Parallel Depth First on GPU

M. Naumov, A. Vrielink and M. Garland, GTC 2017

NVIDIA

AGENDA

Introduction

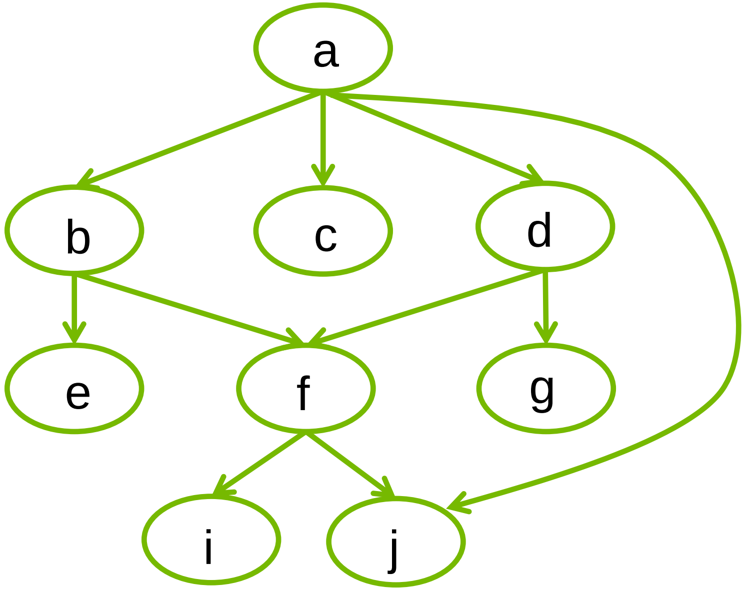
Directed Trees

Directed Acyclic Graphs (DAGs)

- ✓ Path- and SSSP-based variants
- ✓ Optimizations

Performance Experiments

What is DFS?

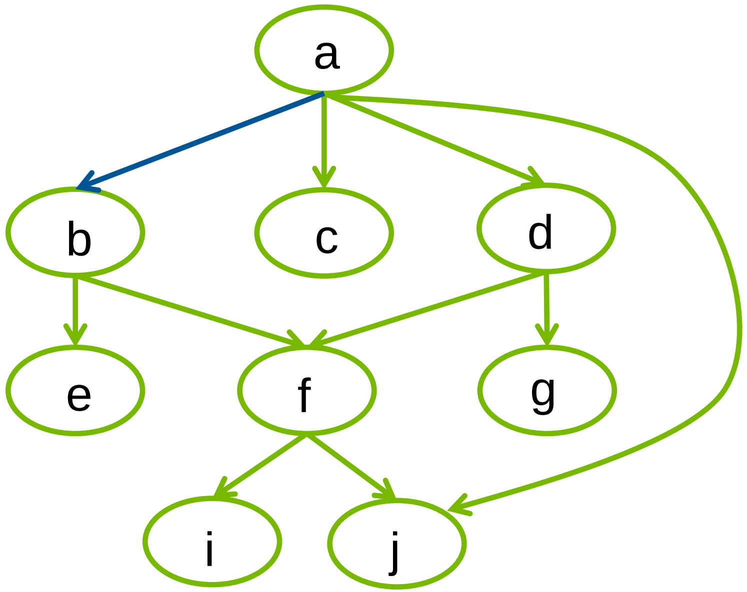


Node: a,b,c,d,e,f,g,i,j
Parent:

Discovery:

Finish:

What is DFS?

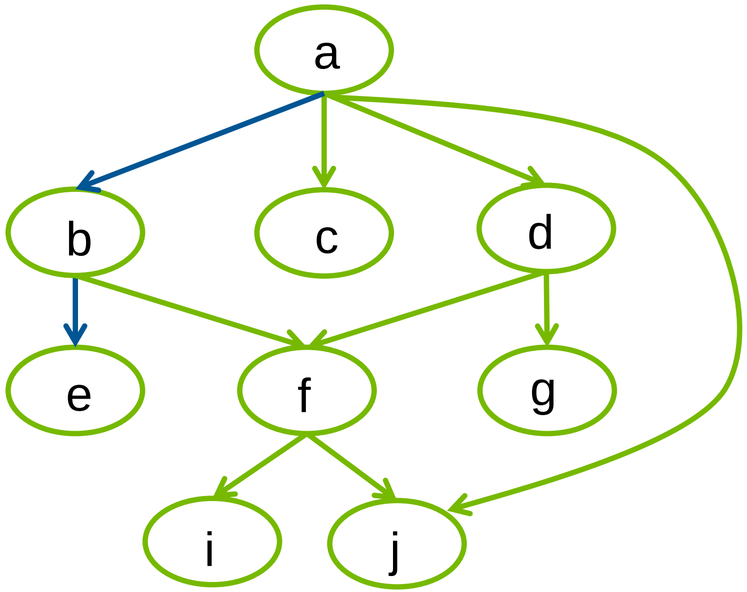


Node: a,b,c,d,e,f,g,i,j
Parent: / ,a

Discovery: a,b

Finish:

What is DFS?

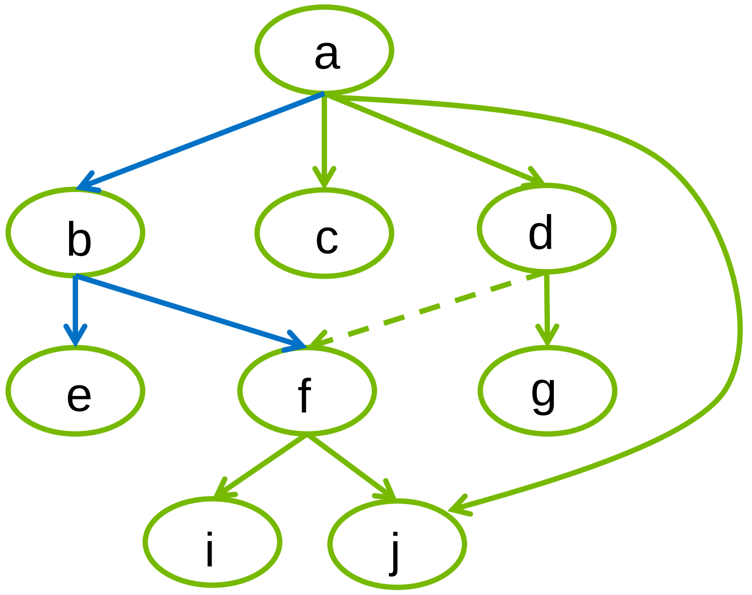


Node: a,b,c,d,e,f,g,i,j
Parent: /,a, b,

Discovery: a,b,e

Finish: e

What is DFS?

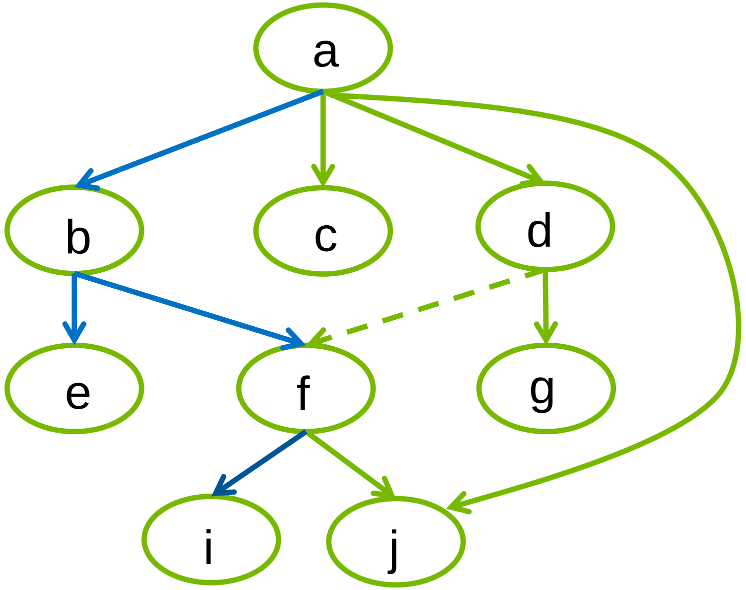


Node: a,b,c,d,e,f,g,i,j
Parent: /,a, b,b

Discovery: a,b,e,f

Finish: e

What is DFS?

