A Random Walk Down Full Memory Lane

Ian Mertz
Charles University

Today

What is . . .

. . . a Random Walk? . . . Full Memory?

How?

What Next?

Today

What is...

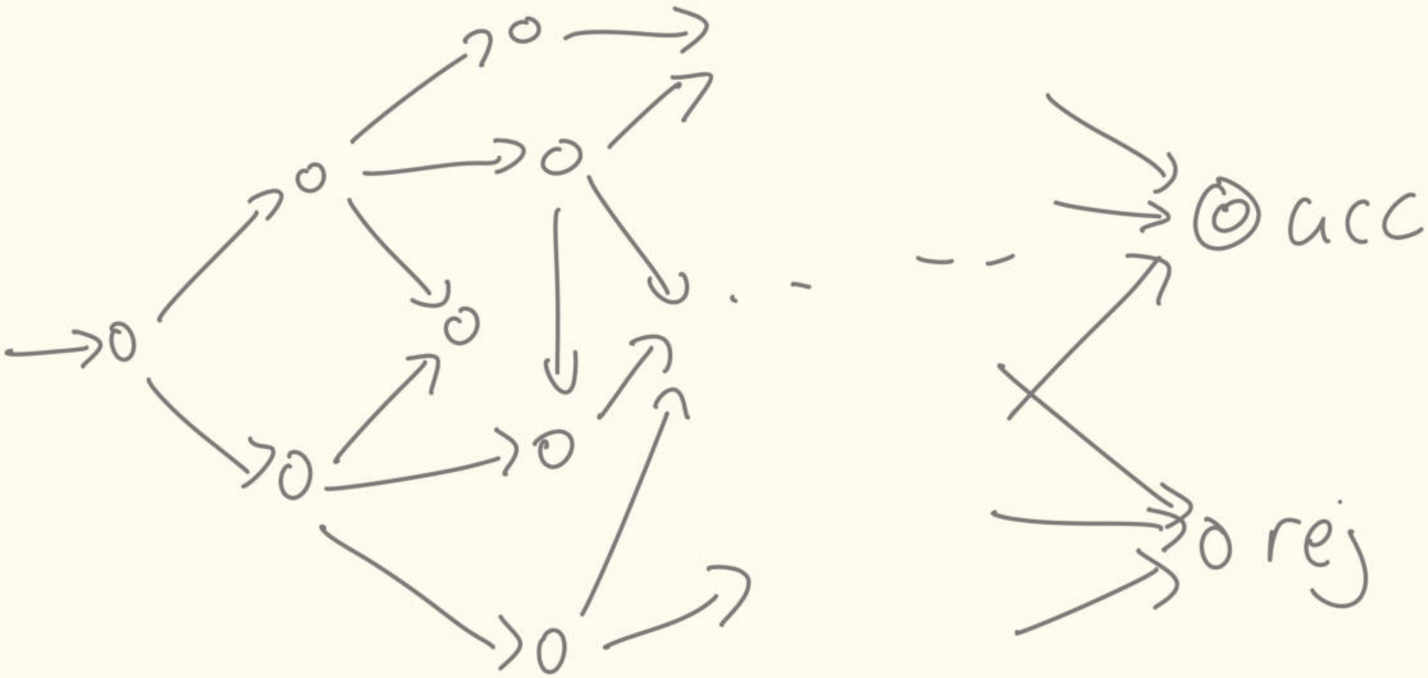
...a Random Walk? ...Full Memory?

How?

What Next?

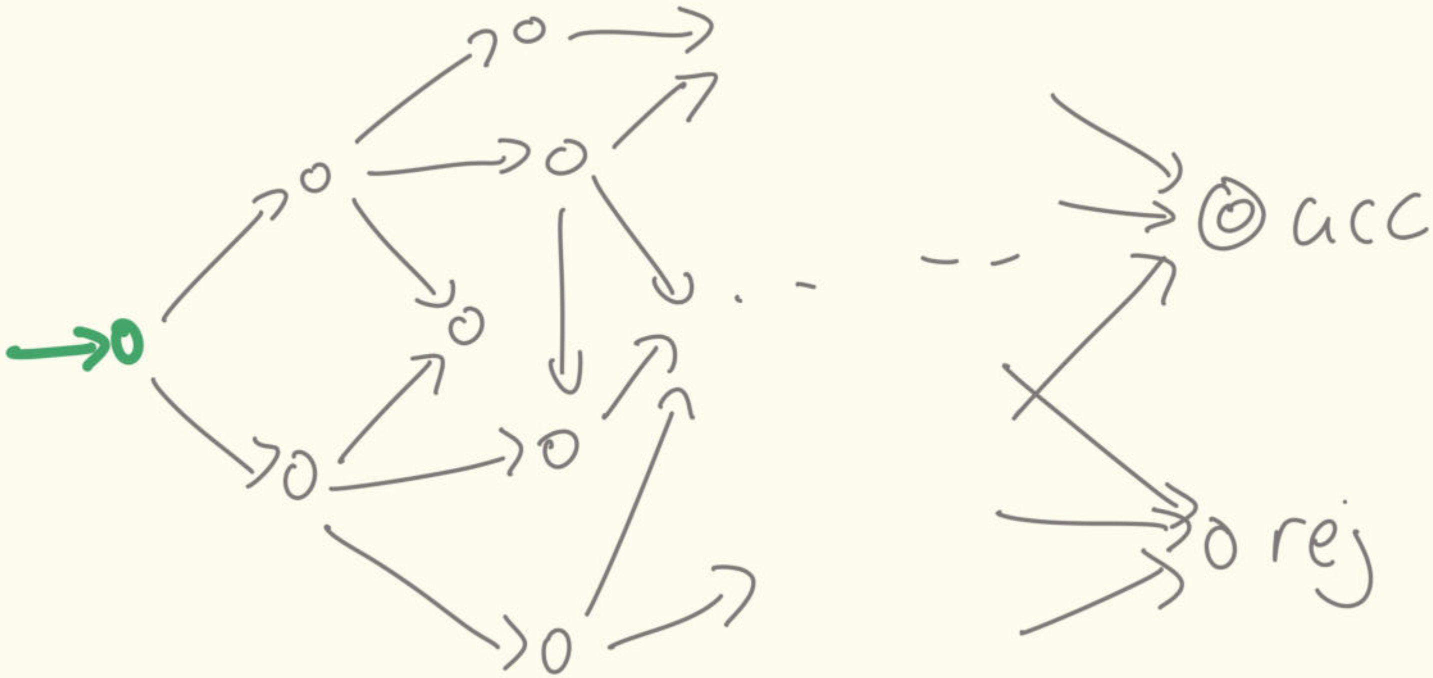
WALKS

G (size poly n)



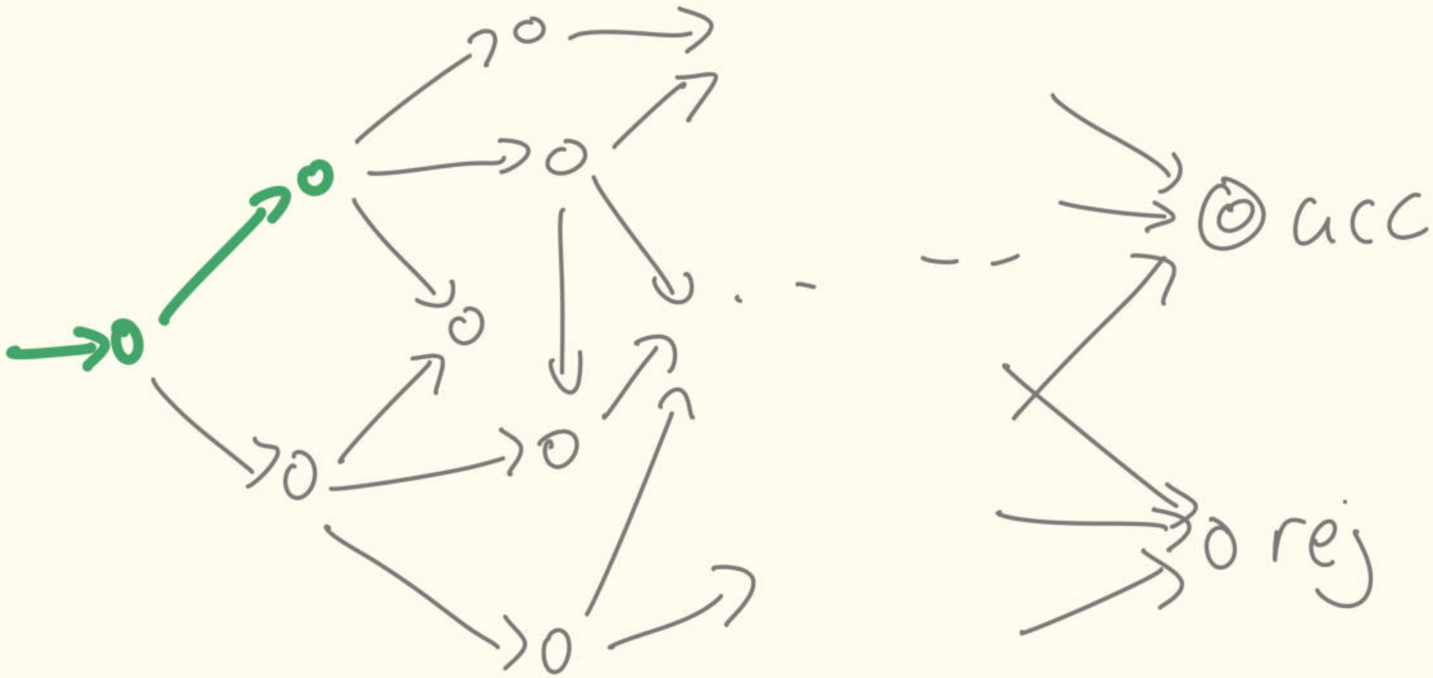
WALKS

G (size poly n)



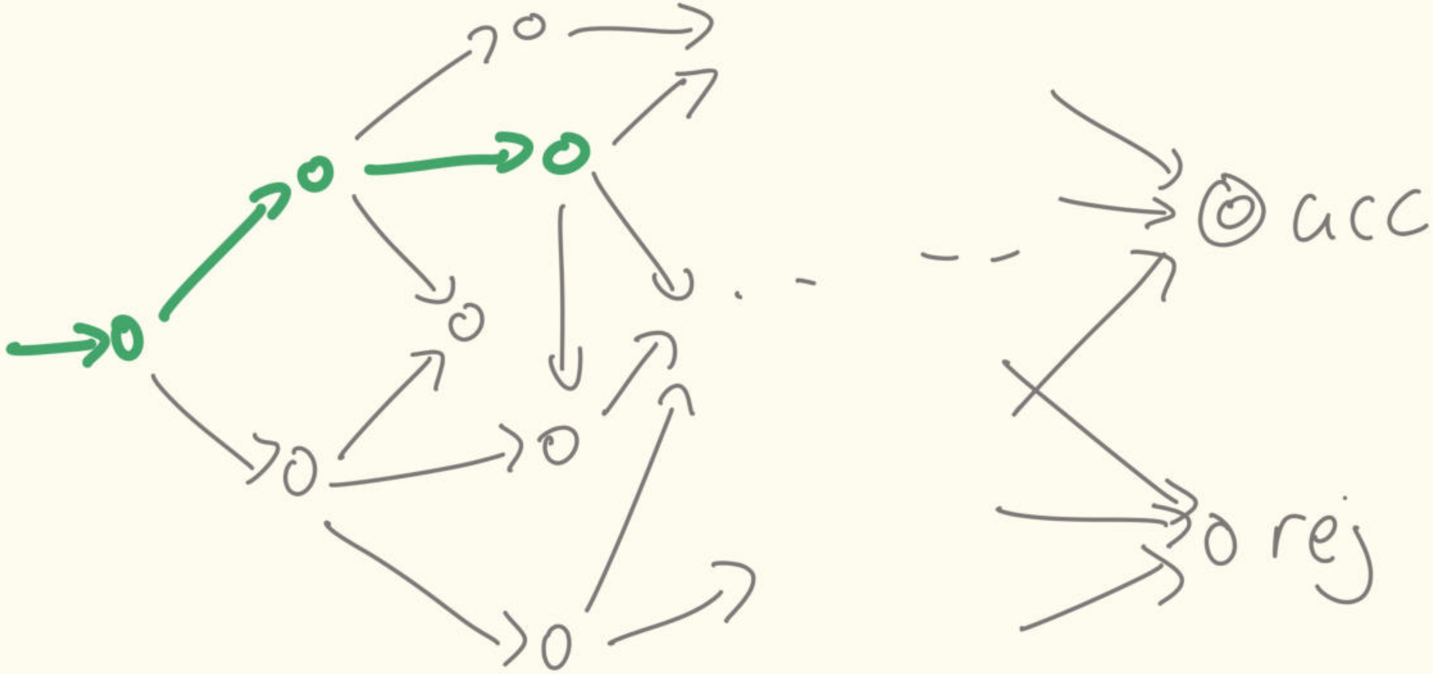
WALKS

G (size poly n)

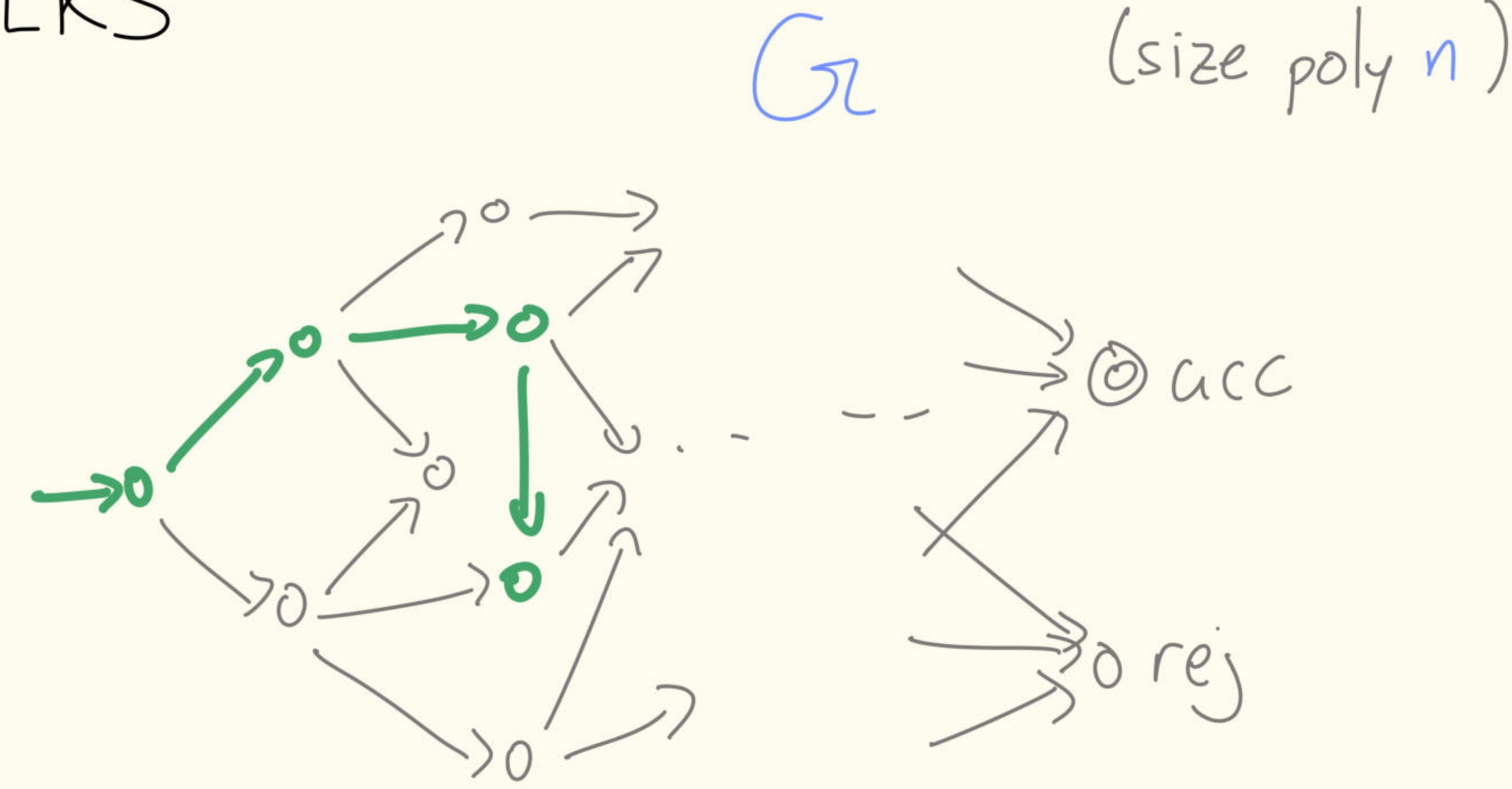


WALKS

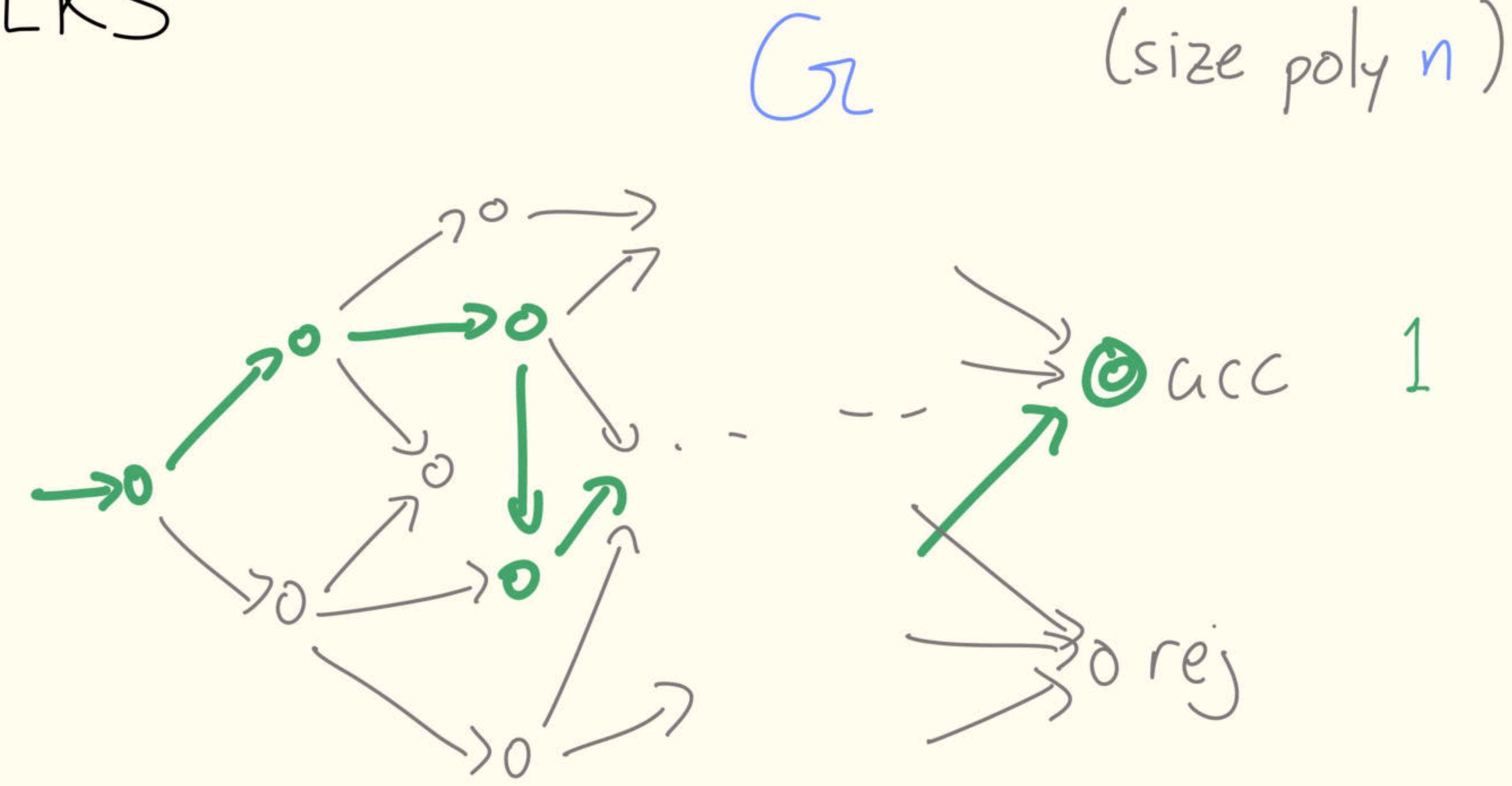
G (size poly n)



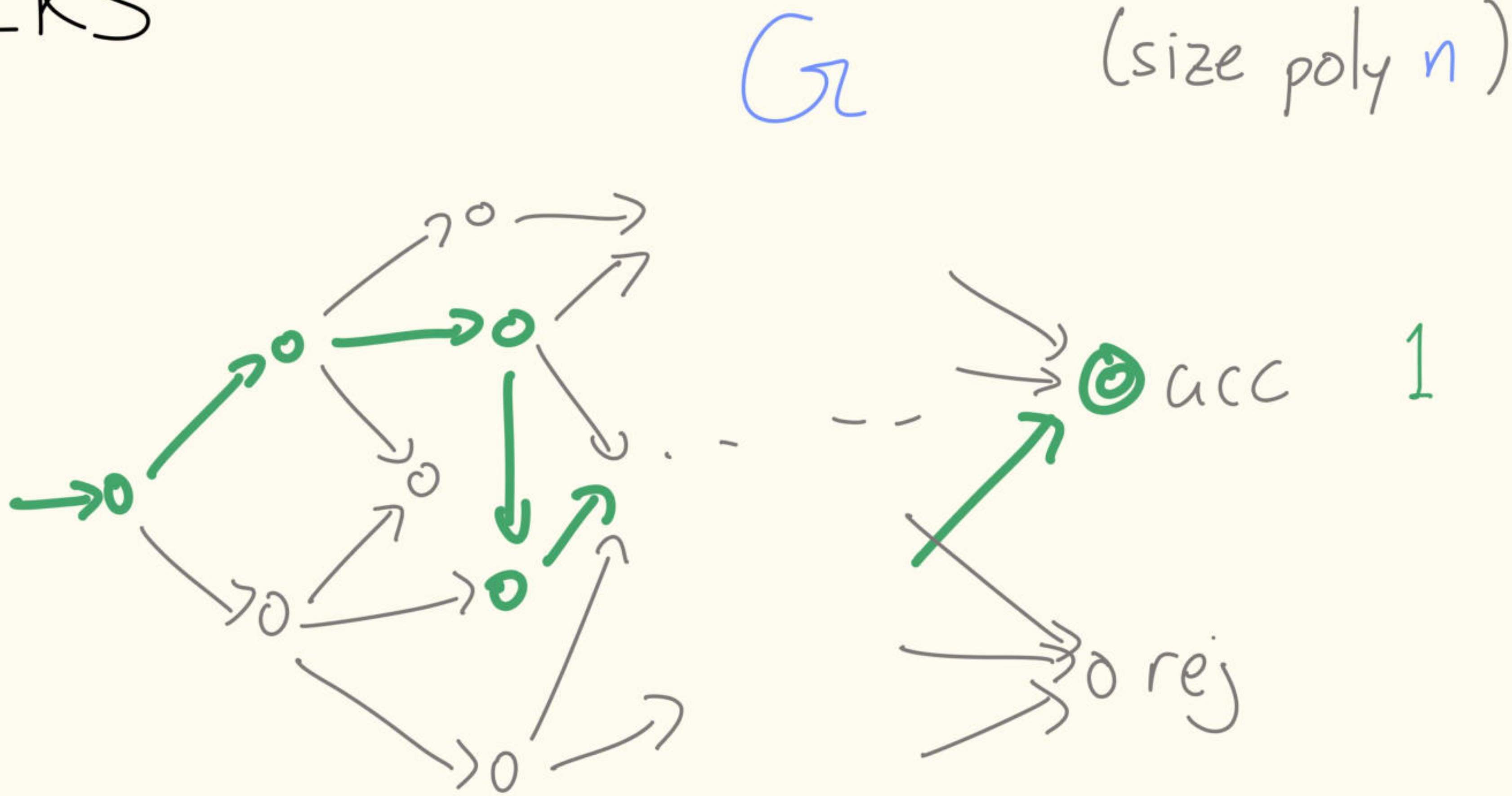
WALKS



WALKS



WALKS

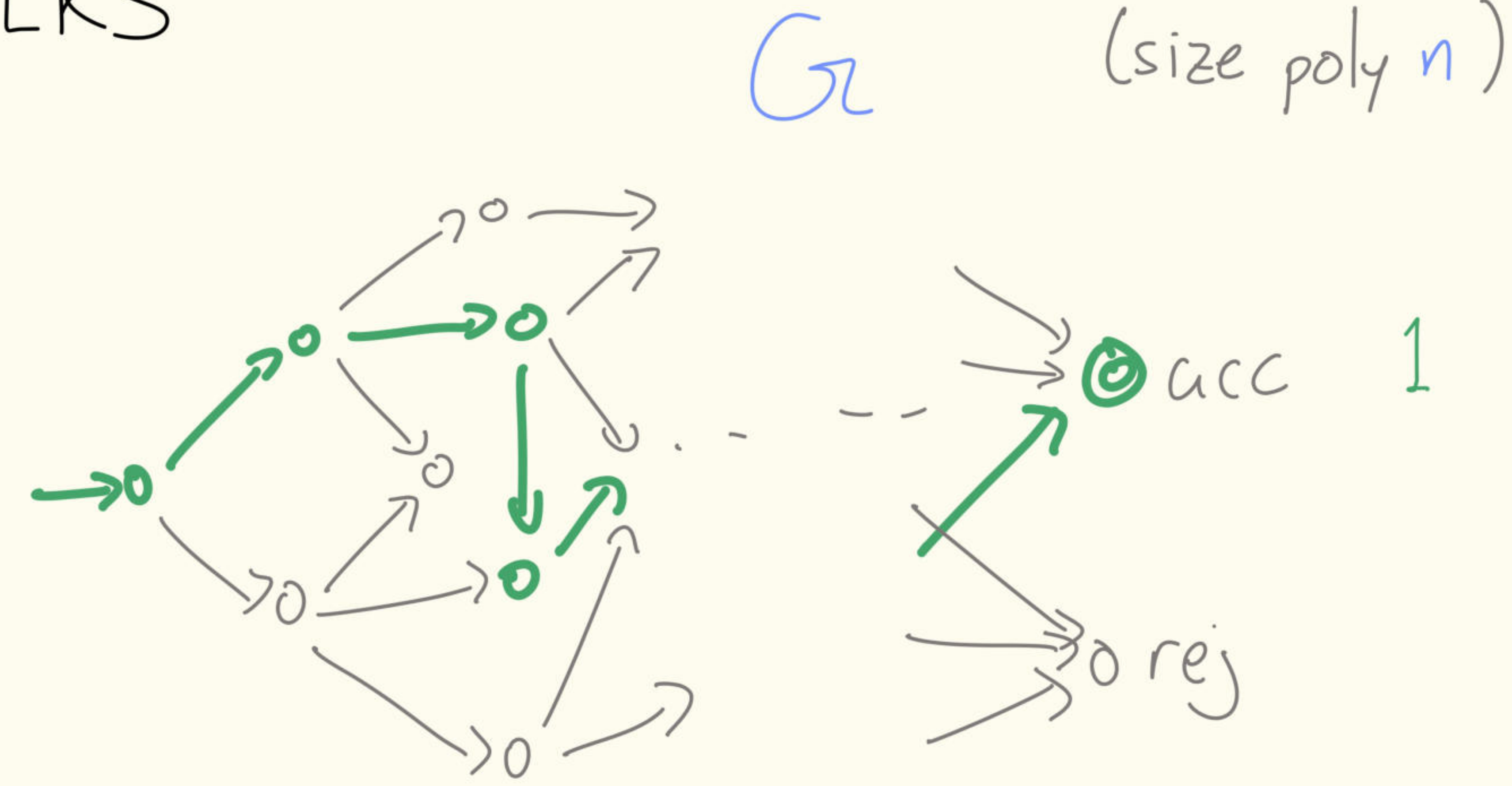


∃ path?

how many paths?

where does a random path go?

WALKS

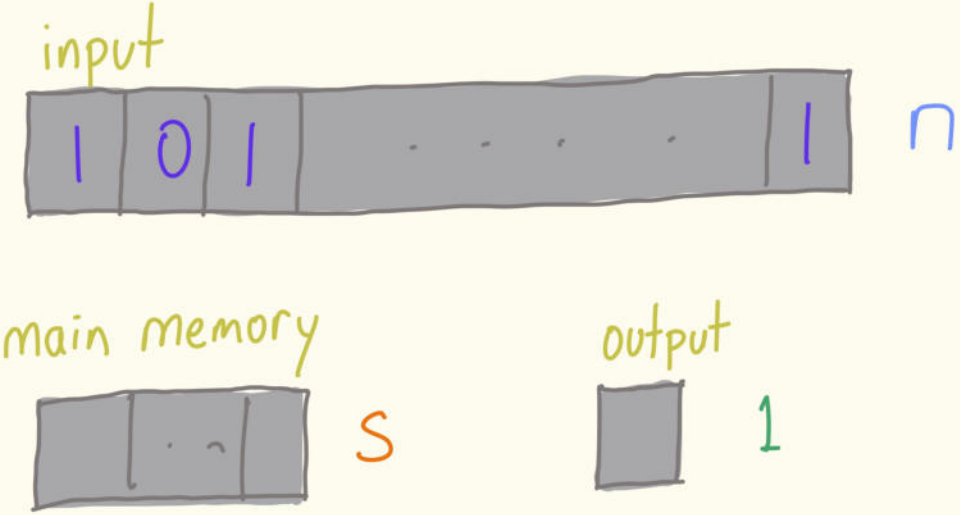


NL
∃ path?

#L
how many paths?

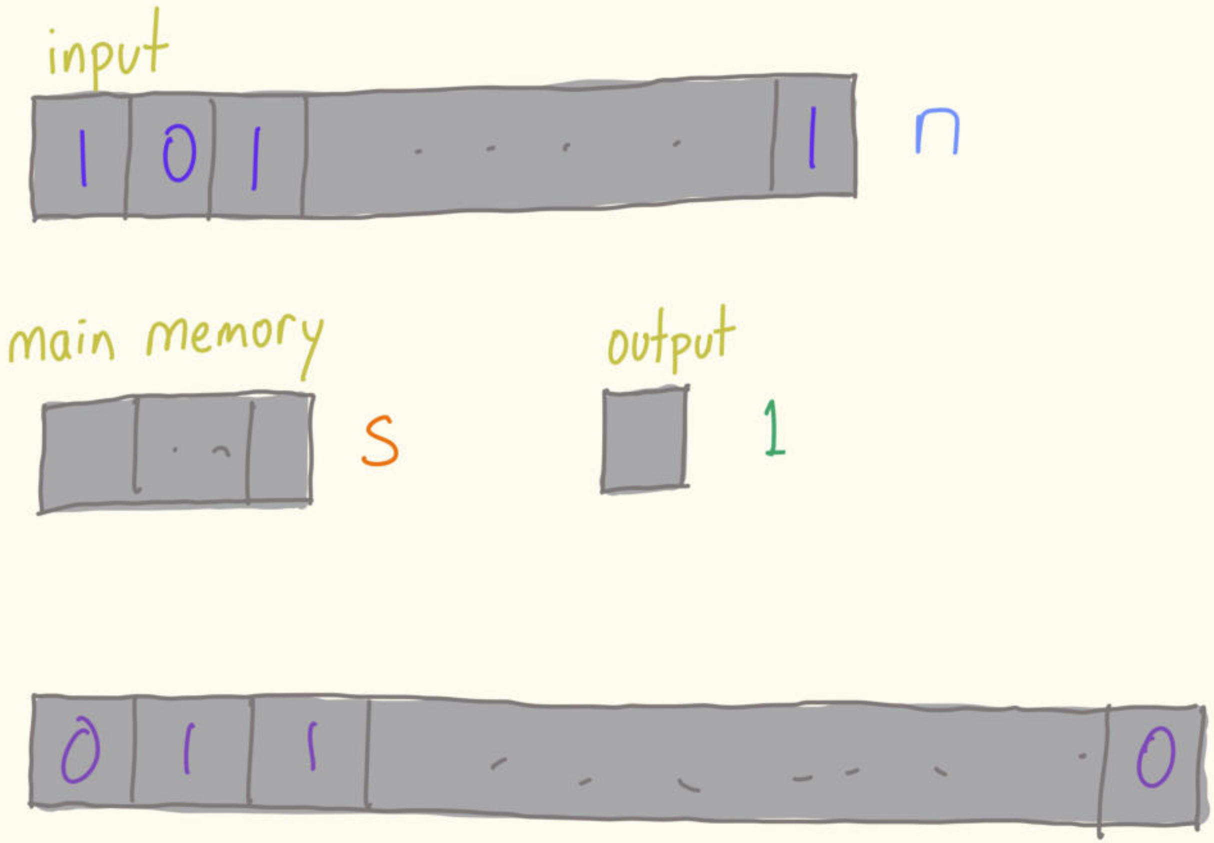
BPL
where does a random path go?

Logspace



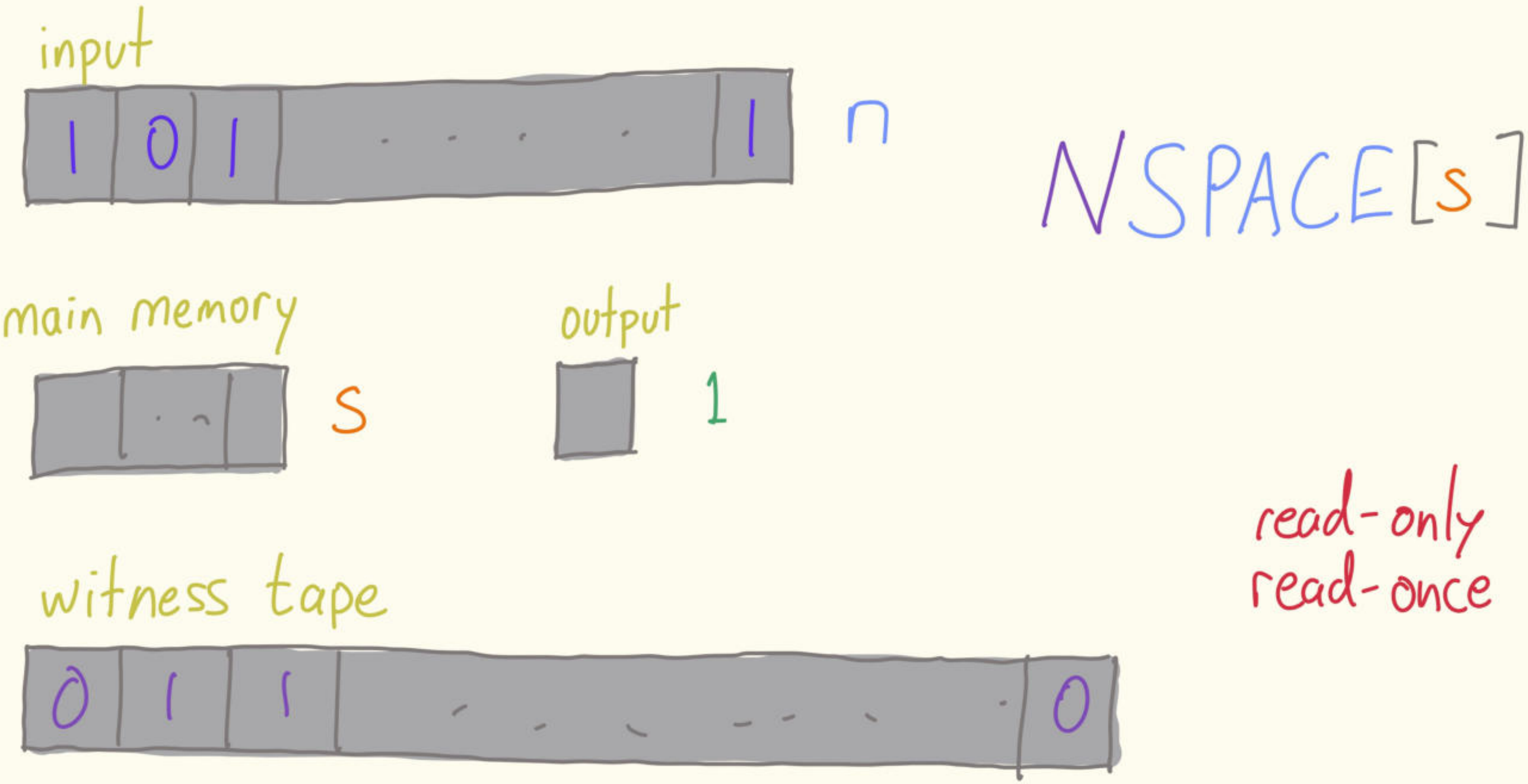
SPACE[s]

LOGSPACE

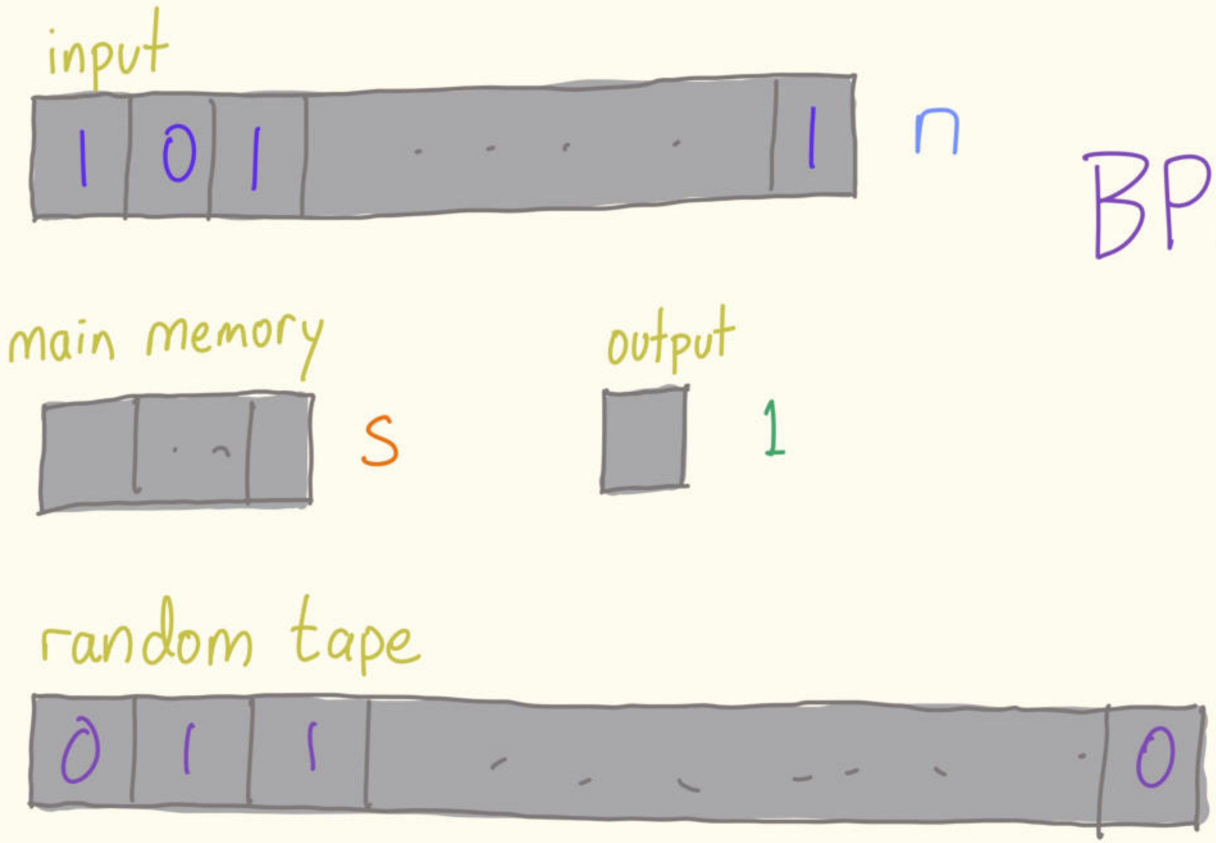


SPACE[s]
+ resources

Logspace


input
1 0 1 1
n
NSPACE[s]
main memory
s
output
1
read-only
read-once
witness tape
0 1 1 0

