Efficient Catalytic Algorithms for All-Pairs Shortest Paths

Alex Liu and Anthony Zou
Mentor: Zoe Xi (MIT)

Green Level High School • River Bluff High School

October 24–25, 2026
MIT PRIMES Conference

Can used memory help you compute?

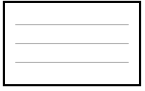
You have a **tiny notebook**: room for only a few numbers.

A friend lends you a **huge hard drive**, completely full of their files.

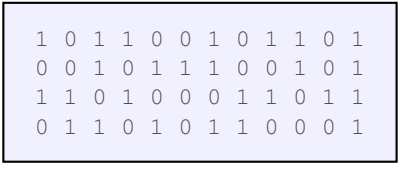
The rules:

- read and write anything you like;
- give it back **exactly** as you found it.

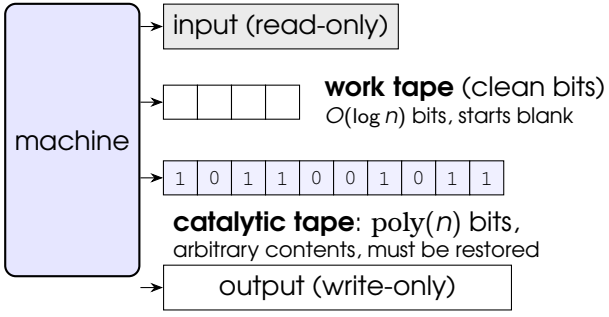
notebook (blank)



hard drive (full of someone's data)



The catalytic model



Catalytic logspace, CL

Problems solvable with

- $O(\log n)$ clean bits,
- a polynomial-size catalytic tape,
- the tape restored at the end.

The key trick: compute, use, uncompute

Registers on the catalytic tape hold **garbage** we are not allowed to lose. Suppose a reversible program P adds a value x into register R.

Step	**OUT**	**R**
start (garbage)	$\mathrm{OUT}_{\text{init}}$	R_{init}
$\mathrm{OUT} \leftarrow \mathrm{OUT} - \mathrm{R}$	$\mathrm{OUT}_{\text{init}} - \mathrm{R}_{\text{init}}$	R_{init}
run P (adds x to R)	$\mathrm{OUT}_{\text{init}} - \mathrm{R}_{\text{init}}$	$\mathrm{R}_{\text{init}} + x$
$\mathrm{OUT} \leftarrow \mathrm{OUT} + \mathrm{R}$	$\mathrm{OUT}_{\text{init}} + x$	$\mathrm{R}_{\text{init}} + x$
run P^{-1} (undo P)	$\mathrm{OUT}_{\text{init}} + x$	R_{init}

Result

$\mathrm{OUT} \leftarrow \mathrm{OUT}_{\text{init}} + x$.

R is restored to R_{init}.

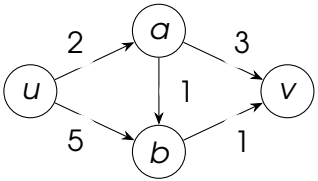
Progression of catalytic computing

2014	Catalytic Logspace (CL) introduced for computation circuits by Buhrman et. al.
2024 – 25	Space-efficient tree evaluation in CL by Cook and Mertz.
2026	First algorithms for graph problems in CL by Cook and Pyne.
Our Work	**Fast catalytic algorithms for shortest paths between all pairs of vertices using new techniques in CL.**

All-Pairs Shortest Paths (APSP)

Problem

Given a directed graph $G = (V, E, w)$, where $w : E \to [W]$ is a weight function, $|V| = n$, and $|E| = m$, find the shortest-path distance $d[u, v]$ for *every* ordered pair of vertices u, v.