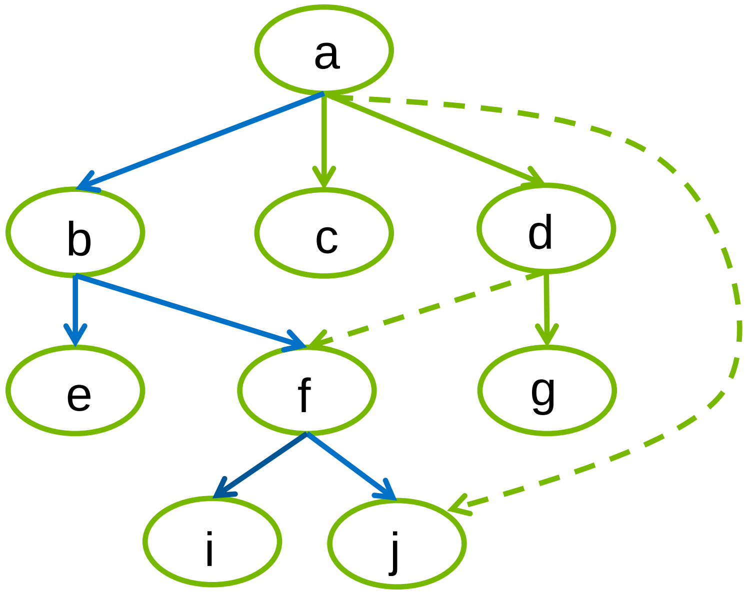


Node: a,b,c,d,e,f,g,i,j
Parent: /,a, b,b, ,f

Discovery: a,b,e,f,i

Finish: e,i

What is DFS?

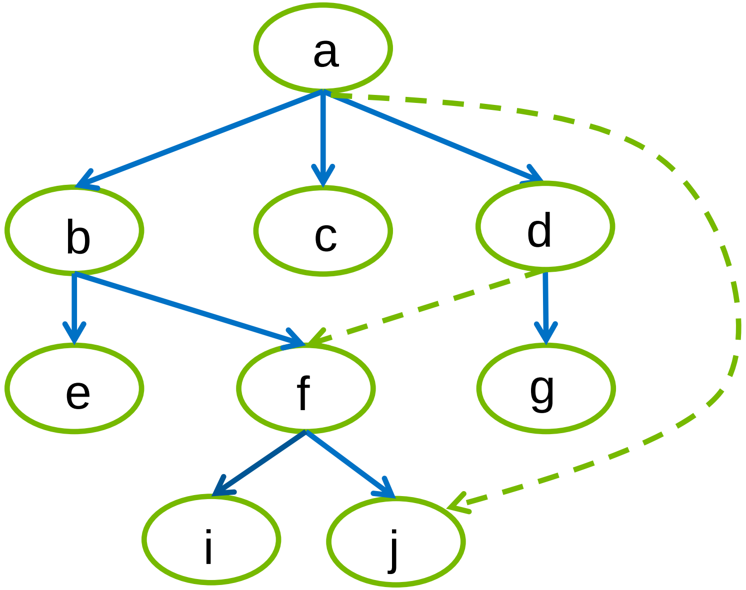


Node: a,b,c,d,e,f,g,i,j
Parent: /,a, b,b, ,f,f

Discovery: a,b,e,f,i,j

Finish: e,i,j

What is DFS?



Node: a,b,c,d,e,f,g,i,j
Parent: /,a,a,a,b,b,d,f,f

Discovery: a,b,e,f,i,j,c,d,g

Finish: e,i,j,f,b,c,g,d,a

Previous Work on DFS

Lexicographic DFS

Planar Graphs

Time $O(\log^2 n)$
Processors $O(n)$

Directed Acyclic Graphs (DAGs)

Time $O(\log^2 n)$
Processors $O(n^\omega / \log n)$

Directed Graphs with Cycles

Time $O(\sqrt{n} \log^{11} n)$
Processors $O(n^3)$

where $\omega < 2.373$ is the matrix multiplication exponent

Previous Work on DFS

Lexicographic DFS

Planar Graphs

Time $O(\log^2 n)$
Processors $O(n)$

Directed Acyclic Graphs (DAGs)

Time $O(\log^2 n)$
Processors $O(n^\omega / \log n)$

Directed Graphs with Cycles

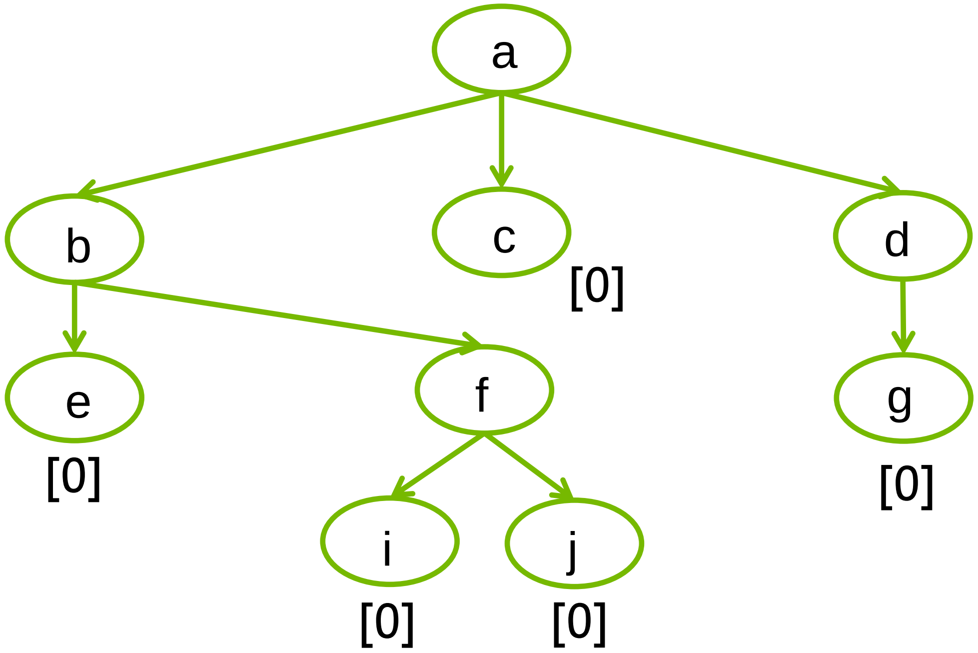
Time $O(\sqrt{n} \log^{11} n)$
Processors $O(n^3)$