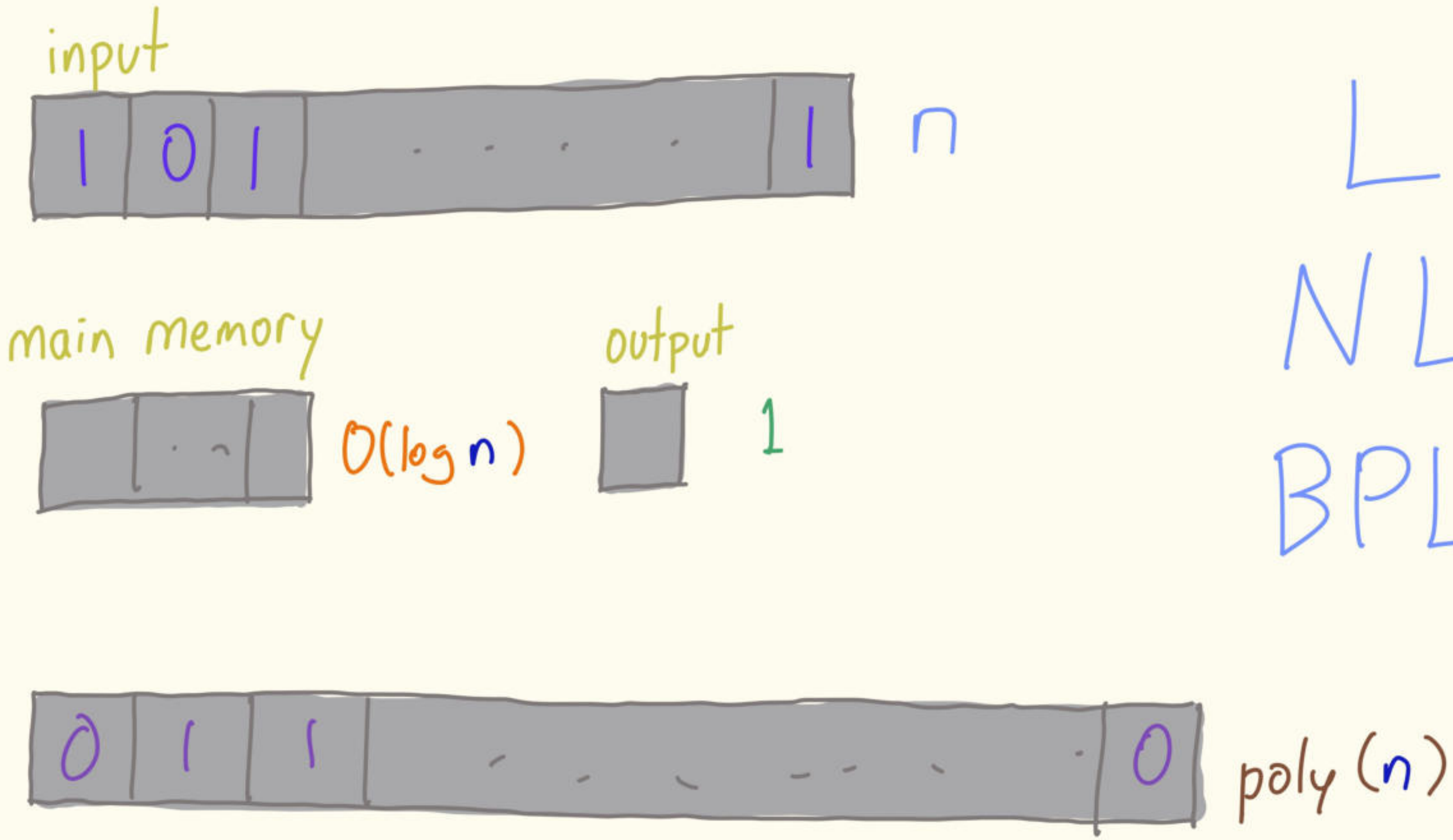
Logspace



BPSPACE[s]

read-only
read-once

LOGSPACE

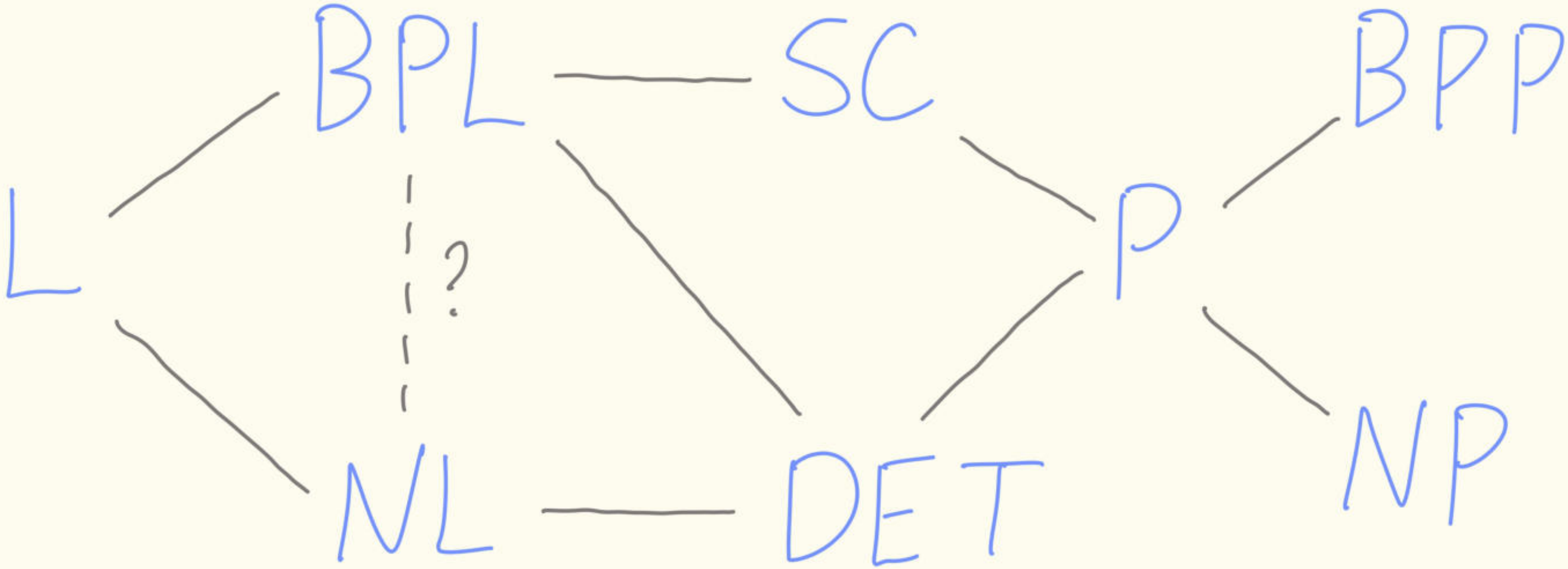


L

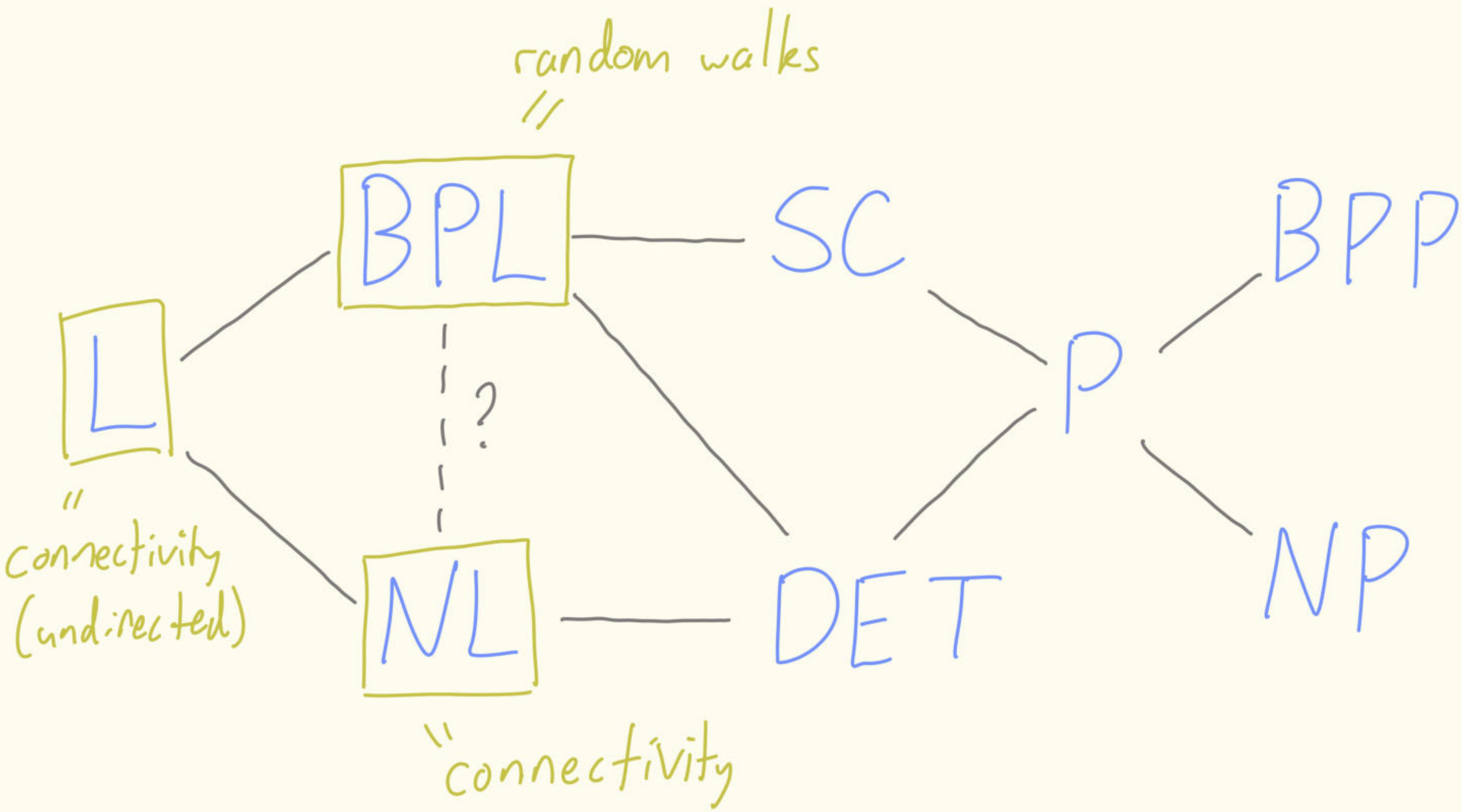
NL

BPL

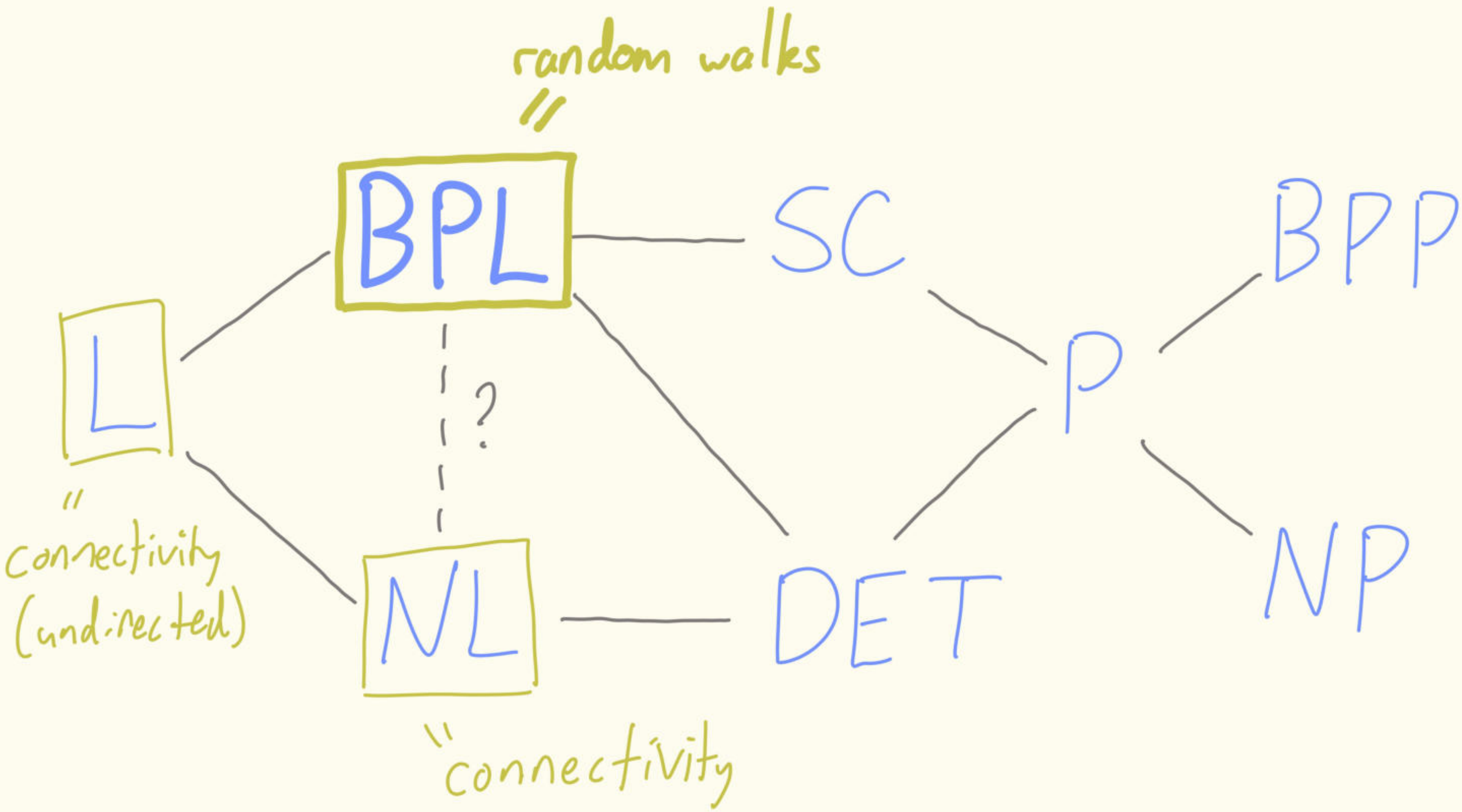
Logspace



Logspace


random walks
BPL
SC
BPP
L
?
P
connectivity
(undirected)
NL
DET
NP
connectivity

Logspace


random walks
"
BPL
SC
BPP
L
?
P
"
connectivity
(undirected)
NL
DET
NP
"
connectivity

Today

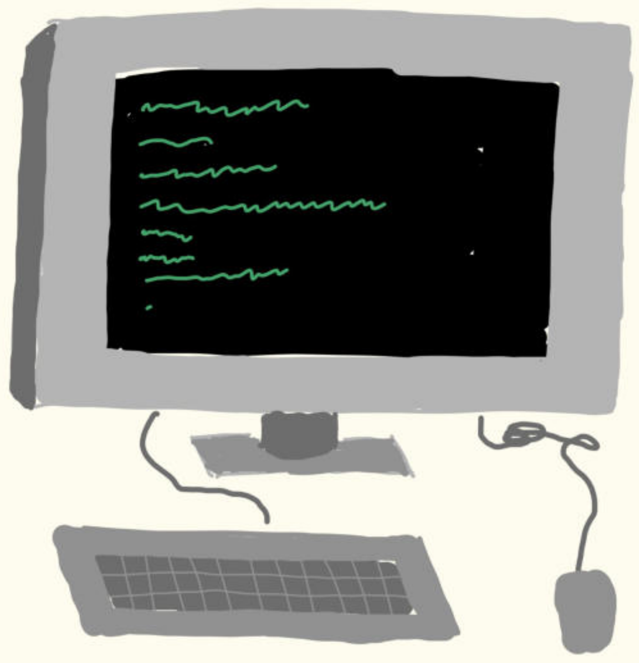
What is...

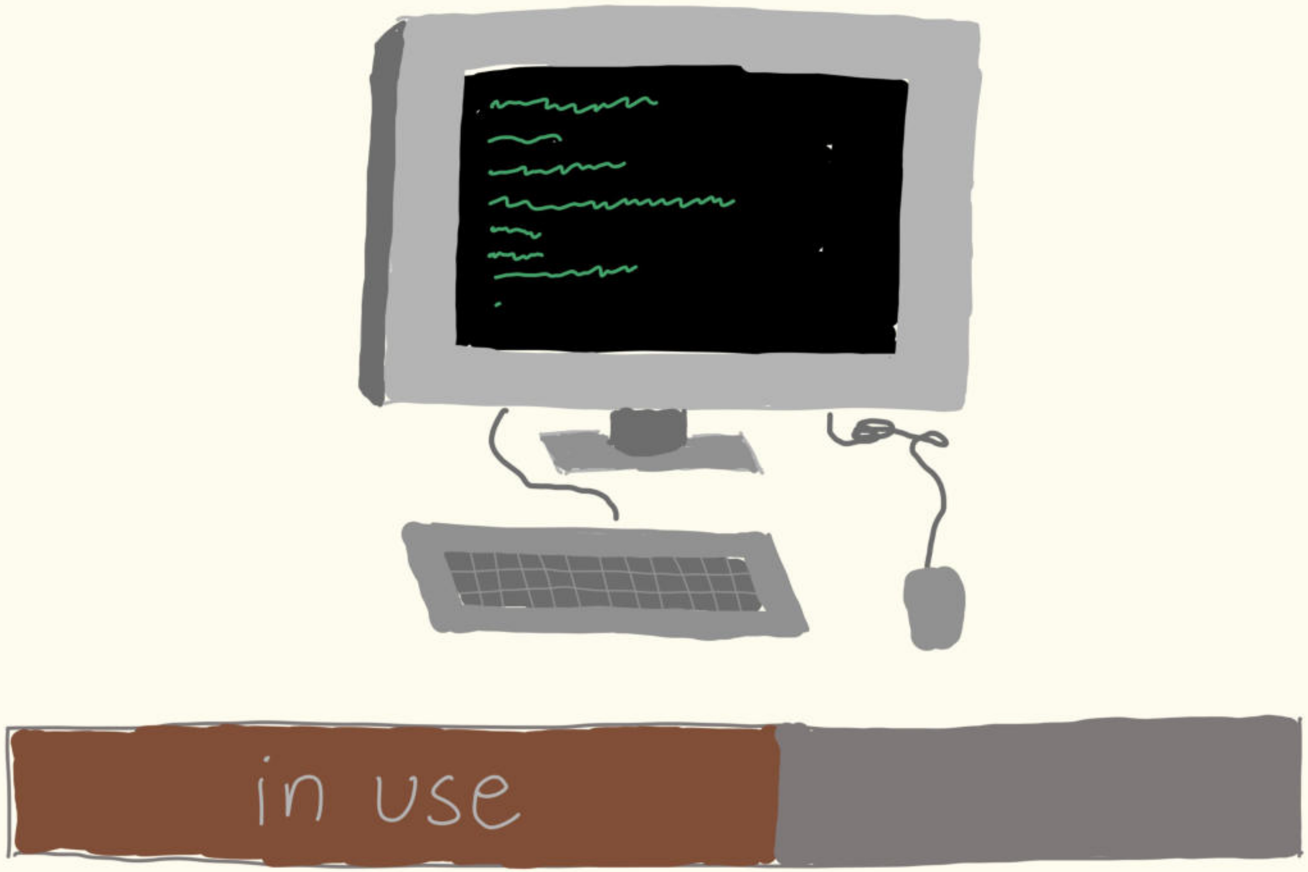
... a Random Walk?

... Full Memory?

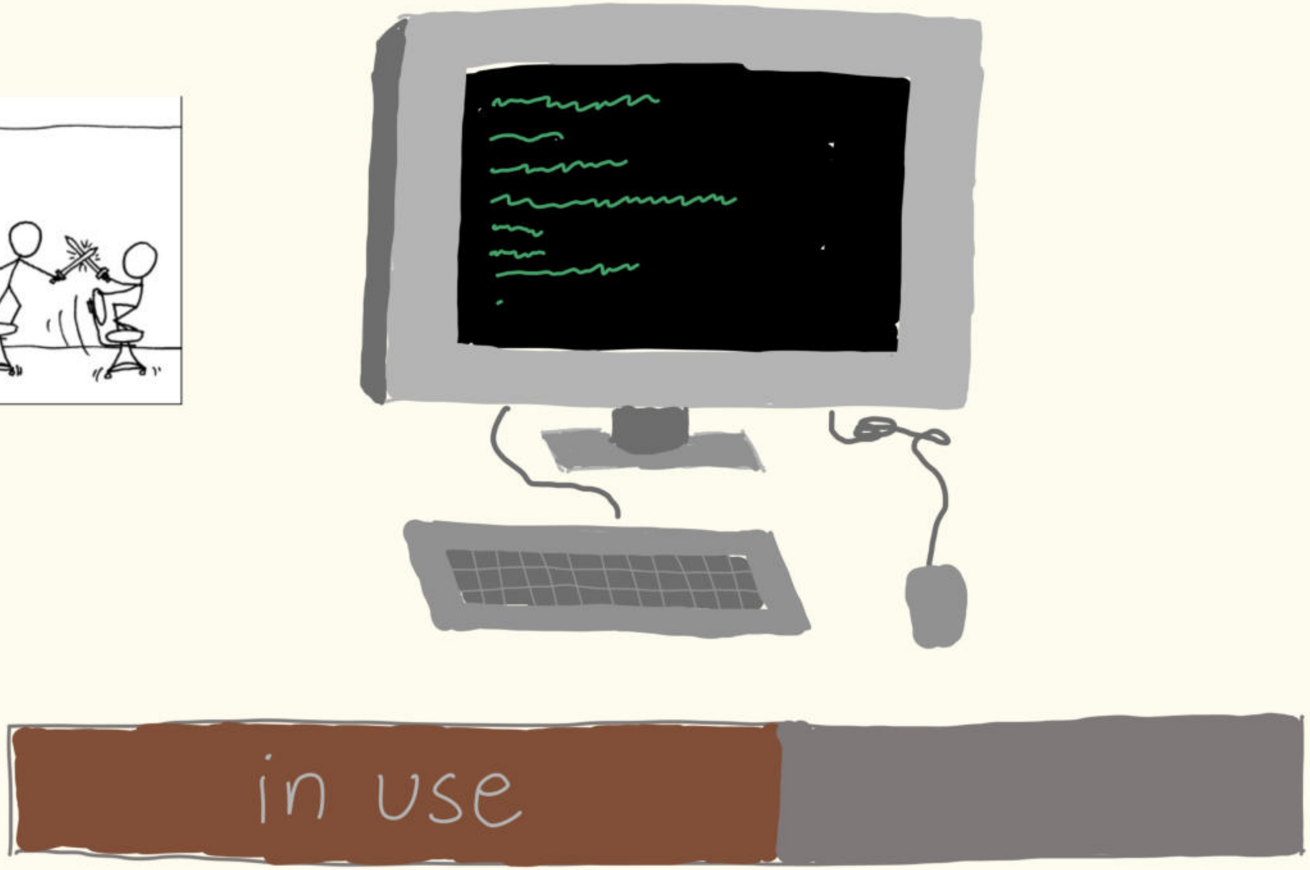
How?

What Next?

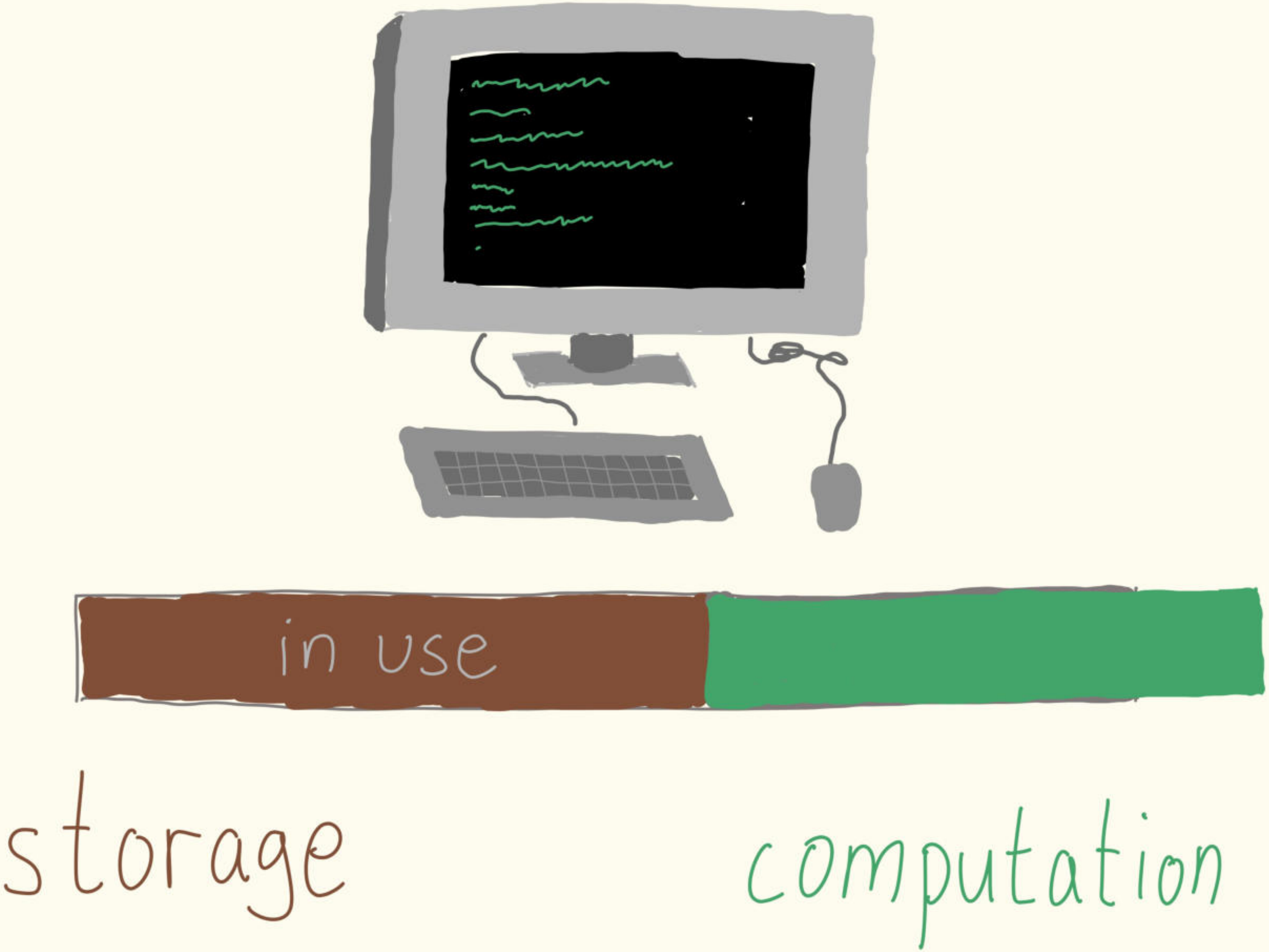


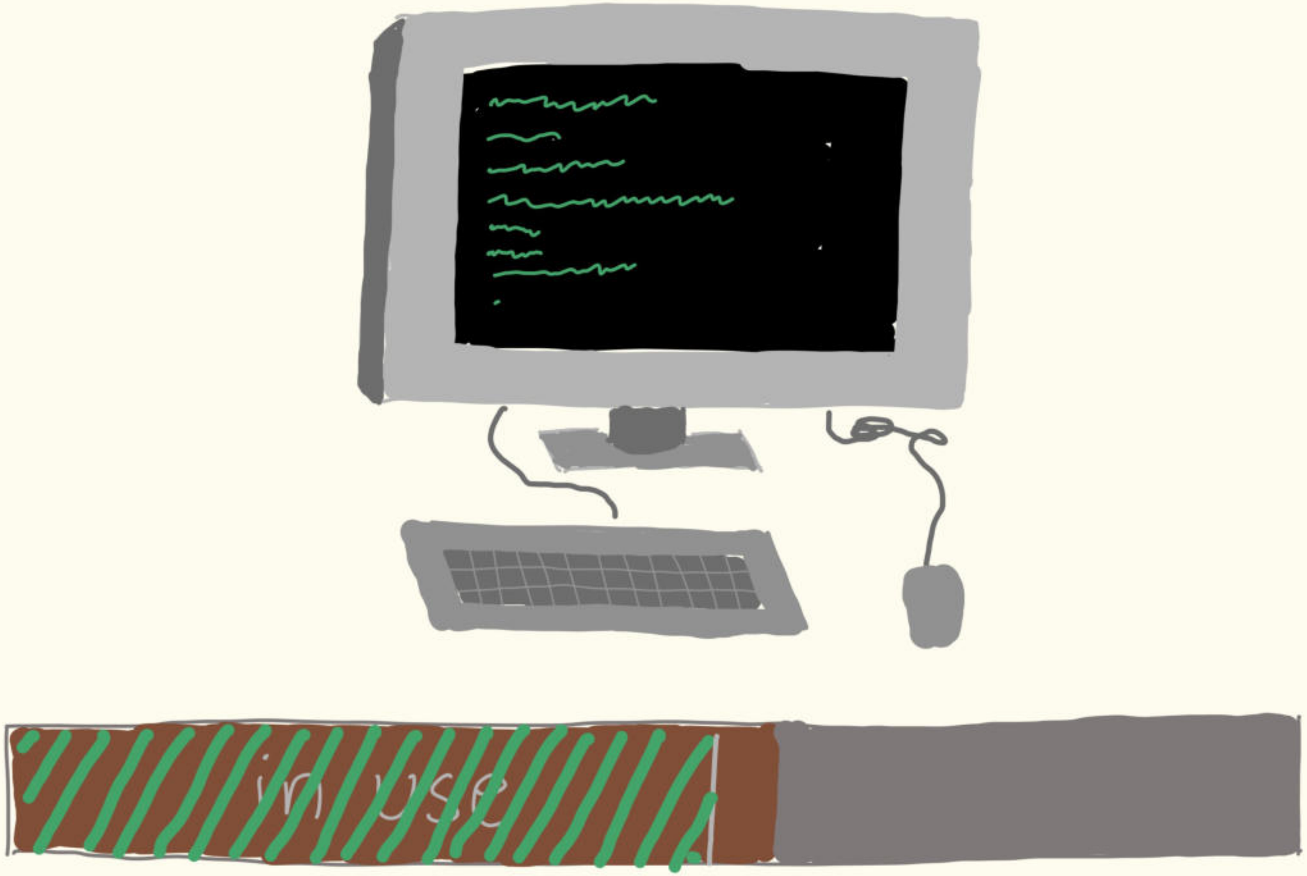

in use


HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.


in use

the next day...

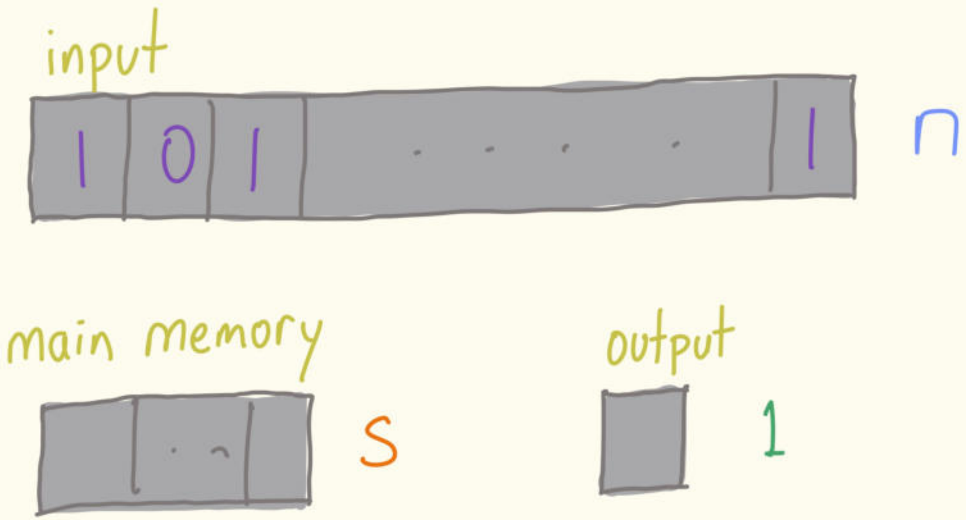

in use
storage
computation



storage + computation

The Study of Reuse

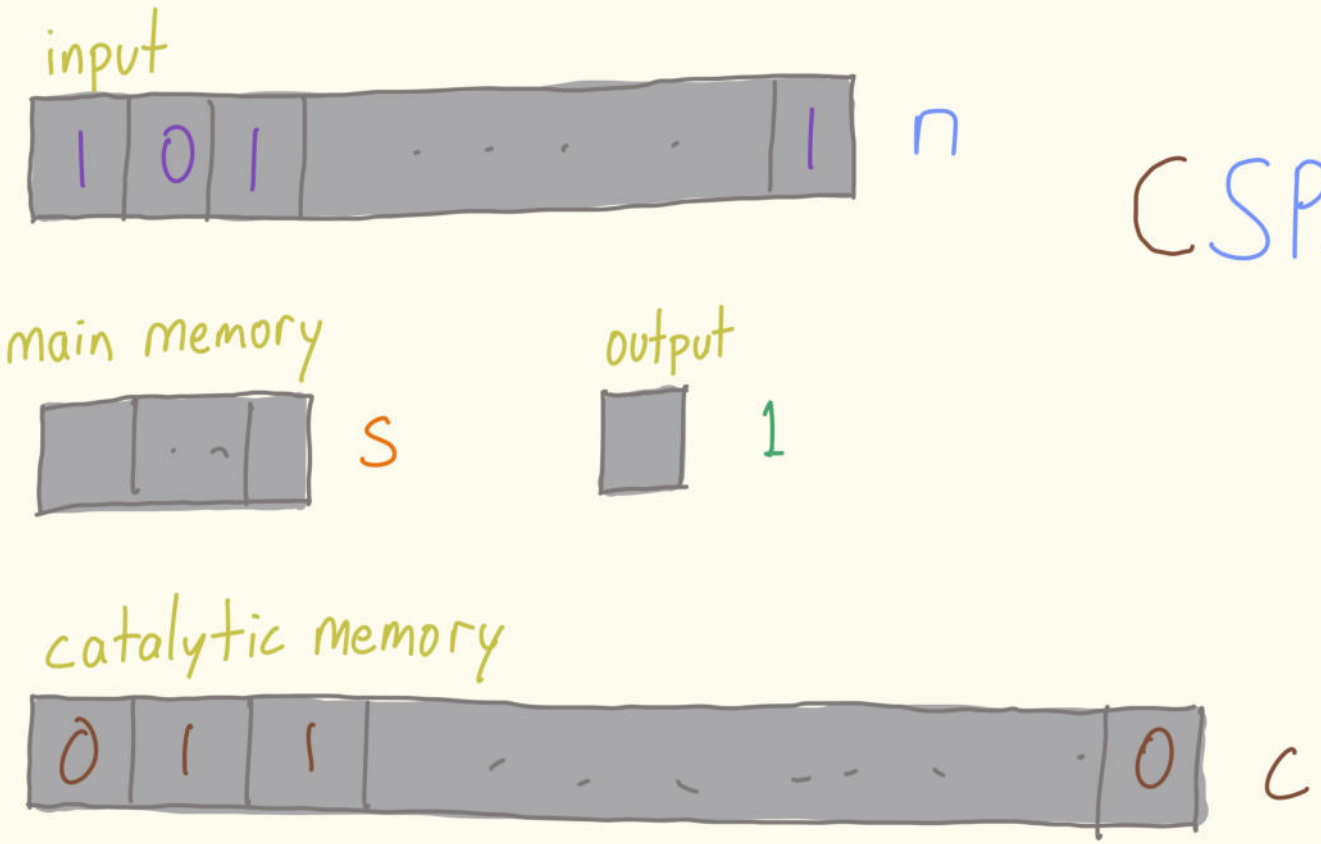
catalytic computing [BCKLS'14]



SPACE[s]

The Study of Reuse

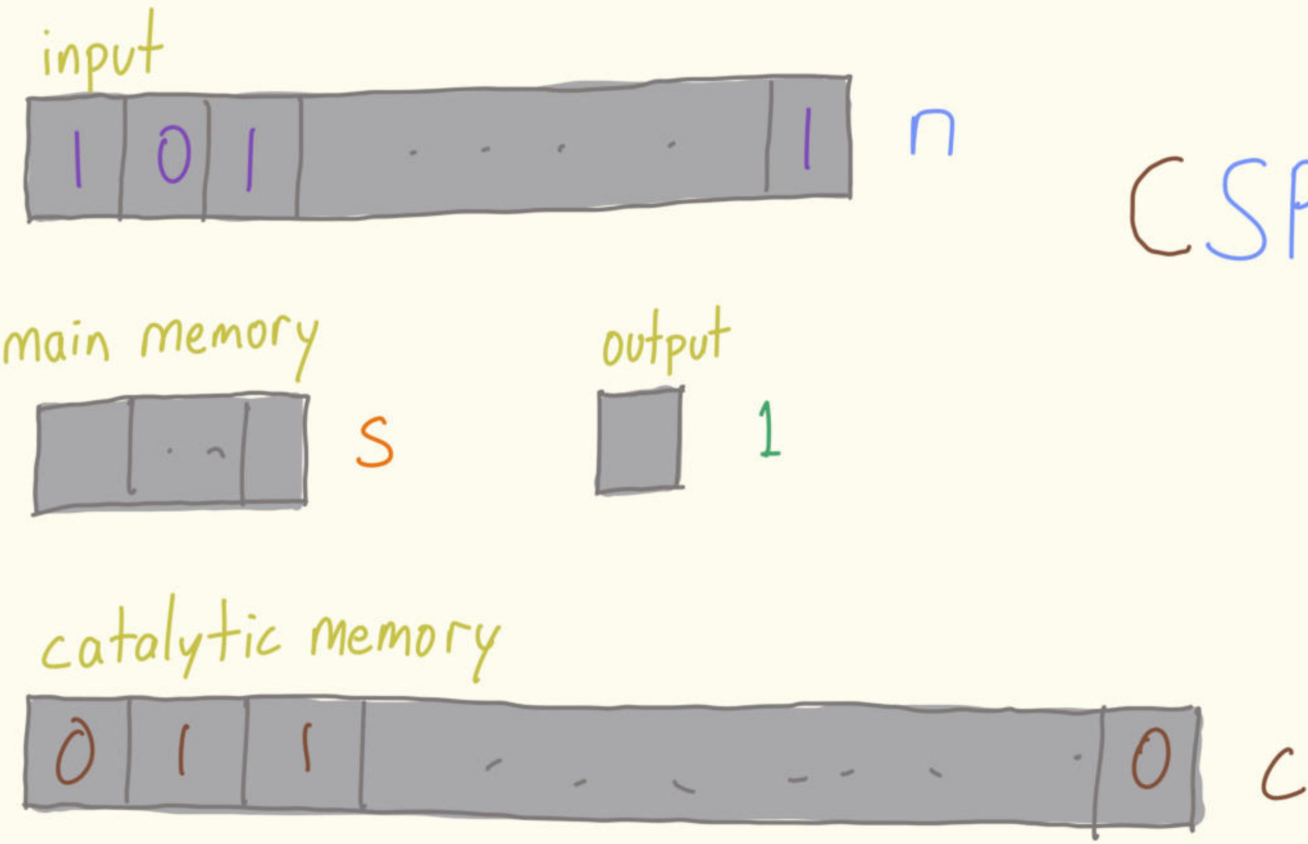
catalytic computing [BCKLS'14]



