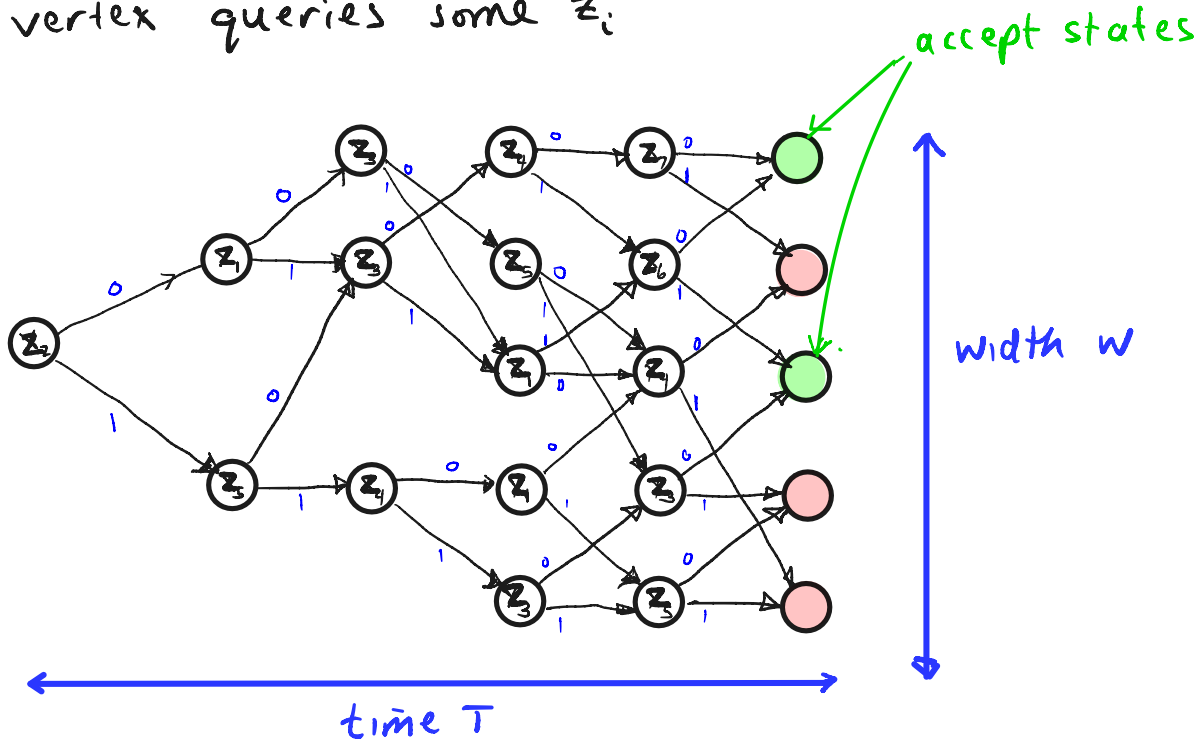
Theorem (Barrington)

Any Boolean formula over n variables
can be evaluated by a branching program in $O(1)$ space,
and time $\text{poly}(|f|)$

Branching Program over $z_1, \dots, z_m$

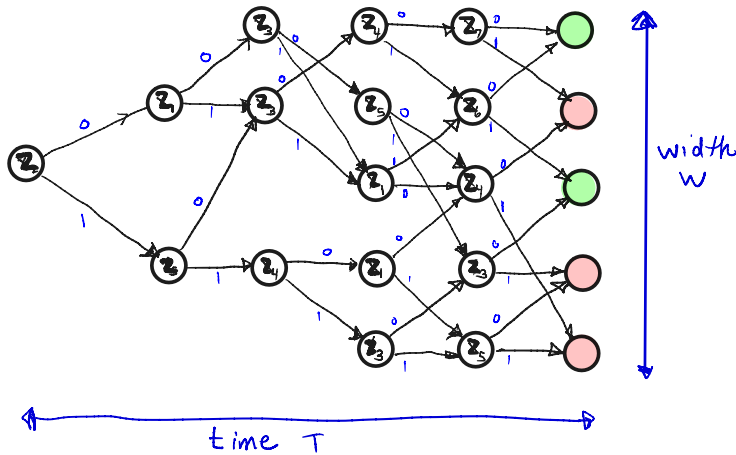
Layered directed graph.

Each vertex queries some z_i .



Branching Program over $z_1, \dots, z_m$

Layered directed graph.
Each vertex queries some z_i .