topological sort, bi-connectivity and planarity testing

where $\omega < 2.373$ is the matrix multiplication exponent

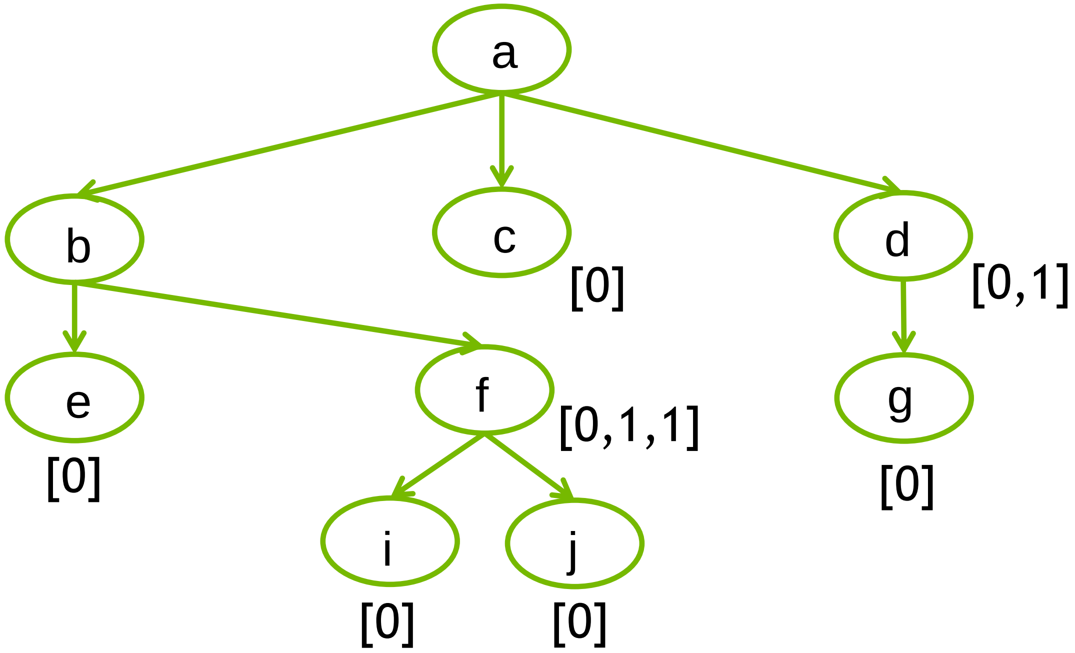
DIRECTED TREES

Directed Tree



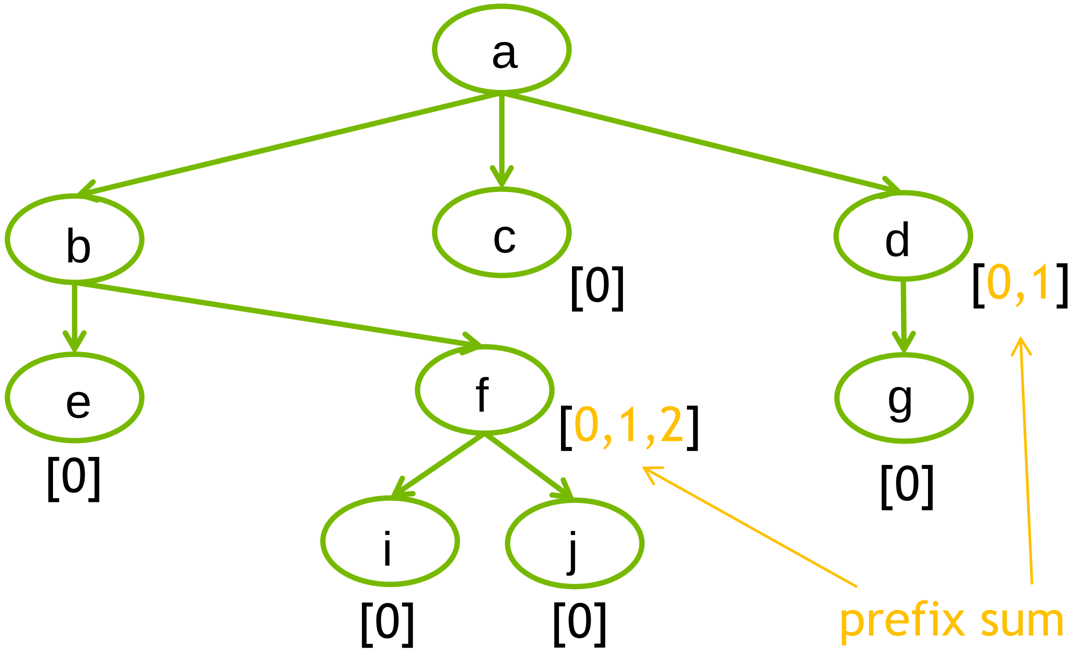
Phase 2: Bottom-Up Traversal

Directed Tree