CSPACE[s, c]

The Study of Reuse

catalytic computing [BCKLS'14]

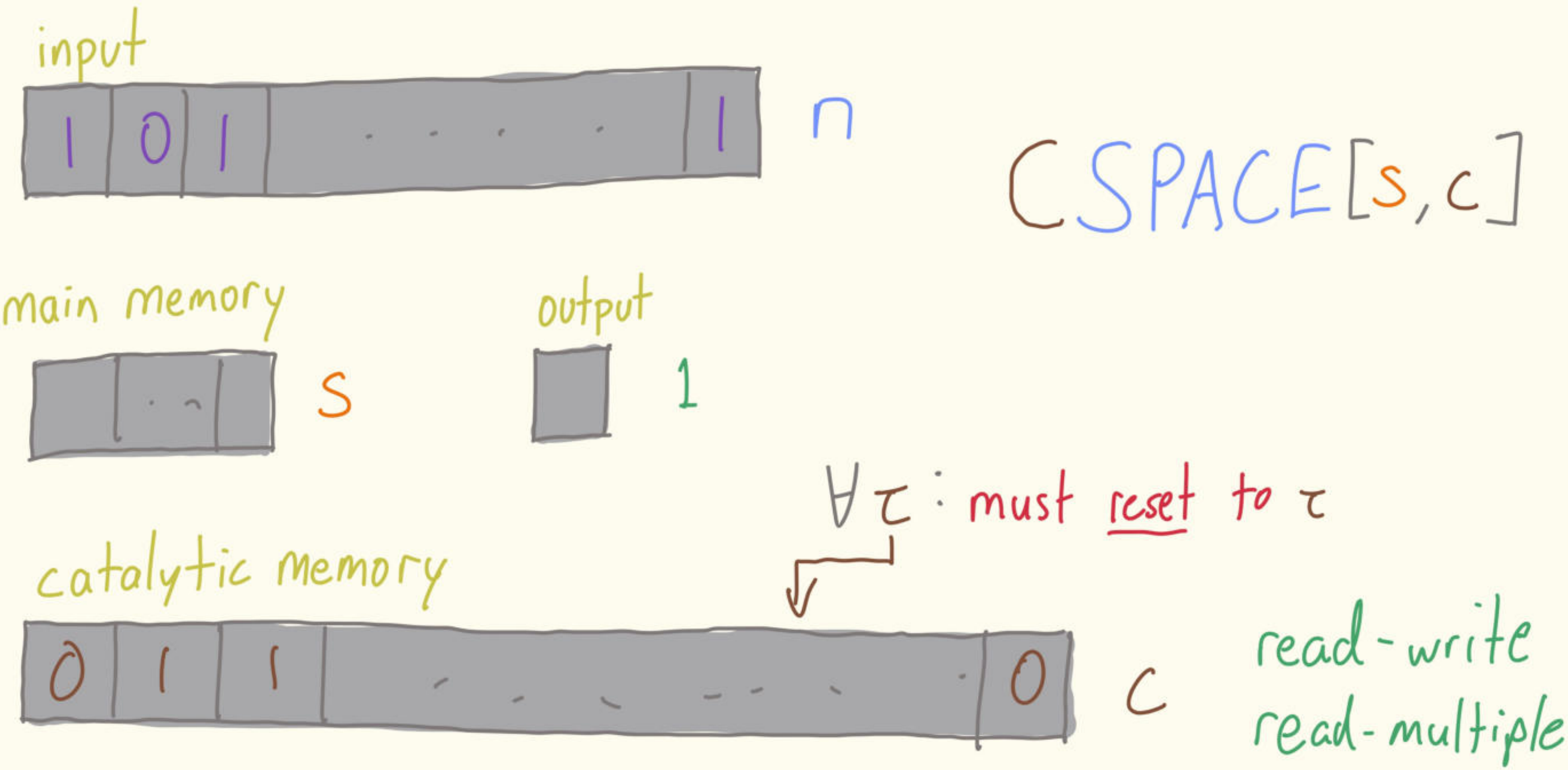


CSPACE[s, c]

read-write
read-multiple

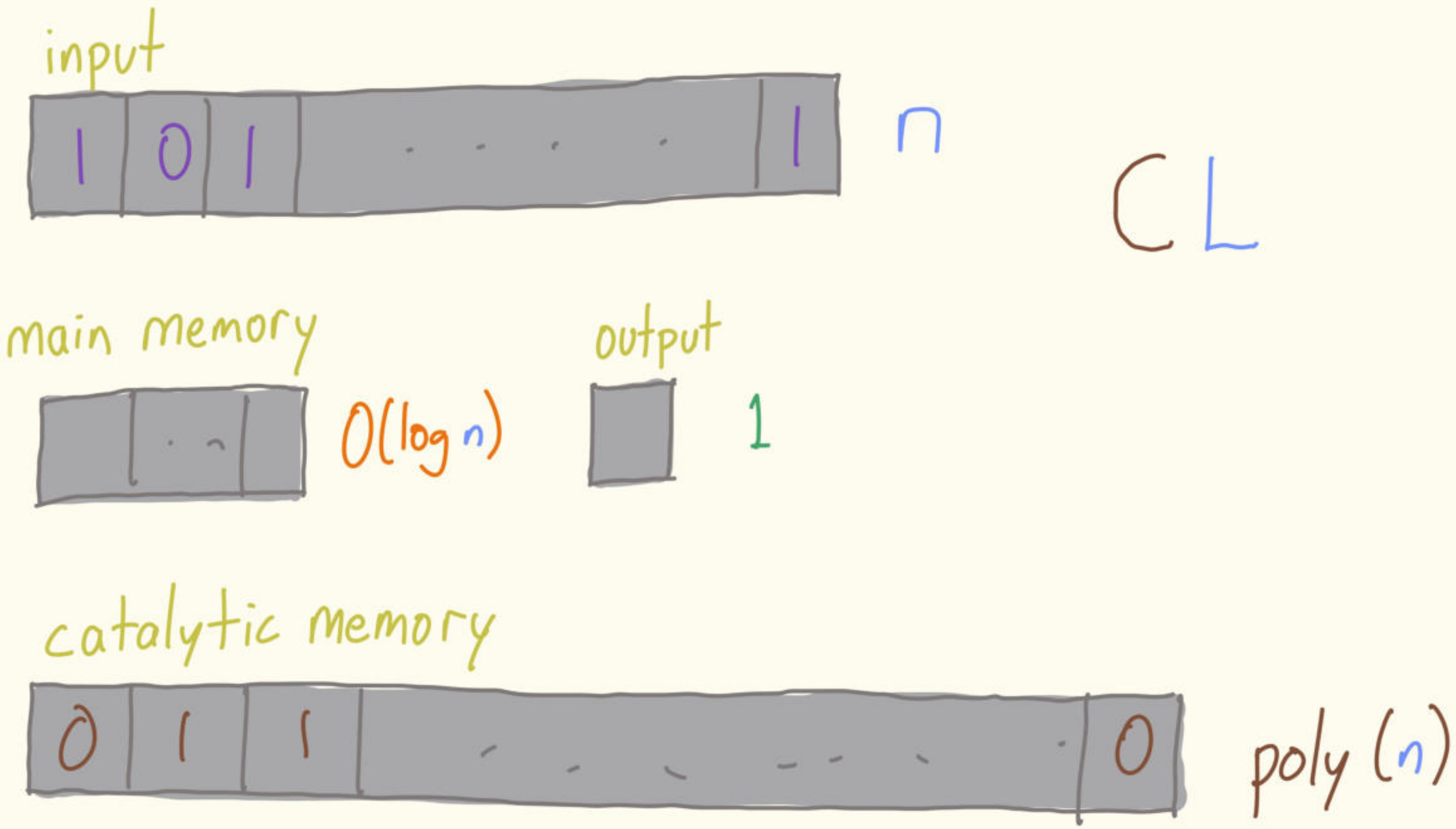
The Study of Reuse

catalytic computing [BCKLS'14]



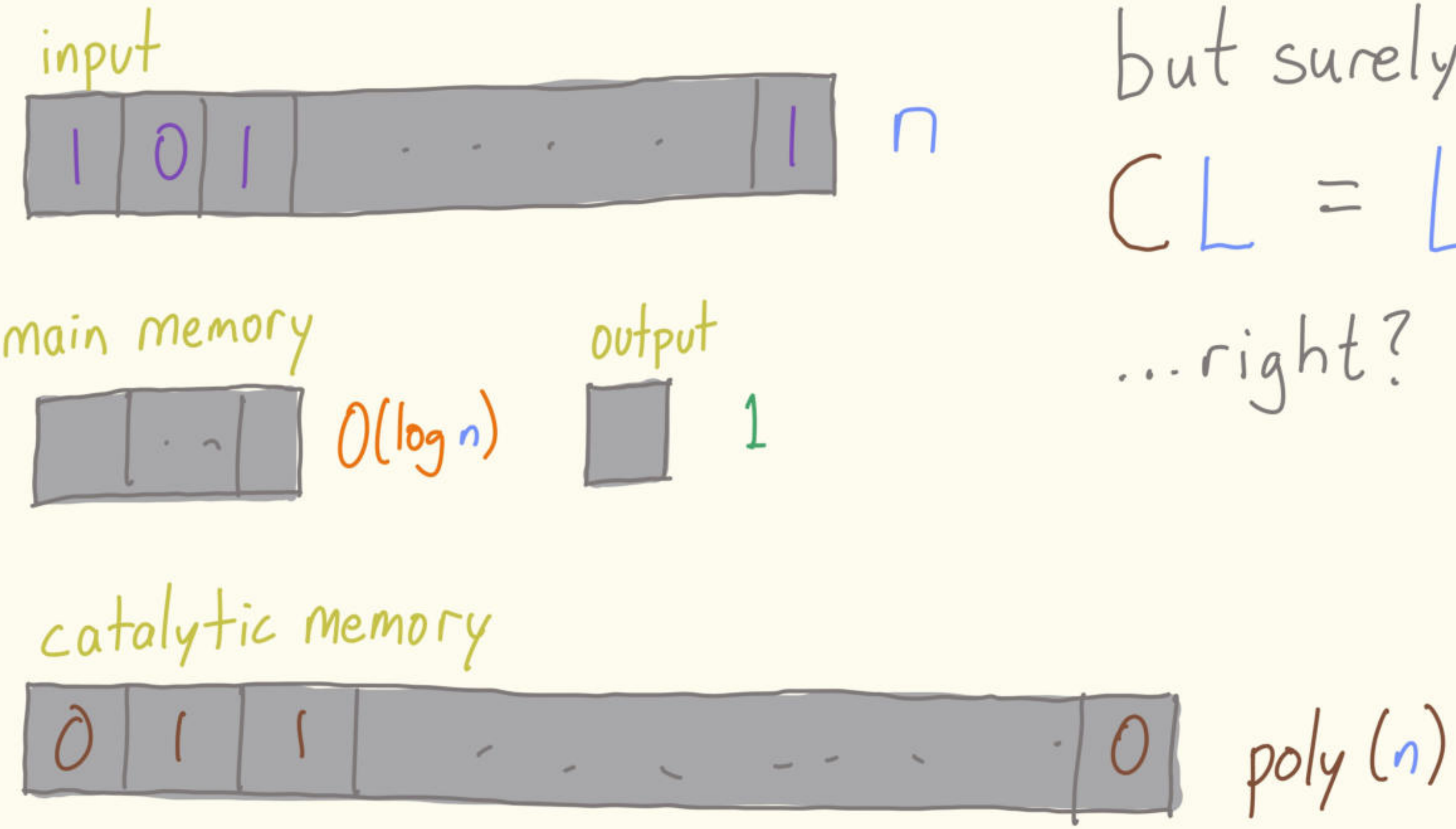
The Study of Reuse

catalytic computing [BCKLS'14]



The Study of Reuse

catalytic computing [BCKLS'14]



but surely

$CL = L$

...right?

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

[BCKLS'14]:

$$CL$$

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

[BCKLS'14]:

$$CL \subseteq ZPP$$

probably $\subseteq P$

The Power of Reuse

trivial:

$$\mathsf{L} \subseteq \mathsf{CL} \subseteq \mathsf{PSPACE}$$

[BCKLS'14]:

$$\mathsf{TC}^1 \subseteq \mathsf{CL} \subseteq \mathsf{ZPP}$$

contains NL, BPL, DET, etc.

probably $\subseteq$ P

The Power of Reuse

most likely:

$$\mathsf{L} \subsetneq \mathsf{CL} \subsetneq \mathsf{PSPACE}$$

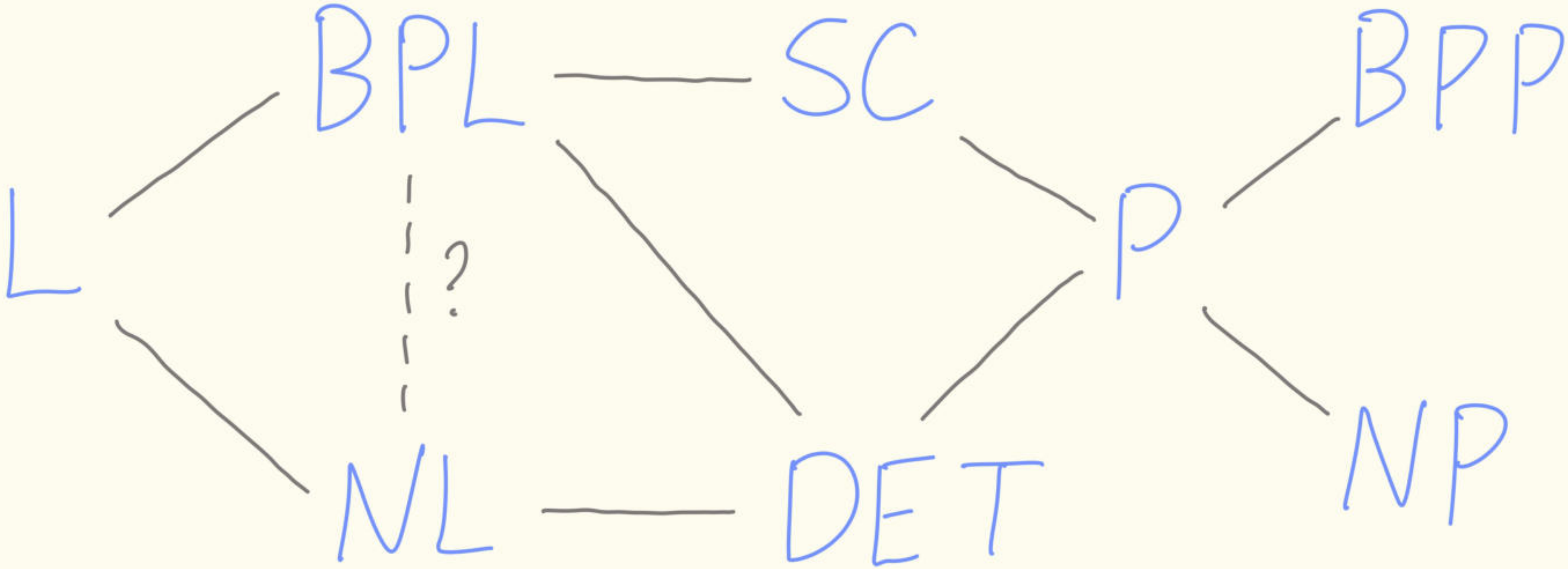
[BCKLS'14]:

$$\mathsf{TC}^1 \subseteq \mathsf{CL} \subseteq \mathsf{ZPP}$$

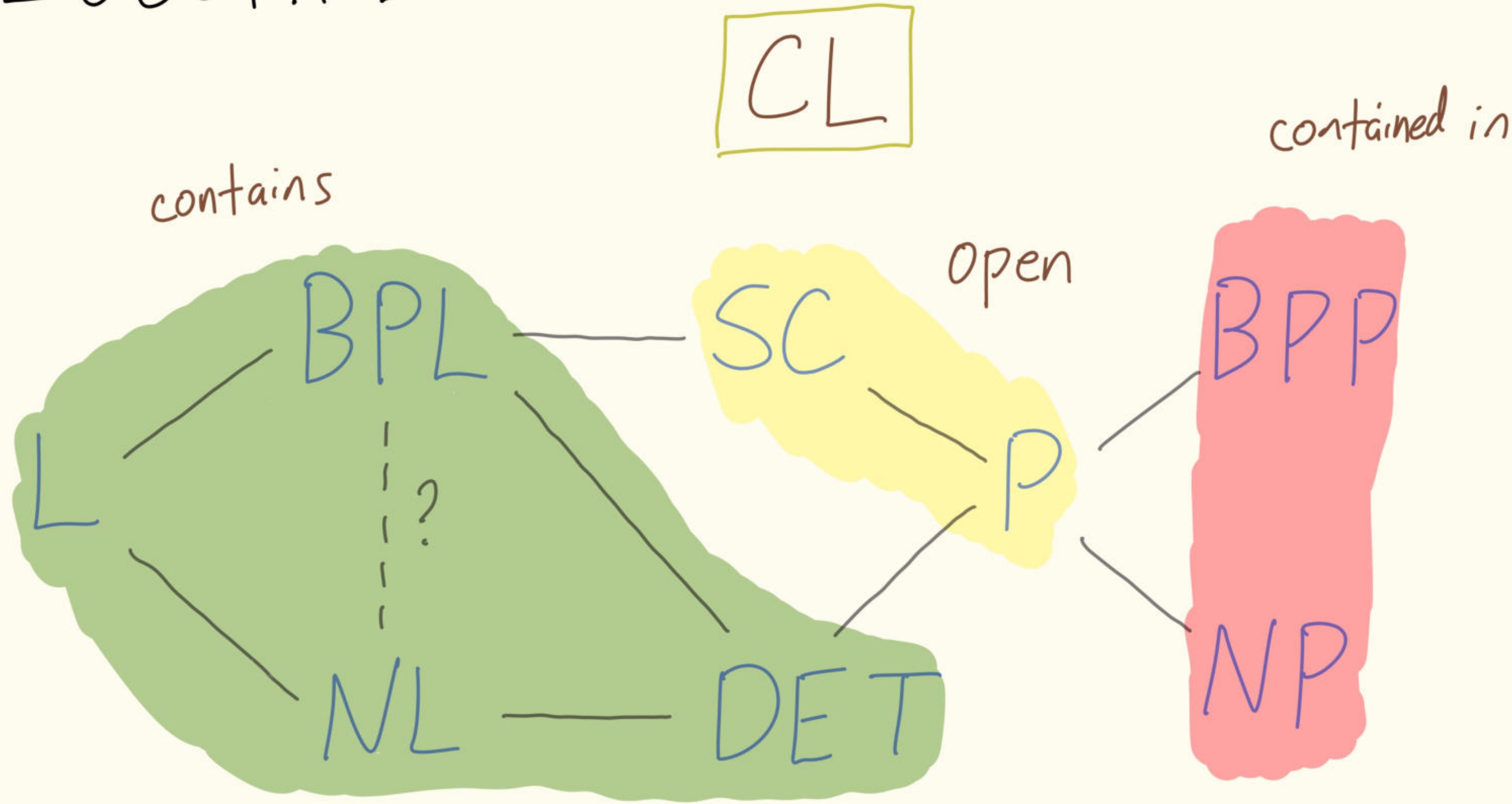
contains NL, BPL, DET, etc.

probably $\subseteq$ P

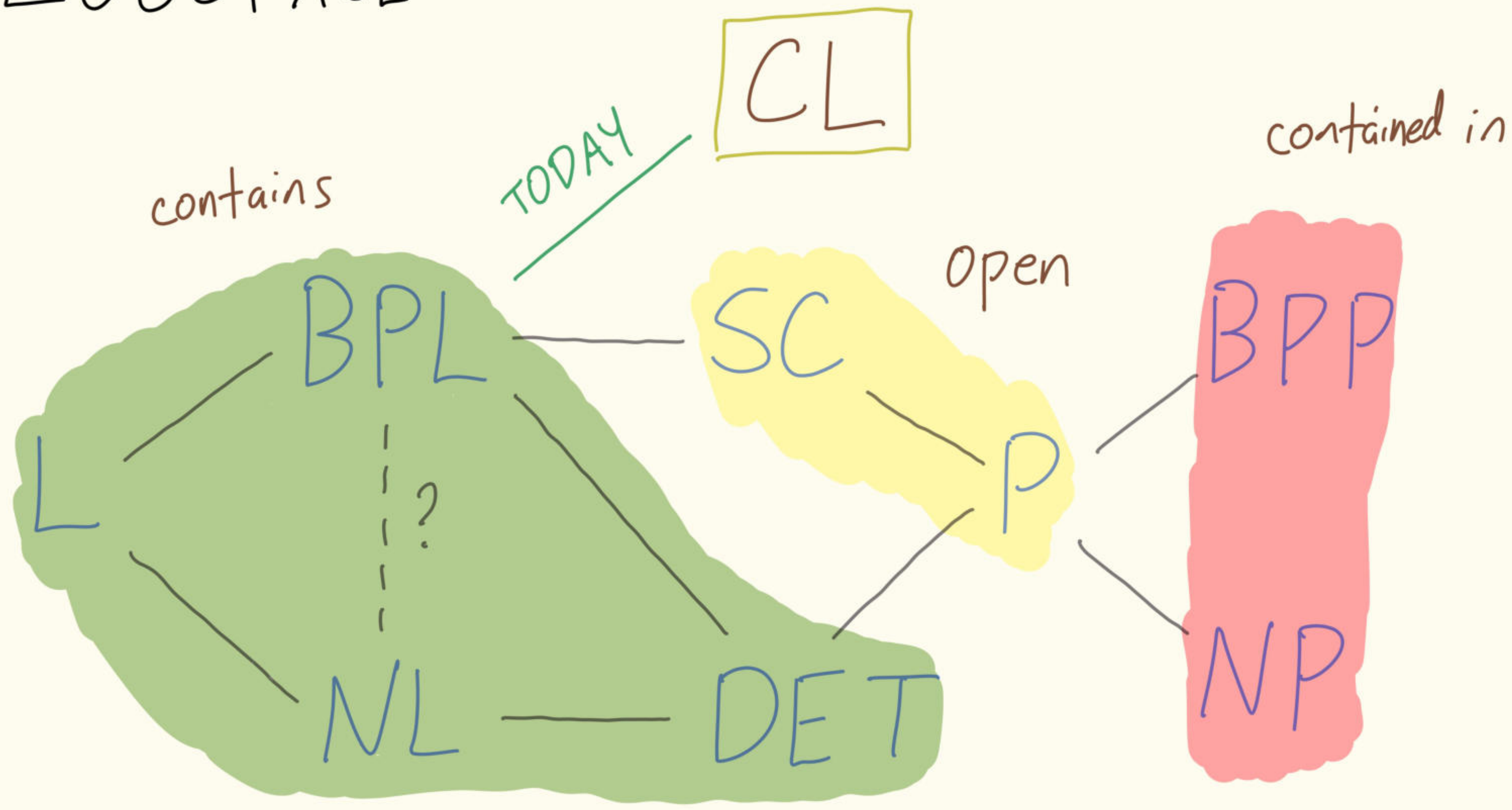
Logspace



Logspace



Logspace



Today

What is . . .

. . . a Random Walk?

. . . Full Memory?

How?

What Next?

TODAY

WHAT IS . . .

. . . A RANDOM WALK ?

. . . FULL MEMORY ?

How?

[Dul`15] [BCKLS`14] [CP`25]

WHAT NEXT ?

TODAY

WHAT IS . . .

. . . A RANDOM WALK? . . . FULL MEMORY?

HOW?

[DUl'15] [BCKLS'14] [CP'25]

COMPRESSION

WHAT NEXT?

Today

What is . . .

. . . a Random Walk?

. . . Full Memory?

How?

[Dul'15]

Compression

[BCKLS'14]

Arithmetic Reversibility

[CP'25]

What Next?

Today

What is . . .

. . . a Random Walk? . . . Full Memory?

How?

[Dul'15] [BCKLS'14] [CP'25]

Compression | Arithmetic Reversibility | The Easy Way

What Next?

Today

What is . . .

. . . a Random Walk? . . . Full Memory?

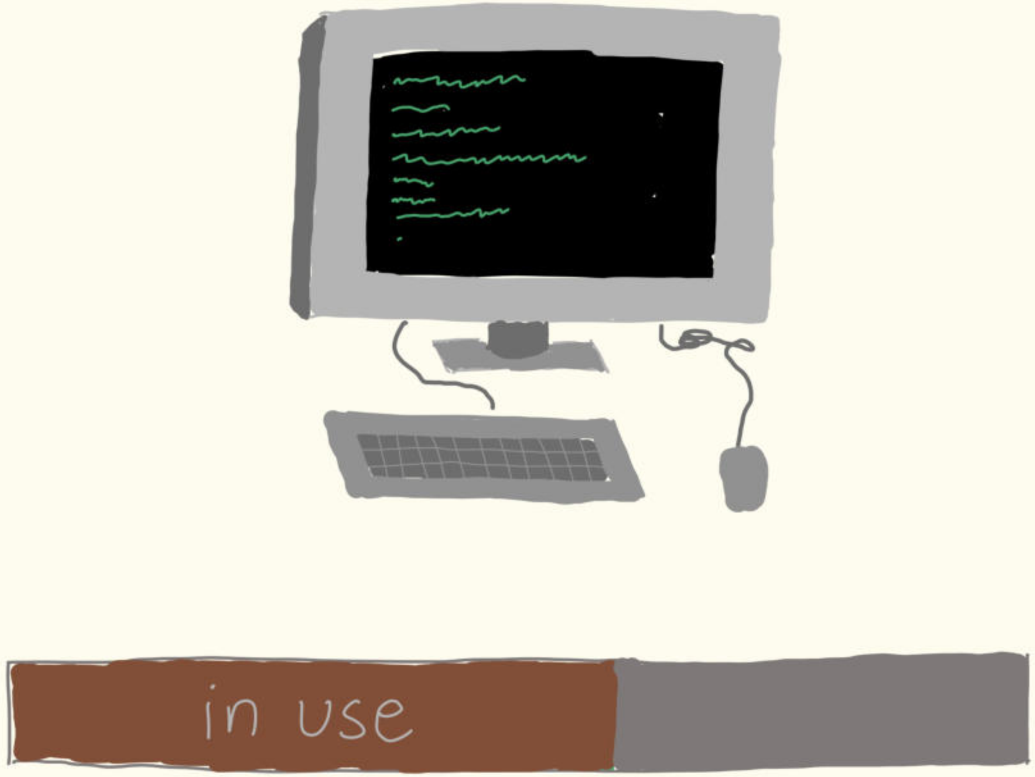
How?

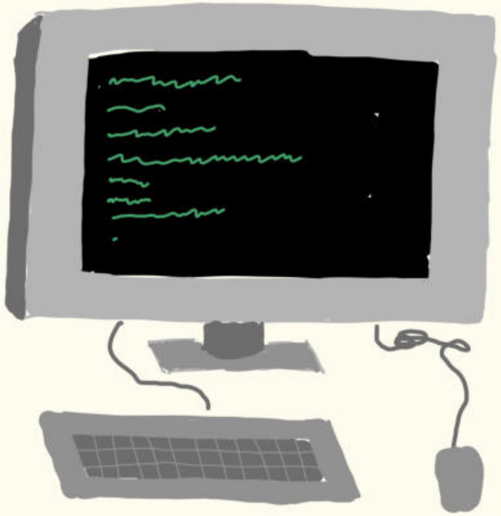
[DuI'15]
Compression

[BCKLS'14]
Arithmetic Reversibility

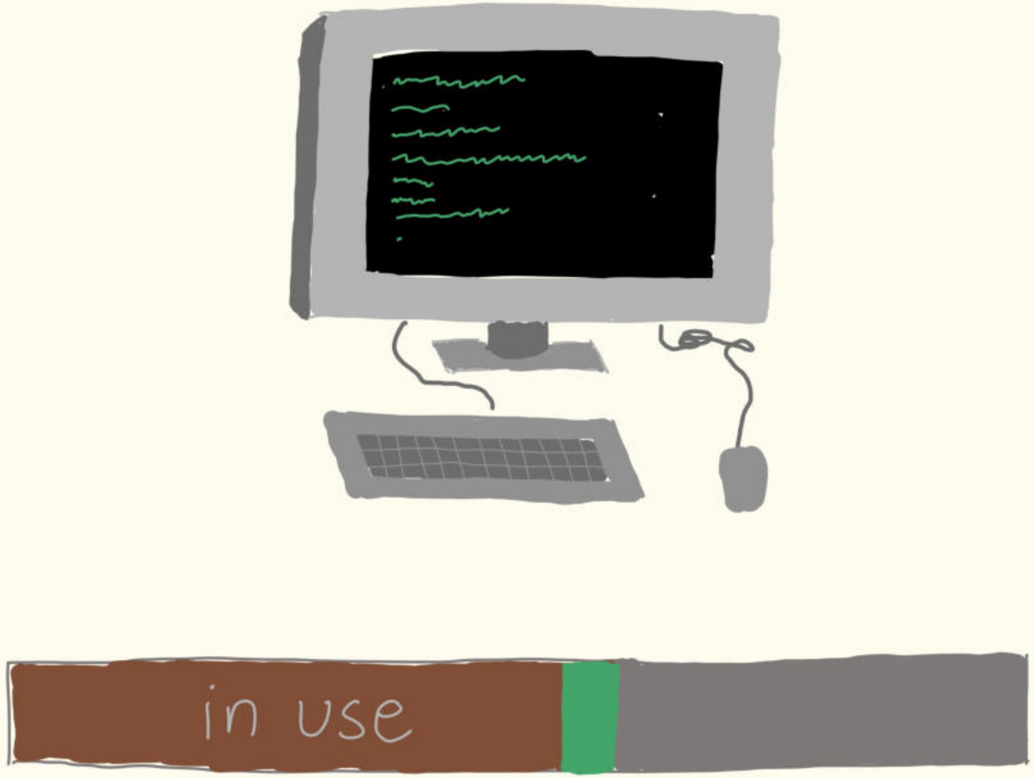
[CP'25]
The Easy Way

What Next?

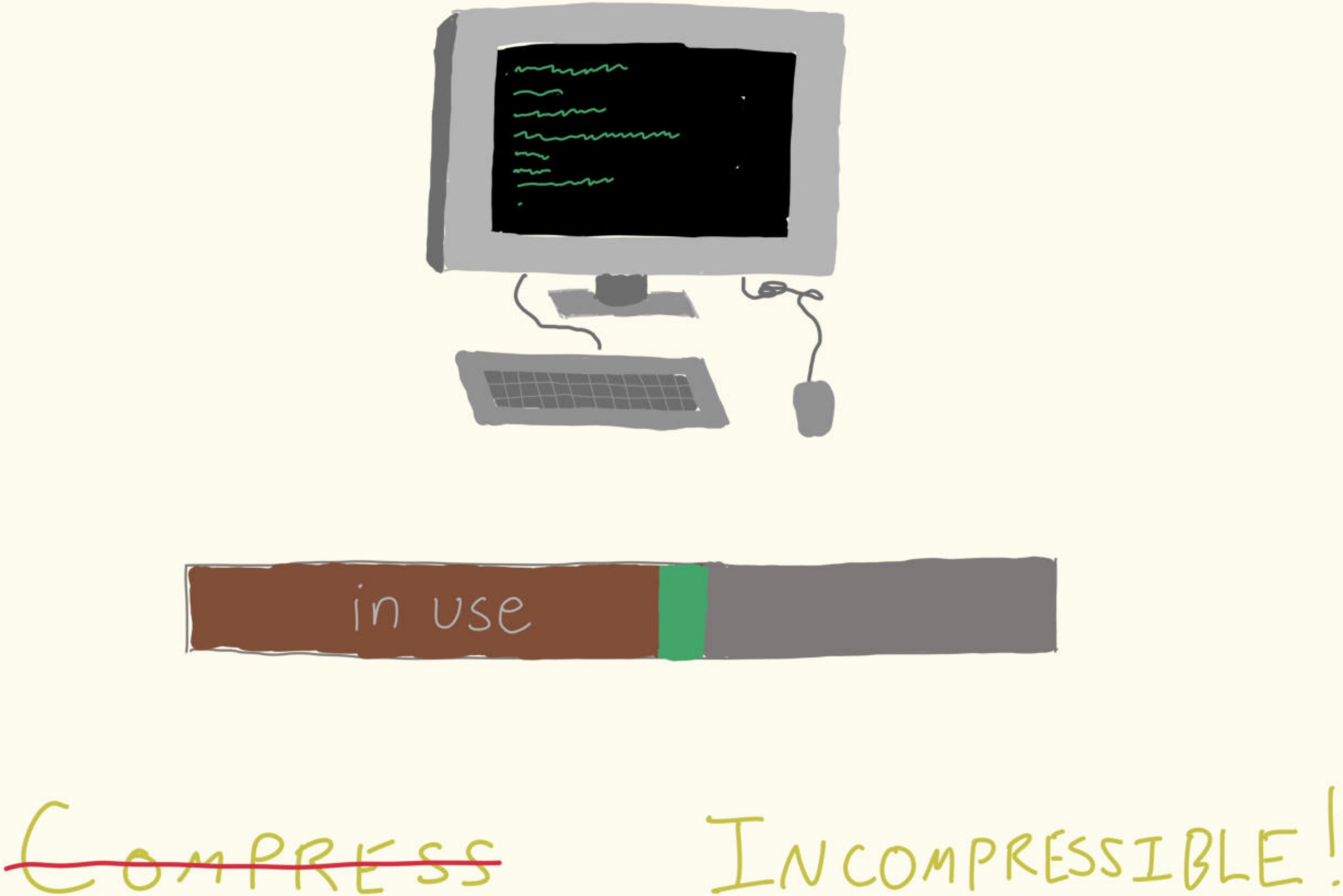

in use



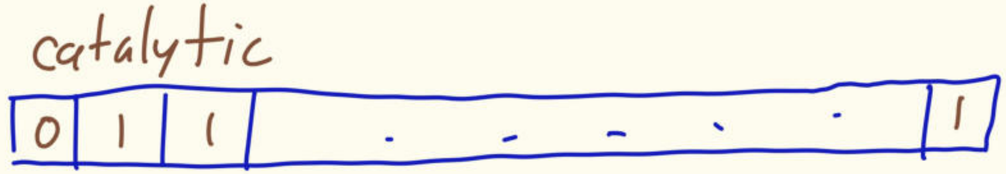
Compress?



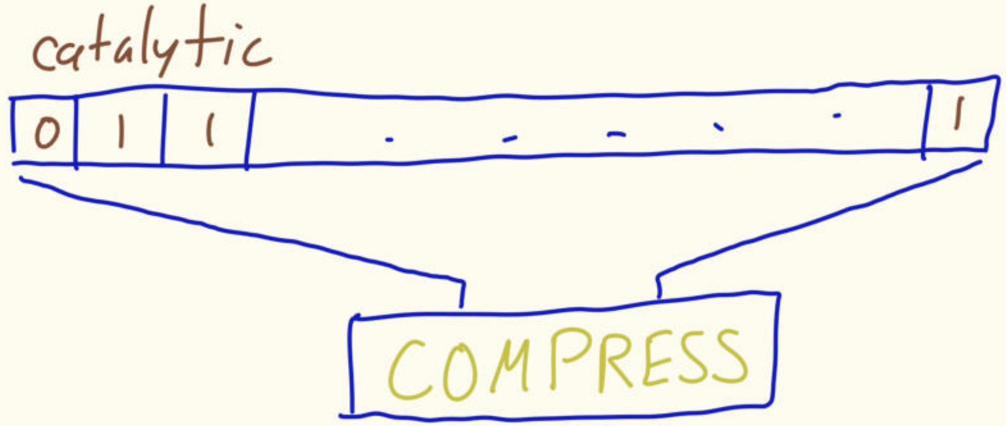
~~COMPRESS~~


in use
~~Compress~~
Incompressible!

