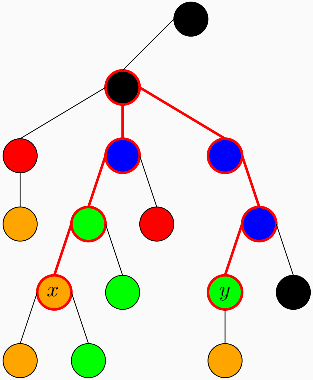
Faster Path Queries in Colored Trees via Sparse Matrix Multiplication and Min-Plus Product

Younan Gao and Meng He

Dalhousie University

Define the Problems



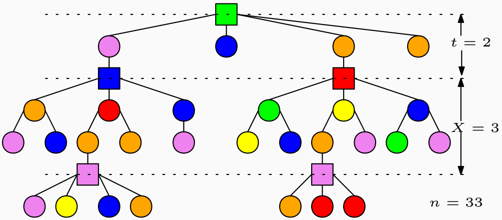
- A tree of n colored nodes;
- colors drawn from $[0..C-1]$;
- color counting: $|C(P_{x,y})| = 4$
- mode on $P_{x,y}$: *blue*
- Least-Frequent on $P_{x,y}$: *black*.

Related Work

		n queries	
		Previous	Ours
Color Counting	Grid	1.40704	
	Tree	1.5	1.40704
Mode	Arrays	1.4805	1.479603
	Tree	1.5	1.483814
Least Frq	Arrays	1.5	1.479603
	Tree	1.5	1.483814

All problems listed above share the same conditional lower bound.

Colored Counting: Paths through the Root



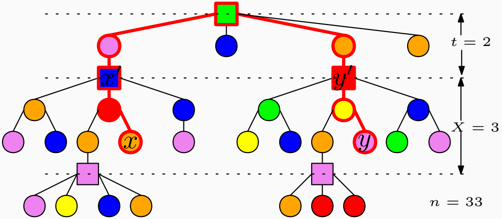
Marking Rules:

- Mark nodes at every X levels,
- starting from some level $t \in [0, X-1]$;
- and mark the root.

Outcomes:

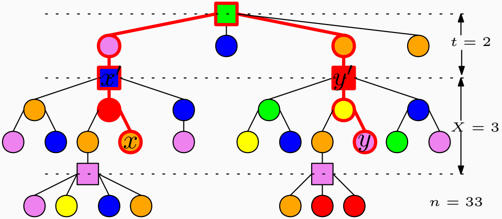
- At most n/X marked nodes;
- $|P_{s,s'}| \leq X + 1$, where node s' is the lowest marked ancestor of node s.

The Solution

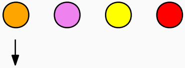


- Suppose that $|C(P_{x',y'})|$ is given.
- For each color $c \in C(P'_{x,x'}) \cup C(P'_{y,y'})$,
- check whether c appears in $P_{x',y'}$.