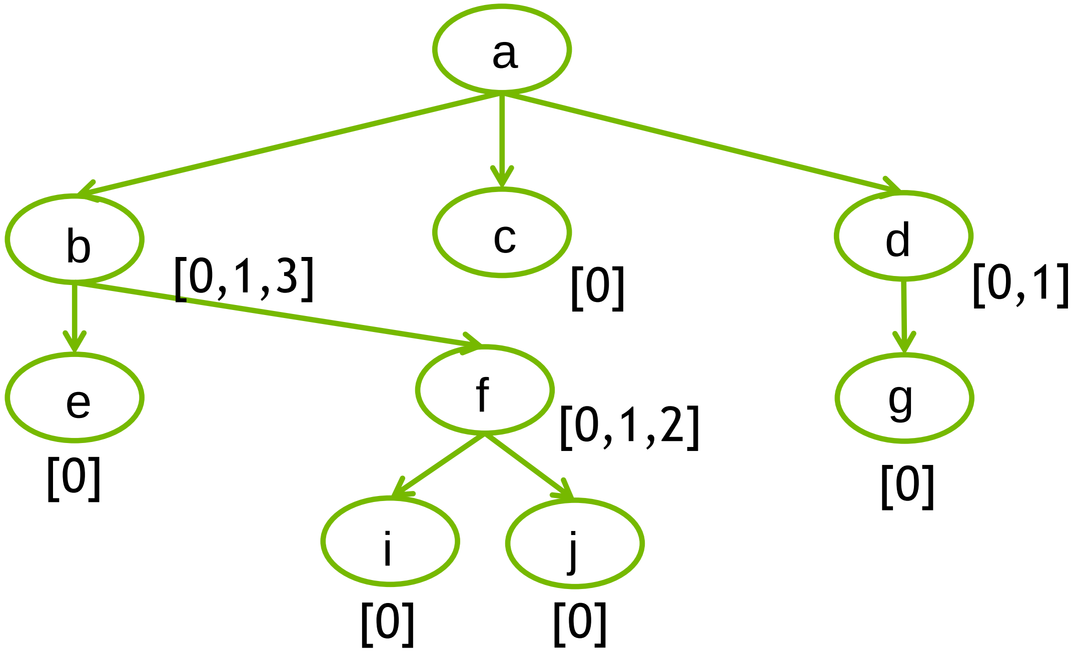
Phase 2: Bottom-Up Traversal

Directed Tree



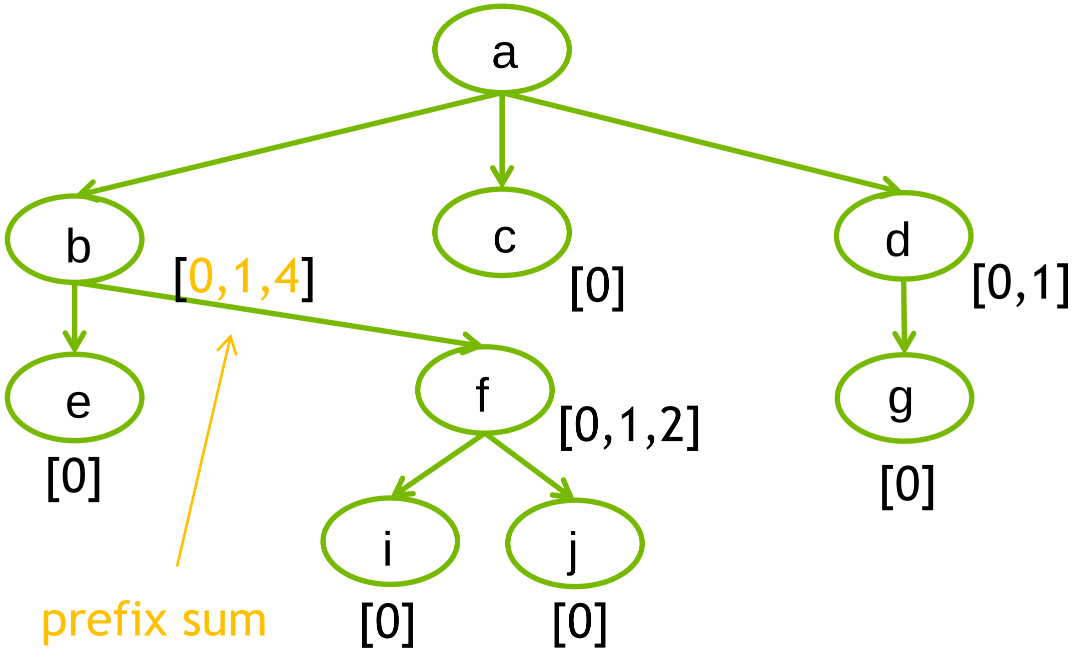
Phase 2: Bottom-Up Traversal

Directed Tree



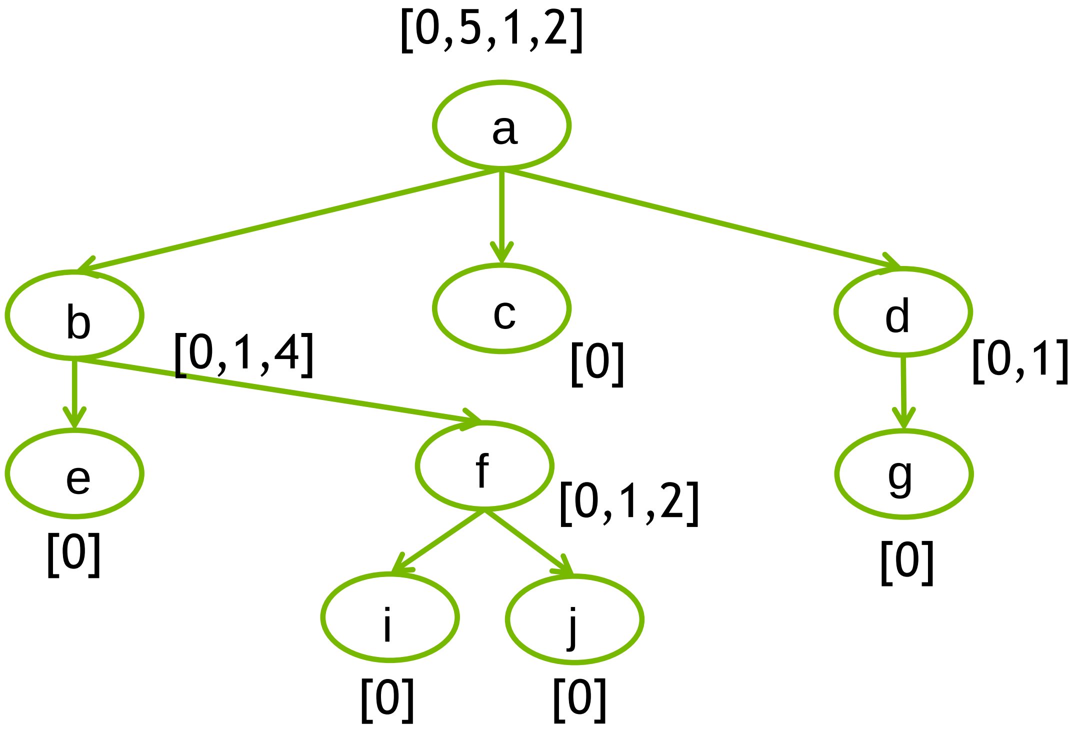
Phase 2: Bottom-Up Traversal

Directed Tree



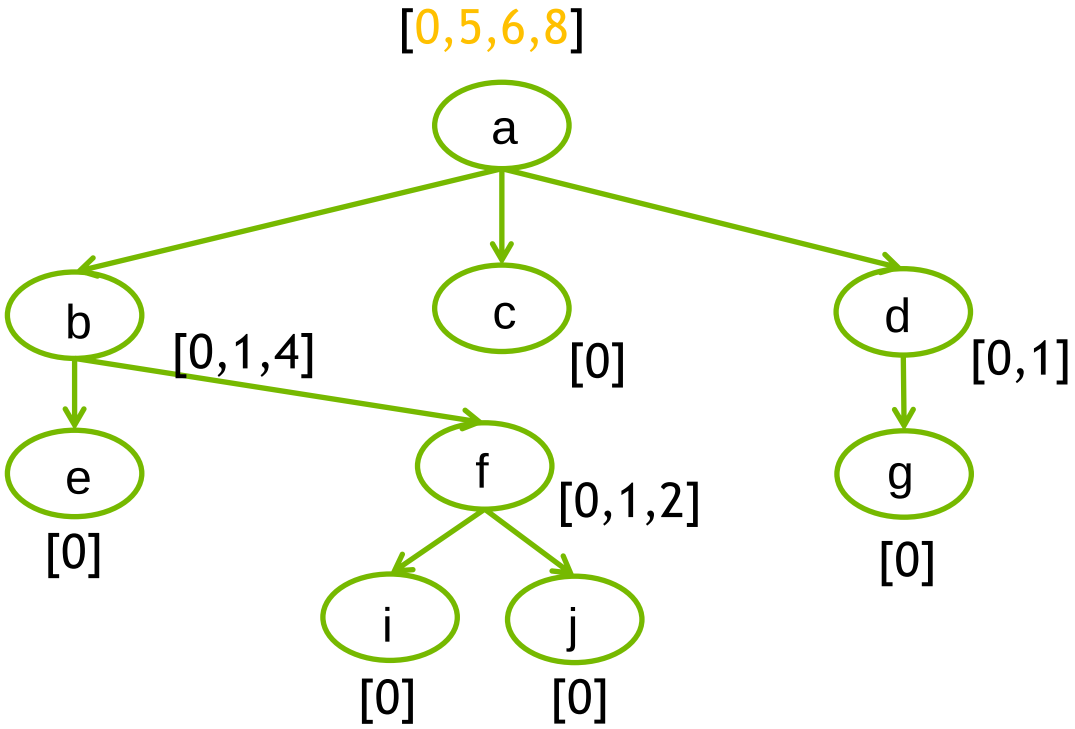
Phase 2: Bottom-Up Traversal