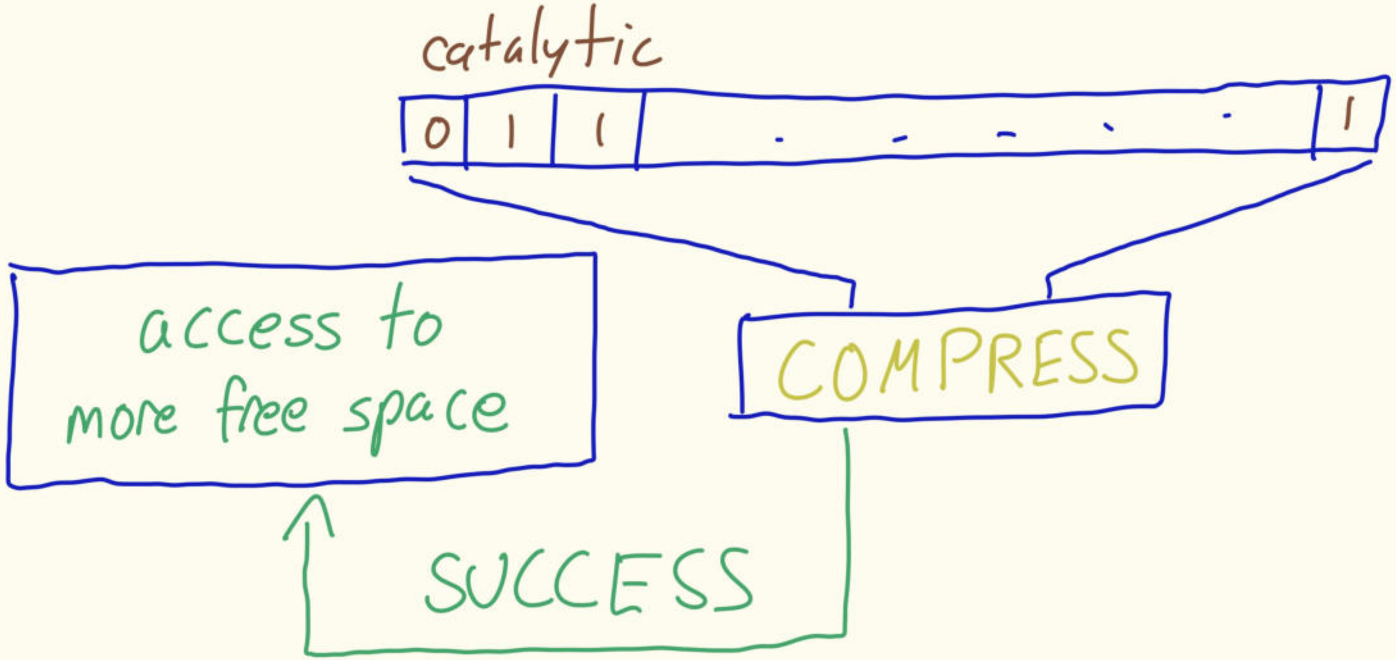
Compress-or-Random



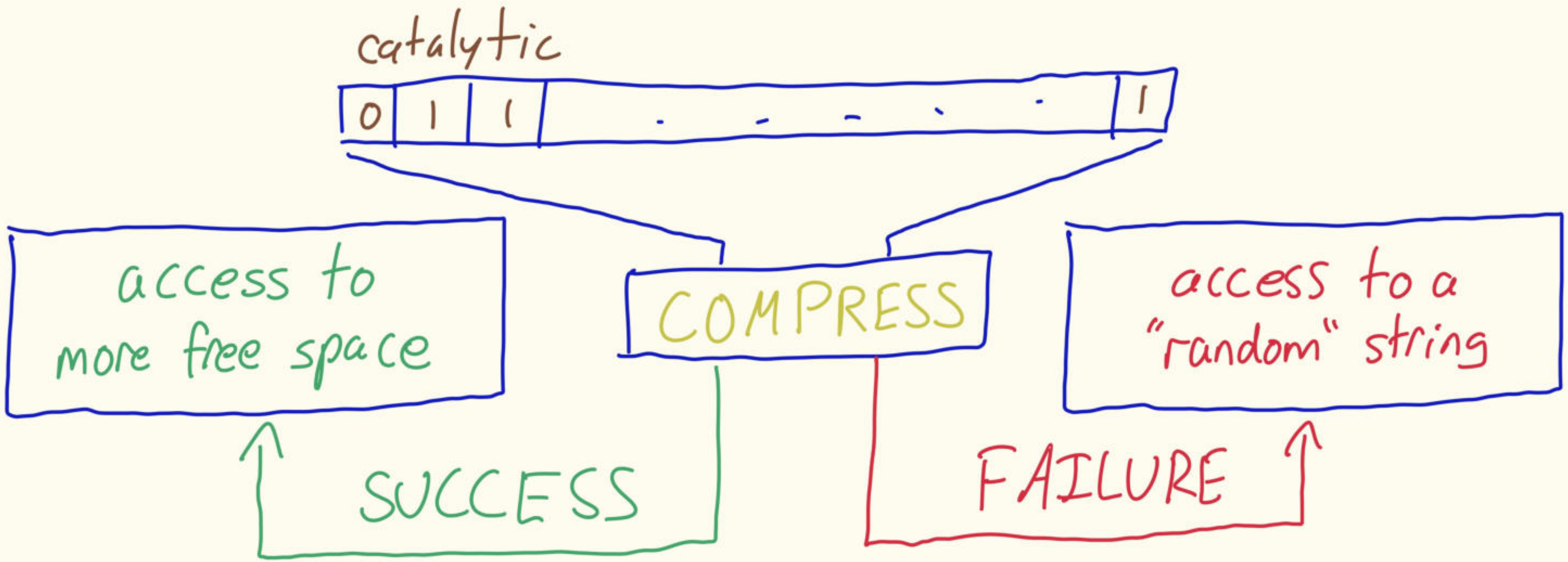
Compress-or-Random



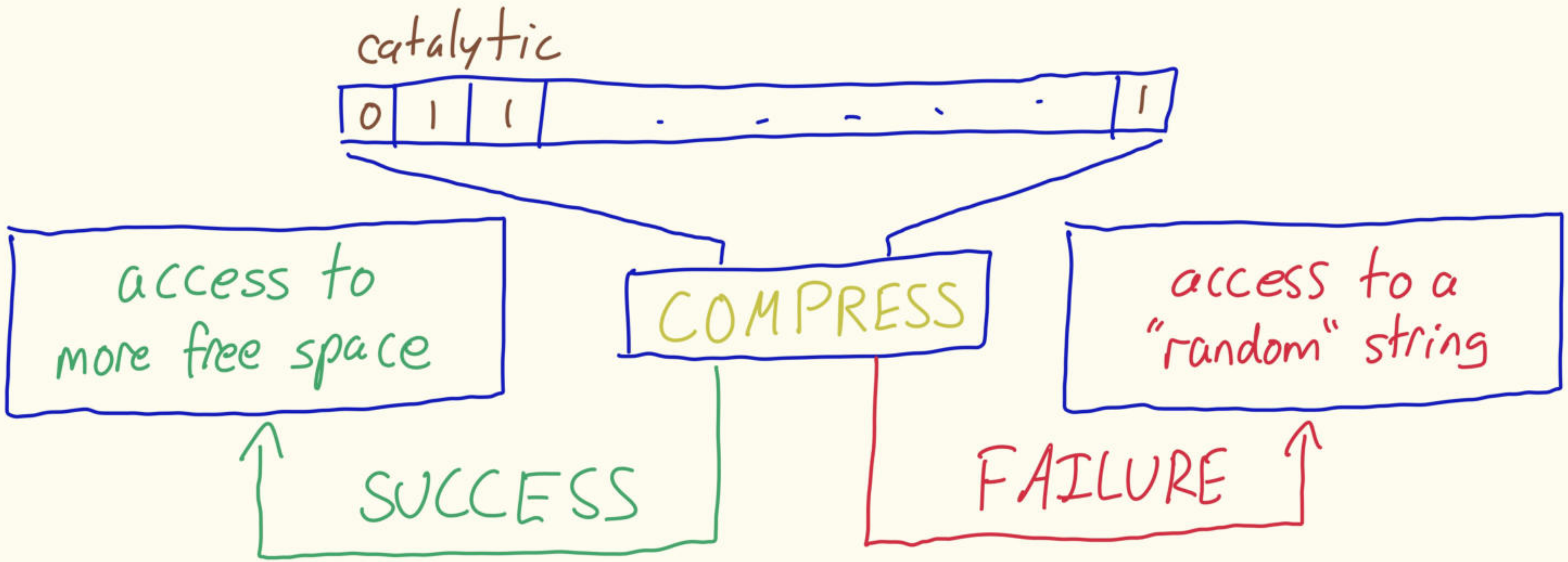
COMPRESS-OR-RANDOM



Compress-or-Random



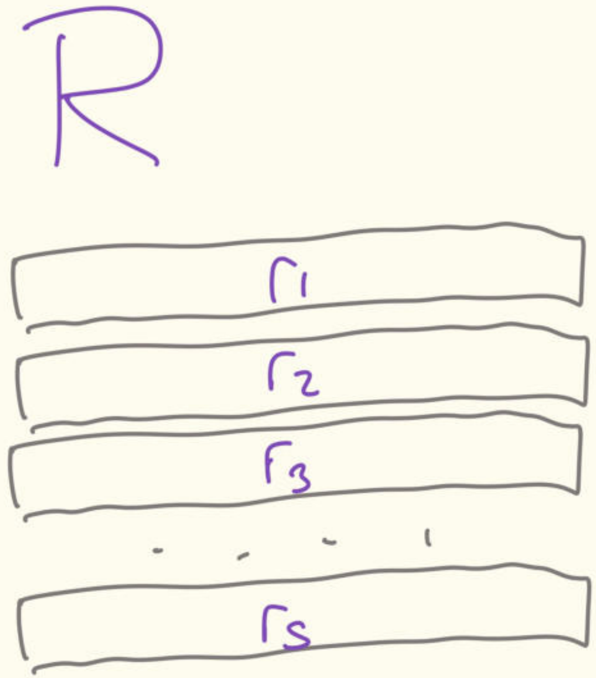
Compress-or-Random

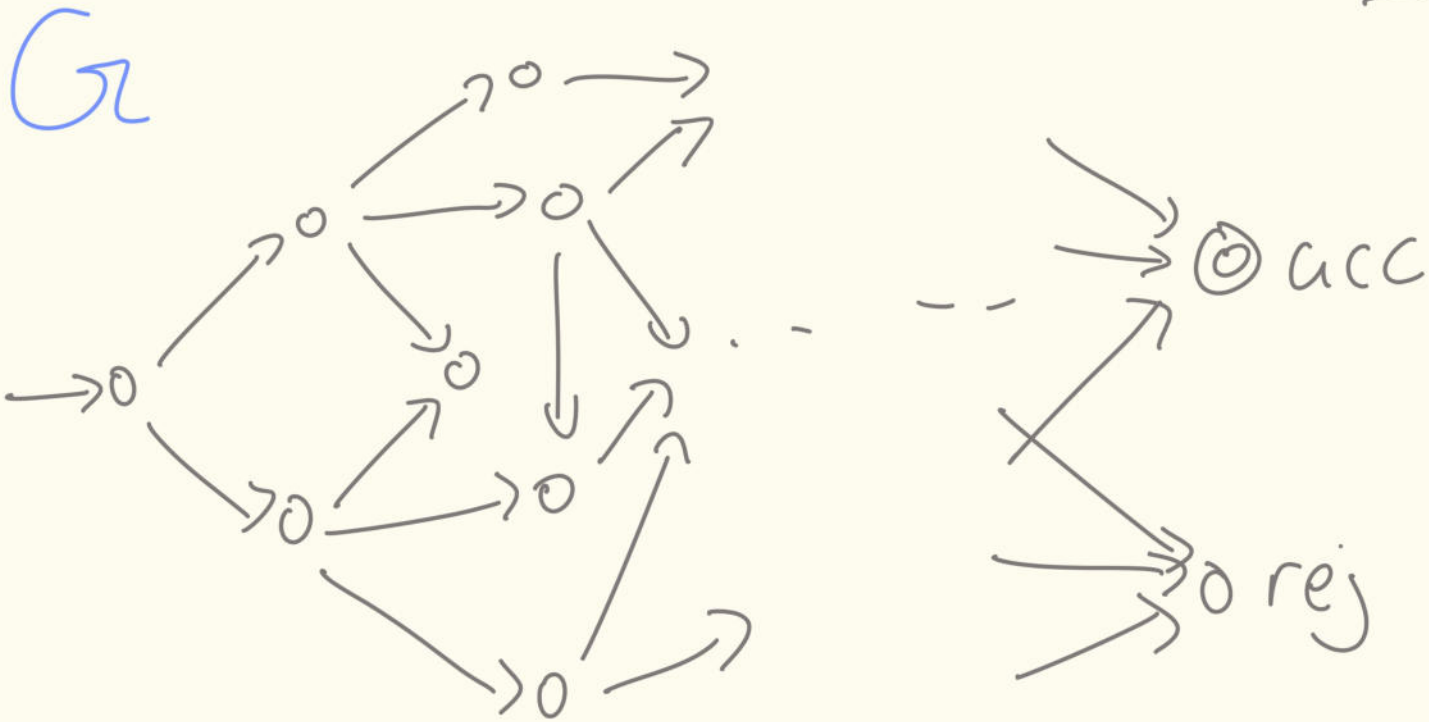


GOAL: 1) COMPRESS, DECOMP $\in$ CL;

2) FAIL $\rightarrow$ M can solve f w/ access to τ

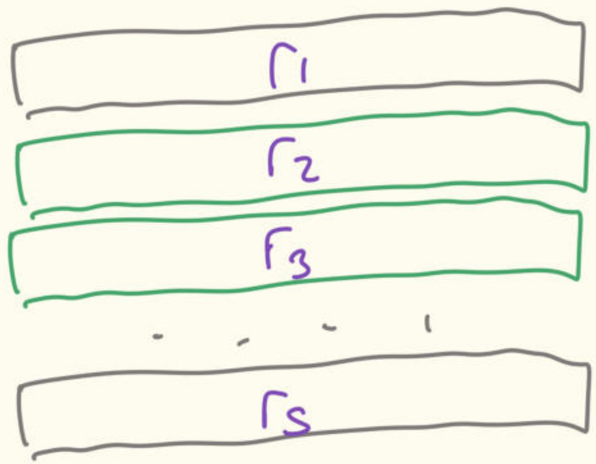
Nisan's Test





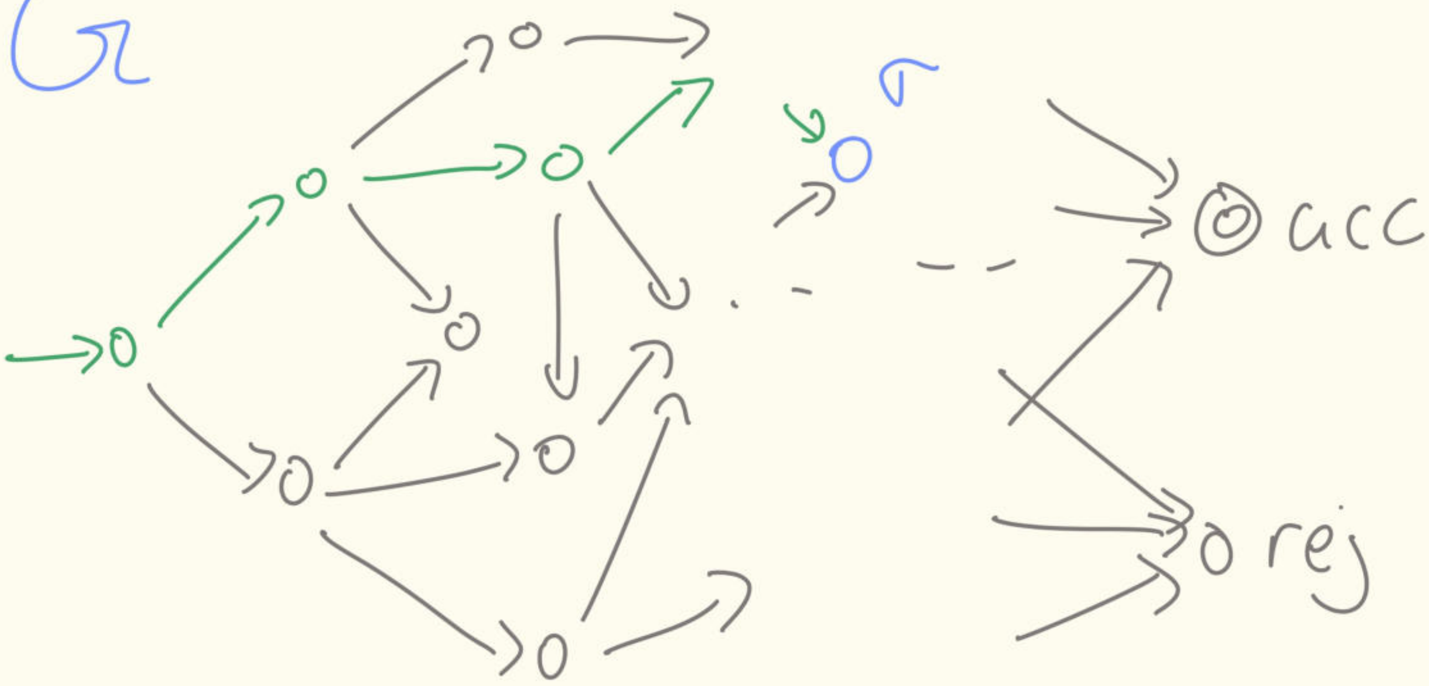
Nisan's Test

R



$R_{\sigma,i} : \{ r : \text{reach } \sigma \text{ at step } i \}$

G

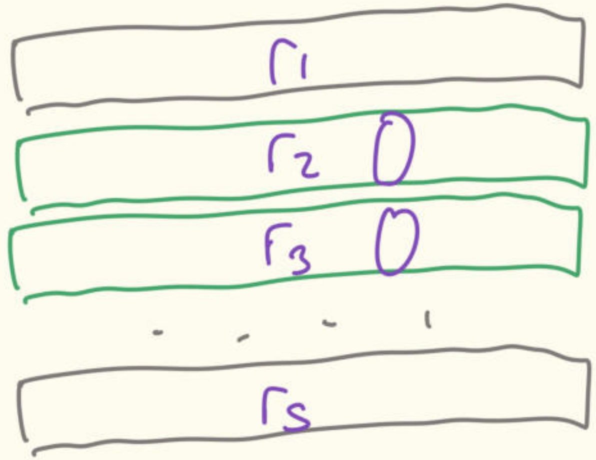


Nisan's Test

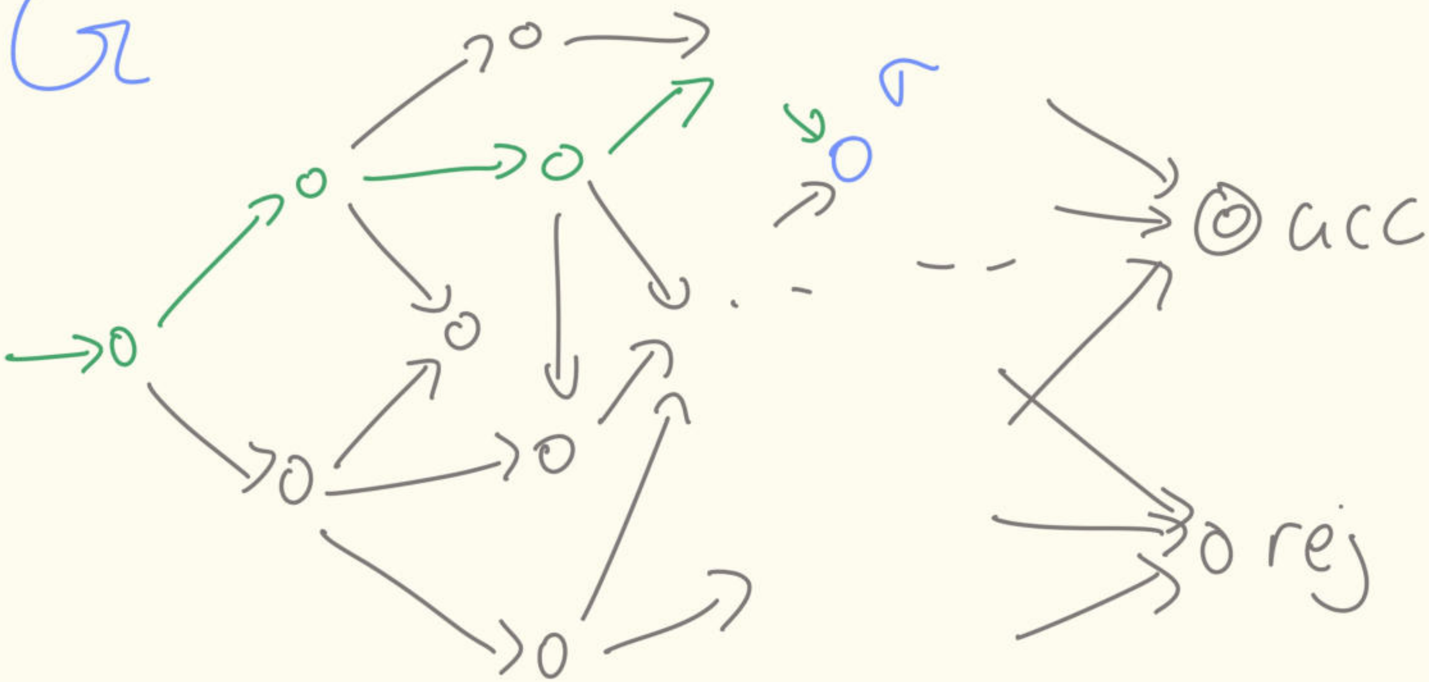
$R_{\sigma,i} : \{ r : \text{reach } \sigma \text{ at step } i \}$

$\rightarrow$ use bit $r[i+1]$ at σ

R



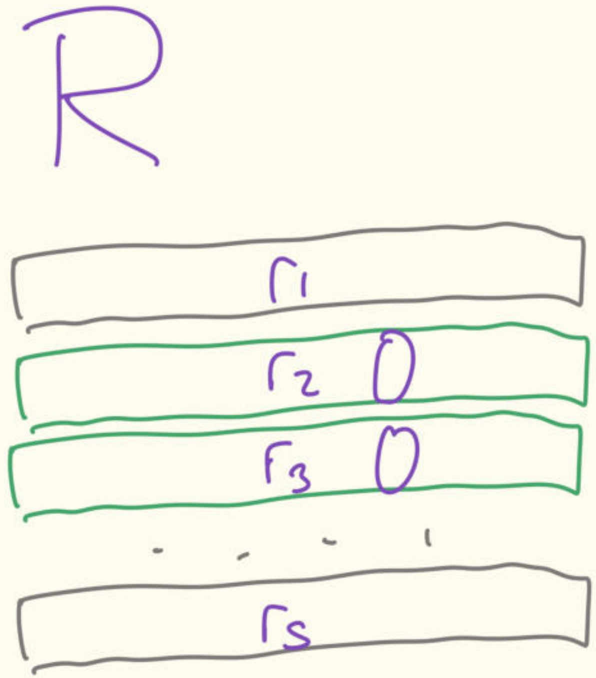
G_σ



Nisan's Test

$R_{\sigma,i} : \{ r : \text{reach } \sigma \text{ at step } i \}$

$\to$ use bit $r[i+1]$ at σ

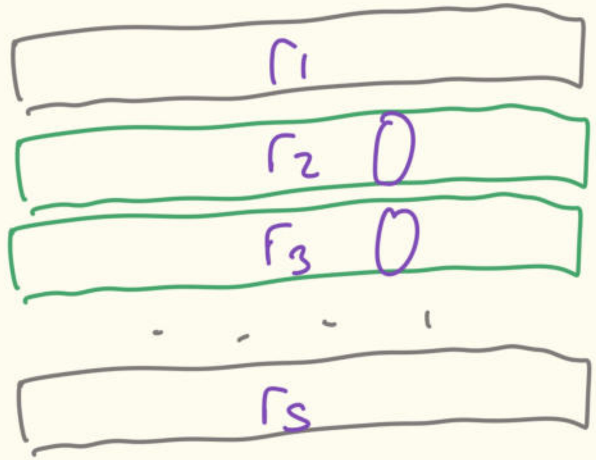


[N'94]:

$\underset{r\sim R}{\mathrm{MAJ}}\ G(r)$ correct

Nisan's Test

R



$R_{\sigma,i}$: $\{ r : \text{reach } \sigma \text{ at step } i \}$

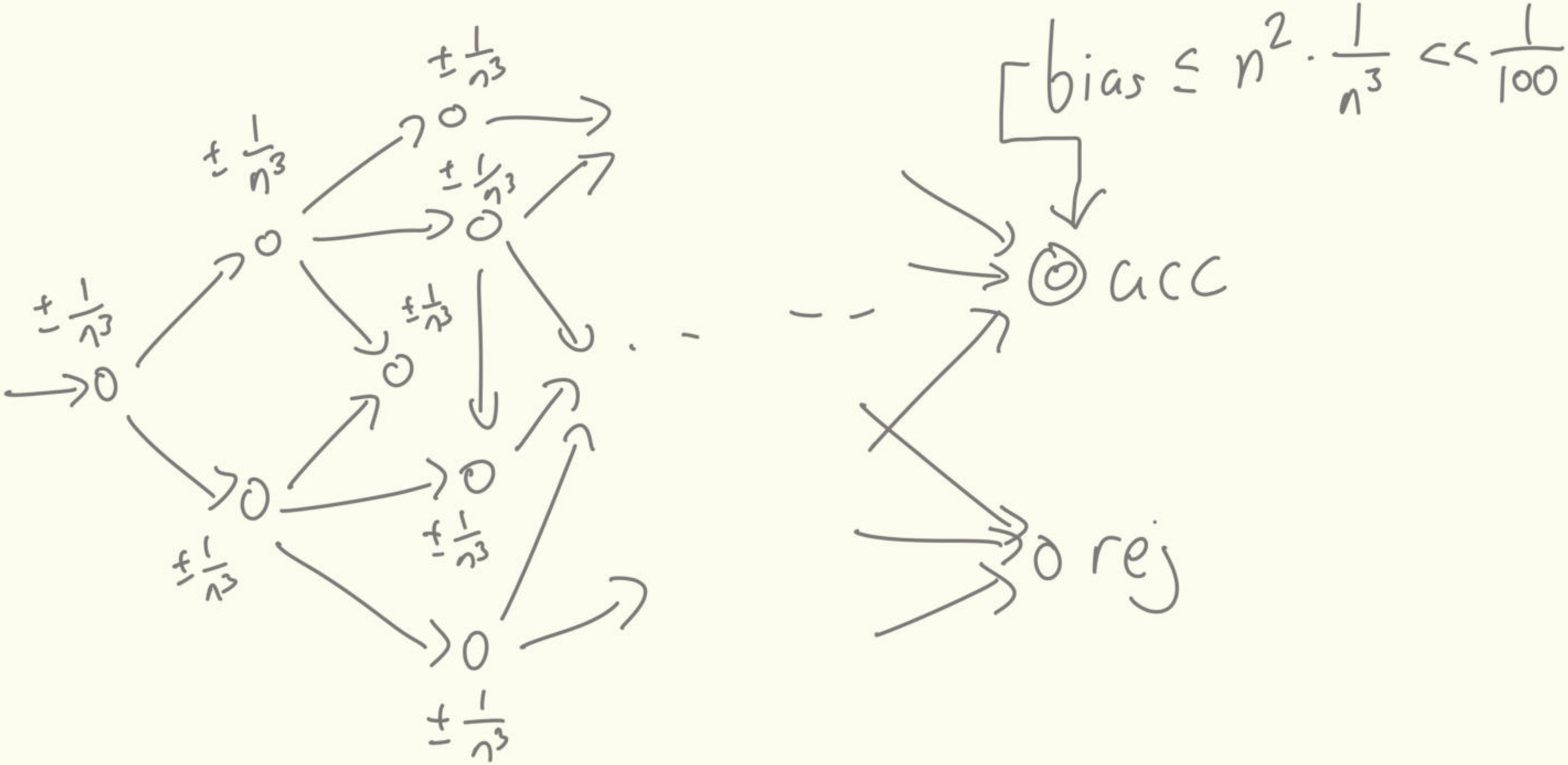
→ use bit $r[i+1]$ at σ

[N'94]:

$$\Pr_{r\sim R_{\sigma,i}}(r[i+1]=0) \approx \Pr_{r\sim R_{\sigma,i}}(r[i+1]=1) \quad \forall \sigma, i$$

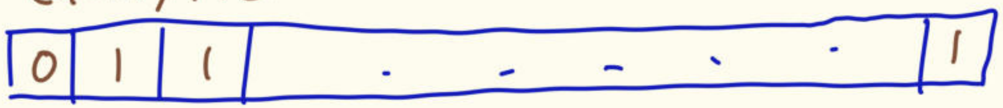
$$\longrightarrow \ \operatorname*{MAJ}_{r\sim R} G(r) \quad \text{correct}$$

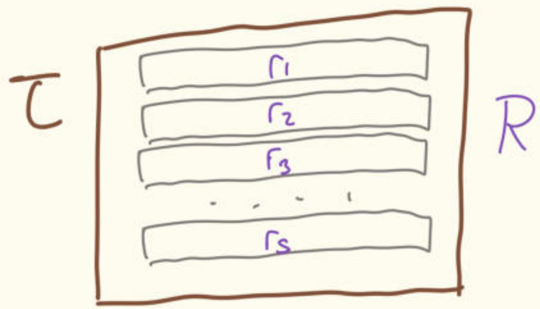
Nisan's Test



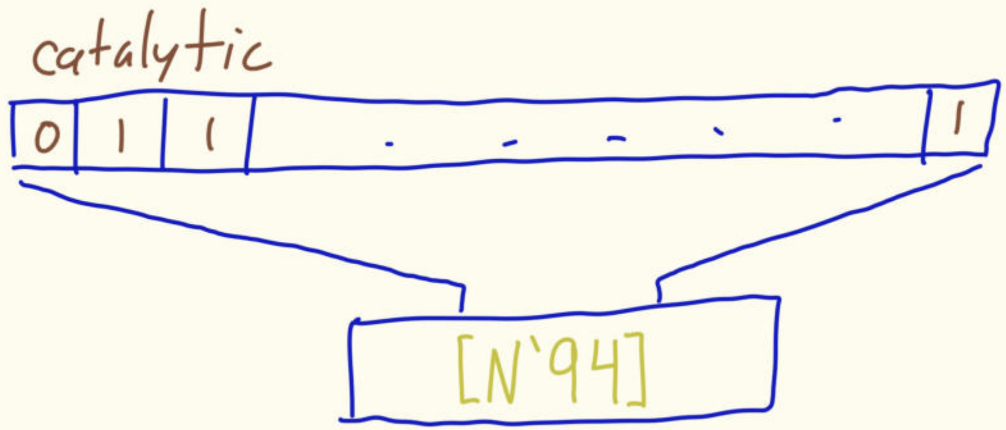
Compress-or-Random

catalytic

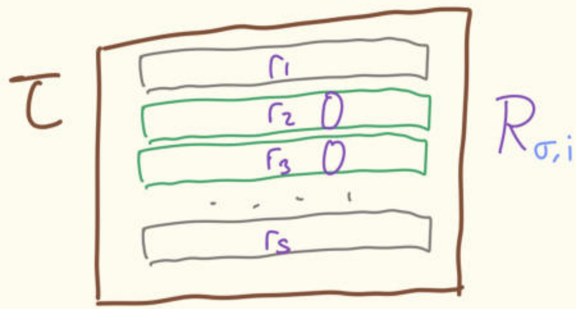




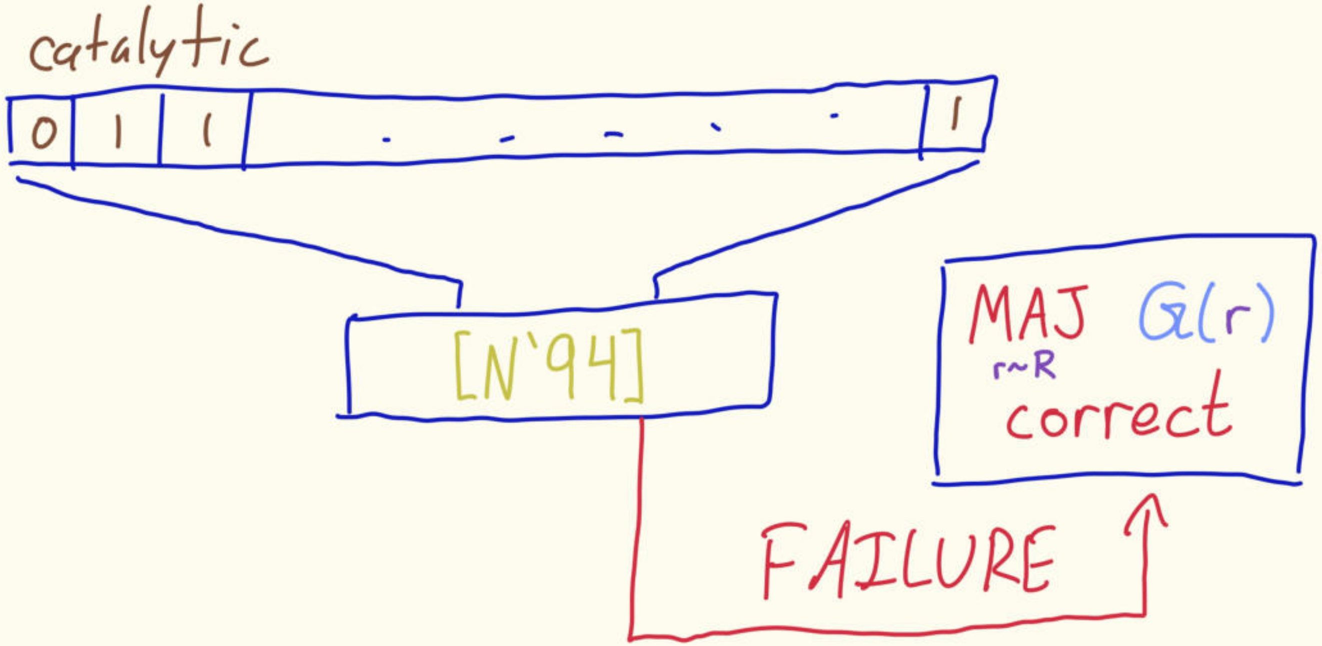
Compress-or-Random



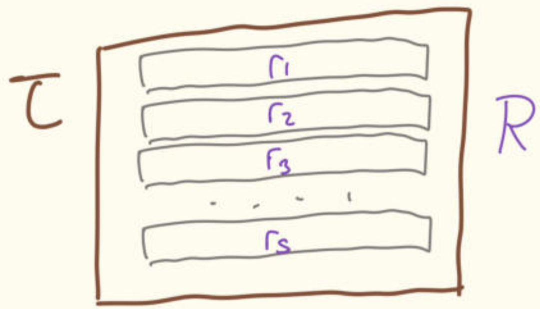
$$\exists \sigma, i \quad \Pr_{r \sim R_{\sigma,i}} \left(r[i+1] = 1 \right) > \frac{1}{2} + \frac{1}{n^3} \ ?$$



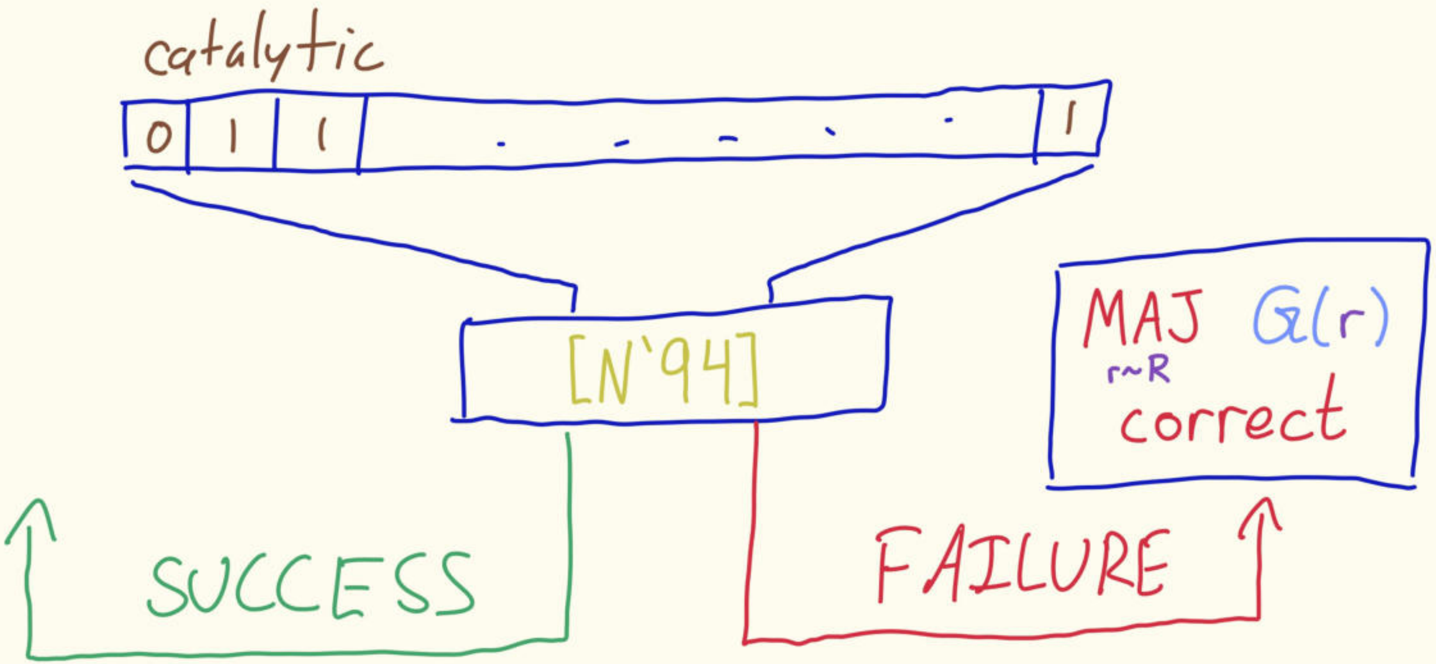
Compress-or-Random



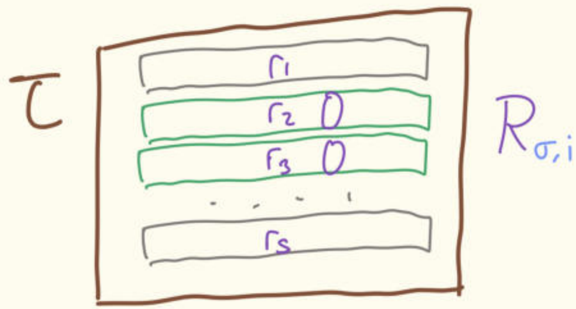
$\exists\, \sigma, i \quad \Pr_{r \sim R_{\sigma,i}} \left(r[i+1] = 1 \right) > \frac{1}{2} + \frac{1}{n^3}$?