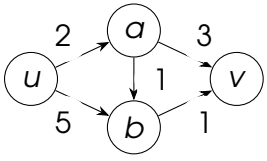
from \ to	u	a	b	v
u	0	2	3	**4**
a	∞	0	1	2
b	∞	∞	0	1
v	∞	∞	∞	0

- Output: $n \times n$ table
- ∞ = no path
- $d[u, v] = 4$: $u \to a \to b \to v$

Path doubling

Let $d_t[u, v]$ = shortest distance from u to v using **at most t edges**. Split a path of $\leq 2t$ edges at a middle vertex k:

$$d_{2t}[u, v] = \min_{k \in V} \Big(d_t[u, k] + d_t[k, v] \Big) \qquad \text{(a \textbf{min-plus product})}$$



$$d_1[u, v] = \infty \quad \text{(no direct edge)}$$

$$d_2[u, v] = \min(\underbrace{2+3}_{\text{via } a}, \underbrace{5+1}_{\text{via } b}) = 5$$

$$d_4[u, v] = \min_k \Big(d_2[u, k] + d_2[k, v] \Big) = \underbrace{3+1}_{k=b} = \mathbf{4}$$

(the path $u \to a \to b \to v$ has 3 edges)

$d_1 \to d_2 \to d_4 \to \cdots \to d_{2^{\lceil \log_2 n \rceil}}$: shortest paths have $< n$ edges

Classical APSP algorithms require lots of memory

Algorithm	Time	Working memory
Floyd–Warshall	$O(n^3)$	$\Theta(n^2)$
Repeated min-plus squaring	$O(n^3 \log n)$	$\Theta(n^2)$

Both keep a whole table of n^2 distances in memory.

Our question

Can a borrowed full memory replace the missing $\Theta(n^2)$ space — and how fast can APSP be in catalytic logspace?

Path doubling on a catalytic tape

Registers

Helpful for computation:

- $R_t[u, v] \mathrel{+}= d_t[u, v]$
- $M_t[u, k, v] \mathrel{+}= \Big(d_t[u, k] + d_t[k, v]\Big)$

Still needed

$$R_{2t}[u, v] \;\mathrel{+}=\; \min_{k \in V} \Big(d_t[u, k] + d_t[k, v]\Big)$$

for all n^2 pairs (u, v).

The difficult part: the minimum

Notice that min *forgets* information ($\min(2, 5) = \min(2, 9)$), so it is **not reversible**.

The catalytic minimum (Sheffield)

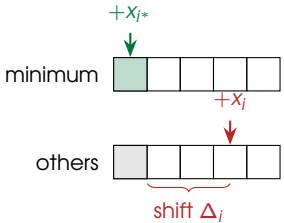
Goal: add $\min_i x_i$ to OUT, using a program P that adds each x_i to a register.

1. Compare pairs: $\delta_{i,j} = (x_i - x_j) + [i > j]$

$x = (3, 1, 1)$	$j = 1$	$j = 2$	$j = 3$	
$i = 1$	0	2	2	$\Delta_1 > 0$
$i = 2$	-1	0	0	$\mathbf{\Delta_2 = 0}$
$i = 3$	-1	1	0	$\Delta_3 > 0$

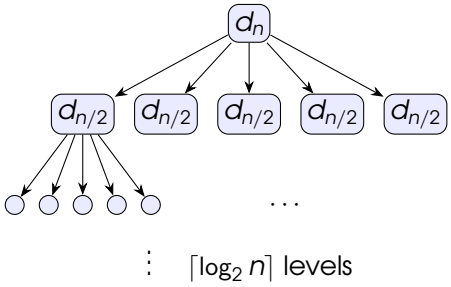
Penalty $\Delta_i = 0$ **only** for the minimum.

2. Deposit x_i, shifted by Δ_i:



Only the minimum hits a **watched** cell.

The price: a factor of 5 at every level



Each level calls the level below **5 times**:

$$5^{\lceil \log_2 n \rceil} = O\left(n^{\log_2 5}\right).$$

Each level also runs n^2 minima of cost $\widetilde{O}(n^2)$

Basic catalytic APSP

Total running time: $n^2 \cdot \widetilde{O}(n^2) \cdot O\left(n^{\log_2 5}\right) < \widetilde{O}\left(n^{6.322}\right)$

Two costs to balance: **the factor 5 per level** and **the n^2 cost of each minimum**.