Directed Tree



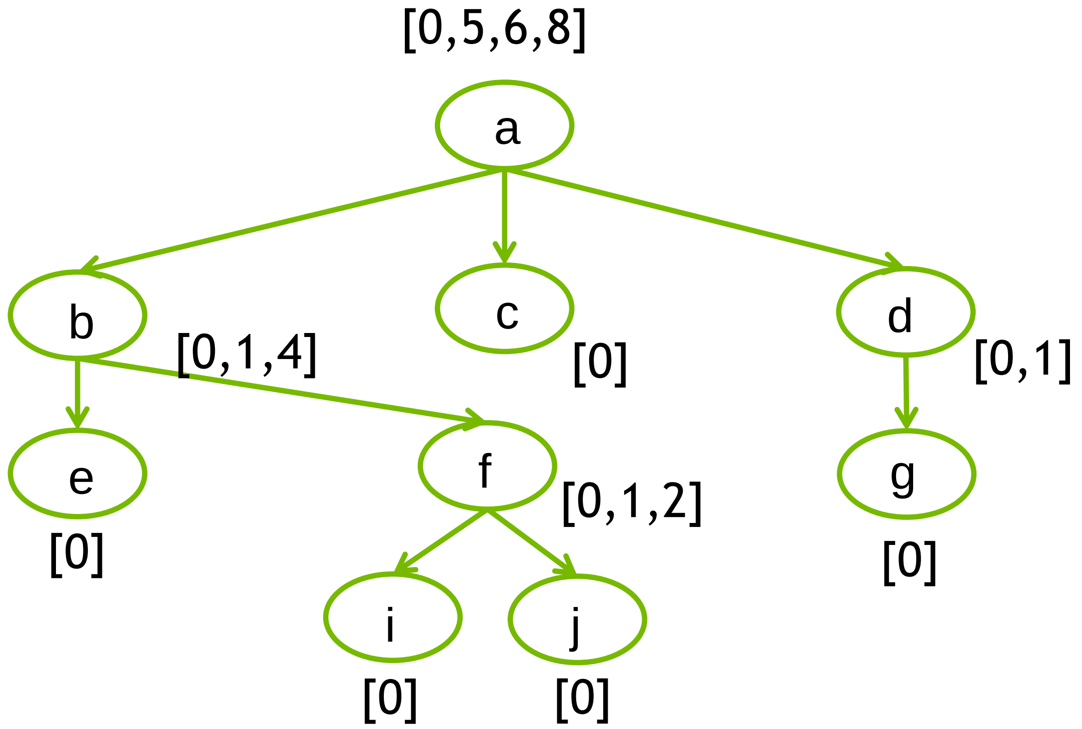
Phase 2: Bottom-Up Traversal

Directed Tree



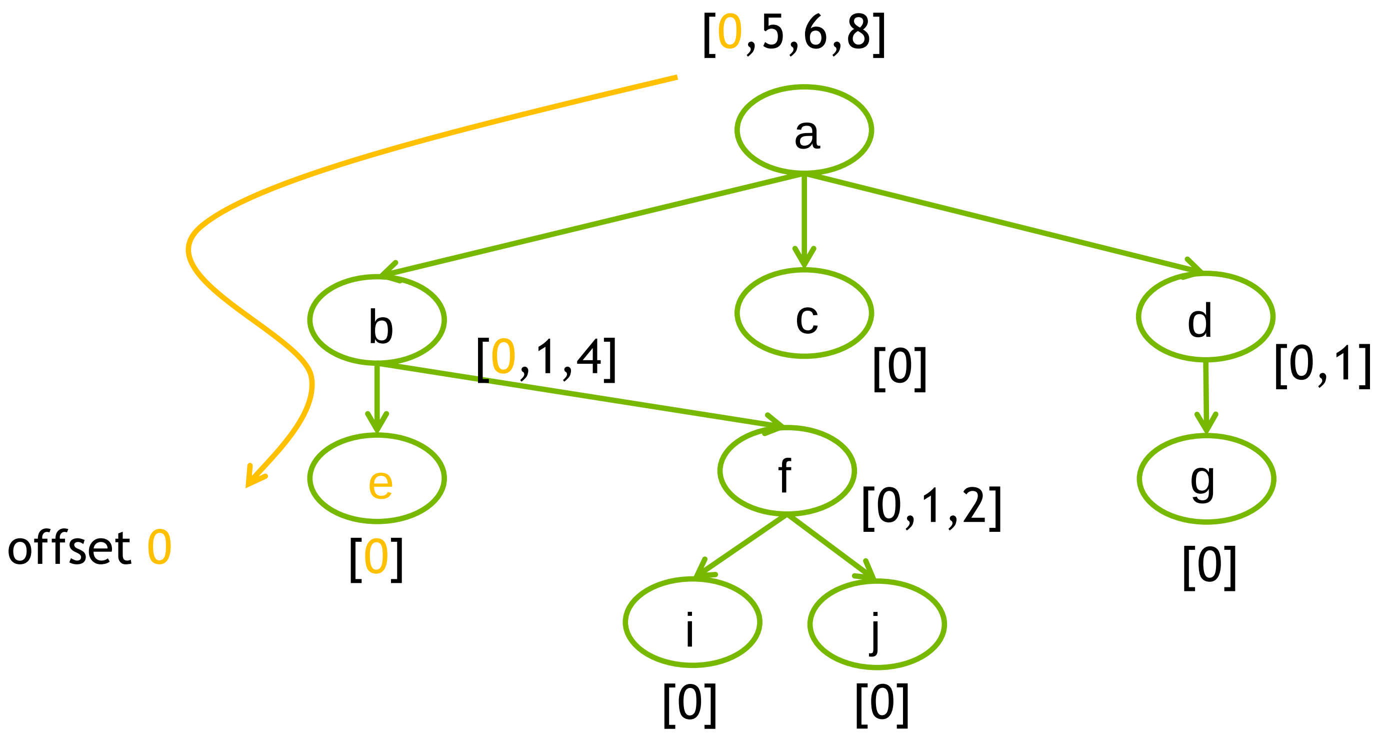
Phase 2: Bottom-Up Traversal

Directed Tree



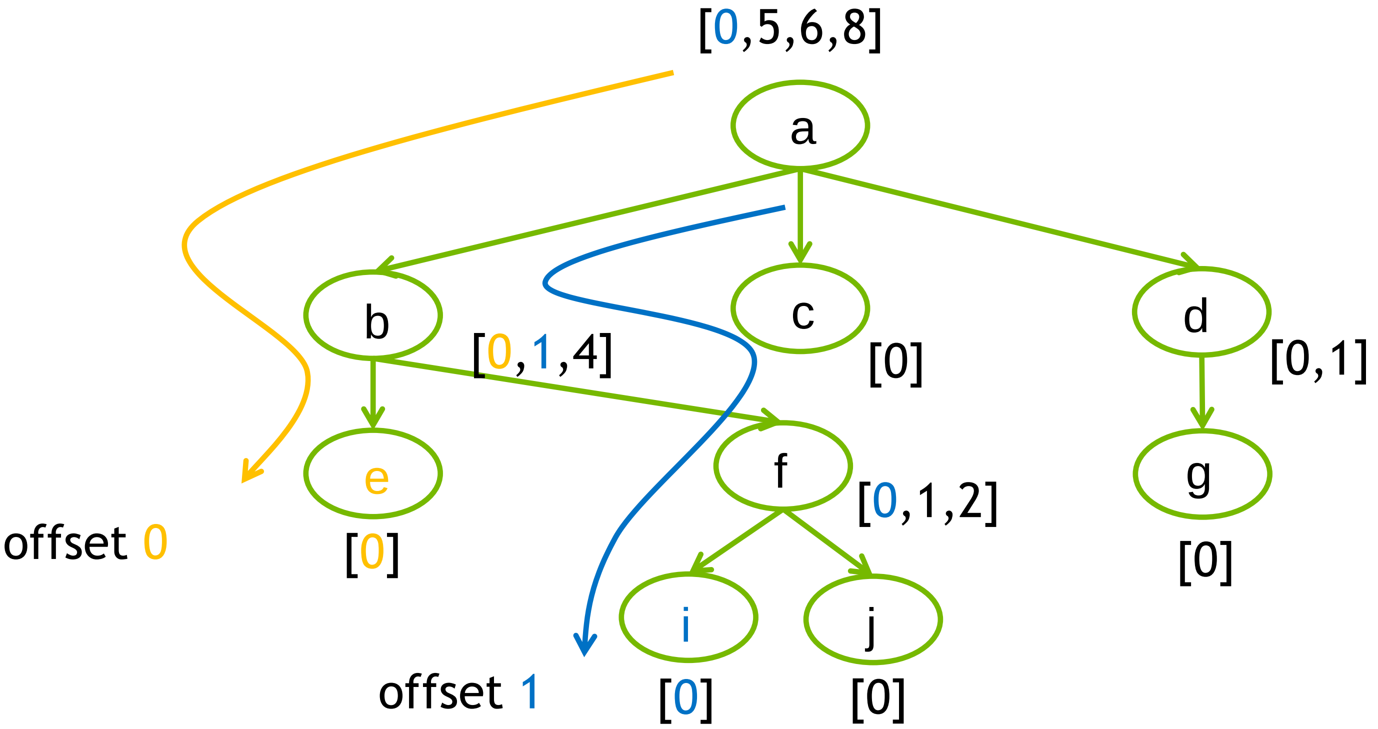
This phase is done, next phase is about to start …