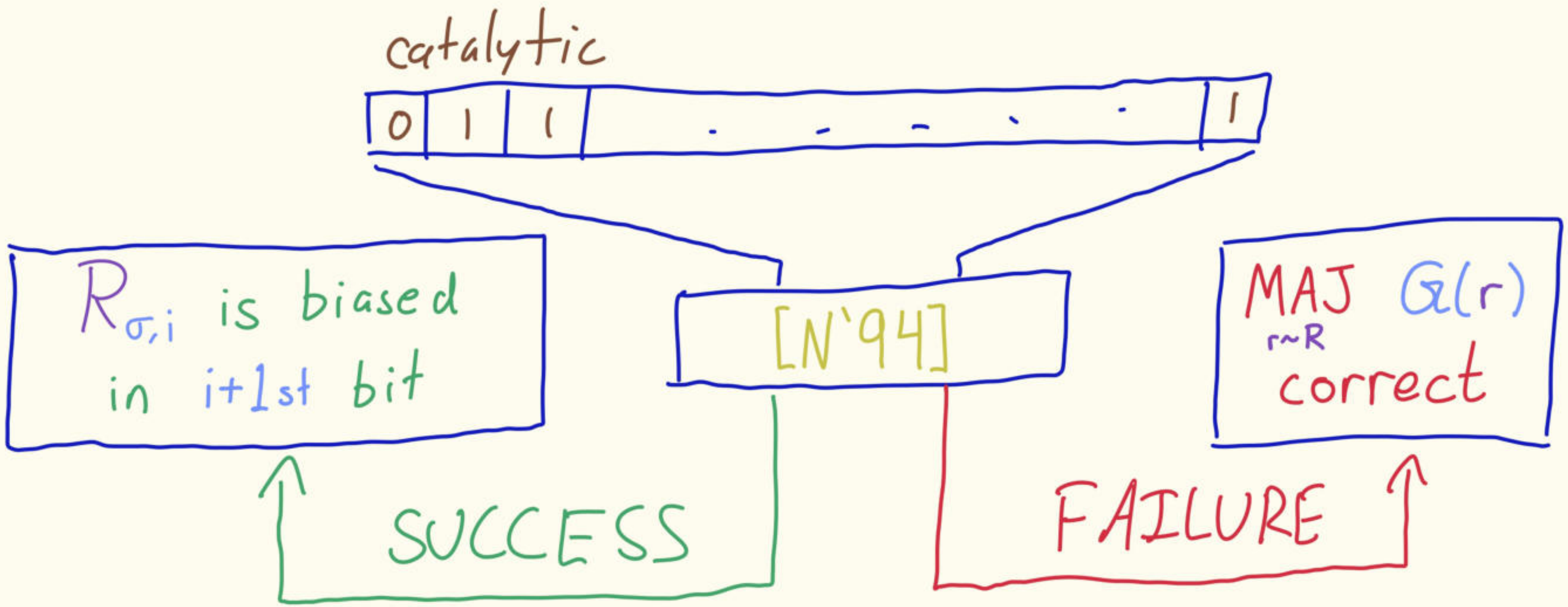
Compress-or-Random



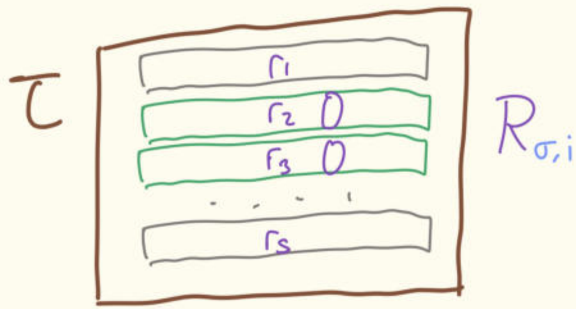
$$\exists \sigma, i \quad \Pr_{r\sim R_{\sigma,i}}\left(r[i+1]=1\right) > \frac{1}{2} + \frac{1}{n^3} \text{ ?}$$



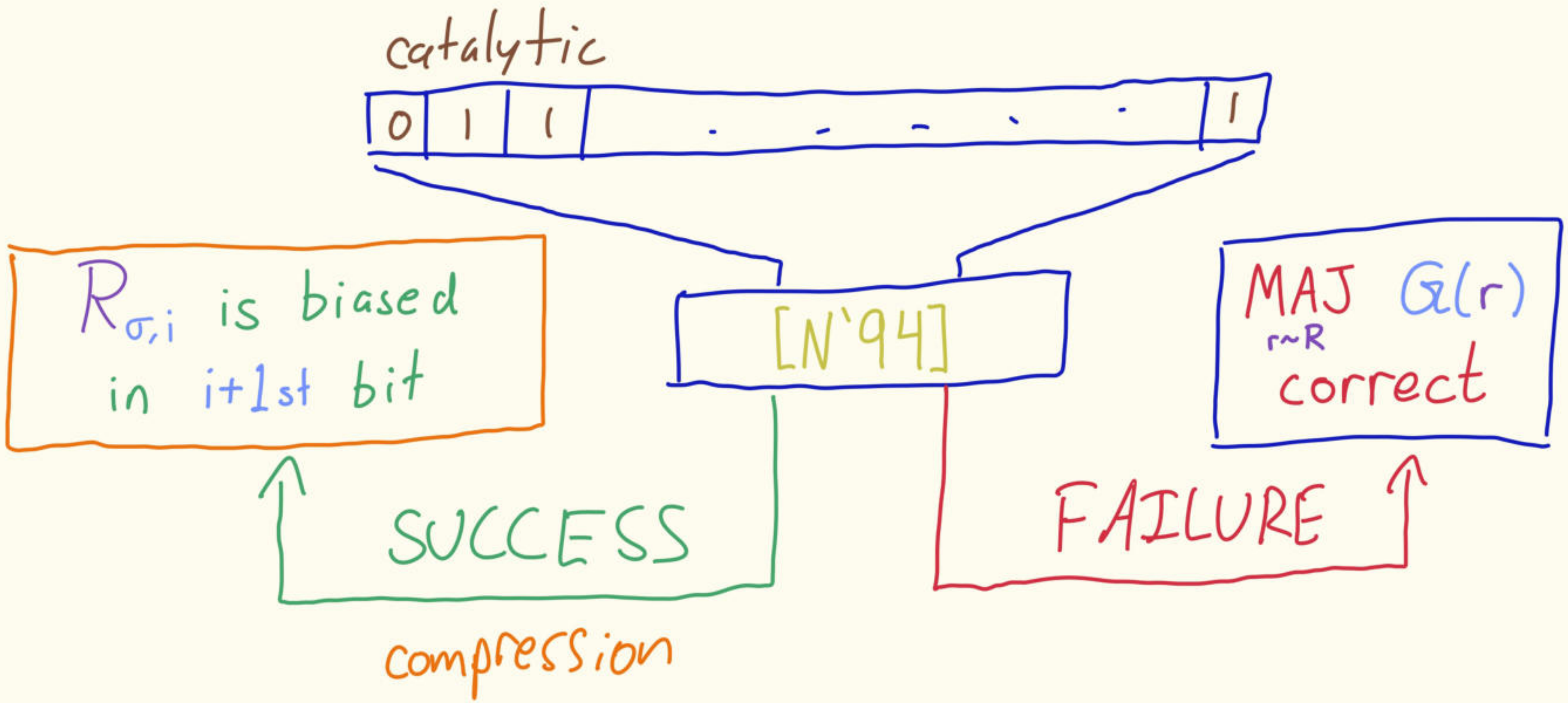
Compress-or-Random



$$\exists\, \sigma, i \quad \Pr_{r\sim R_{\sigma,i}}\left(r[i+1]=1\right) > \frac{1}{2}+\frac{1}{n^3}\ ?$$



Compress-or-Random



Exercise #1

The Power of Reuse

[Pyn'24]: $\mathsf{BPL} \subseteq \mathsf{CSPACE}[\log n, \log^2 n]$

bad hash functions
for Nisan's generator

The Power of Reuse

[Pyn'24]: $\mathsf{BPL} \subseteq \mathsf{CSPACE}[\log n, \log^2 n]$

[DPTW'25]: "$\mathsf{BPL} \subseteq \mathsf{NL}$ or $\mathsf{NL} \subseteq \mathsf{SC}^2$"

memory configuration of
P-time simulations of NL

The Power of Reuse

[Pyn'24]: $\mathsf{BPL} \subseteq \mathsf{CSPACE}[\log n, \log^2 n]$

[DPTW'25]: "$\mathsf{BPL} \subseteq \mathsf{NL}$ or $\mathsf{NL} \subseteq \mathsf{SC}^2$"

[CLMP'25]: $\mathsf{CBPL} \subseteq \mathsf{CL}$ large configuration graphs of CL

[KMPS'25]: $\mathsf{CNL} \subseteq \mathsf{CL}$ (+ AMCL hierarchy)

The Power of Reuse

[Pyn'24]: $\mathsf{BPL} \subseteq \mathsf{CSPACE}[\log n, \log^2 n]$

[DPTW'25]: "$\mathsf{BPL} \subseteq \mathsf{NL}$ or $\mathsf{NL} \subseteq \mathsf{SC}^2$"

[CLMP'25]: $\mathsf{CBPL} \subseteq \mathsf{CL}$

[KMPS'25]: $\mathsf{CNL} \subseteq \mathsf{CL}$ (+ AMCL hierarchy)

[AM'25]: $\textsc{Match} \in \mathsf{CL}$ isolation lemma

and more!

Today

What is...

... a Random Walk? ... Full Memory?

How?

[Dul'15]
Compression

[BCKLS'14]
Arithmetic Reversibility

[CP'25]
The Easy Way

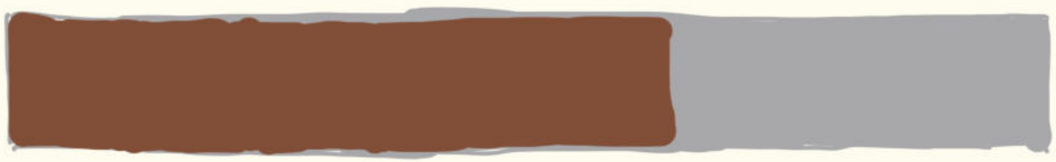
What Next?

storage + computation

Reusing Space

Given: Want:

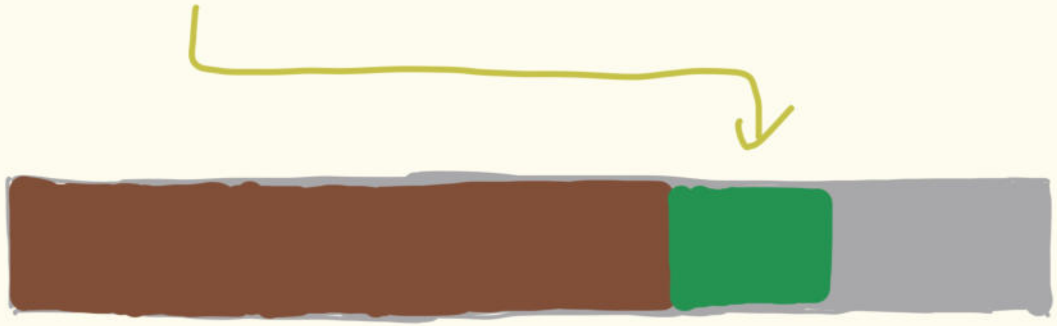
x $f(x)$



Reusing Space

Given: x

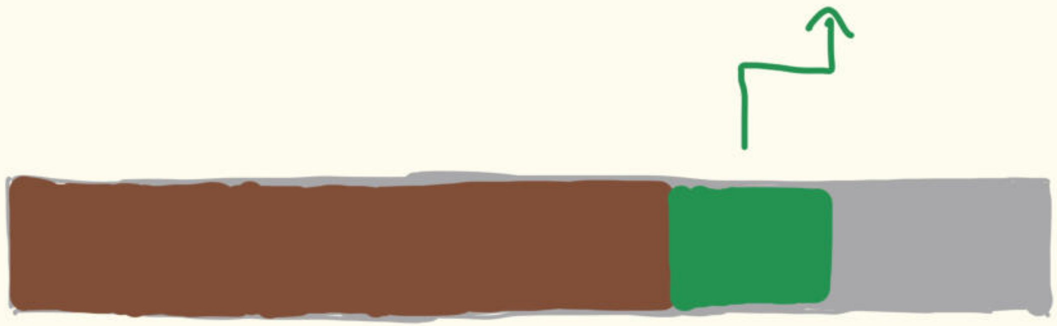
Want: $f(x)$



Reusing Space

Given: Want:

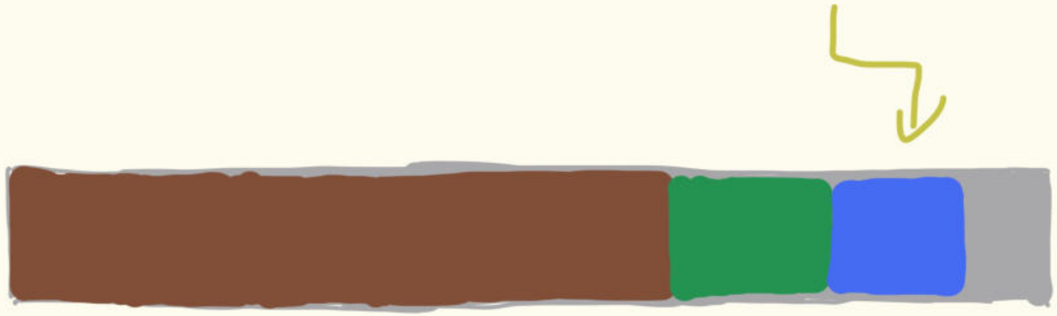
x $f(x)$



Reusing Space

Given: x

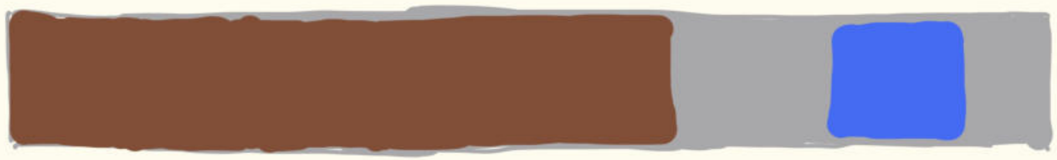
Want: $f(x)$



Reusing Space

Given: Want:

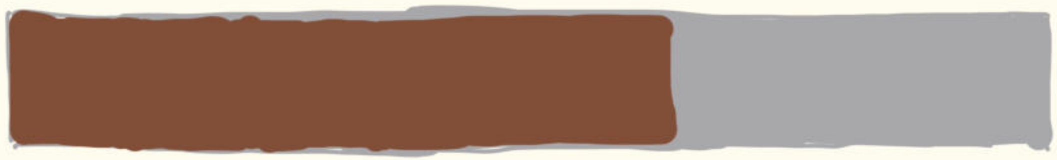
x $f(x)$



REUSING SPACE

GIVEN: WANT:

x $f(x)$



storage + computation

Reusing Space

Given: Want:

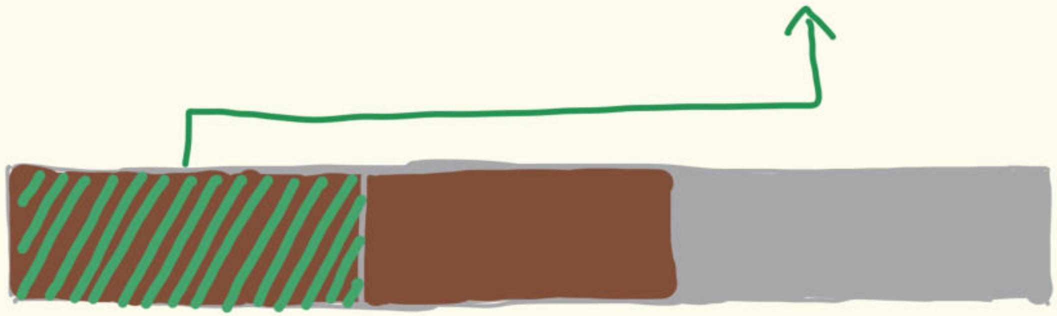
$+ x$ $f(x)$

storage + computation

Reusing Space

Given: Want:

$+\ x$ $f(x)$

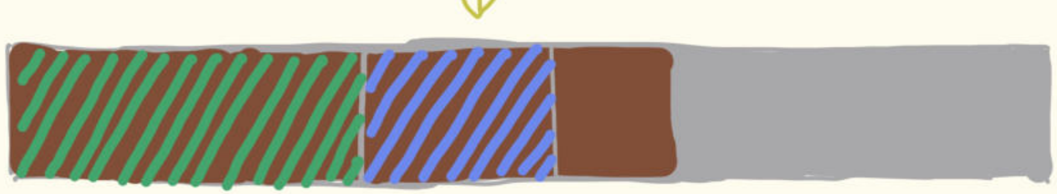


storage + computation

REUSING SPACE

GIVEN: WANT:

+ x + $f(x)$

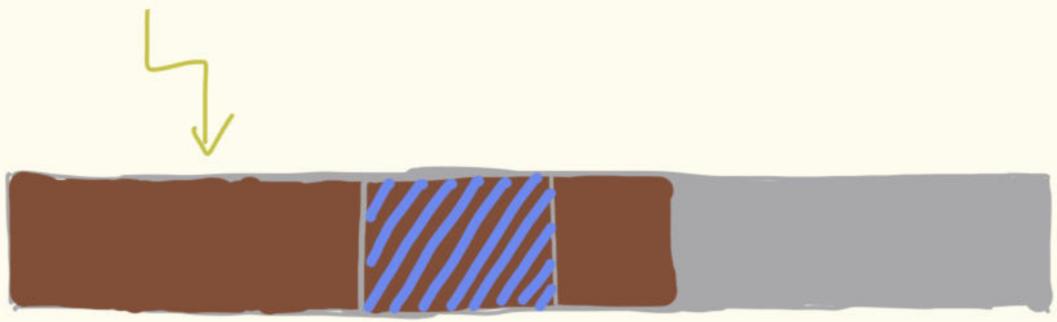


storage + computation

Reusing Space

Given: Want:

$\pm x$ $\pm f(x)$



storage + computation

REUSING SPACE

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

Reusing Space

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$$P_x^{(\cdot 1)}: \quad R_1 \underset{(-)}{+}= x$$

$$P_y^{(\cdot 1)}: \quad R_2 \underset{(-)}{+}= y$$

Reusing Space

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$$P_x^{(\cdot 1)}: \; R_1 \underset{(-)}{+}= x \qquad\longmapsto\qquad P_{ADD}^{(\cdot 1)}: \; R_3 \underset{(-)}{+}= x + y$$

$$P_y^{(\cdot 1)}: \; R_2 \underset{(-)}{+}= y \qquad\qquad\qquad P_{MULT}^{(\cdot 1)}: \; R_3 \underset{(-)}{+}= x\, y$$

Reusing Space

$$P_x^{(-1)} : \; R_1 \underset{(-)}{+}= x$$

$$P_y^{(-1)} : \; R_2 \underset{(-)}{+}= y$$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$$P_{ADD}^{(-1)} : \; R_3 \underset{(-)}{+}= x + y$$

REUSING SPACE

$P_x^{(-1)}: \ R_1 \underset{(-)}{+}= x$

$P_y^{(-1)}: \ R_2 \underset{(-)}{+}= y$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$-\tau_1$

$-\tau_2$

P_{ADD}:

1. $R_3 \mathrel{-}= R_1 + R_2$

Reusing Space

$$\boxed{\begin{array}{l} P_x^{(-1)} : R_1 \underset{(-)}{+}= x \\ P_y^{(-1)} : R_2 \underset{(-)}{+}= y \end{array}}$$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$+x$	$+y$	$-\tau_1$ $-\tau_2$

P_{ADD}:

1. $R_3 \mathrel{-}= R_1 + R_2$

2. P_x, P_y

Reusing Space

$P_x^{(-1)}: \; R_1 \underset{(-)}{+}= x$

$P_y^{(-1)}: \; R_2 \underset{(-)}{+}= y$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3
$+x$	$+y$	$-\cancel{\tau_1} + (\cancel{\tau_1} + x)$ $-\cancel{\tau_2} + (\cancel{\tau_2} + y)$

P_{ADD}:

1. $R_3 \mathrel{-}= R_1 + R_2$
2. P_x, P_y
3. $R_3 \mathrel{+}= R_1 + R_2$

Reusing Space

$$P_x^{(-1)}: \; R_1 \underset{(-)}{+}= x$$
$$P_y^{(-1)}: \; R_2 \underset{(-)}{+}= y$$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3
$\cancel{+x}$	$\cancel{+y}$	$+x+y$
$\cancel{-x}$	$\cancel{-y}$	

P_{ADD} :

1. $R_3 \mathrel{-}= R_1 + R_2$
2. $P_x \;, \quad P_y$
3. $R_3 \mathrel{+}= R_1 + R_2$
4. $P_x^{-1} \;, \quad P_y^{-1}$

Reusing Space

$P_x^{(-1)}: R_1 \underset{(-)}{+}= x$

$P_y^{(-1)}: R_2 \underset{(-)}{+}= y$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$$P_{MULT}^{(-1)}: R_3 \underset{(-)}{+}= xy$$

Reusing Space

$$\boxed{\begin{array}{l} P_x^{(-1)} : R_1 \underset{(-)}{+}= x \\ P_y^{(-1)} : R_2 \underset{(-)}{+}= y \end{array}}$$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$$\boxed{P_{MULT}^{(-1)} : R_3 \underset{(-)}{+}= xy}$$

Exercise #2

Reusing Space

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$$P_x^{(-1)}: \; R_1 \underset{(-)}{+}= x \qquad\qquad P_{ADD}^{(-1)}: \; R_3 \underset{(-)}{+}= x+y$$

$$\longmapsto$$

$$P_y^{(-1)}: \; R_2 \underset{(-)}{+}= y \qquad\qquad P_{MULT}^{(-1)}: \; R_3 \underset{(-)}{+}= xy$$

Reusing Space

[BC'92]:

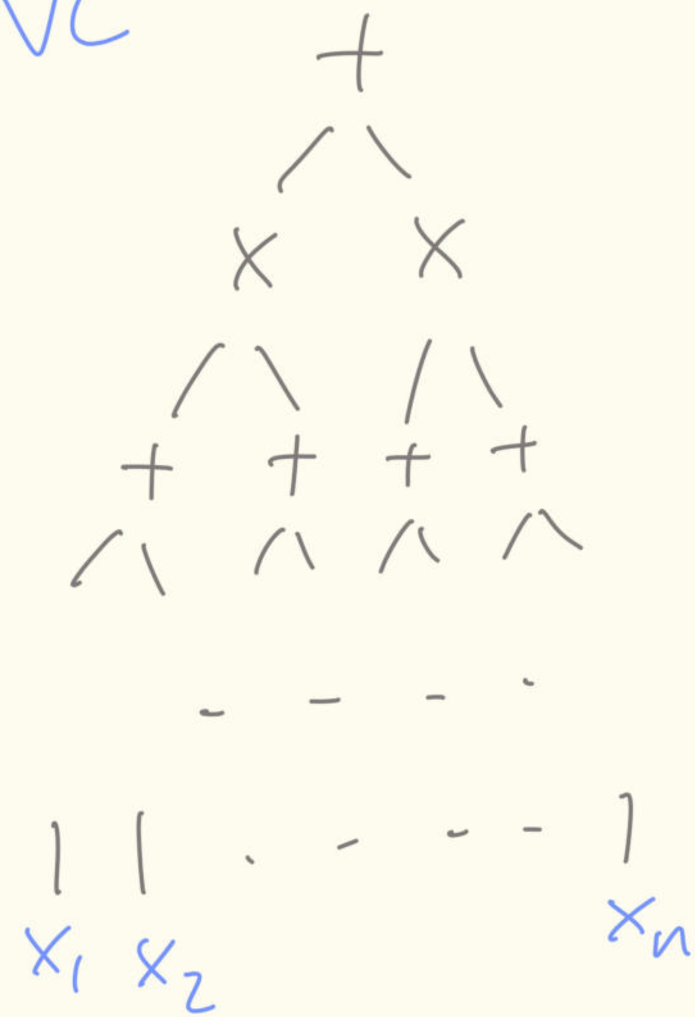
R_1	R_2	R_3
τ_1	τ_2	τ_3

$$P_x^{(-1)}: \ R_1 \underset{(-)}{+}= x \qquad P_{ADD}^{(-1)}: \ R_3 \underset{(-)}{+}= x+y$$

$$\longmapsto$$

$$P_y^{(-1)}: \ R_2 \underset{(-)}{+}= y \qquad P_{MULT}^{(-1)}: \ R_3 \underset{(-)}{+}= xy$$

$\#NC^1$



Reusing Space

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	$\boxed{\tau_3}$
$+g_1$	$+g_2$	$+g_1+g_2$

$$P_x^{(-1)}: R_1 \underset{(-)}{+}= x$$

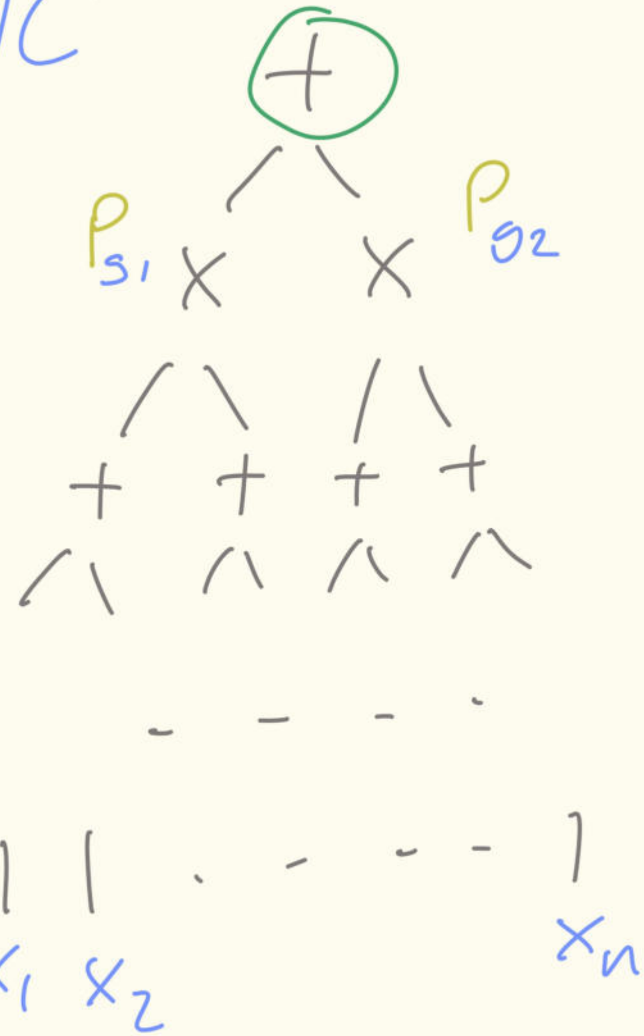
$$P_y^{(-1)}: R_2 \underset{(-)}{+}= y$$

$$\mapsto$$

$$\boxed{P_{ADD}^{(-1)}: R_3 \underset{(-)}{+}= x+y}$$

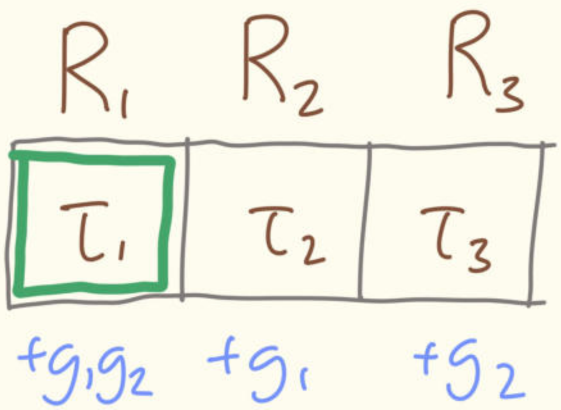
$$P_{MULT}^{(-1)}: R_3 \underset{(-)}{+}= xy$$

$\#NC^1$



REUSING SPACE

[BC'92]:



$P_x^{(-1)}: R_1 \underset{(-)}{+}= x$

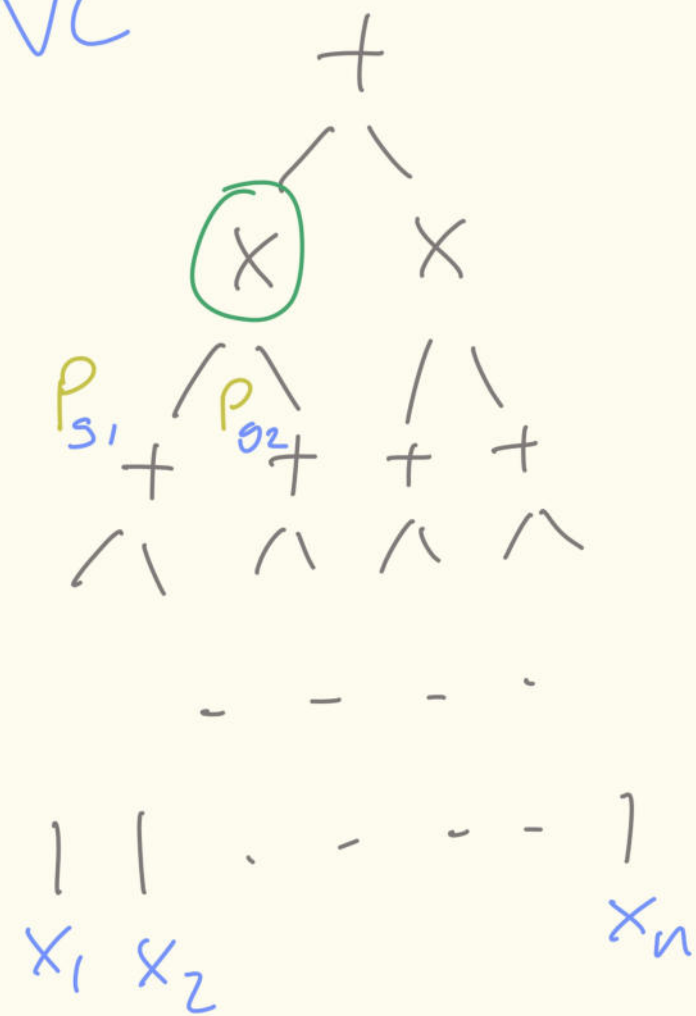
$P_y^{(-1)}: R_2 \underset{(-)}{+}= y$

$\mapsto$

$P_{ADD}^{(-1)}: R_3 \underset{(-)}{+}= x+y$

$\boxed{P_{MULT}^{(-1)}: R_3 \underset{(-)}{+}= xy}$

$\#NC^1$



Reusing Space

[BC'92]:

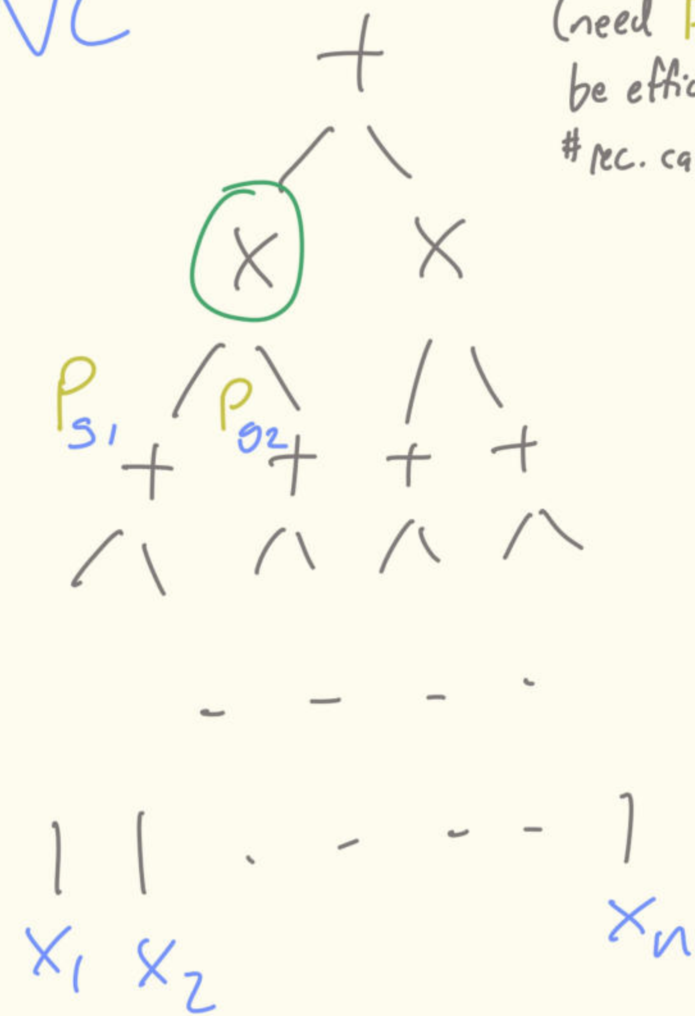
R_1	R_2	R_3
τ_1	τ_2	τ_3

values are a <u>mess</u> (based on current stack)

$\#NC^1$

recursive

(need P to be efficient in #rec. calls)



Reusing Space

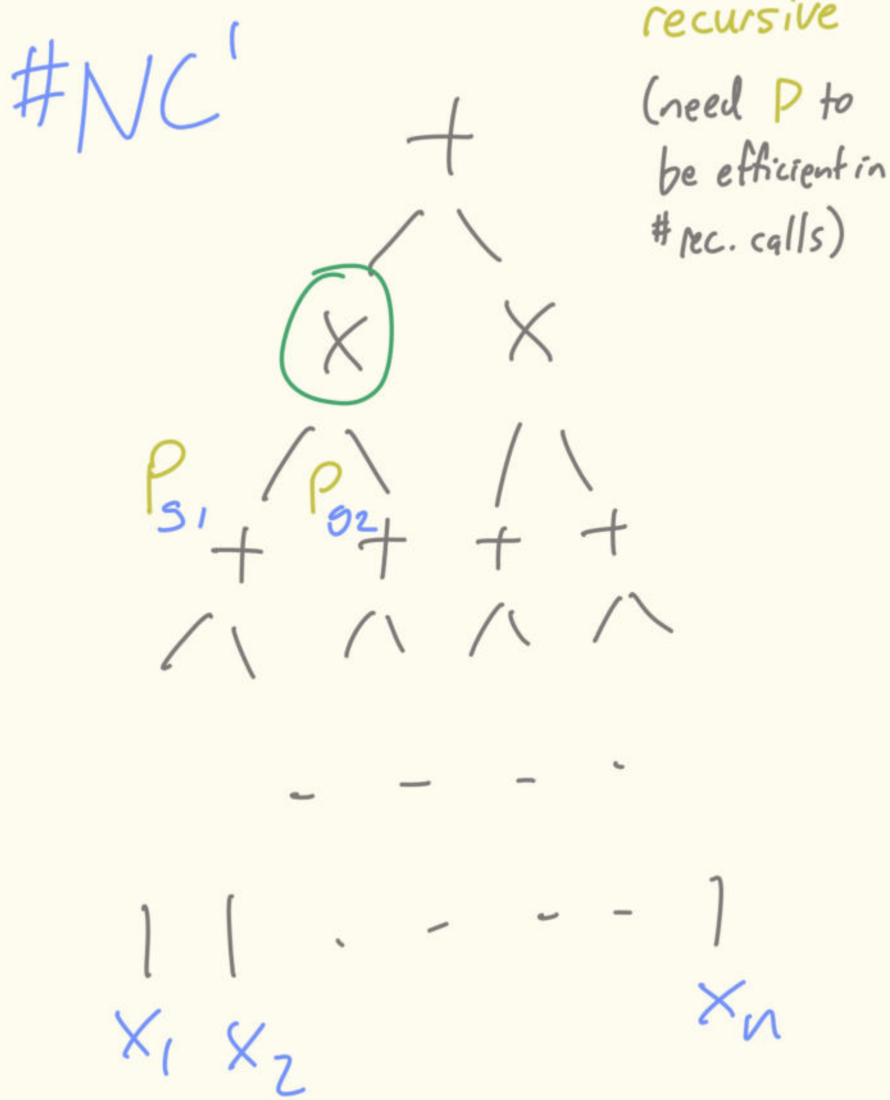
[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

values are a mess (based on current stack)

... but P_{ADD}, P_{MULT} work for any τ values

→ can always use same registers (rotating off target) everywhere in the recursion



Reusing Space

[BCKLS'14]:

R_1	R_2	R_3		R_s	R_{out}
τ_1	τ_2	τ_3	- - - -	τ_s	τ_{out}

$$P^{(\cdot 1)}_{\vec{x}} : \vec{R} \underset{(-)}{+}= \vec{x} \longrightarrow P^{(\cdot 1)}_{ADD} : R_{out} \underset{(-)}{+}= \sum x$$

Reusing Space

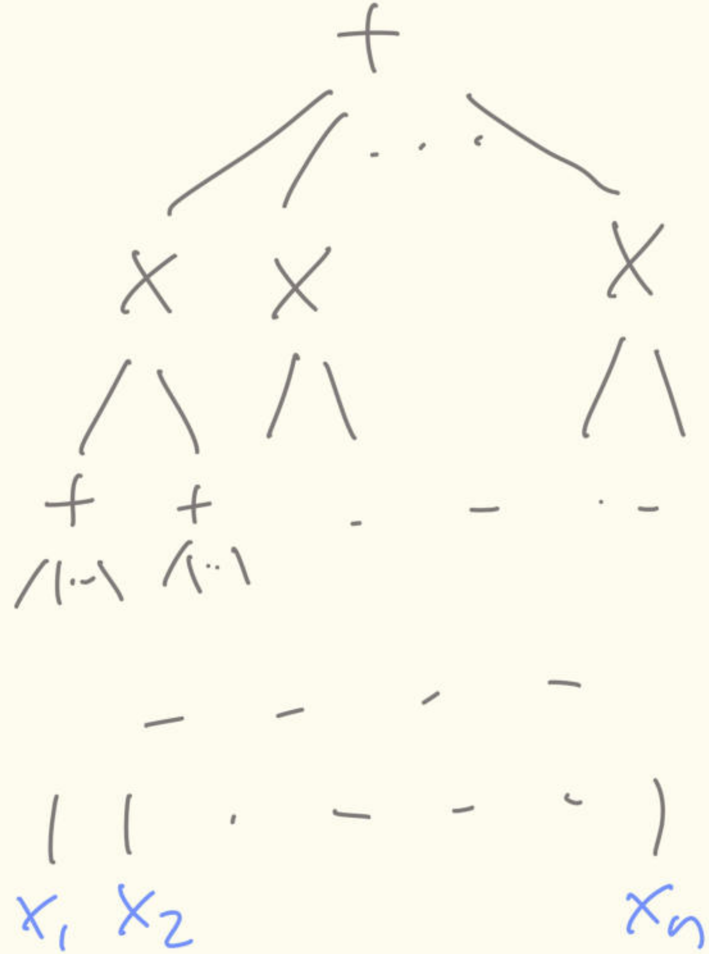
[BCKLS'14]:

$P^{(-1)}_{\vec{x}}: \vec{R} \underset{(-)}{+}= \vec{x} \rightarrow$

$P^{(-1)}_{ADD}: R_{out} \underset{(-)}{+}= \sum x$

$P^{(-1)}_{MULT}: R_{out} \underset{(-)}{+}= x\,y$

VP



Reusing Space

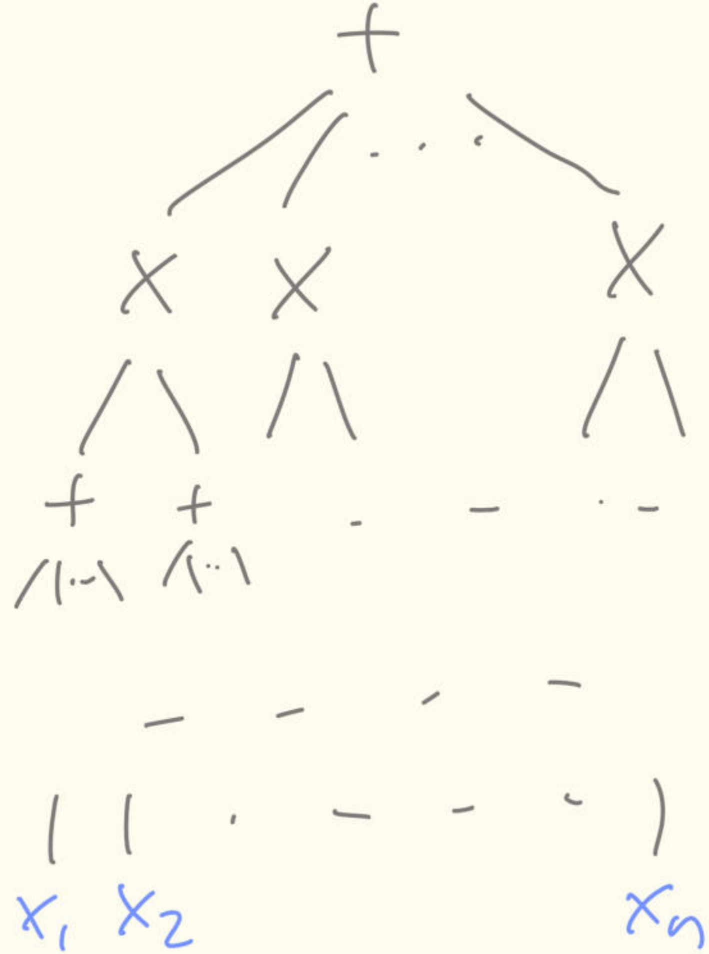
[BCKLS'14]:

$$P^{(-1)}_{\vec{x}} : \vec{R} \underset{(-)}{+}= \vec{x} \quad \to \quad \begin{array}{l} P^{(-1)}_{ADD} : R_{out} \underset{(-)}{+}= \sum x \\ P^{(-1)}_{MULT} : R_{out} \underset{(-)}{+}= x\,y \end{array}$$

DET

$\subseteq$

VP



Reusing Space

[BCKLS'14]:

$P^{(\cdot 1)}_{\vec{x}}: \vec{R} \underset{(-)}{+}= \vec{x} \rightarrow$

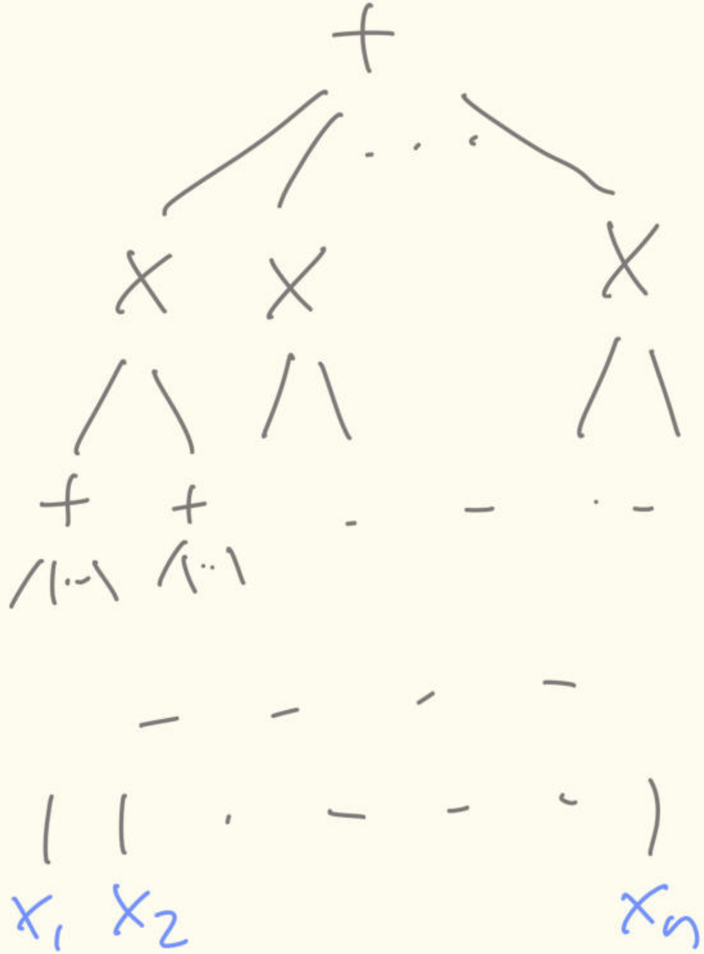
$P^{(\cdot 1)}_{ADD}: R_{out} \underset{(-)}{+}= \sum x$

$P^{(\cdot 1)}_{MULT}: R_{out} \underset{(-)}{+}= x\,y$

$\text{DET} \supseteq \text{BPL}$

$\cap|$

VP



The Power of Reuse

[B'89]: $x \wedge y$

[BC'92]: $x \times y$

[BCKLS'14]: $\mathrm{MAJ}(\vec{x})$

[AFMSV'25]: M^k

The Power of Reuse

[B'89]: $NC^1 \subseteq PBP[5]$

[BC'92]: $\#NC^1 \subseteq L$

[BCKLS'14]: $TC^1 \subseteq CL$

[AFMSV'25]: $SAC^2 \subseteq CSPACE[o(\log^2 n)]$

The Power of Reuse

(worse efficiency)

[CM'20, 21, 24]: <u>any</u> $f(x_1 \ldots x_n)$

The Power of Reuse

(worse efficiency)

[CM'20, 21, 24]: <u>any</u> $f(x_1, \dots x_n)$

[CM'24]: $\mathsf{TEP} \subseteq \mathsf{SPACE}[\log n \cdot \log\log n]$

The Power of Reuse

[CM'20, 21, 24]: **any** $f(x_1 \ldots x_n)$ (worse efficiency)

[CM'24]: $\mathsf{TEP} \subseteq \mathsf{SPACE}[\log n \cdot \log \log n]$

[Wil'25]: $\mathsf{TIME}[t] \subseteq \mathsf{SPACE}[\sqrt{t \log t}]$

Today

What is...