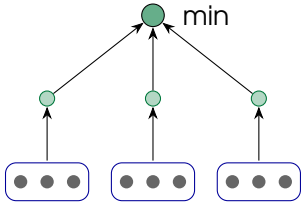
Our results

Algorithm	**Type**	**Time**
Basic catalytic path doubling	deterministic	$\widetilde{O}\left(n^{6.322}\right)$
Previous best (Sheffield, 2025)	best of either type	$\widetilde{O}\left(n^{5.967}\right)$
Ours: minimum trees	deterministic	$\boldsymbol{\widetilde{O}\left(n^{5.895}\right)}$
Ours: + random hitting sets	randomized	$\widetilde{O}\left(n^{5.754}\right)$
Ours: + merged levels	randomized	$\boldsymbol{\widetilde{O}\left(n^{5.120}\right)}$

Idea 1: minimum trees

Instead of a simple minimum of n^2 pairs, take minima of **batches**, then the minimum of the batch winners.

Depth 2: $n^{2/3}$ batches of size $n^{1/3}$, then one batch of the $n^{2/3}$ winners: $\widetilde{O}(n^{4/3})$ per round.



n candidates in batches (schematic)

depth	1	2	3	...	q
extra time (per minimum)	n^2	$n^{4/3}$	$n^{8/7}$	...	$n^{2^q/(2^q-1)}$
calls to the previous program	5	25	125	...	5^q

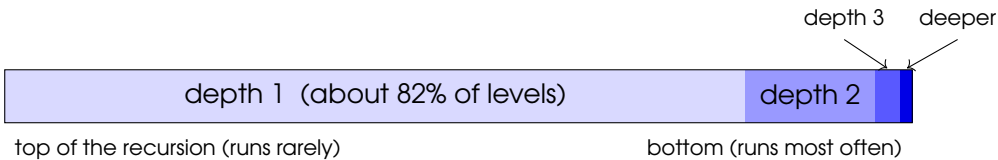
Tradeoff

Deeper tree: cheaper minimum, but **5× more calls** per extra level of depth.

Idea 1: where to spend the depth

Choose a minimum-tree depth q_ℓ separately for **each** level of path doubling.

- Extra depth at a level multiplies the work of **that level and every level below it** by 5.
- So extra depth is cheapest near the **bottom**, where minima also run most often.



Depths are chosen so that **no level dominates**.

Deterministic result

Theorem (deterministic catalytic APSP)

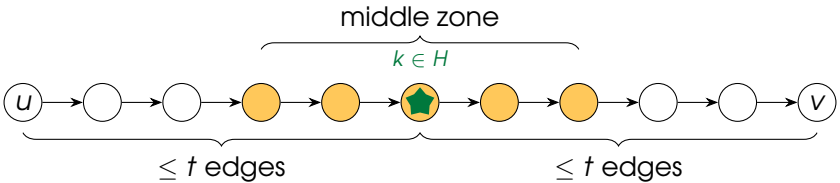
There is a deterministic catalytic algorithm that adds the APSP matrix of a weighted directed graph $G = (V, E, w)$ to the catalytic tape. It uses

$$O(\log n) \text{ workspace}, \qquad \widetilde{O}\left(n^5 \cdot W\right) \text{ catalytic space}, \qquad \widetilde{O}\left(n^{5.895}\right) \text{ time}.$$

$$\underset{\text{basic}}{n^{6.322}} \longrightarrow \underset{\text{previous best}}{n^{5.967}} \longrightarrow \underset{\text{ours}}{\boldsymbol{n^{5.895}}}$$

Idea 2: hitting sets

Grow paths a bit **slower than doubling**: from t edges to about $(2-\alpha)t$ edges, with small $\alpha > 0$.



Example: $t = 7$, path with 10 edges. Any k in the middle set splits it into two pieces of $\leq t$ edges.