Directed Tree



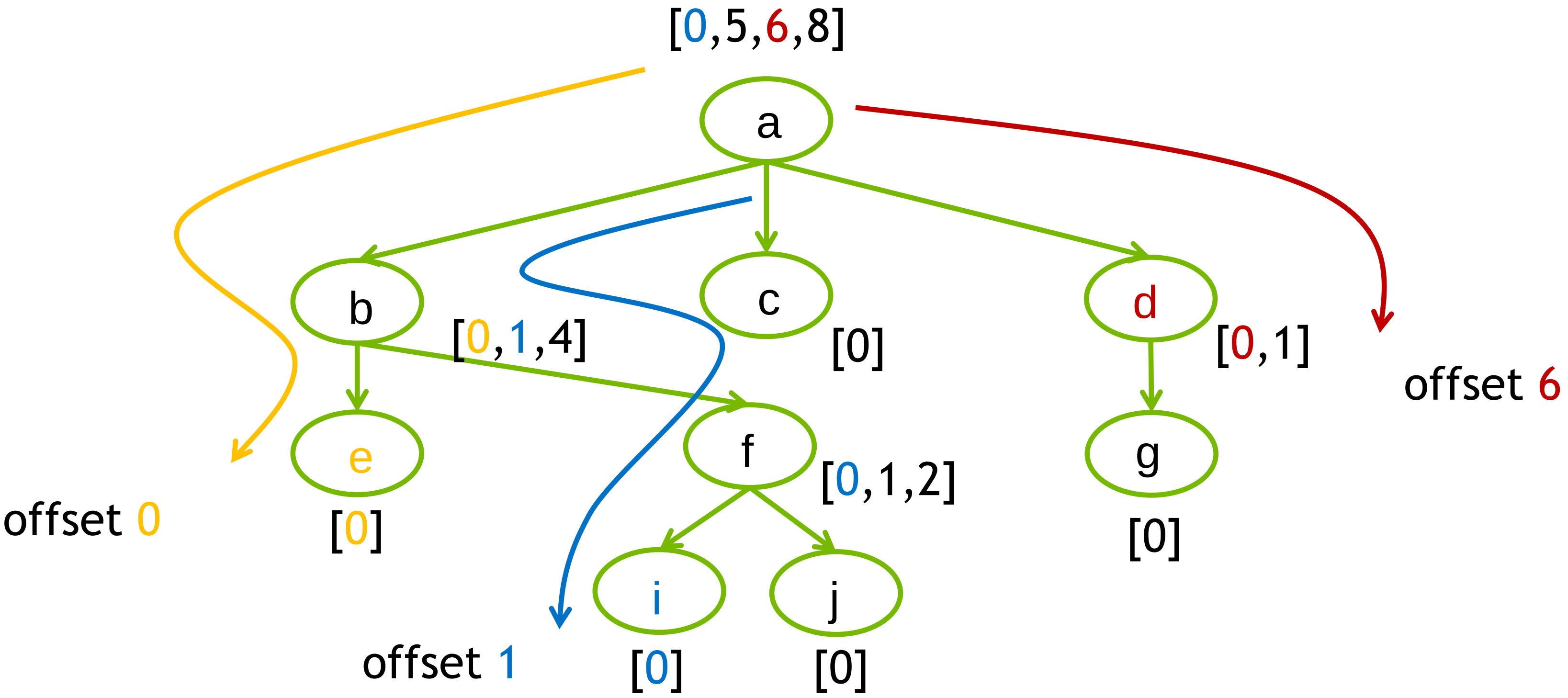
Phase 3: Top-down Traversal

Directed Tree



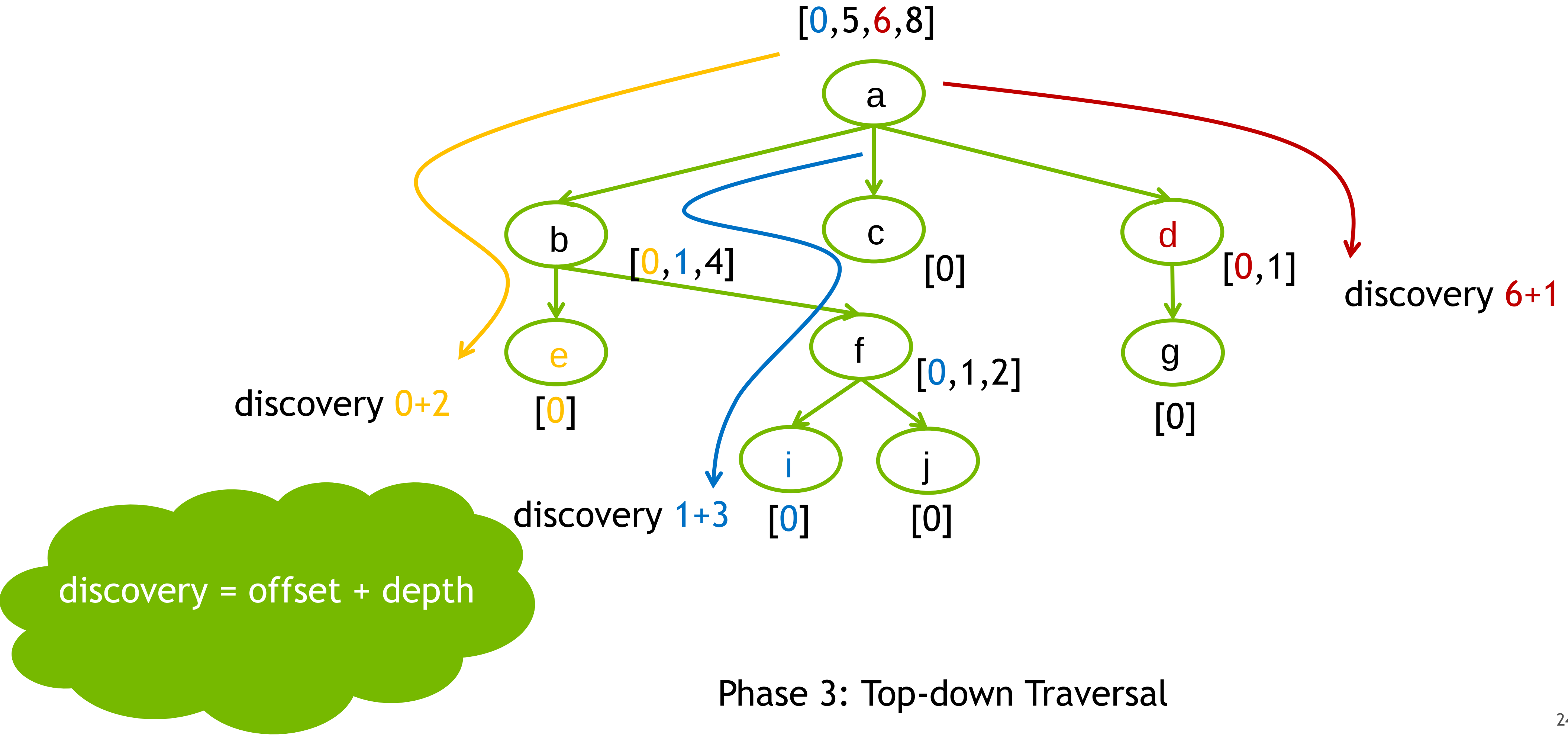
Phase 3: Top-down Traversal

Directed Tree



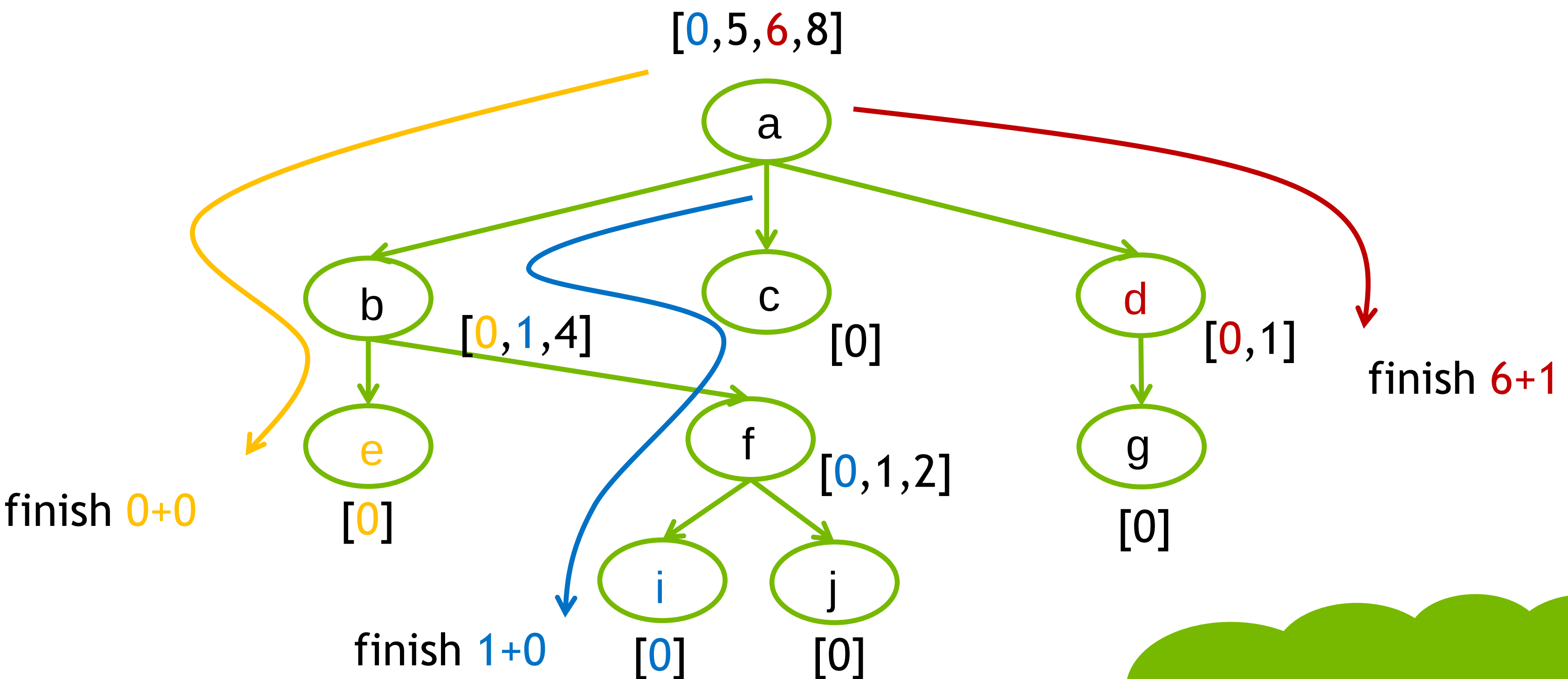
Phase 3: Top-down Traversal

Directed Tree



Phase 3: Top-down Traversal