... a Random Walk? ... Full Memory?

How?

[Dul'15] Compression

[BCKLS'14] Arithmetic Reversibility

[CP'25] The Easy Way

What Next?

Deterministic Random Walks

Compress-or-Random

Arithmetic Reversibility

Deterministic Random Walks

Compress-or-Random

[N'94]: if $\forall \sigma, i \quad \Pr_{r \sim R_{\sigma,i}} (r[i+1] = 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r \sim R}\ \mathcal{G}(r)$ correct

Arithmetic Reversibility

Deterministic Random Walks

COMPRESS-OR-RANDOM

[N'94]: if $\forall \sigma, i \quad \Pr_{r \sim R_{\sigma,i}} (r[i+1] = 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r \sim R}\ \mathcal{G}(r)$ correct

ARITHMETIC REVERSIBILITY

[BC'92]: $P_x^{(-1)} : R_1 \underset{(-)}{+}= x$

$P_y^{(-1)} : R_2 \underset{(-)}{+}= y$

$\longmapsto$

$P_{ADD}^{(-1)} : R_3 \underset{(-)}{+}= x + y$

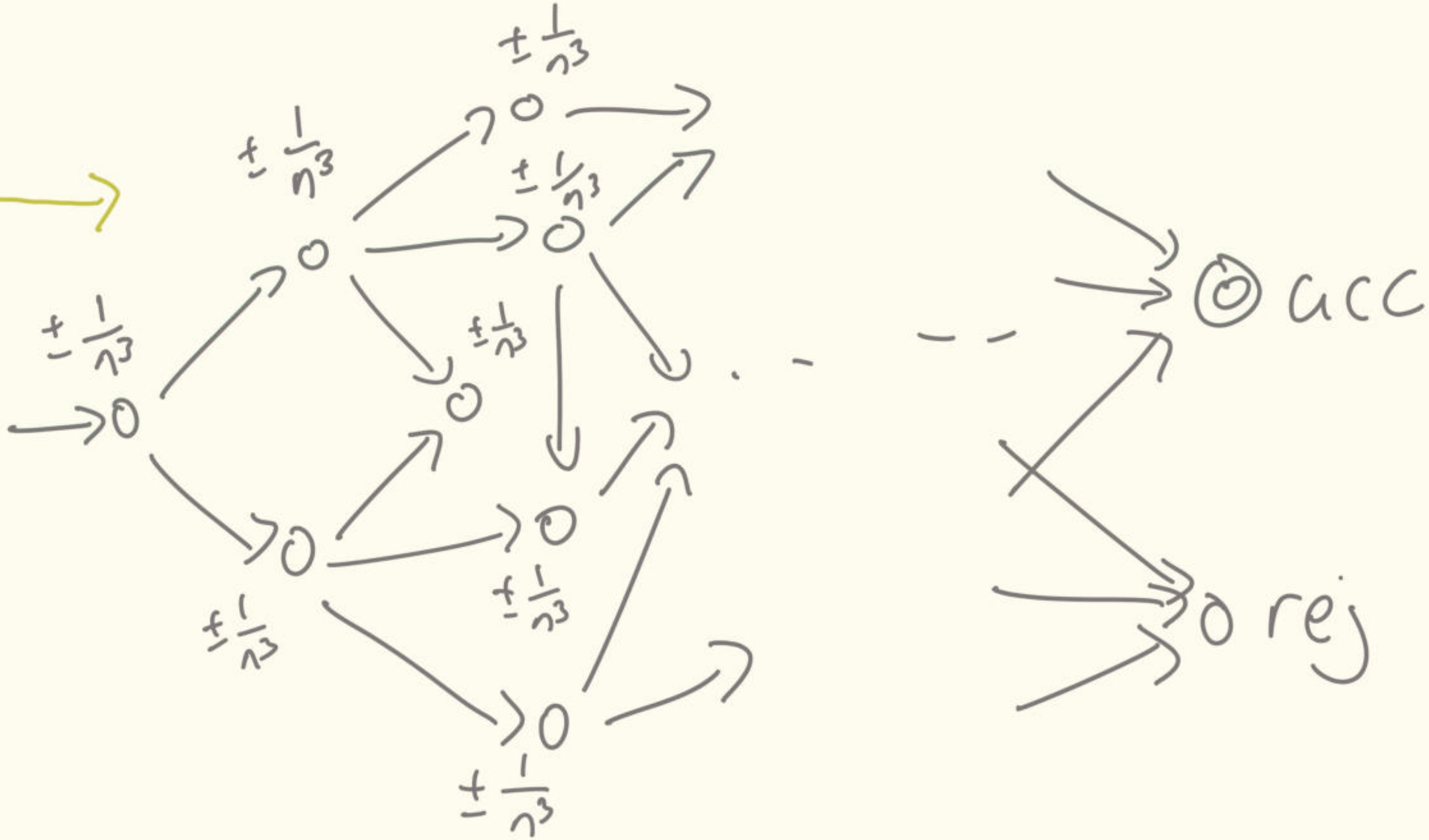
$P_{MULT}^{(-1)} : R_3 \underset{(-)}{+}= x y$