The Solution

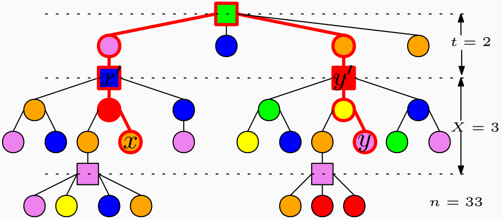


$result = |C(P_{x',y'})|$

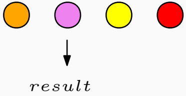


$result$

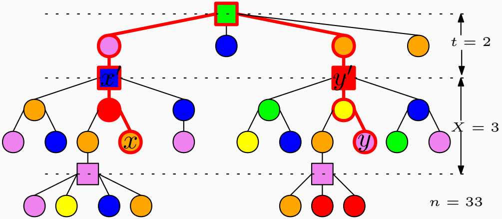
The Solution



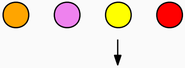
$result = |C(P_{x',y'})|$



The Solution

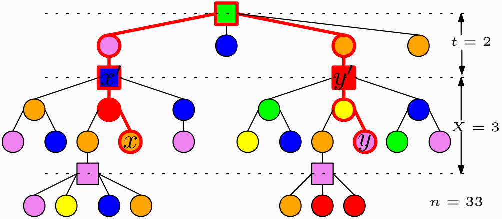


$$result = |C(P_{x',y'})|$$

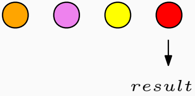


$$result++$$

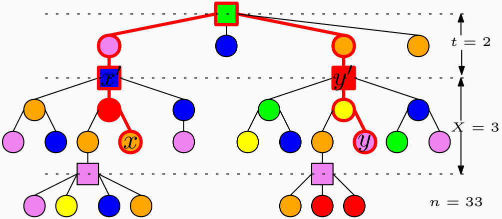
The Solution



$result = |C(P_{x',y'})|$



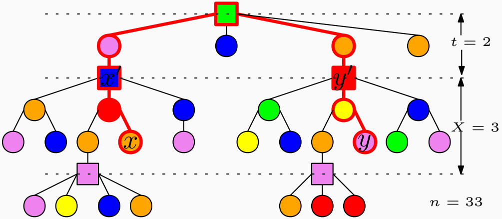
The Solution



$result = |C(P_{x',y'})|$

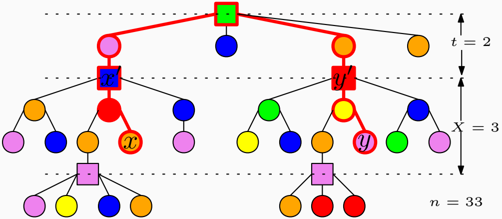
return result

The Solution



- in $O(X \text{ polylog } n)$ time,
- using $O((\frac{n}{X})^2 + n)$ words.
- How to compute $|C(P_{x',y'})|$ for all pairs of x' and y' efficiently?

The Matrices



- Construct an $n/X * C$ matrix, A.
- Entry $A[i, c]$ stores 1 if color c appears in path $P_{x_i,\perp}$; 0 otherwise.

A						
0	0	1	0	0	0	0
1	0	1	1	0	1	0
2	0	1	1	1	1	1
3	0	1	0	1	0	1
4	1	1	0	1	1	1

The Matrices

A						
0	0	1	0	0	0	0
1	0	1	1	0	1	0
2	0	1	1	1	1	1
3	0	1	0	1	0	1
4	1	1	0	1	1	1

*

A^T	*0*	*1*	*2*	*3*	*4*
	0	0	0	0	1
	1	1	1	1	1
	0	1	1	0	0
	0	0	1	1	1
	0	1	1	0	1
	0	0	1	1	1

=

M	*0*	*1*	*2*	*3(y')*	*4*
0	1	1	1	1	1
1(x')	1	3	3	1	2
2	1	3	5	3	4
3	1	1	3	3	3
4	1	2	4	3	5

$$M[1,3] = |C(P_{x',\perp}) \cap C(P_{y',\perp})|$$

$$\begin{aligned} |C(P_{x',y'})| &= |C(P_{x',\perp}) \cup C(P_{y',\perp})| \\ &= |C(P_{x',\perp})| + |C(P_{y',\perp})| - |C(P_{x',\perp}) \cap C(P_{y',\perp})| \\ &= |C(P_{x',\perp})| + |C(P_{y',\perp})| - M[1,3] \end{aligned}$$

The Matrices

A						
0	0	1	0	0	0	0
1	0	1	1	0	1	0
2	0	1	1	1	1	1
3	0	1	0	1	0	1
4	1	1	0	1	1	1

*

A^T	*0*	*1*	*2*	*3*	*4*
	0	0	0	0	1
	1	1	1	1	1
	0	1	1	0	0
	0	0	1	1	1
	0	1	1	0	1
	0	0	1	1	1

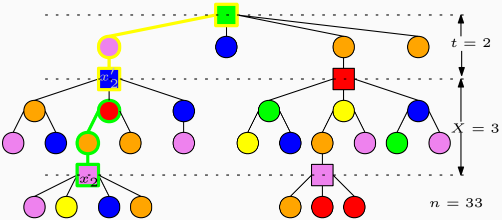
=

M	*0*	*1*	*2*	*3(y')*	*4*
0	1	1	1	1	1
1(x')	1	3	3	1	2
2	1	3	5	3	4
3	1	1	3	3	3
4	1	2	4	3	5

However,

- Matrix A could have as many as Cn/X non-zero entries;
- and computing matrix M could take $C \cdot (\frac{n}{X})^2$ time.

The Updated Matrices



- Define $\hat{C}(x_i)$ to be $C(P'_{x_i,x'_i}) \setminus C(P_{x'_i,\perp})$.
- Entry $\hat{A}[i, c]$ stores 1 if color c appears in $\hat{C}(x_i)$ and 0 otherwise.