Directed Tree



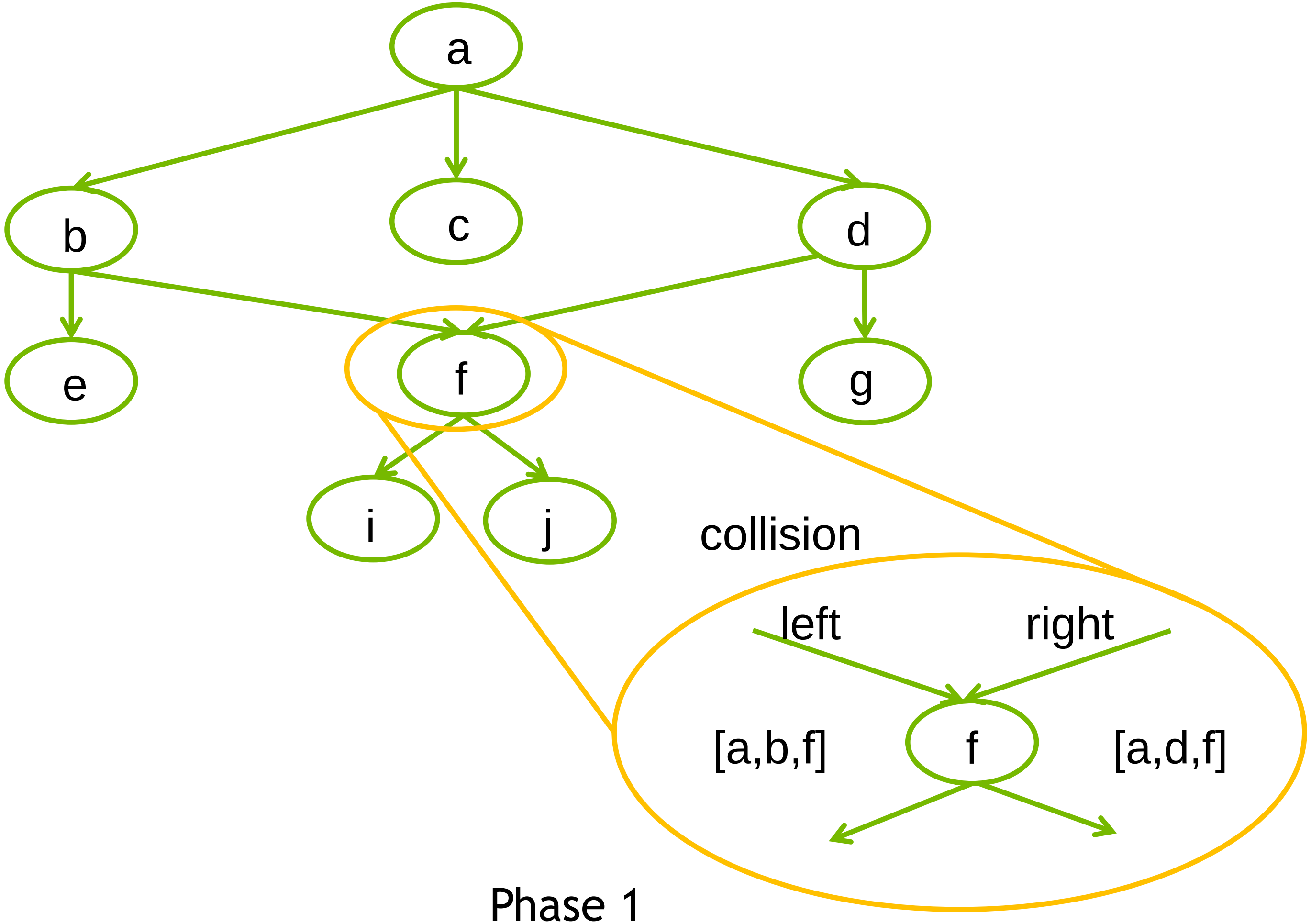
finish = offset + sub-tree size

Phase 3: Top-down Traversal

DIRECTED ACYCLIC GRAPHS

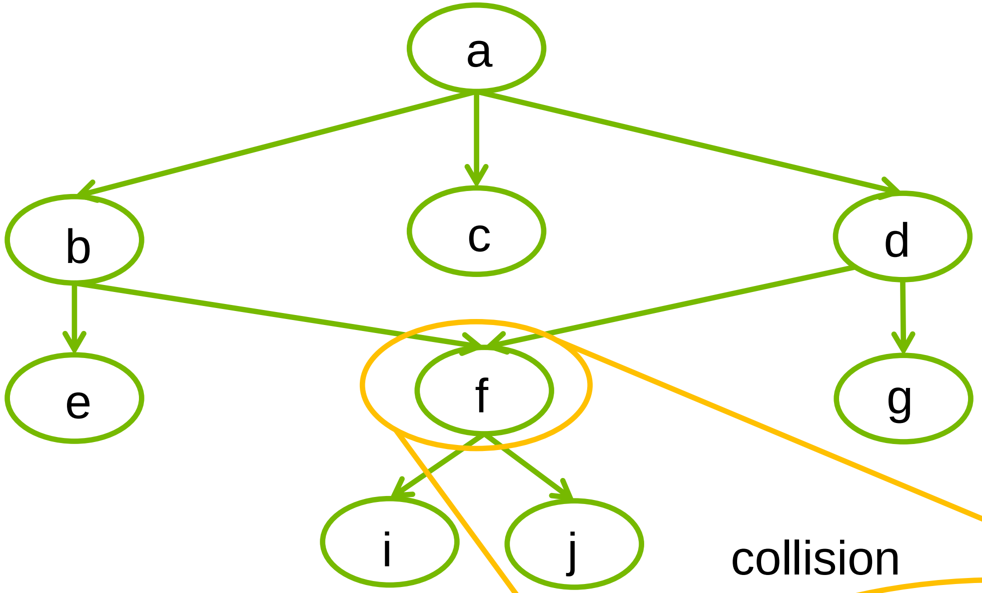
PATH-BASED VARIANT

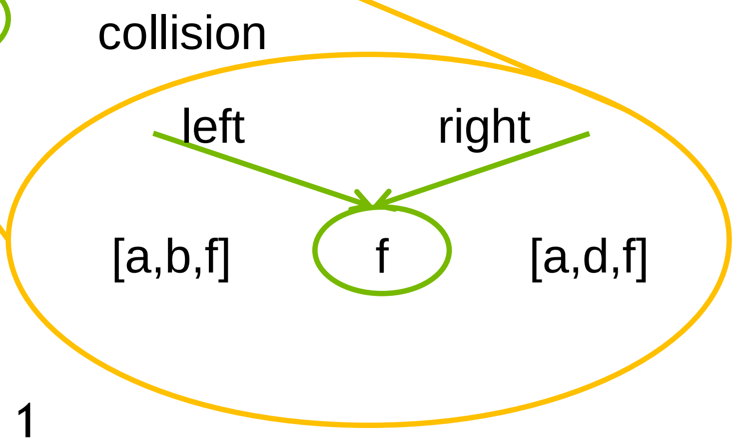
Path-Based (for DAGs)



Phase 1

Path-Based (for DAGs)