A (uniformly constructible)
Width w , time T Branching Program
is like a "Random Access" TM:

- Any input bit can be accessed in one step
- $\log w$ is "space"

Simulation by ordinary TM:

$$\text{TM space} \approx \log w + \underbrace{O(\log w)}$$

↑
To keep track of time
and to access input x_i .

$$\text{TM time} \approx \text{poly}(T)$$

Theorem (Barrington)

Any Boolean formula over n variables
can be evaluated by a branching program in $O(1)$ space,
and time $\text{poly}(|f|)$

Barrington original Thm: space : 5 states = $\lceil \log_2 5 \rceil = 3$ bits
we'll do a simpler proof by Ben-OR cleve : 8 states = 3 bits

* If f has $\text{poly}(n)$ size, can solve by a TM in
 $O(1)$ space AND time $\text{poly}(n)$

Main step: computing $P_{F_1 \wedge F_2}$ (inductively)

- Assume there already exists a 3-register Program P_{F_1} :

Initial Register Contents of P_{F_1} :

t_1	t_2	t_3
-------	-------	-------

$R1 \qquad R2 \qquad R3$

Final Register Contents of P_{F_1} :

t_1	t_2	$t_3 \oplus F_1$
-------	-------	------------------

$R1 \qquad R2 \qquad R3$

- Also assume there already exists a 3-register Program P_{F_2}

Main Step: Computing $P_{F_1 \wedge F_2}$ (inductively)

- Assume there already exists a 3-register Program P_{F_1} :

Initial Register Contents of P_{F_1} :

t_1	t_2	t_3
R1	R2	R3

Final Register Contents of P_{F_1} :

t_1	t_2	$t_3 \oplus F_1$
R1	R2	R3

- Also assume there already exists a 3-register Program P_{F_2}

Naively we need more registers:

	R1	R2	R2	R4	R5
Initial content	t_1	t_2	t_3	t_4	t_5
Copy $t_2 : R_4 \leftarrow R_2$	t_1	t_2	t_3	t_2	t_5
Copy $t_3 : R_5 \leftarrow R_3$	t_1	t_2	t_3	t_2	t_3
Reset R2, R3 : $R_2, R_3 \leftarrow 0$	t_1	0	0	t_2	t_3
RUN P_{F_1} :	t_1	0	F_1	t_2	t_3
RUN P_{F_2} into R2 :	t_1	F_2	F_1	t_2	t_3
$R3 \leftarrow R4 + R2 \times R3$	t_1	F_2	$t_3 + F_1 F_2$	t_2	t_3
$R2 \leftarrow R4$	t_1	t_2	$t_3 + F_1 F_2$	t_2	t_3

all t 's are mod 2

Main step: computing $P_{F_1 \wedge F_2}$ (inductively)

<u>Program</u>	R_1	R_2	R_3
Initial:	t_1	t_2	t_3
① call P_{F_1}, P_{F_2}	$t_1 + F_1$	$t_2 + F_2$	t_3
② $R_3 \leftarrow R_3 + R_1 R_2$	$t_1 + F_1$	$t_2 + F_2$	$t_3 + (t_1 + F_1)(t_2 + F_2)$
③ call F_1	t_1	$t_2 + F_2$	$t_3 + (t_1 + F_1)(t_2 + F_2)$
④ $R_3 \leftarrow R_3 + R_1 R_2$	t_1	$t_2 + F_2$	$t_3 + (t_1 + F_1)(t_2 + F_2) + t_1(t_2 + F_2) = t_3 + t_2 F_1 + F_1 F_2$
⑤ P_{F_1}, P_{F_2}	$t_1 + F_1$	t_2	$t_3 + t_2 F_1 + F_1 F_2$
⑥ $R_3 \leftarrow R_3 + R_1 R_2$	$t_1 + F_1$	t_2	$t_3 + t_2 F_1 + F_1 F_2 + (t_1 + F_1)t_2 = t_3 + t_1 t_2 + F_1 F_2$
⑦ P_{F_1}	t_1	t_2	$t_3 + t_1 t_2 + F_1 F_2$
⑧ $R_3 \leftarrow R_3 + R_1 R_2$	t_1	t_2	$t_3 + t_1 t_2 + F_1 F_2 + t_1 t_2 = t_3 + F_1 F_2$

all t 's are mod 2

Main step: computing $P_{F_1 \wedge F_2}$ (inductively)