$\hat{A}$						
0	0	0	0	0	0	0
1	0	0	1	0	1	0
2	0	0	0	1	0	1
3	0	0	0	1	0	1
4	1	0	0	0	1	0

The Updated Matrix

$\hat{A}$						
0	0	0	0	0	0	0
1	0	0	1	0	1	0
2	0	0	0	1	0	1
3	0	0	0	1	0	1
4	1	0	0	0	1	0

*

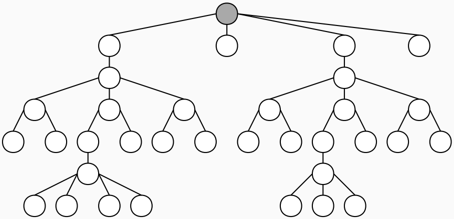
$\hat{A}^T$	*0*	*1*	*2*	*3*	*4*
	0	0	0	0	1
	0	0	0	0	0
	0	1	0	0	0
	0	0	1	1	0
	0	1	0	0	1
	0	0	1	1	0

=

$\widehat{M}$	*0*	*1*	*2*	*3(y')*	*4*
0	0	0	0	0	0
1(x')	0	2	0	0	1
2	0	0	2	2	0
3	0	0	2	2	0
4	0	1	0	0	2

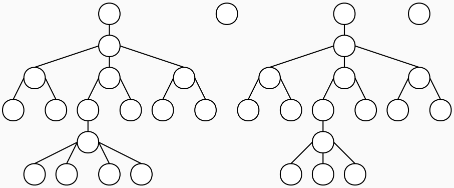
- Matrix $\hat{A}$ has at most $O(n)$ non-zero entries:
  - At most n/X rows and at most X non-zero entries per row.
- Matrix $\hat{M}$ can be computed in $O(n^{(\omega+1)/2}/X^{(\omega-1)/2})$ time.
  - for any $X \in [n^{(\omega-1)/(\omega+1)}, n]$, using SRMM.
- Entry $\hat{M}[i,j]$ stores $|\hat{C}(x_i) \cap \hat{C}(x_j)|$;
- $\hat{M}$ can be turned into M in $O((\frac{n}{X})^2)$ time.

Colored Counting: Arbitrary Path



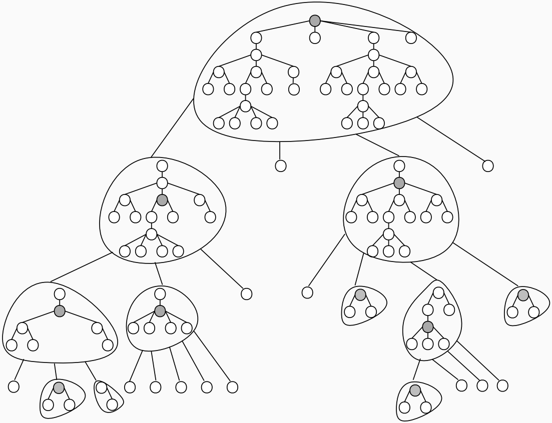
- Apply centroid decomposition.
- After removing the centroid, each subtree contains at most $n/2$ nodes.
- A query path is either contained within a subtree or through the centroid.

Colored Counting: Arbitrary Path



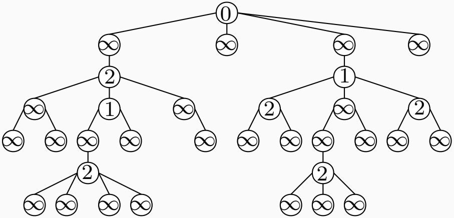
- Apply centroid decomposition.
- After removing the centroid, each subtree contains at most $n/2$ nodes.
- A query path is either contained within a subtree or through the centroid.

Process Each Subtree Recursively



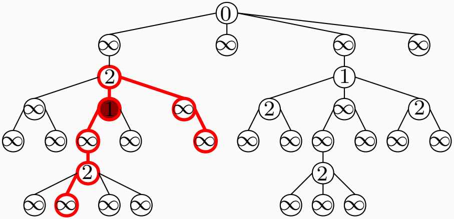
- Process each subtree recursively.
- The recursive tree contains $\lceil \lg \frac{n}{X} \rceil$ levels.
- Each base component contains at most X nodes.