Idea 2: randomized path growing

D_t: correct for pairs whose shortest path has $\leq t$ edges, otherwise never too small.

$$D_{t'}[u,v] = \min_{k \in \{u\} \cup H} \Big(D_t[u,k] + D_t[k,v] \Big), \qquad t' = \lceil (2-\alpha)t \rceil$$

	Candidates per pair	**Path length grows**
Full min-plus product	n	$t \mapsto 2t$
Hitting-set product	$\widetilde{O}(n/t)$	$t \mapsto (2-\alpha)t$

- **Regenerate** hitting set H from one short random seed.
- Hitting sets shrink as paths get longer, so later levels are cheaper.

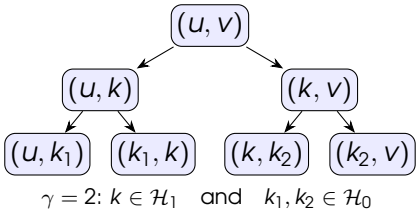
  With re-balanced minimum trees: $\widetilde{O}\left(n^{5.754}\right)$ **time**, $\widetilde{O}\left(n^5 \cdot W\right)$ catalytic bits.

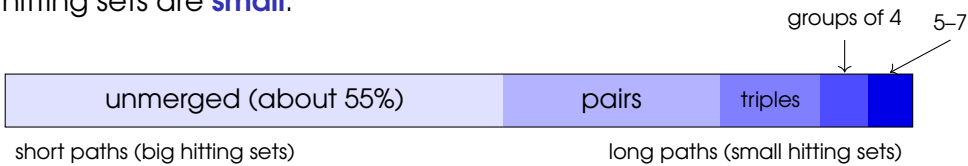
Idea 3: merge several levels into one

Every separate level costs a **factor of 5**.

Merge γ levels into one step: split the path again and again using γ hitting sets, and **pay the factor 5 once**.

The catch: Candidates now **multiply** across the tree, so merging only pays off when hitting sets are **small**.





Every group size costs the same $\Longrightarrow$ **randomized** $\widetilde{O}\left(n^{5.120}\right)$ **time**

Randomized result

Theorem (randomized catalytic APSP)

There is a catalytic algorithm that, with high probability, adds the APSP matrix of a weighted directed graph $G = (V, E, w)$ to the catalytic tape. It uses

$O(\log n)$ workspace, $\widetilde{O}\left(n^{6.12} \cdot W\right)$ catalytic space, $\widetilde{O}\left(n^{5.120}\right)$ time.

$n^{5.967}$ previous best $\longrightarrow$ $n^{5.754}$ ours: hitting sets $\longrightarrow$ $\boldsymbol{n^{5.120}}$ ours: + merging

Takeaways

1. **Catalytic computing:** Full memory is still useful!
2. **Catalytic minimums:** The bottleneck in reversible computation.
3. **Deterministic**: Using depth-based trees, we can optimize.
4. **Randomized**: Hitting sets trade perfect accuracy for speed.

Open questions

- Catalytic space reduction
- Hitting set schedule optimizations
- Other space-efficient graph algorithms

Acknowledgments

We thank our mentor, **Zoe Xi**, for her guidance.

We also thank the **MIT PRIMES** program for the opportunity to carry out this research.

Questions?

References

- H. Buhrman, R. Cleve, M. Koucký, B. Loff, and F. Speelman. *Computing with a Full Memory: Catalytic Space*. STOC, 2014.
- J. Cook and I. Mertz. *Tree Evaluation is in Space $O(\log n \cdot \log \log n)$*. STOC, 2024.
- R. R. Williams. *Simulating Time With Square-Root Space*. STOC, 2025.
- J. Cook and E. Pyne. *Efficient Catalytic Graph Algorithms*. ITCS, 2026.
- N. Sheffield. *A Sketch of a Catalytic Algorithm for APSP*. Unpublished manuscript, 2025.
- U. Zwick. *All Pairs Shortest Paths Using Bridging Sets and Rectangular Matrix Multiplication*. JACM, 2002.

Full proofs, pseudocode, and related work: A. Liu and A. Zou, *Efficient Catalytic Algorithms for All-Pairs Shortest Paths*.