Deterministic Random Walks

R_1	R_2	R_3		R_s
τ_1	τ_2	τ_3	- - - -	τ_s

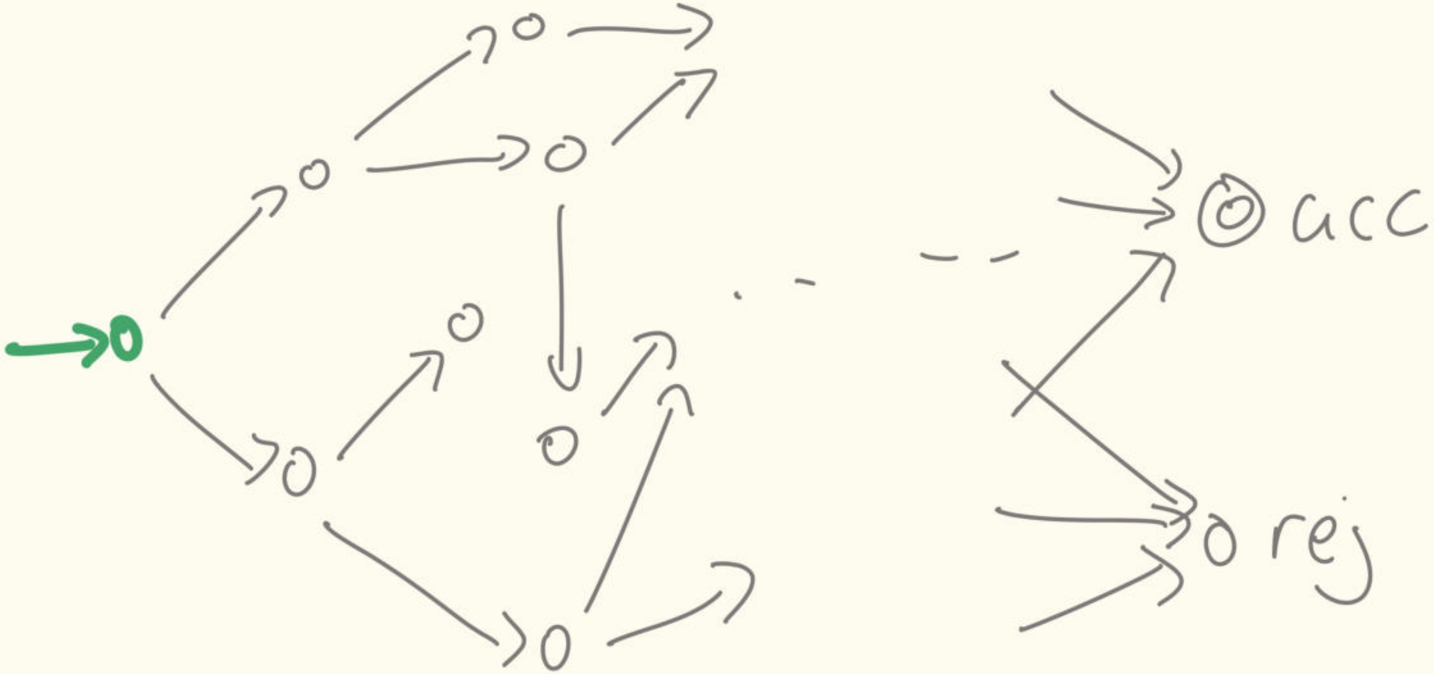
GOAL

set of walks with $\leq \frac{1}{n^3}$ bias/node



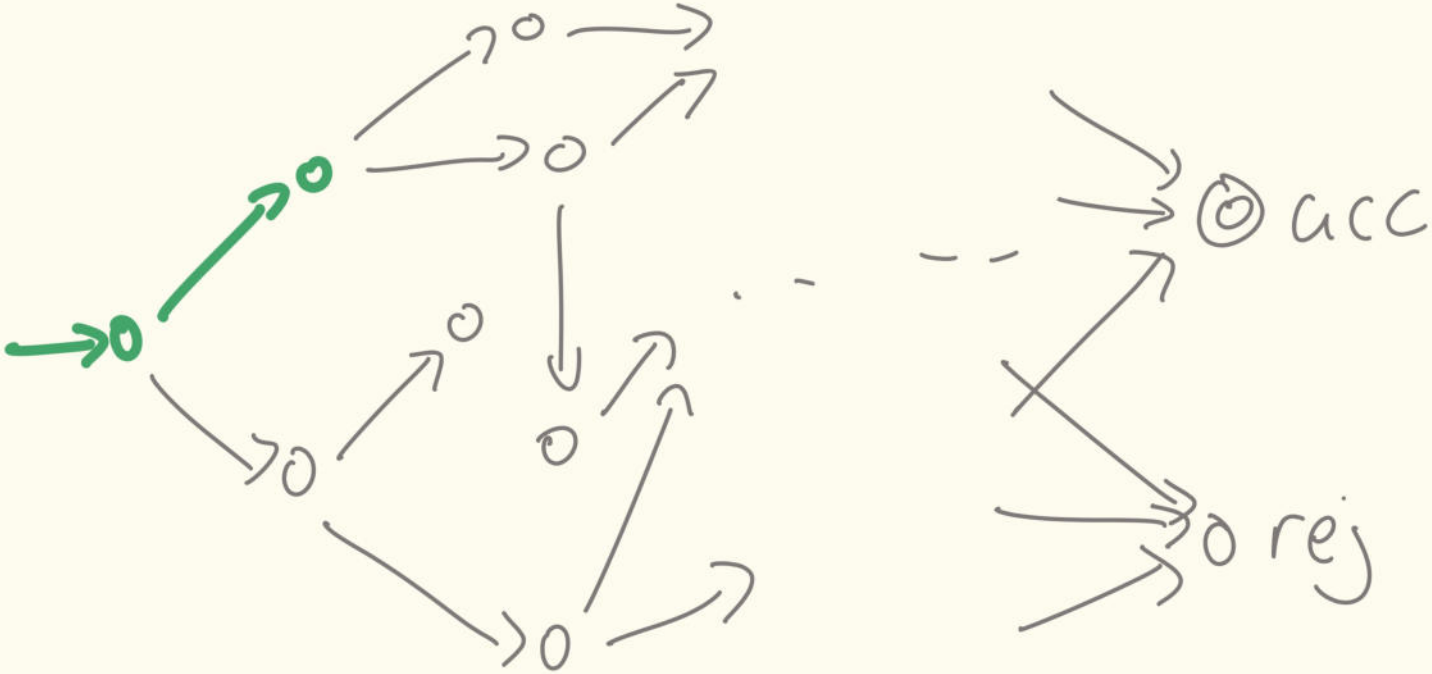
Deterministic Random Walks

R_1	R_2	R_3		R_s
0	1	1	- - - -	1



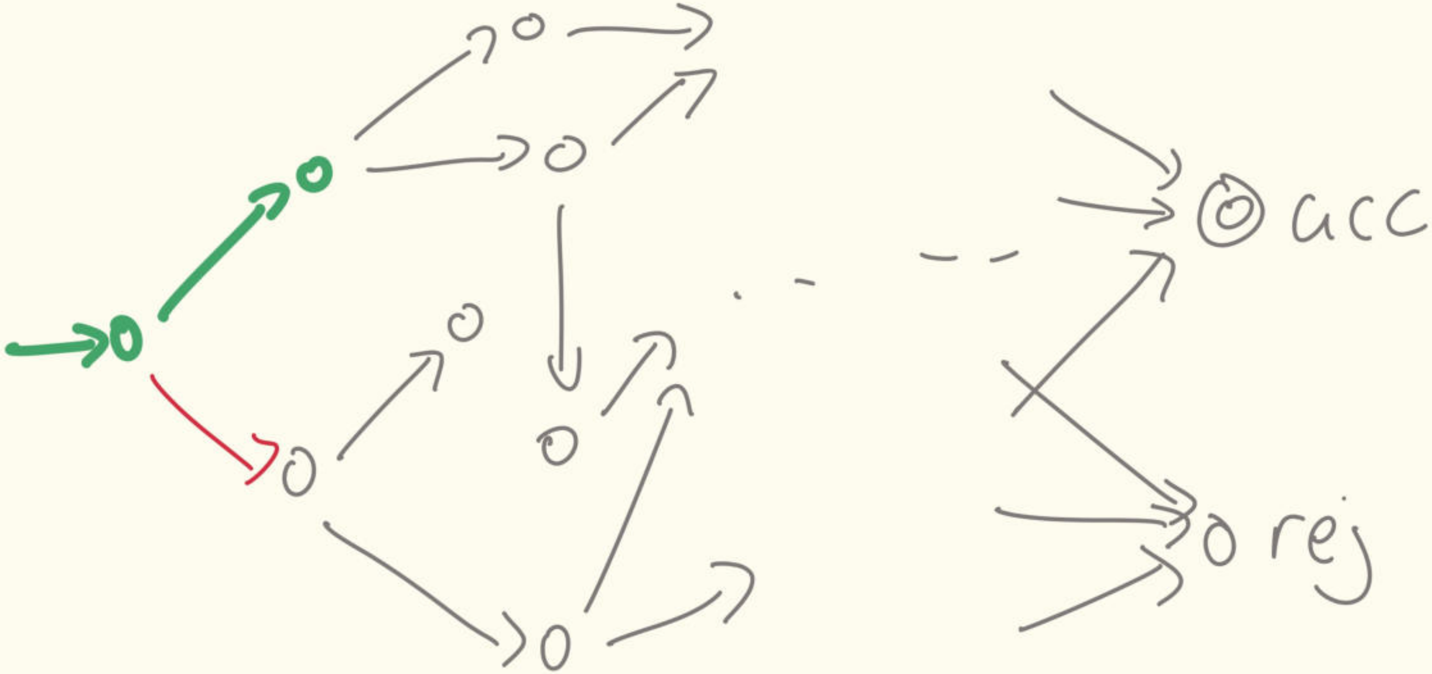
Deterministic Random Walks

R_1	R_2	R_3		R_s
0	1	1	- - - -	1

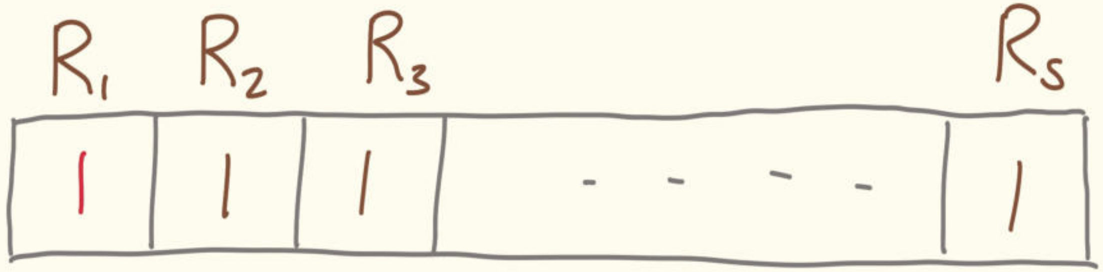


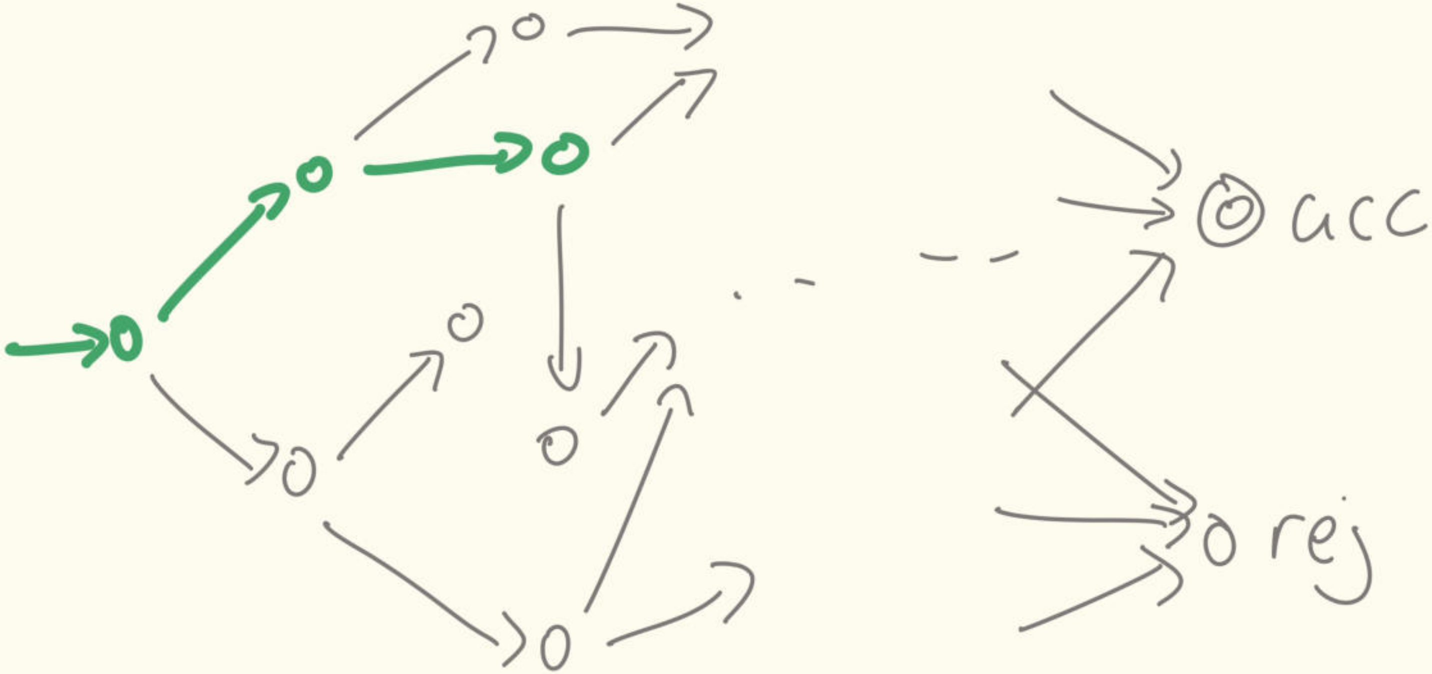
Deterministic Random Walks

R_1	R_2	R_3		R_s
1	1	1	- - - -	1

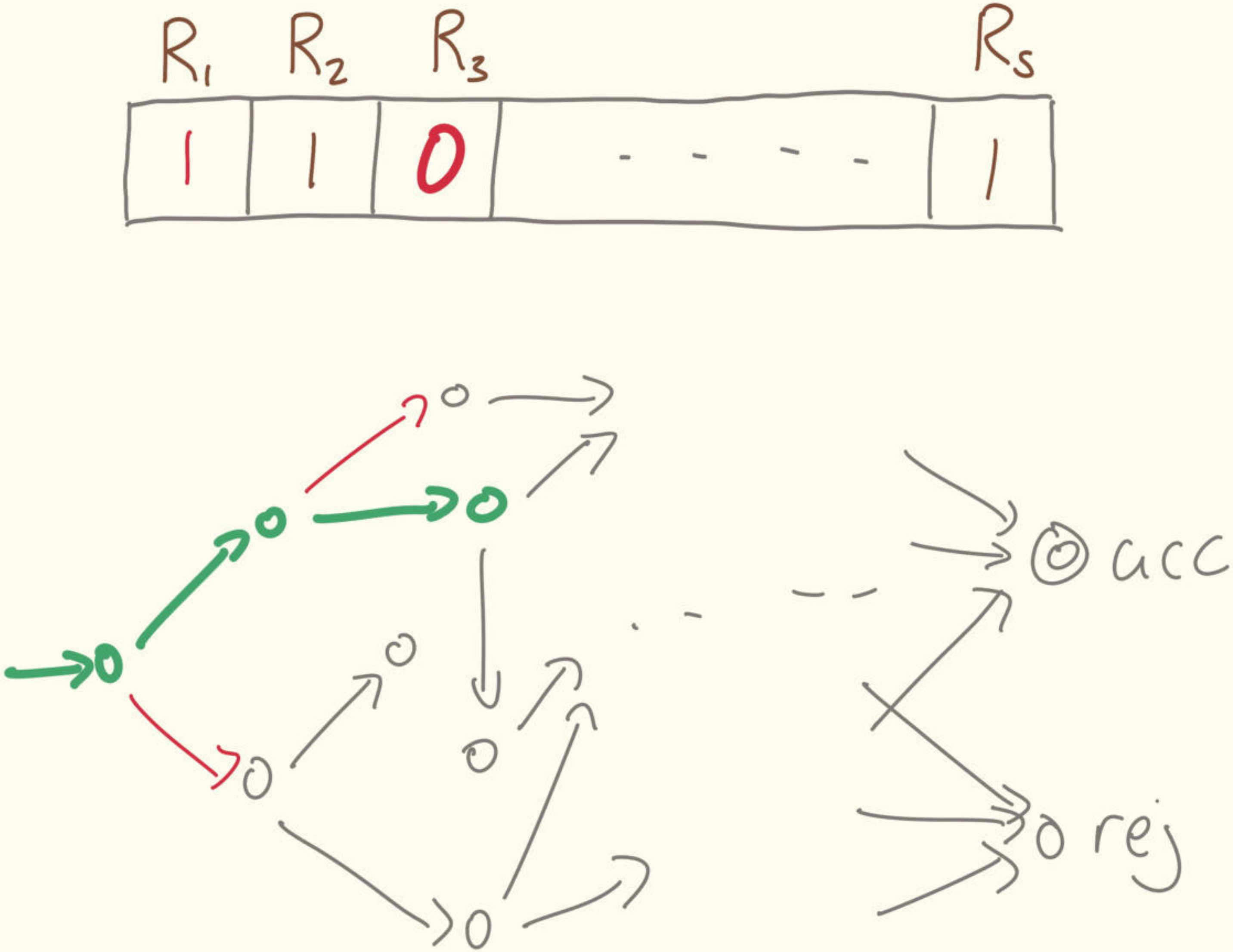


Deterministic Random Walks



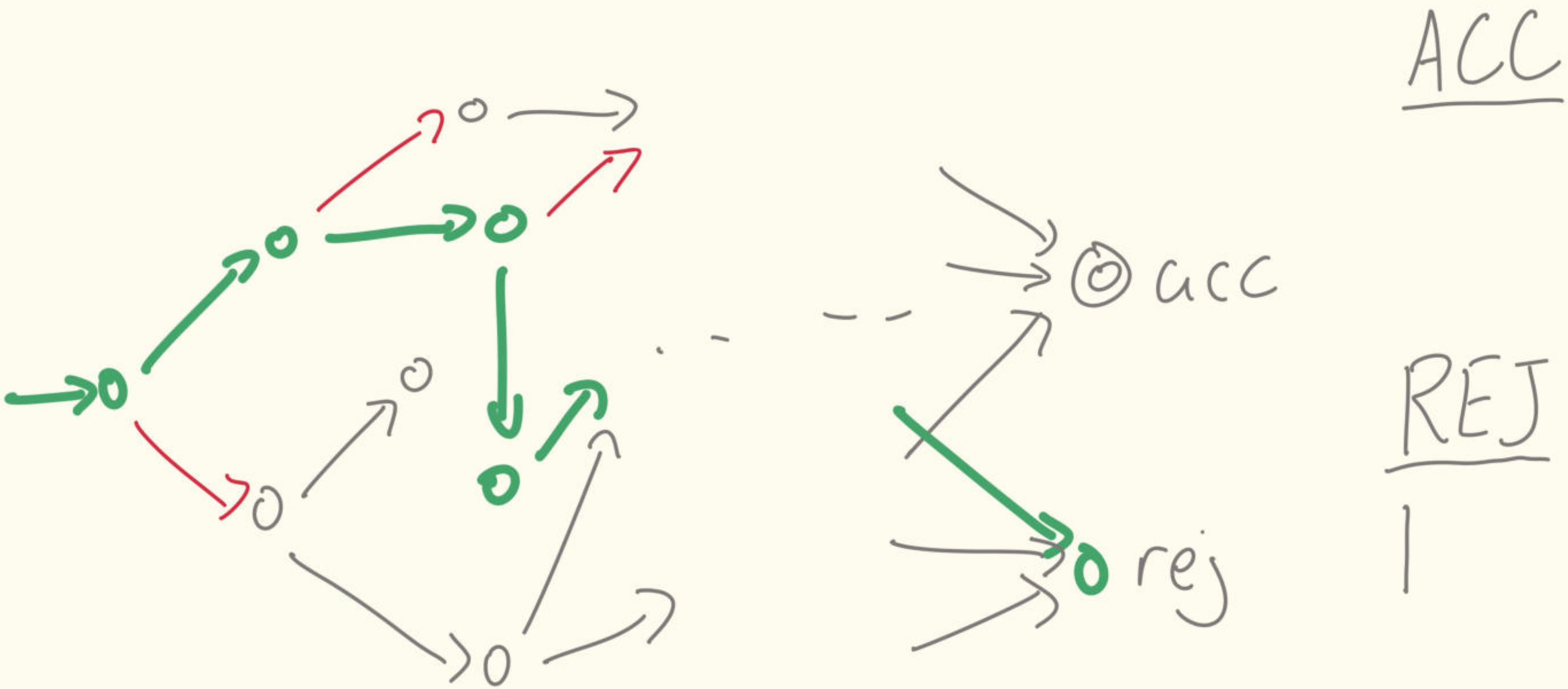


Deterministic Random Walks



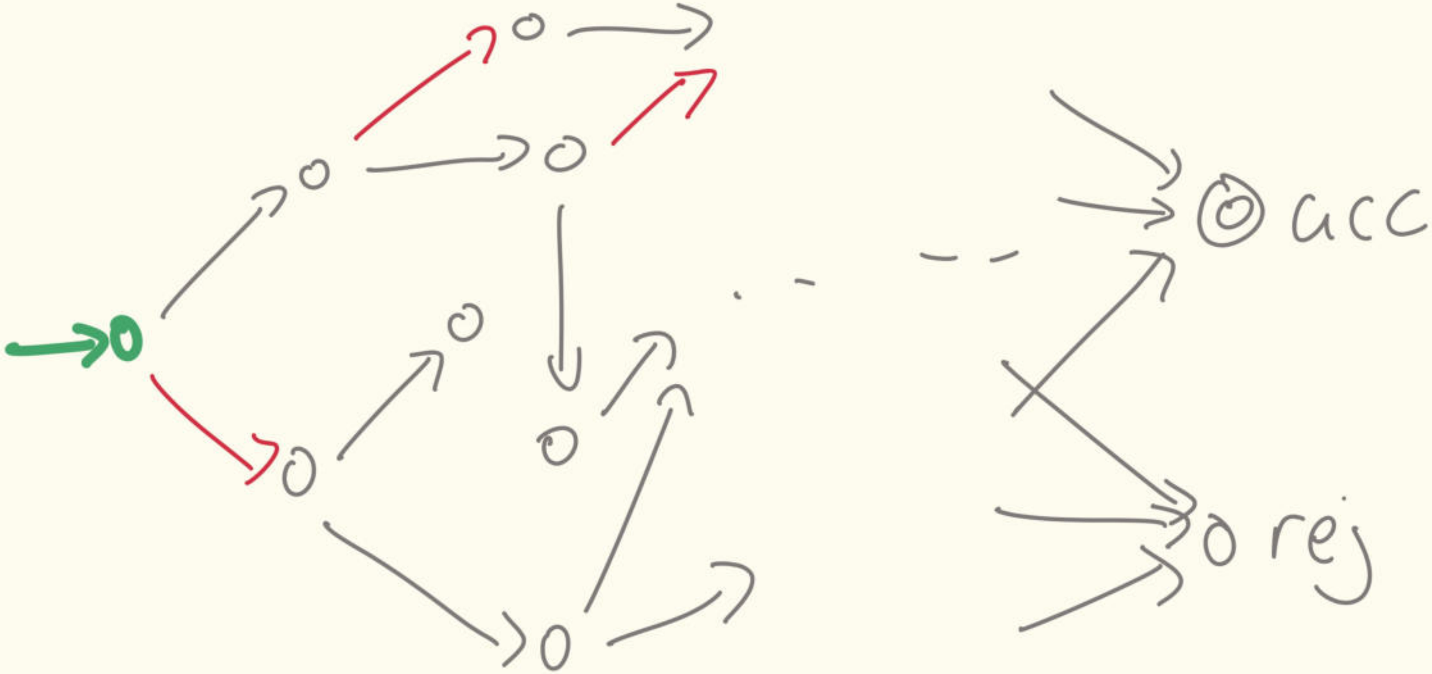
Deterministic Random Walks

R_1	R_2	R_3		R_s
1	1	0	- - - -	0



Deterministic Random Walks

R_1	R_2	R_3		R_s
1	1	0	- - - -	0

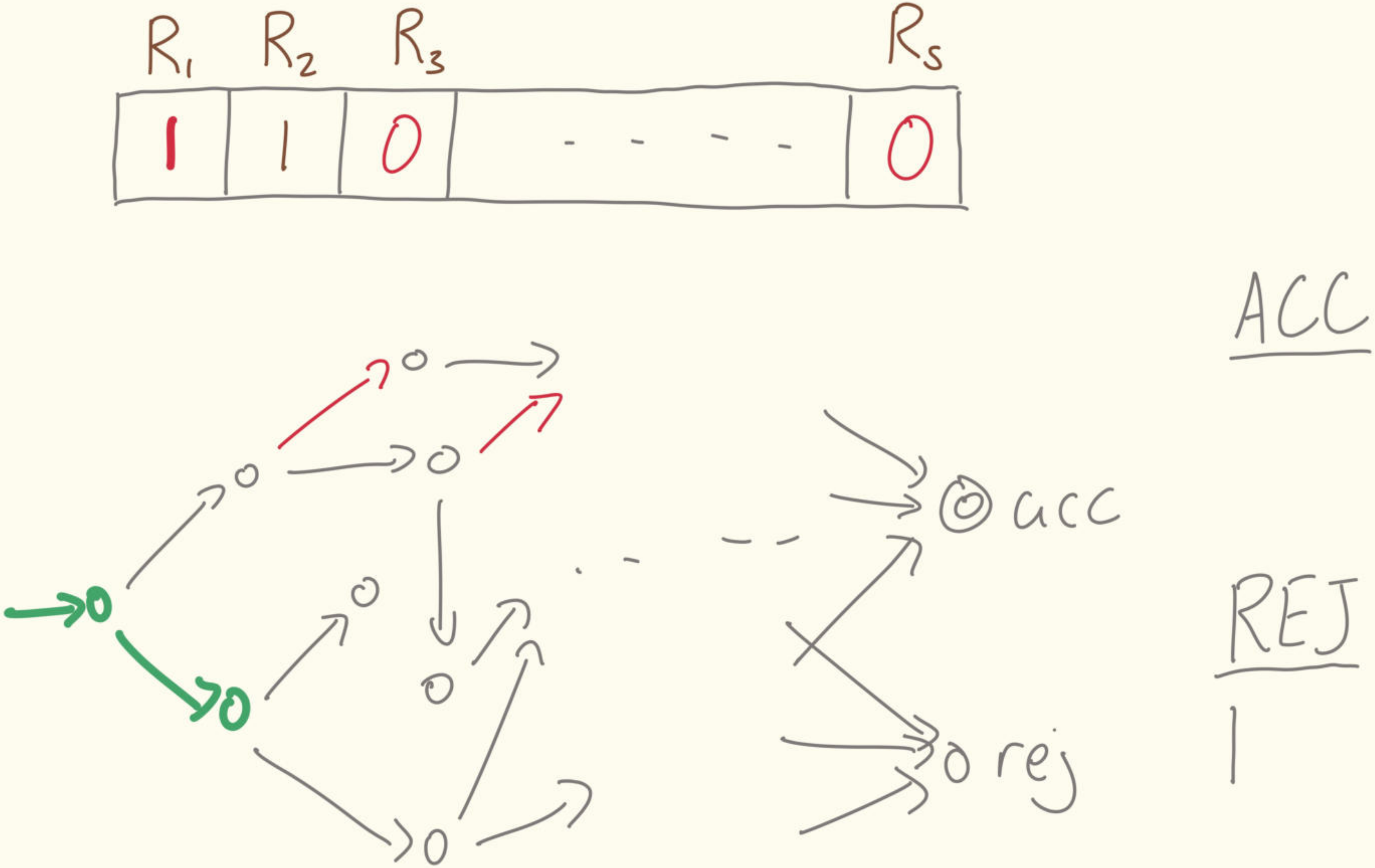


ACC

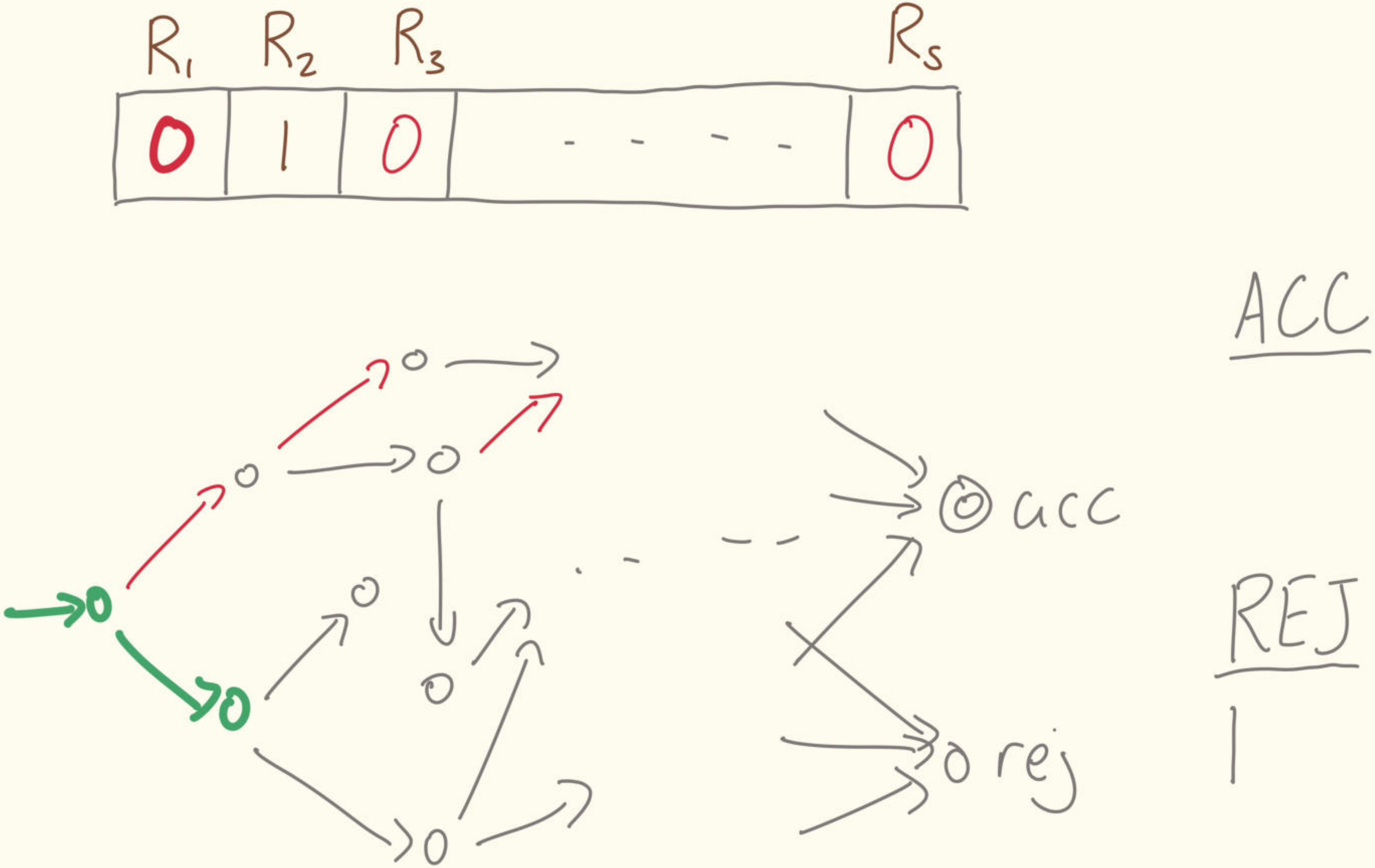
REJ

|

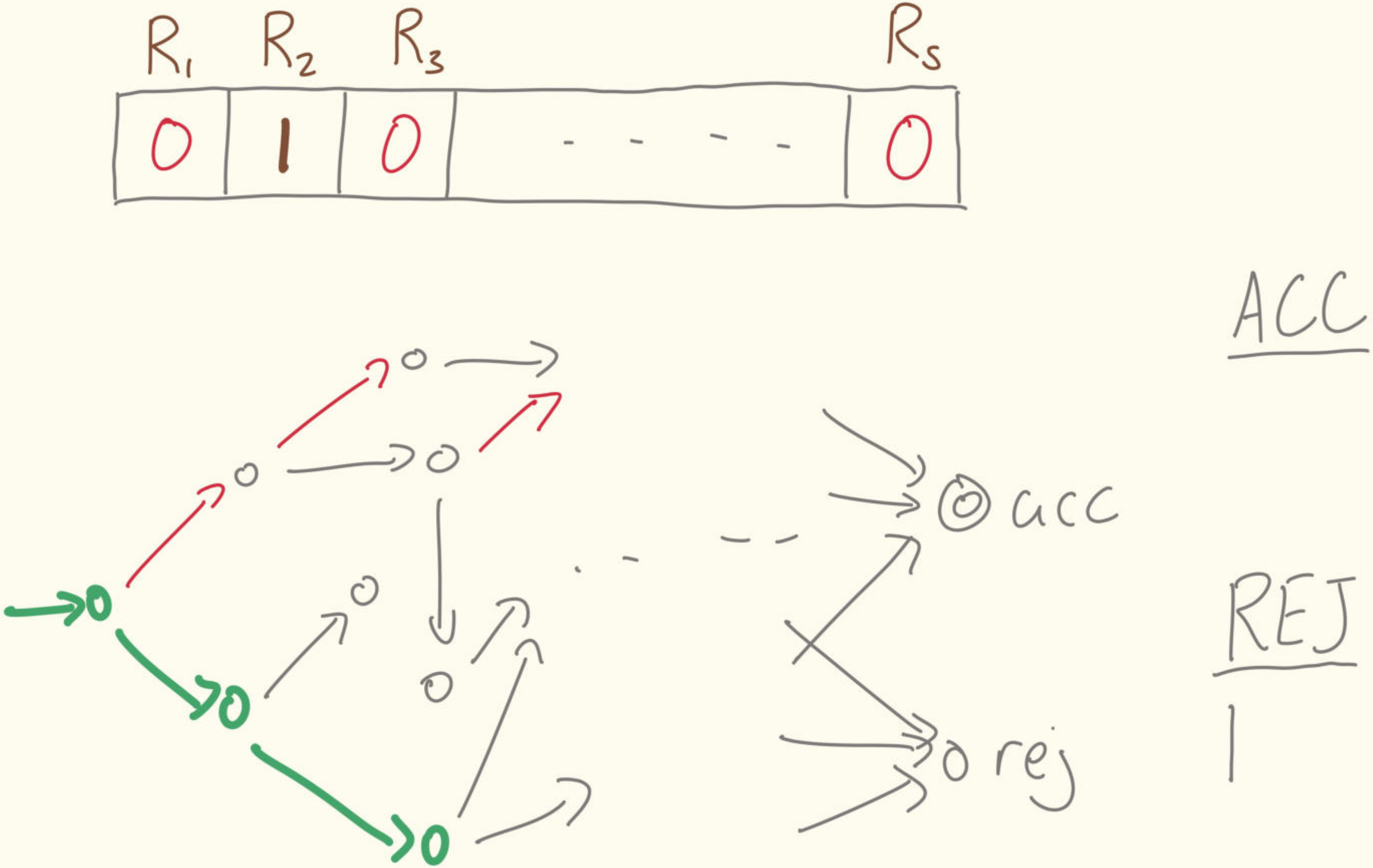
Deterministic Random Walks



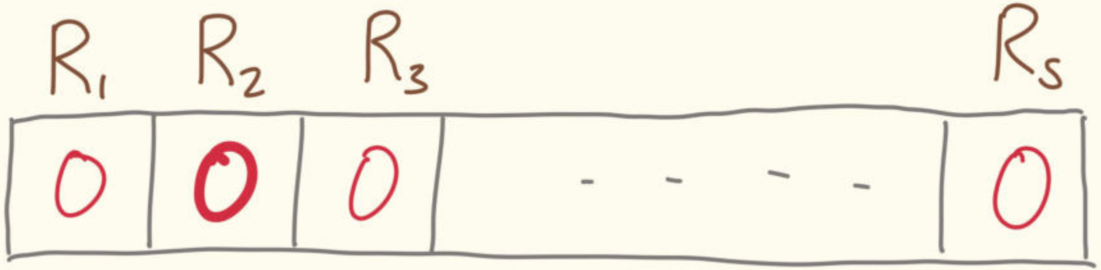
Deterministic Random Walks

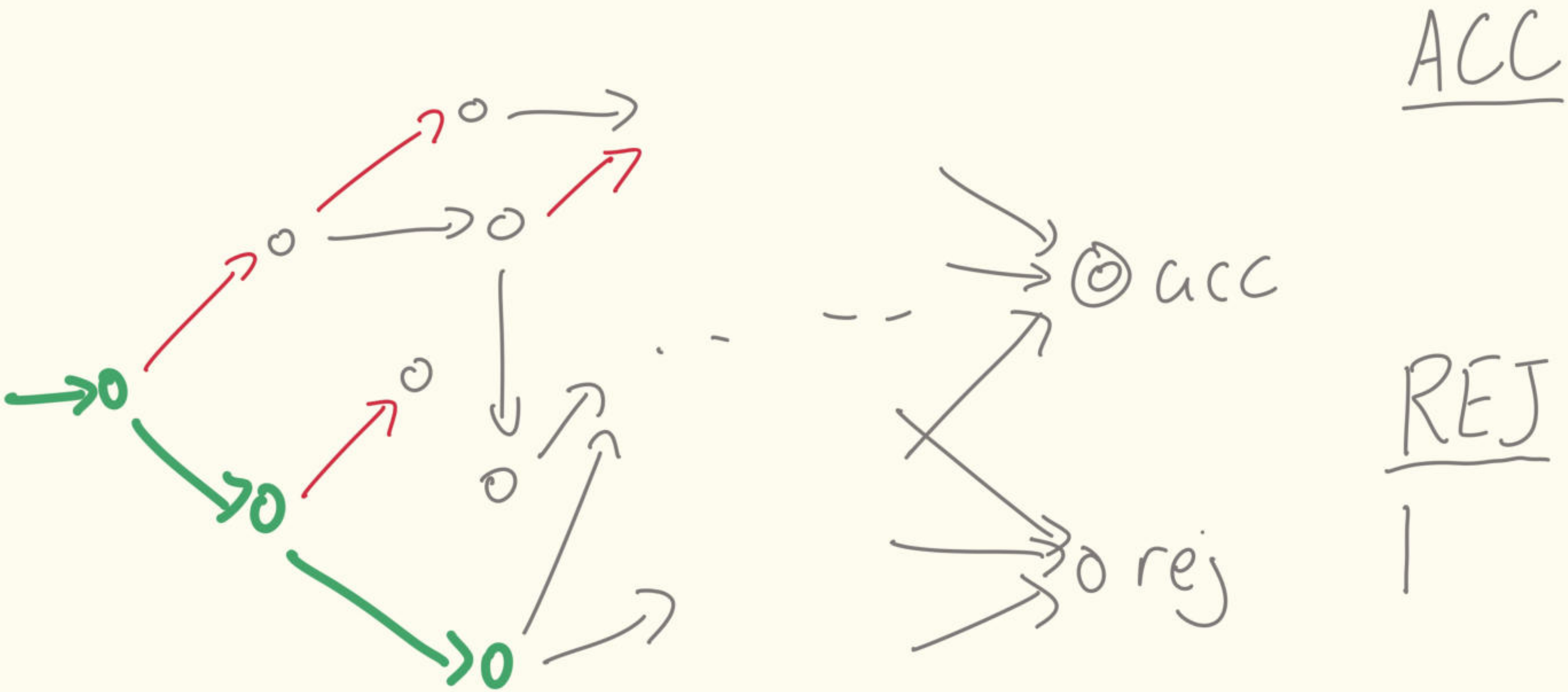


Deterministic Random Walks

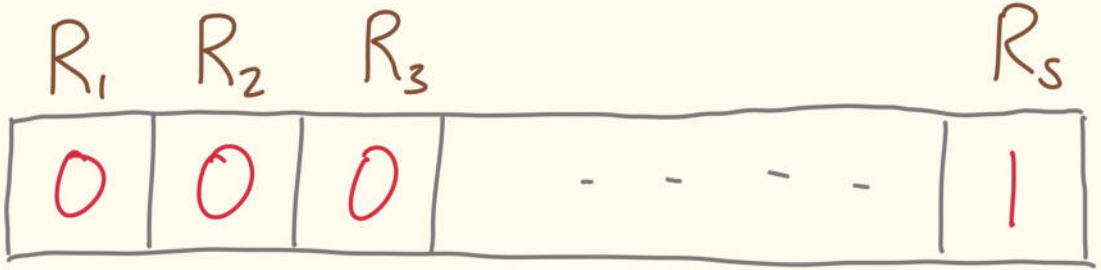


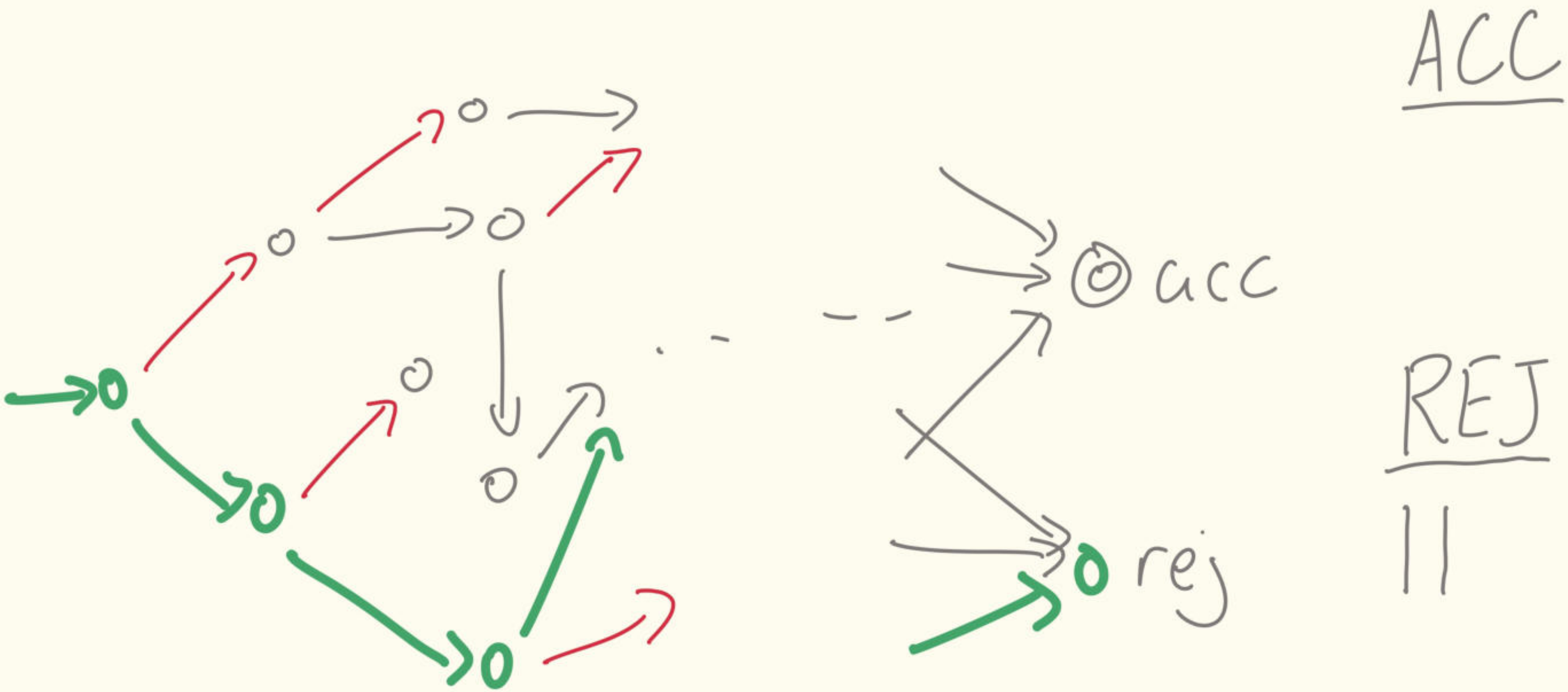
Deterministic Random Walks



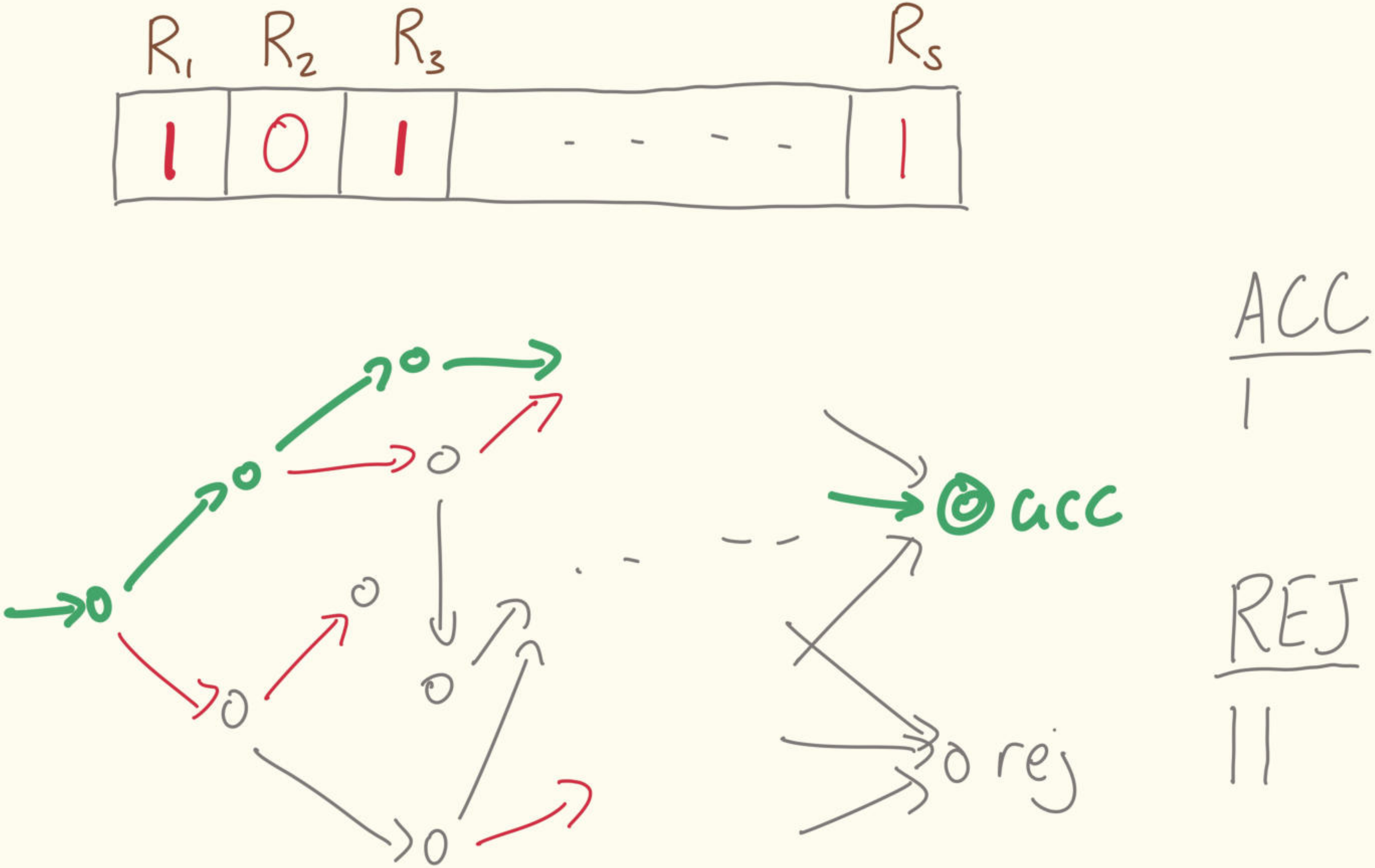


Deterministic Random Walks

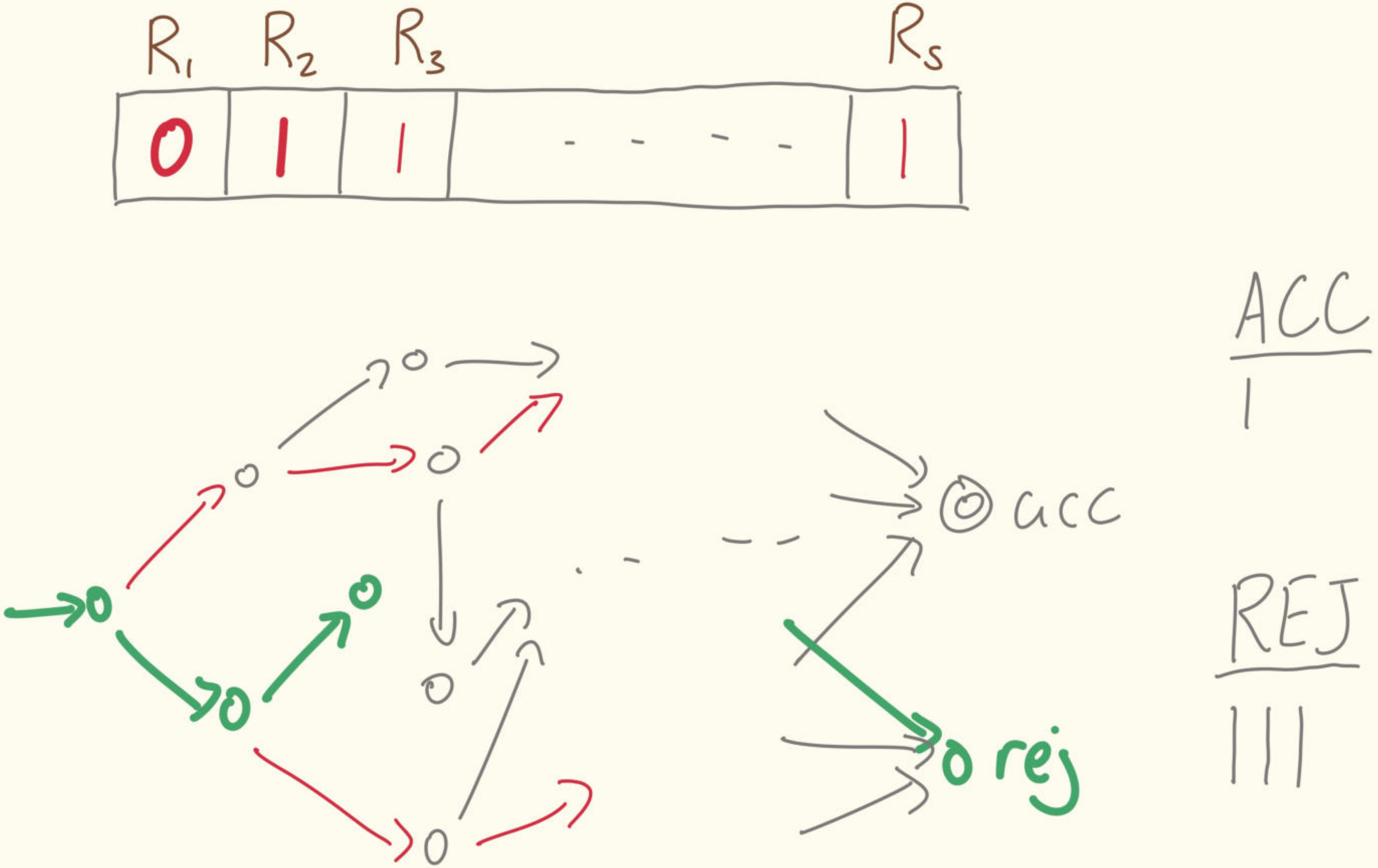




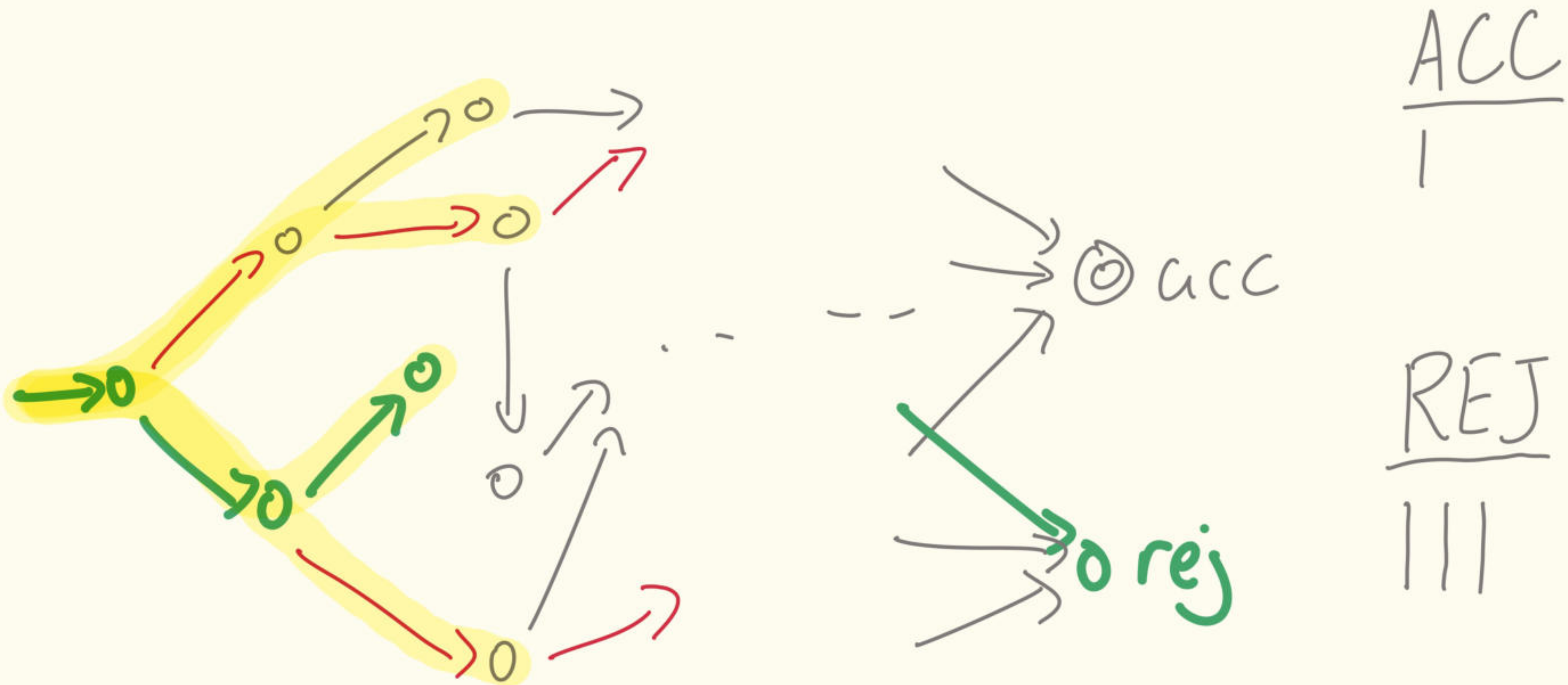
Deterministic Random Walks



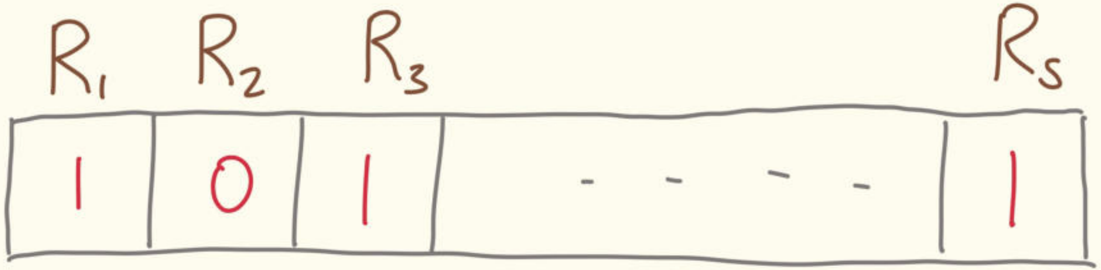
Deterministic Random Walks

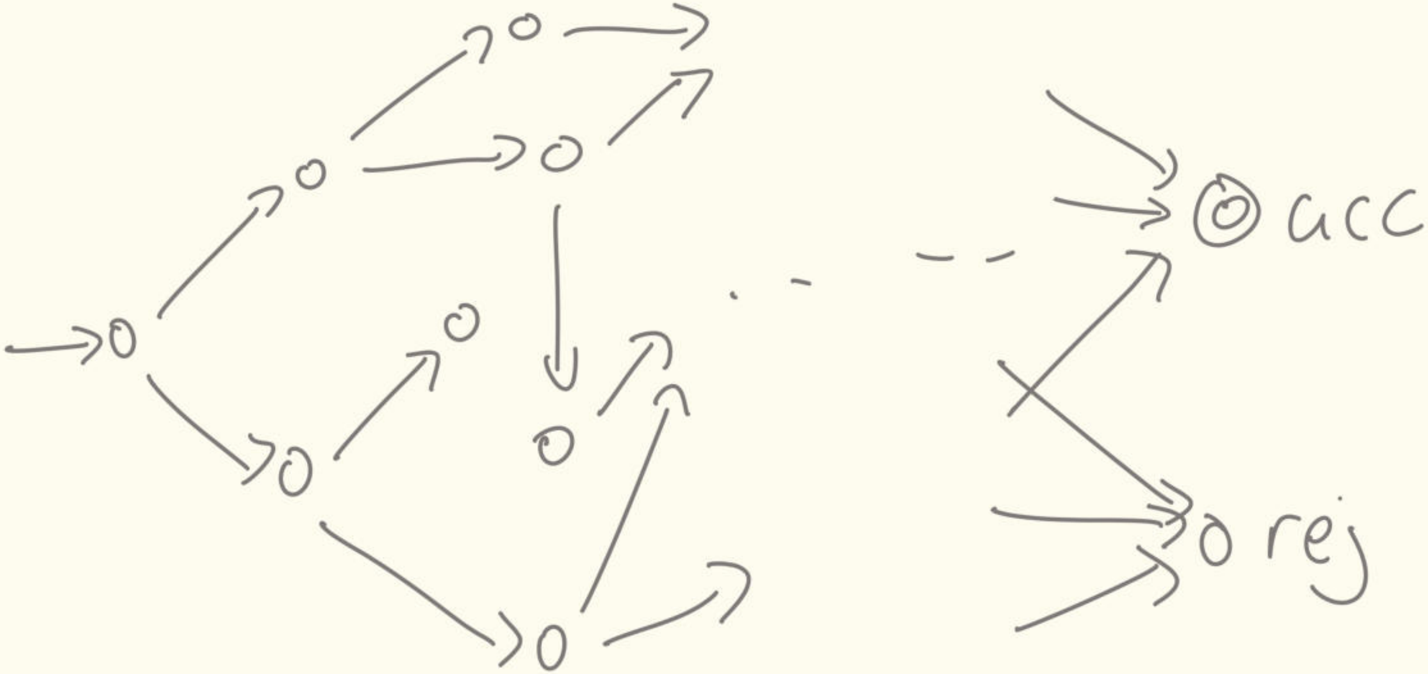


Deterministic Random Walks



Deterministic Random Walks





ACC

REJ

Deterministic Random Walks

Correctness	Resetting

Deterministic Random Walks

Correctness

if $\forall \sigma \; \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\text{MAJ}_{r \sim R} \; \mathcal{A}(r)$ correct

Resetting

Deterministic Random Walks

Correctness

if $\forall \sigma \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\text{MAJ}_{r \sim R}\ G(r)$ correct

---

next bit of r is current R_σ

Resetting

Deterministic Random Walks

Correctness

if $\forall \sigma \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\text{MAJ}_{r \sim R}\ \mathcal{G}(r)$ correct

---

next bit of r is current R_σ

$\rightarrow R_\sigma$ flipped next time

Resetting

DETERMINISTIC RANDOM WALKS

CORRECTNESS

if $\forall \sigma \ \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r \sim R}\ \mathcal{G}(r)$ correct

---

next bit of r is current R_σ

$\rightarrow$ R_σ flipped next time

$\rightarrow$ bias is basically 0 ∎

RESETTING

Deterministic Random Walks

Correctness

if $\forall_\sigma \Pr_{r\sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r\sim R}\ \mathcal{G}(r)$ correct

---

next bit of r is current R_σ

→ R_σ flipped next time

→ bias is basically 0 ■

Resetting

walk loop:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

flip $R_{\sigma_{curr}}$

Deterministic Random Walks

Correctness

if $\forall \sigma \; \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r \sim R} \; \mathcal{A}(r)$ correct

---

next bit of r is current R_σ

$\rightarrow$ R_σ flipped next time

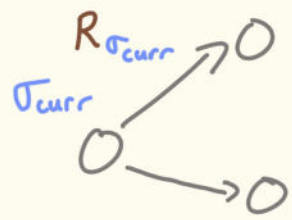
$\rightarrow$ bias is basically 0 ∎

Resetting

walk loop:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

flip $R_{\sigma_{curr}}$



Deterministic Random Walks

Correctness

if $\forall \sigma \; \Pr_{r\sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r\sim R}\, \mathcal{G}(r)$ correct

---

next bit of r is current R_σ

→ R_σ flipped next time

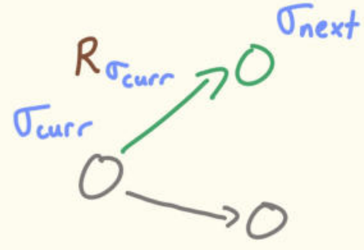
→ bias is basically 0 ∎

Resetting

walk loop:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

flip $R_{\sigma_{curr}}$



Deterministic Random Walks

Correctness

if $\forall \sigma \ \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r \sim R}\ G(r)$ correct

---

next bit of r is current R_σ

$\rightarrow$ R_σ flipped next time

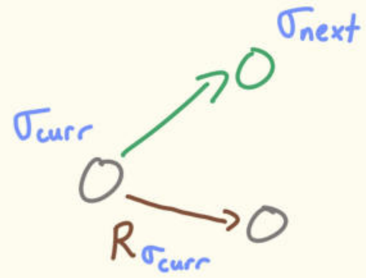
$\rightarrow$ bias is basically 0 ∎

Resetting

walk loop:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

flip $R_{\sigma_{curr}}$



Deterministic Random Walks

Correctness

if $\forall \sigma \ \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r \sim R}\ \mathcal{A}(r)$ correct

---

next bit of r is current R_σ

→ R_σ flipped next time

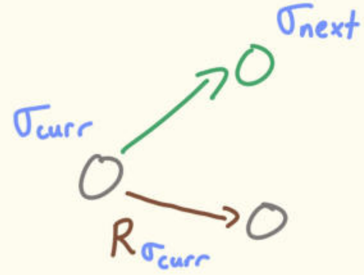
→ bias is basically 0 $\blacksquare$

Resetting

walk loop:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

flip $R_{\sigma_{curr}}$



reset loop:

flip $R_{\sigma_{curr}}$

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

Deterministic Random Walks

Correctness

if $\forall \sigma \; \Pr_{r\sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r\sim R}\; G(r)$ correct

---

next bit of r is current R_σ

→ R_σ flipped next time

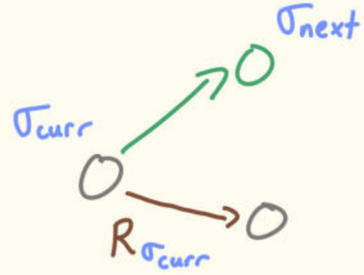
→ bias is basically 0 ∎

Resetting

walk loop:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

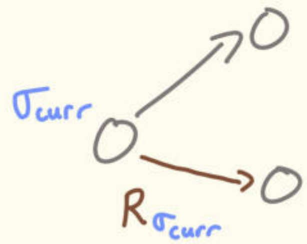
flip $R_{\sigma_{curr}}$



reset loop:

flip $R_{\sigma_{curr}}$

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$



Deterministic Random Walks

Correctness

if $\forall \sigma \; \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\text{MAJ}_{r \sim R} \; \mathcal{A}(r)$ correct

---

next bit of r is current R_σ

→ R_σ flipped next time

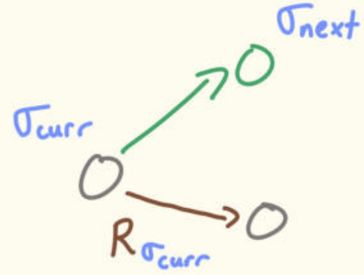
→ bias is basically 0 ∎

Resetting

walk loop:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

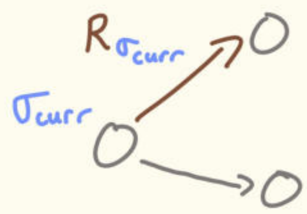
flip $R_{\sigma_{curr}}$



reset loop:

flip $R_{\sigma_{curr}}$

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$



Deterministic Random Walks

Correctness

if $\forall_\sigma \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r \sim R}\ \mathcal{A}(r)$ correct

---

next bit of r is current R_σ

→ R_σ flipped next time

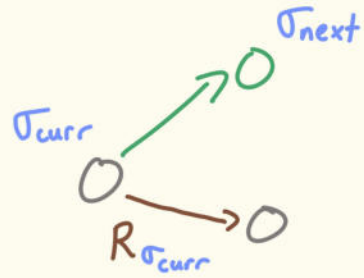
→ bias is basically 0 ∎

Resetting

walk loop:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

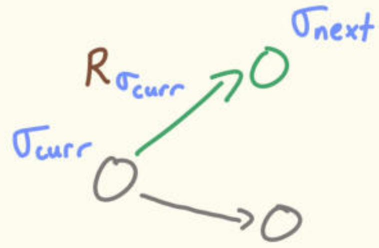
flip $R_{\sigma_{curr}}$



reset loop:

flip $R_{\sigma_{curr}}$

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$



Deterministic Random Walks

Correctness

if $\forall_\sigma \Pr_{r\sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\mathrm{MAJ}_{r\sim R}\ G(r)$ correct

---

next bit of r is current R_σ

$\rightarrow$ R_σ flipped next time

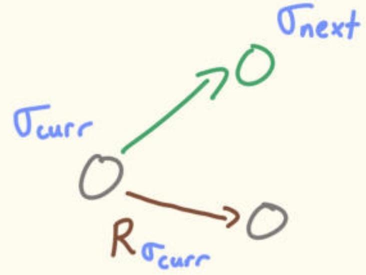
$\rightarrow$ bias is basically 0 ∎

Resetting

repeat until sink:

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

flip $R_{\sigma_{curr}}$

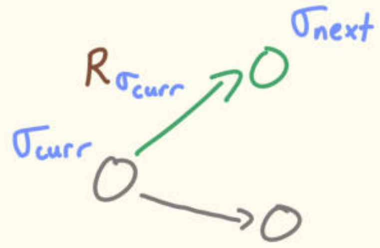


$\Uparrow$ RESET ✓

repeat until sink:

flip $R_{\sigma_{curr}}$

$\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$



Deterministic Random Walks

Correctness

if $\forall \sigma \ \Pr_{r \sim R_\sigma}(\text{next bit of } r \text{ is } 1) = \frac{1}{2} \pm \frac{1}{n^3}$

then $\text{MAJ}_{r \sim R}\ \mathcal{A}(r)$ correct

---

next bit of r is current R_σ

→ R_σ flipped next time

→ bias is basically 0 ∎

Resetting

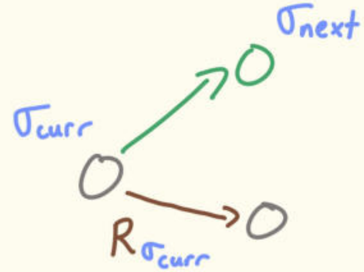
repeat n^3 times:

  repeat until sink:

    $\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$

    flip $R_{\sigma_{curr}}$

  count[sink]++

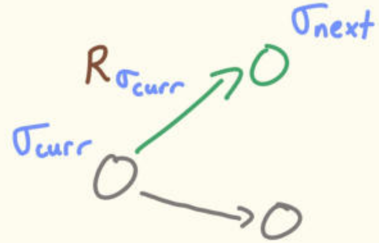


repeat n^3 times:

  repeat until sink:

    flip $R_{\sigma_{curr}}$

    $\sigma_{next} \leftarrow \sigma_{curr}[R_{\sigma_{curr}}]$



return $\text{MAJ}_{b \in \{0,1\}}(\text{count}[b])$ ∎

The Power of Reuse

[CP'25]:

CL algorithms for NL, #L, etc.

The Power of Reuse

[CP'25]:

CL algorithms for NL, #L, etc.

[Coo'21, She'25 (blogs)]:

CL algorithms for TC', MM

The Power of Reuse

[CP'25]:

CL algorithms for NL, #L, etc.

[Coo'21, She'25 (blogs)]:

CL algorithms for TC', MM

[CGMPS'25]:

in-place algorithms, structuring catalysts

Today

What is...

... a Random Walk? ... Full Memory?

How?

[Dul'15] [BCKLS'14] [CP'25]

Compression | Arithmetic Reversibility | The Easy Way

What Next?