Weighted Tree

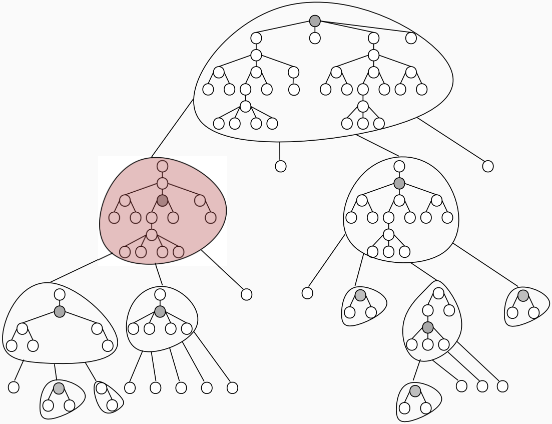


- Centroids at level ℓ are assigned weight-ℓ.
- Non-centroids in the base components are assigned to weight-∞.
- Construct a data structure for path minimum queries.

Path Minimum Queries

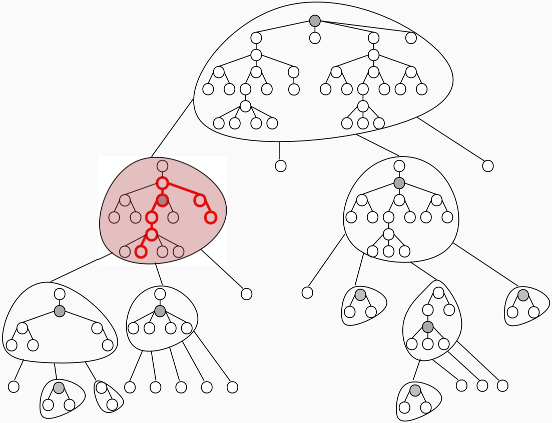


- Find the node v carrying the minimum weight on the query path.



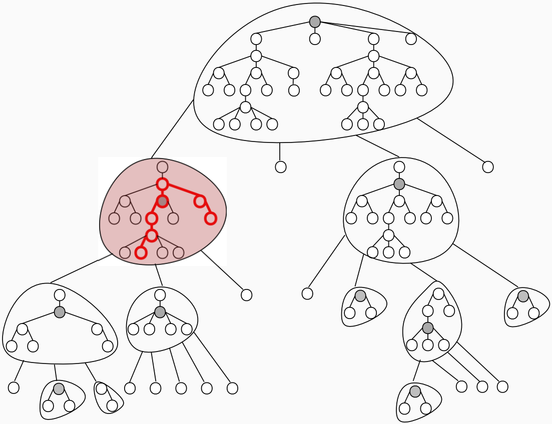
- Find the node v carrying the minimum weight on the query path.
- Identify the component s that has v as the centroid.

Path Minimum Queries



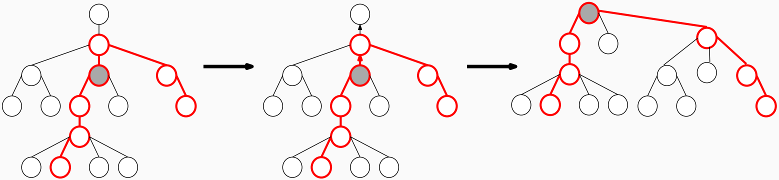
- Find the node v carrying the minimum weight on the query path.
- Identify the component s that has v as the centroid.
- The query path in s must contain the centroid.

Path Minimum Queries



- Find the node v carrying the minimum weight on the query path.
- Identify the component s that has v as the centroid.
- The query path in s must contain the centroid.
- If the minimum weight is ∞, then the query path is within some base component.

Colored Counting



- Turn the query path to be a path through the root of the component.
- The space cost is increased by a $\lg n$ factor, i.e., $O(((\frac{n}{X})^2 + n) \times \lg n)$ words,
- while the query time bound is maintained, which is $O(X \text{ polylog } n)$.

Conclusions

- Breaking the bound $n^{3/2}$:
  - † Batched colored path counting;
  - † batched path mode;
    - * reducing to computing the Min-plus Product;
    - * applying the special structure inherited from tree topology.
  - † batched least-frequent queries;
    - * $A_{i,k^*} + B_{k^*,j} = \min\{A_{i,k} + B_{k,j}\}$
    - * and $A_{i,k^{**}} + B_{k^{**},j} = \min\{A_{i,k} + B_{k,j} : A_{i,k} + B_{k,j} > A_{i,k^*} + B_{k^*,j}\}$
- Their respective dynamization, breaking the bound $n^{2/3}$.
- Open Problem:
  - † batched mode queries on arrays: $O(n^{1.479603})$ time;
  - † batched path mode queries: $O(n^{1.483814})$

Questions?