Program	R_1	R_2	R_3
Initial:	t_1	t_2	t_3
① call P_{F_1}, P_{F_2}	$t_1 + F_1$	$t_2 + F_2$	t_3
② $R_3 \leftarrow R_3 + R_1 R_2$	$t_1 + F_1$	$t_2 + F_2$	$t_3 + (t_1 + F_1)(t_2 + F_2)$
③ call P_{F_1}	t_1	$t_2 + F_2$	$t_3 + (t_1 + F_1)(t_2 + F_2)$
④ $R_3 \leftarrow R_3 + R_1 R_2$	t_1	$t_2 + F_2$	$t_3 + (t_1 + F_1)(t_2 + F_2) + t_1(t_2 + F_2) = t_3 + t_2 F_1 + F_1 F_2$
⑤ P_{F_1}, P_{F_2}	$t_1 + F_1$	t_2	$t_3 + t_2 F_1 + F_1 F_2$
⑥ $R_3 \leftarrow R_3 + R_1 R_2$	$t_1 + F_1$	t_2	$t_3 + t_2 F_1 + F_1 F_2 + (t_1 + F_1)t_2 = t_3 + t_1 t_2 + F_1 F_2$
⑦ P_{F_1}	t_1	t_2	$t_3 + t_1 t_2 + F_1 F_2$
⑧ $R_3 \leftarrow R_3 + R_1 R_2$	t_1	t_2	$t_3 + t_1 t_2 + F_1 F_2 + t_1 t_2 = t_3 + F_1 F_2$

6 inductive calls

Let f be arbitrary formula over AND, $\neg$, and let $x_1 \dots x_n$ be input. Then using just 3 bits of work tape we can compute $f(x_1 \dots x_n)$

Inductive statement: For any formula f , and for any initial contents of 1-bit registers

t_1	t_2	t_3
R_1	R_2	R_3

we can compute:

t_1	t_2	$t_3 + f$
-------	-------	-----------

Note: similarly can compute
and also

$t_1 + f$	t_2	t_3
-----------	-------	-------

t_1	$t_2 + f$	t_3
-------	-----------	-------

Let f be arbitrary formula over AND, $\neg$, and let $x_1 \dots x_n$ be input. Then using just 3 bits of work tape we can compute $f(x_1 \dots x_n)$

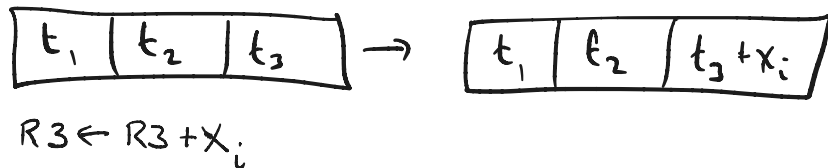
Inductive statement: For any formula f , and for any initial contents of 1-bit registers

t_1	t_2	t_3
R_1	R_2	R_3

we can compute:

t_1	t_2	$t_3 + f$
-------	-------	-----------

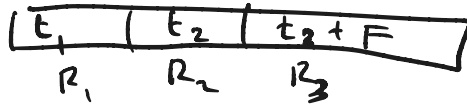
Base case $f = x_i$



Inductive statement: For any formula F , and for any initial contents of 1-bit registers



we can compute:



Inductive step:

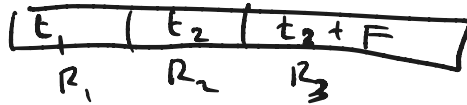
Case I: $F = \neg F_1$: Assume we have P_{F_1}

Program P_{F_1} :	Program	R_1	R_2	R_3
	Initial:	t_1	t_2	t_3
	① Call P_{F_1}	t_1	t_2	$t_3 + F_1$
	② $R_3 \leftarrow R_3 + 1$	t_1	t_2	$t_3 + F_1 + 1 = t_3 + F$

Inductive statement: For any formula F , and for any initial contents of 1-bit registers



we can compute:



Inductive step:

Case I: $F = \neg F_1$: Assume we have P_{F_1}

Case II: $F = F_1 \wedge F_2$: Assume P_{F_1}, P_{F_2}

We already did this case!

Theorem (Barrington)

Any Boolean formula over n variables
can be evaluated by a branching program in $O(1)$ space,
and time $\text{poly}(|f|)$

We showed f has a 3-bit program for any formula f !

Runtime : Assume wlog that f is balanced so
 F has size s , and depth $O(\log s)$.

Each subformula f_i of depth d_i : makes ≤ 6 recursive
calls to depth $d_i - 1$ subformulas.

$$\therefore \text{runtime } f_i \sim 6^{d_i}$$

$$\therefore \text{runtime } f \sim 2^{O(d)} = 2^{O(\log s)} = \text{poly}(s)$$