SUMMING UP

To take random walks with full memory,

SUMMING UP

To take random walks with full memory,

... compress non-random memory

Summing Up

To take random walks with full memory,

... compress non-random memory

... use arithmetic reversibility

Summing Up

To take random walks with full memory,

... compress non-random memory

... use arithmetic reversibility

... add some cleverness

SUMMING UP

To take random walks with full memory,

... compress non-random memory

... use arithmetic reversibility

... add some cleverness

and probably more!

SUMMING UP

To take random walks with full memory,

OUR TWO MAIN CATALYTIC TECHNIQUES

... compress non-random memory

... use arithmetic reversibility

... add some cleverness

and probably more!

SUMMING UP

To take random walks with full memory,

OUR TWO MAIN CATALYTIC TECHNIQUES

... compress non-random memory

... use arithmetic reversibility

A NEW FRONTIER

... add some cleverness

and probably more!

The Power of Reuse

* $S := \mathsf{SPACE}$
$CS := \mathsf{CSPACE}$

	Classical	Catalytic
Classical	$\mathsf{TEP} \in S[\log n \cdot \log\log n]$ $\mathsf{TIME}[t] \subseteq S[\sqrt{t \log t}]$ "$\mathsf{BPL} \subseteq \mathsf{NL}$ or $\mathsf{NL} \subseteq \mathsf{SC}^2$"	$\mathsf{CL} \subseteq \mathsf{ZPP}$ $\mathsf{CL} \leq$ Lossy-Code $\mathsf{QCL} \subseteq \mathsf{EQP}$
Catalytic	$\mathsf{TC}^1 \subseteq \mathsf{CL}$ Match $\in \mathsf{CL}$ $\mathsf{BPL} \subseteq CS[\log n, \log^2 n]$ $\exists O: \mathsf{EXP}^O \subseteq \mathsf{CL}^O$	$\mathsf{CBPL} \subseteq \mathsf{CL}$ $\mathsf{CNL} \subseteq \mathsf{CL}$ $\mathsf{CPrL} \subseteq \mathsf{CL}$ $\mathsf{LCS}[s, c, e] = CS[s + e \log c, c]$

Resources

Resources

1. Give a simple, direct proof of uSTConn $\in$ L.
2. Give a simple, direct proof of uSTConn $\in$ CL.
3. Give a simple, direct proof of STConn $\in$ CL.
4. Try to improve Savich's Theorem: prove NSPACE$(s) \subseteq$ SPACE$(o(s^2))$.
5. Improve the deterministic space complexity of BPSPACE(s).
6. Decide the space complexity of TreeEval.
7. Give a register program for computing any polynomial $p(x_1 \dots x_n)$ using $O(n)$ registers over a constant size ring $\mathcal{R}$ and $O(1)$ recursive calls to the input x.
8. Show that for any branching program B of sufficiently large width $w = \Omega(1)$ and length ℓ, there exists a branching program B' of width $w/2$ and length $O(\ell)$ computing the same function.
9. Show that for any branching program B of sufficiently large width w and length ℓ, there exists a branching program B' of width $w - 1$ and length poly (ℓ) computing the same function.
10. Find any function whose optimal space algorithm can be made almost entirely catalytic, i.e. a function requiring—or even that we only know how to do in—SPACE(s) but which is computable in CSPACE$(\ll s, \approx s)$.
11. Prove CL $\subsetneq$ P.
12. Show that P $\not\subseteq$ L/poly implies CL $\subsetneq$ P.
13. Show that CL $\subsetneq$ P would give strong evidence ZPP $\subsetneq$ P.
14. Show that NC^2, or even any circuit of $\omega(\log n)$ depth, can be computed in CL.
15. Give a register program for computing x^k in the non-commutative setting using linear space and a constant number of recursive calls to x.
16. Show that BPNC^1 $\subseteq$ CL.
17. Design a catalytic branching program with $2^{O(n)}$ start nodes and total size $2^{O(n)} \cdot O(n)$ for any function f.
18. What is the power of CL$/poly$, and does it have a natural syntactic characterization?
19. Show the existence of an oracle D such that CL^D = EXP^D.
20. Extend the BPL $\subseteq$ CL simulation to show CBPL $\subseteq$ CL.
21. Show that CL is equivalent even if we allow $\omega(1)$ many errors on the catalytic tape at the end, or alternatively if we allow $O(1)$ such errors in expectation over all inputs x and catalytic tapes τ.
22. Utilize non-determinism in conjunction with catalytic computing in a non-trivial way.
23. Prove CNSPACE$(s, c) \subseteq$ CSPACE(s^2, c^2).
24. Implement a catalytic algorithm such that it is actually useful.
25. What does quantum catalytic space look like?
26. Devise a register program using basic instructions inspired by unitary computation, and use it to show non-trivial results for e.g. BQP.
27. Devise a circuit that uses known results from space reuse and catalytic computing to efficiently solve some problem in a way that we do not know how to do directly.
28. Show TC^1 $\subseteq$ VP.
29. Is the network coding conjecture true or false?
30. Prove or disprove the network coding conjecture when all nodes are restricted to sending linear transformations of their incoming messages.
31. Is there a meaningful notion of a catalytic data structure, or is there anything to be gained from a data structure stored in catalytic memory?
32. Show CL is contained in some subclass of P, perhaps NC, given a believable cryptographic assumption.
33. Show evidence against objects in cryptography based on techniques in reusing space.
34. Show the existence, conditional or otherwise, of a natural class of cryptographic objects by using clean computation.
35. Prove that the existence of one-way functions in CL, or even any one-way function computable by a poly-size poly-length register program, implies the existence of one-way functions in NC^0.

source: [Mer'23]

Resources

Reusing Space: Techniques and Open Problems

Ian Mertz*

Abstract

In the world of space-bounded complexity, there is a strain of results showing that space can, somewhat paradoxically, be used for multiple purposes at once. Touchstone results include Barrington's Theorem and the recent line of work on catalytic computing. We refer to such techniques, in contrast to the usual notion of reclaiming space, as *reusing space*.

In this survey we will dip our toes into the world of reusing space. We do so in part by studying techniques, viewed through the lens of a few highlight results, but our main focus will be the wide variety of open problems in the field.

In addition to the broader and more challenging questions, we aim to provide a number of questions that are fairly simple to state, have clear practical and theoretical implications, and, most importantly, that a newcomer with little background experience can still sit down and play with for a while.

Catalytic computation

Michal Koucký*
Computer Science Institute
Charles University, Prague
koucky@iuuk.mff.cuni.cz

Abstract

Catalytic computation was defined by Buhrman et al. (STOC, 2014). It addresses the question whether memory, that already stores some unknown data that should be preserved for later use, can be meaningfully used for computation. Buhrman et al. provide an intriguing answer to this question by giving examples where the occupied memory can be used to perform computation. In this expository article we survey what is known about this problem and how it relates to other problems.

Resources

iuuk.mff.cuni.cz/~iwmertz/

introduction [my main interests] | previous work [past activities] | methodology [teaching & students]
main results [publications] | acknowledgements [funding, positions, & visits] | appendix [the fun stuff]

catalytic computation & reusing space

How useful is *full memory* as a computational resource? Imagine trying to solve some functions on a computer with only limited memory, but now you are also given additional access to a *massive hard drive* which it can freely use, provided it *keeps all the initial data on the hard drive intact* at the end of its computation. Considering this data could be arbitrary—obviously this memory has nothing to do with the problem at hand—does this hard drive give us *any additional power*?

The surprising answer is that this hard drive, which we call *catalytic memory*, is very powerful. First, it gives us at least as much power as any other well-studied resource, be it randomness or non-determinism; in fact, *catalytic memory alone* is as powerful as being given *catalytic memory, randomness, and non-determinism simultaneously*. Second, the techniques and subroutines developed for this catalytic computation model, which I (uncreatively) refer to as *reusing space*, have given *major breakthrough results in the ordinary space-bounded setting*, mostly notably Williams' recent simulation of time t in space $\sqrt{t \log t}$.

My central goal is to understand and characterize this catalytic model, as well as to further use the techniques developed therein to solve longstanding open questions about space.

[survey (EATCS)] [techniques] **(more resources coming soon)**

techniques (catalytic computing)

Here is an overview of some of the major arguments/techniques that appear in the catalytic literature. If you're here I'm assuming you know the setup for the model; if not then check out a survey article such as mine or Michal Koucky's, or even the original paper by Buhrman et al. (a great read!) to get oriented. For more info on specifics, I would suggest checking out the resources page for suggested papers, talks, etc.

Take a click on whatever strikes your fancy!

compress-or-random

register programs

structure in catalytic space

introduction

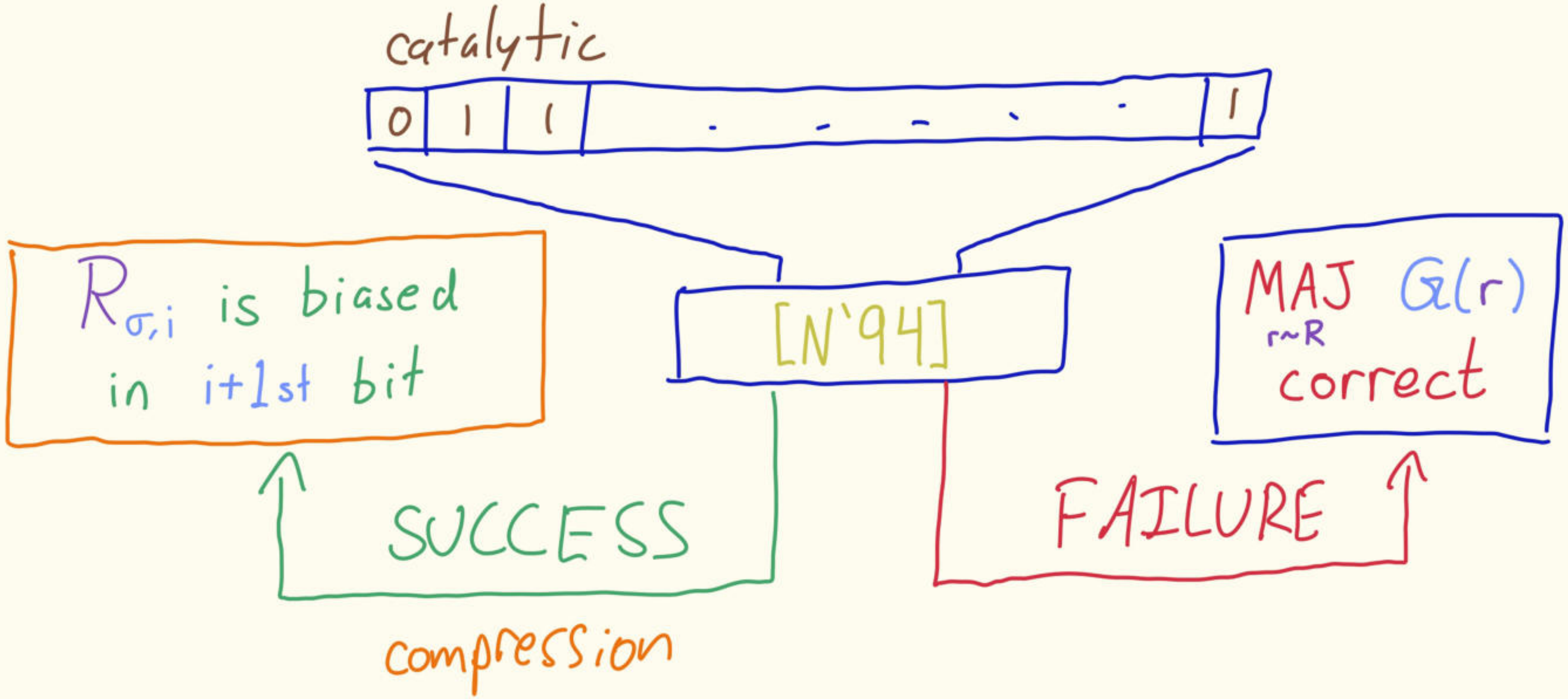
The only trivial upper bound on catalytic space is an equal amount of pure space. In order to improve this result, we need to look deeper into the structure of catalytic algorithms. We will borrow from two fundamental results on ordinary space: first, the straightforward fact that space s algorithms can be simulated in *time* $\exp(s)$; and second, the much more intriguing (and much more recently discovered) fact that all algorithms can be made *reversible* with only a constant amount of extra space.

open problems database to come...

THANKS!

Exercise Solutions

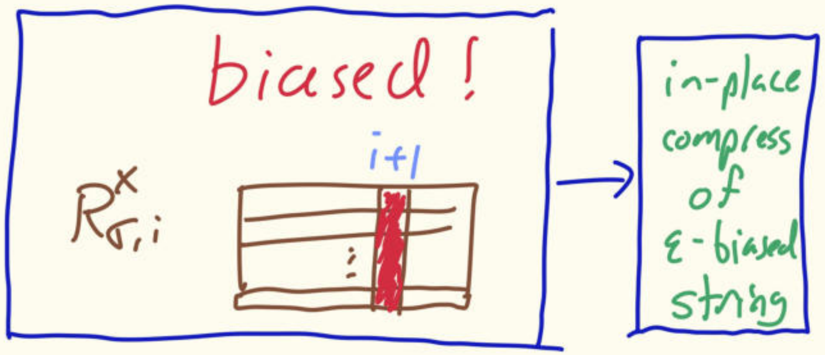
Compress-or-Random



Exercise #1

Compress-or-Random

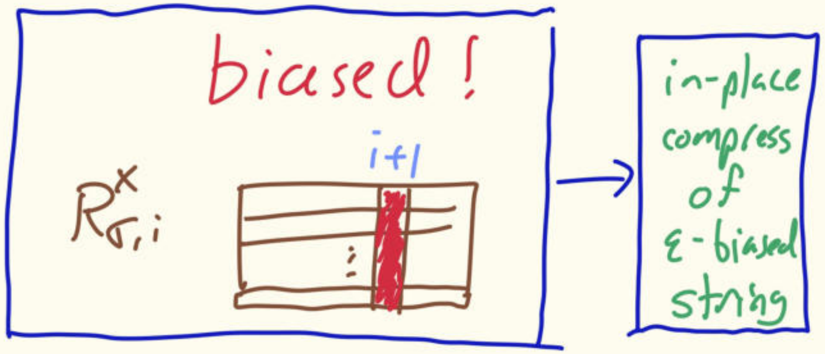
Goal: $\|\tau\| = k \leq \frac{n}{2} - m \rightarrow |\mathrm{COMP}(\tau)| \leq n - \frac{m^2}{n}$



COMPRESS-OR-RANDOM

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

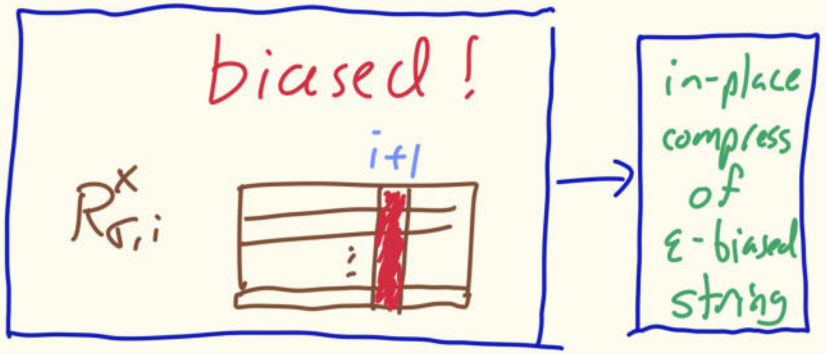
Goal: $\underbrace{\|\tau\| = k}_{} \leq \frac{n}{2} - m \rightarrow |\text{COMP}(\tau)| \leq n - \frac{m^2}{n}$



Compress-or-Random

Goal: $\|\tau\| = k \le \frac{n}{2} - m \to |\mathrm{COMP}(\tau)| \le \overbrace{n - \frac{m^2}{n}}^{\text{index into set } \{\tau' \in \{0,1\}^n \,:\, \|\tau'\| = k\}}$

COMPRESS	DECOMPRESS



Compress-or-Random

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

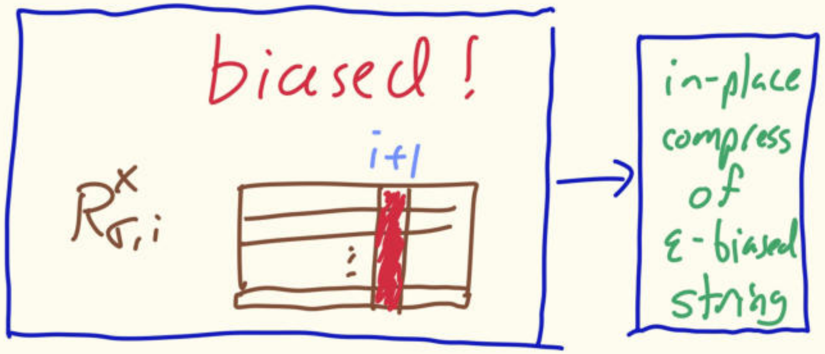
Goal : $\|\tau\| = k \le \frac{n}{2} - m \to |\mathrm{COMP}(\tau)| \le n - \frac{m^2}{n}$

COMPRESS

$k' \leftarrow \tau_1$ // # ones so far

$\tau_1 \leftarrow 0$ // base case

DECOMPRESS



COMPRESS-OR-RANDOM

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal : $\|\tau\| = k \le \frac{n}{2} - m \to |\mathrm{COMP}(\tau)| \le n - \frac{m^2}{n}$

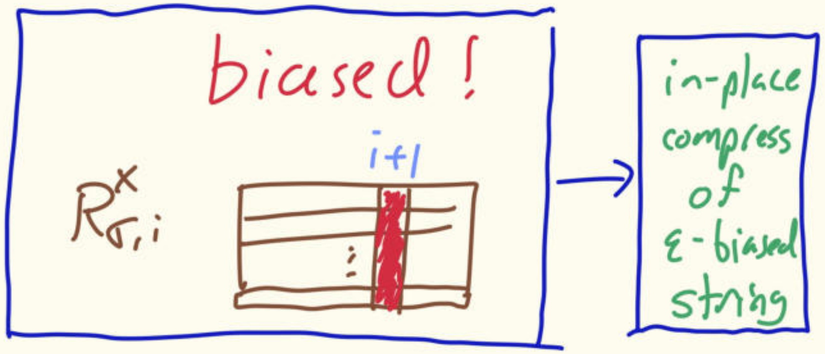
COMPRESS

$k' \leftarrow \tau_1$ // # ones so far

$\tau_1 \leftarrow 0$ // base case

for $i = 2 \ldots n$:

DECOMPRESS



COMPRESS-OR-RANDOM

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal : $\|\tau\| = k \le \frac{n}{2} - m \rightarrow |\mathrm{COMP}(\tau)| \le n - \frac{m^2}{n}$

COMPRESS

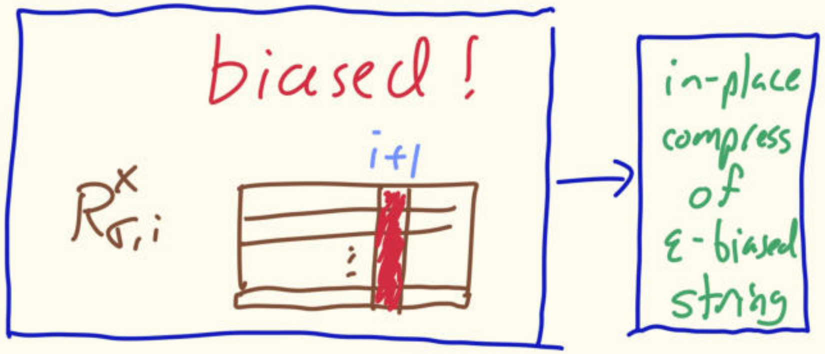
$k' \leftarrow \tau_1$ // # ones so far

$\tau_1 \leftarrow 0$ // base case

for $i = 2 \ldots n$:

if $\tau_i = 0$, do nothing

DECOMPRESS



Compress-or-Random

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal: $\|\tau\| = k \leq \frac{n}{2} - m \rightarrow |\mathrm{COMP}(\tau)| \leq n - \frac{m^2}{n}$

COMPRESS

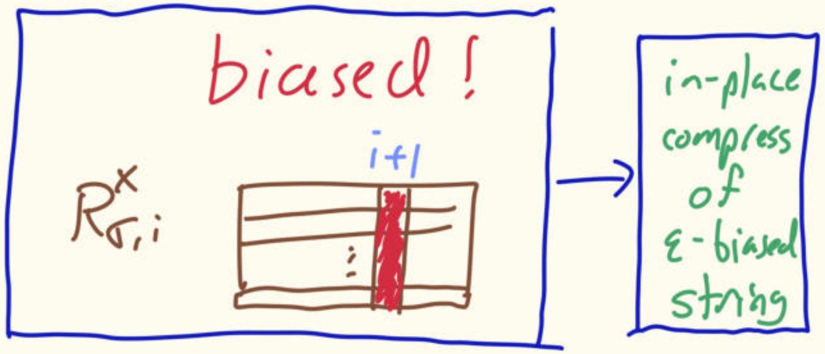
$k' \leftarrow \tau_1$ // # ones so far

$\tau_1 \leftarrow 0$ // base case

for $i = 2 \ldots n$:

- if $\tau_i = 0$, do nothing
- if $\tau_i = 1$:

DECOMPRESS



COMPRESS-OR-RANDOM

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal: $\|\tau\| = k \leq \frac{n}{2} - m \rightarrow |\mathrm{COMP}(\tau)| \leq n - \frac{m^2}{n}$

COMPRESS

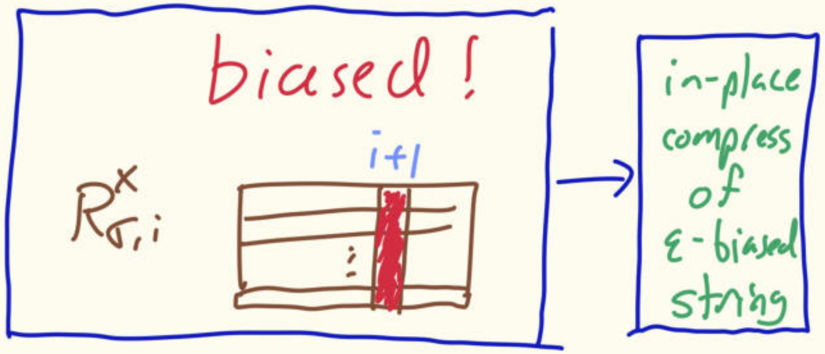
$k' \leftarrow \tau_1$ // # ones so far

$\tau_1 \leftarrow 0$ // base case

for $i = 2 \ldots n$:

- if $\tau_i = 0$, do nothing
- if $\tau_i = 1$:
  - k'++

DECOMPRESS



COMPRESS-OR-RANDOM

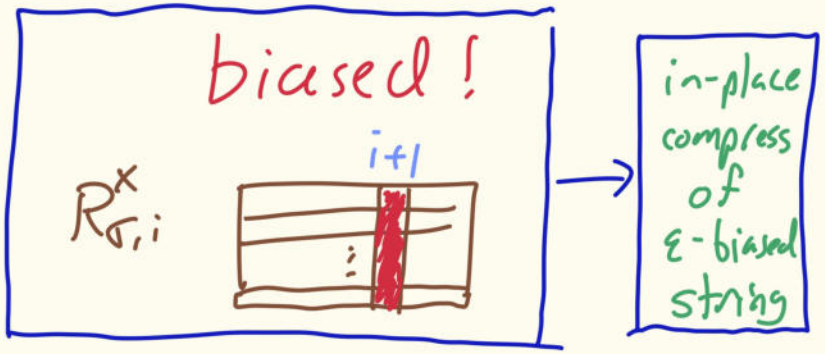
index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal: $\|\tau\| = k \leq \frac{n}{2} - m \rightarrow |\mathrm{COMP}(\tau)| \leq n - \frac{m^2}{n}$

COMPRESS

```
k' ← τ_1          // # ones so far
τ_1 ← 0           // base case
for i = 2...n :
    if τ_i = 0 , do nothing
    if τ_i = 1 :
        k' ++
        τ_i ← 0
```

DECOMPRESS



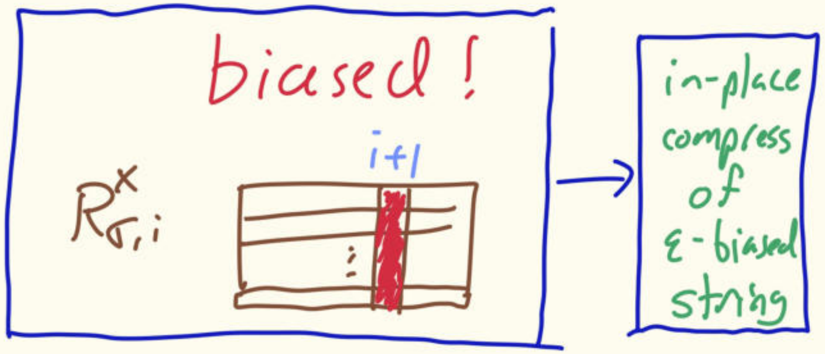
Compress-or-Random

$$\text{Goal}: \|\tau\| = k \le \frac{n}{2} - m \rightarrow \Big|\overbrace{\mathrm{COMP}(\tau)}^{\text{index into set } \{\tau' \in \{0,1\}^n \,:\, \|\tau'\| = k\}}\Big| \le n - \frac{m^2}{n}$$

COMPRESS

```
k' ← τ_1        // # ones so far
τ_1 ← 0         // base case
for i = 2...n :
    if τ_i = 0 , do nothing
    if τ_i = 1 :
        k'++
        τ_i ← 0
        τ += (i-1 choose k')
```

DECOMPRESS



Compress-or-Random

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal : $\|\tau\| = k \le \frac{n}{2} - m \to |\mathrm{COMP}(\tau)| \le n - \frac{m^2}{n}$

COMPRESS

$k' \leftarrow \tau_1$ // # ones so far

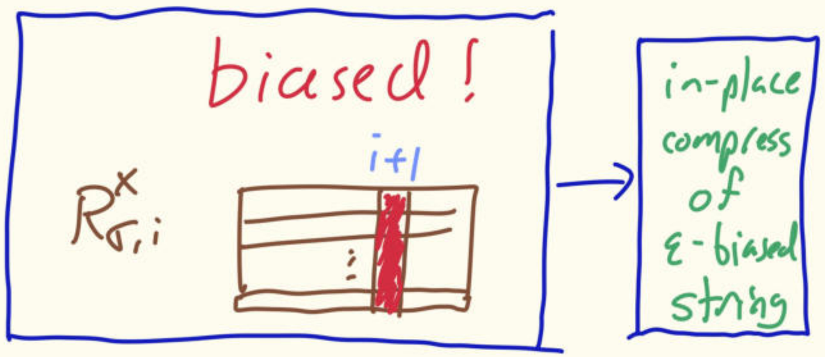
$\tau_1 \leftarrow 0$ // base case

for $i = 2 \ldots n$:

- if $\tau_i = 0$, do nothing
- if $\tau_i = 1$:
  - k'++
  - $\tau_i \leftarrow 0$
  - $\tau \mathrel{+}= \binom{i-1}{k'}$

return τ // $\tau_i = 0 \;\; \forall i \ge n - \frac{m^2}{n}$

DECOMPRESS



COMPRESS-OR-RANDOM

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal: $\|\tau\| = k \le \frac{n}{2} - m \rightarrow |\text{COMP}(\tau)| \le n - \frac{m^2}{n}$

COMPRESS

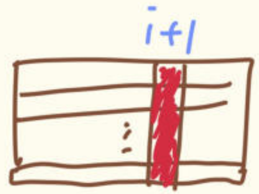
```
k' ← τ_1          // # ones so far
τ_1 ← 0           // base case
for i = 2 ... n :
    if τ_i = 0 , do nothing
    if τ_i = 1 :
        k' ++
        τ_i ← 0
        τ += (i-1 choose k')
return τ          // τ_i = 0  ∀ i ≥ n - m²/n
```

DECOMPRESS

$k' \leftarrow k$ // # ones unaccounted for so far

biased!

$R^x_{\sigma,i}$



in-place compress of ε-biased string

COMPRESS-OR-RANDOM

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal: $\|\tau\| = k \le \frac{n}{2} - m \rightarrow |\text{COMP}(\tau)| \le n - \frac{m^2}{n}$

COMPRESS

$k' \leftarrow \tau_1$ // # ones so far

$\tau_1 \leftarrow 0$ // base case

for $i = 2 \ldots n$:

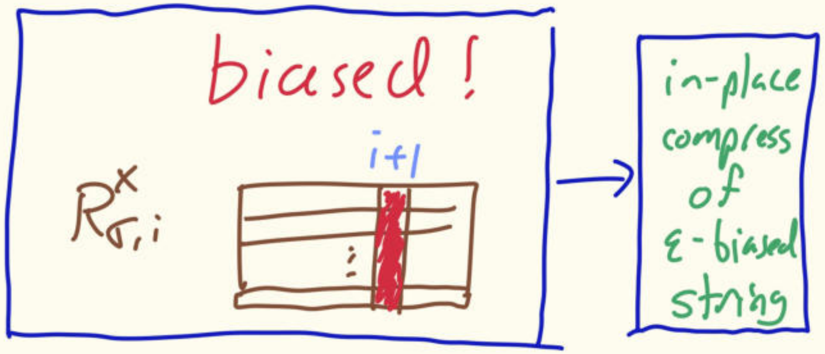
- if $\tau_i = 0$, do nothing
- if $\tau_i = 1$:
  - k'++
  - $\tau_i \leftarrow 0$
  - $\tau \mathrel{+}= \binom{i-1}{k'}$

return τ // $\tau_i = 0 \ \forall i \ge n - \frac{m^2}{n}$

DECOMPRESS

$k' \leftarrow k$ // # ones unaccounted for so far

for $i = n \ldots 2$:



COMPRESS-OR-RANDOM

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal : $\|\tau\| = k \leq \frac{n}{2} - m \rightarrow |\text{COMP}(\tau)| \leq n - \frac{m^2}{n}$

COMPRESS

$k' \leftarrow \tau_1$ // # ones so far

$\tau_1 \leftarrow 0$ // base case

for $i = 2 \ldots n$:

  if $\tau_i = 0$, do nothing

  if $\tau_i = 1$:

    k'++

    $\tau_i \leftarrow 0$

    $\tau \mathrel{+}= \binom{i-1}{k'}$

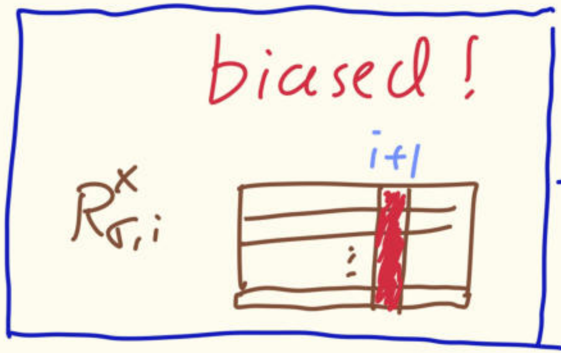
return τ // $\tau_i = 0 \quad \forall i \geq n - \frac{m^2}{n}$

DECOMPRESS

$k' \leftarrow k$ // # ones unacconted for so far

for $i = n \ldots 2$:

  if $\binom{i-1}{k'} > \tau$, do nothing



$\rightarrow$ in-place compress of ε-biased string

Compress-or-Random

index into set
$\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

Goal: $\|\tau\| = k \le \frac{n}{2} - m \rightarrow |\mathrm{COMP}(\tau)| \le n - \frac{m^2}{n}$

COMPRESS

$k' \leftarrow \tau_1$ // # ones so far

$\tau_1 \leftarrow 0$ // base case

for $i = 2 \ldots n$:

- if $\tau_i = 0$, do nothing
- if $\tau_i = 1$:
  - k' ++
  - $\tau_i \leftarrow 0$
  - $\tau \mathrel{+}= \binom{i-1}{k'}$

return τ // $\tau_i = 0 \;\; \forall i \ge n - \frac{m^2}{n}$

DECOMPRESS

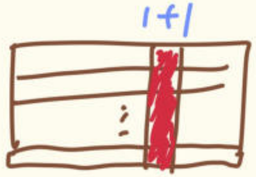
$k' \leftarrow k$ // # ones unaccounted for so far

for $i = n \ldots 2$:

- if $\binom{i-1}{k'} > \tau$, do nothing
- if $\binom{i-1}{k'} \le \tau$:

biased!

$R^x_{\sigma, i}$



$\rightarrow$ in-place compress of ε-biased string

Compress-or-Random

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

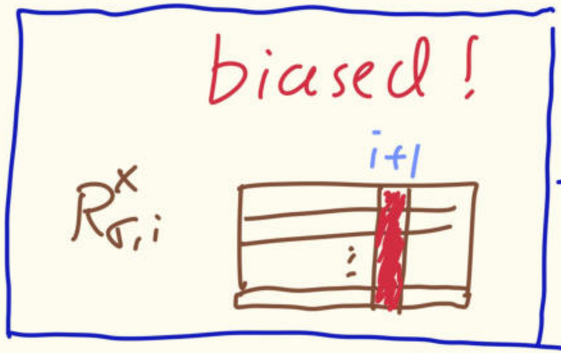
$$\text{Goal}: \underbrace{\|\tau\|}_{} = k \le \frac{n}{2} - m \rightarrow |\text{COMP}(\tau)| \le n - \frac{m^2}{n}$$

COMPRESS

```
k' ← τ_1          // # ones so far
τ_1 ← 0           // base case
for i = 2 ... n :
 | if τ_i = 0 , do nothing
 | if τ_i = 1 :
 |  | k' ++
 |  | τ_i ← 0
 |  | τ += (i-1 choose k')
return τ   // τ_i = 0  ∀ i ≥ n - m²/n
```

DECOMPRESS

```
k' ← k            // # ones unacconted for so far
for i = n ... 2 :
 | if (i-1 choose k') > τ , do nothing
 | if (i-1 choose k') ≤ τ :
 |  | τ -= (i-1 choose k')
 |  | τ_i ← 1
 |  | k' --
```



in-place compress of ε-biased string

COMPRESS-OR-RANDOM

index into set $\{\tau' \in \{0,1\}^n : \|\tau'\| = k\}$

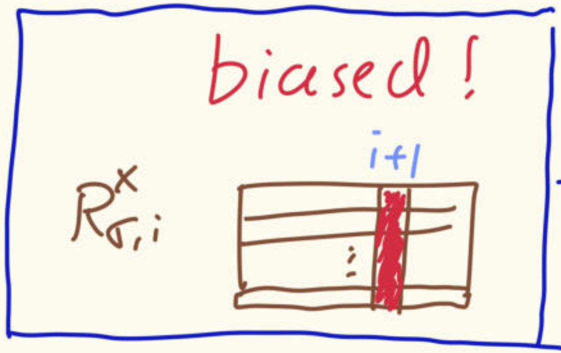
Goal: $\|\tau\| = k \leq \frac{n}{2} - m \rightarrow |\mathrm{COMP}(\tau)| \leq n - \frac{m^2}{n}$

COMPRESS

```
k' ← τ_1        // # ones so far
τ_1 ← 0         // base case
for i = 2...n :
  | if τ_i = 0 , do nothing
  | if τ_i = 1 :
  |   | k' ++
  |   | τ_i ← 0
  |   | τ += (i-1 choose k')
return τ        // τ_i = 0  ∀ i ≥ n - m²/n
```

DECOMPRESS

```
k' ← k          // # ones unaccounted for so far
for i = n...2 :
  | if (i-1 choose k') > τ, do nothing
  | if (i-1 choose k') ≤ τ :
  |   | τ -= (i-1 choose k')
  |   | τ_i ← 1
  |   | k' --
τ_1 ← k'        // base case; k' ∈ {0,1}
return τ
```



in-place compress of ε-biased string

Reusing Space

$$P_x^{(-1)} : R_1 \underset{(-)}{+}= x$$

$$P_y^{(-1)} : R_2 \underset{(-)}{+}= y$$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$$\boxed{P_{MULT}^{(-1)} : R_3 \underset{(-)}{+}= xy}$$

Exercise #2

Reusing Space

$P_x^{(\cdot 1)}$: $R_1 \underset{(-)}{+}= x$

$P_y^{(\cdot 1)}$: $R_2 \underset{(-)}{+}= y$

[BC'92]:

$P_{MULT}^{(\cdot 1)}$:

R_1	R_2	R_3
τ_1	τ_2	τ_3
$+x$	$+y$	

1. P_x, P_y

Reusing Space

$P_x^{(-1)}$: $R_1 \overset{+}{\underset{(-)}{}}= x$

$P_y^{(-1)}$: $R_2 \overset{+}{\underset{(-)}{}}= y$

[BC'92]:

$P_{MULT}^{(-1)}$:

R_1	R_2	R_3
τ_1	τ_2	τ_3
$+x$	$+y$	

1. P_x, P_y
2. $R_3 \mathrel{+}= R_1 R_2$

$+\tau_1\tau_2 + x\tau_2$

$+\tau_1 y + xy$

Reusing Space

$P_x^{(-1)}$: $R_1 \underset{(-)}{+}= x$

$P_y^{(-1)}$: $R_2 \underset{(-)}{+}= y$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3
$+x$		

$P_{MULT}^{(-1)}$:

1. P_x, P_y
2. $R_3 \mathrel{+}= R_1 R_2$
3. P_y^{-1}
4. $R_3 \mathrel{-}= R_1 R_2$

$$+\cancel{\tau_1\tau_2} + \cancel{x\tau_2}$$
$$+\tau_1 y + xy$$
$$-\cancel{\tau_1\tau_2} - \cancel{x\tau_2}$$

Reusing Space

$P_x^{(\cdot 1)}: \; R_1 \underset{(-)}{+}= x$

$P_y^{(\cdot 1)}: \; R_2 \underset{(-)}{+}= y$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$+y$

$P_{MULT}^{(\cdot 1)}$:

1. P_x, P_y
2. $R_3 \mathrel{+}= R_1 R_2$
3. P_y^{-1}
4. $R_3 \mathrel{-}= R_1 R_2$
5. P_x^{-1}, P_y
6. $R_3 \mathrel{-}= R_1 R_2$

$$+\cancel{T_1 T_2} + \cancel{x T_2}$$
$$+\cancel{T_1 y} + xy$$
$$-\cancel{T_1 T_2} - \cancel{x T_2}$$
$$-T_1 T_2 - \cancel{T_1 y}$$

REUSING SPACE

$$P_x^{(-1)}: \; R_1 \underset{(-)}{+}= x$$
$$P_y^{(-1)}: \; R_2 \underset{(-)}{+}= y$$

[BC'92]:

R_1	R_2	R_3
τ_1	τ_2	τ_3

$P_{MULT}^{(-1)}$:

1. P_x, P_y
2. $R_3 \mathrel{+}= R_1 R_2$
3. P_y^{-1}
4. $R_3 \mathrel{-}= R_1 R_2$
5. P_x^{-1}, P_y
6. $R_3 \mathrel{-}= R_1 R_2$
7. P_y^{-1}
8. $R_3 \mathrel{+}= R_1 R_2$

$$+\cancel{\tau_1\tau_2} + \cancel{x\tau_2}$$
$$+\cancel{\tau_1 y} + xy$$
$$-\cancel{\tau_1\tau_2} - \cancel{x\tau_2}$$
$$-\cancel{\tau_1\tau_2} - \cancel{\tau_1 y}$$
$$+\cancel{\tau_1\tau_2}$$