- wait until all paths to a node are traversed
- align path sequences

  left [a,b,f]

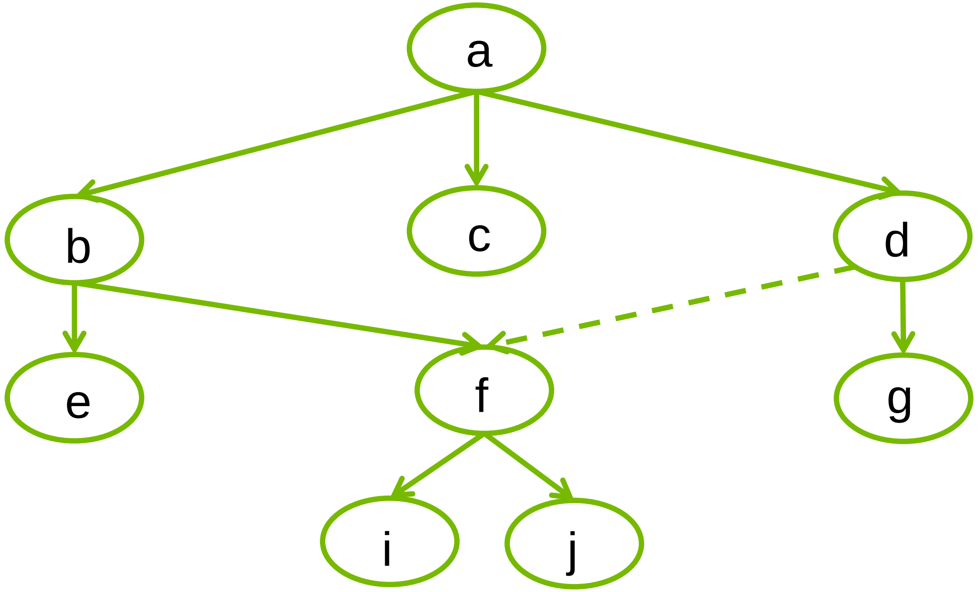
  right [a,d,f]

  (lexicographically smallest)
- compare left-to-right and choose smallest

resolution

Phase 1

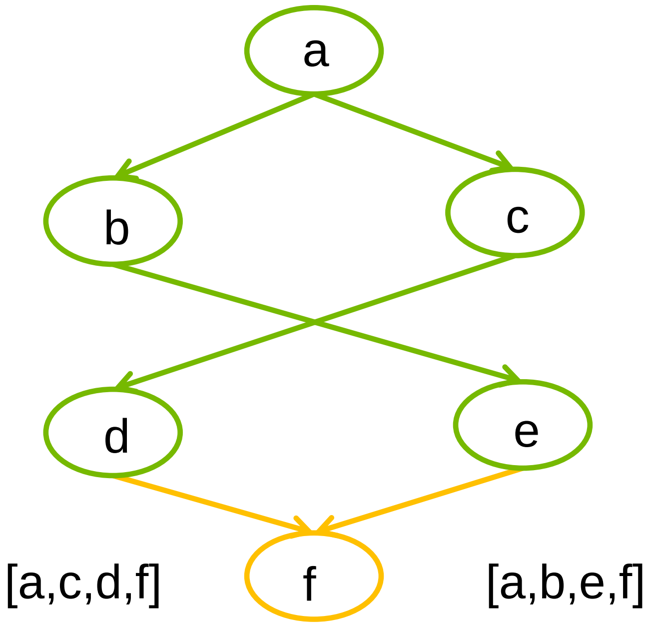
Path-Based (for DAGs)



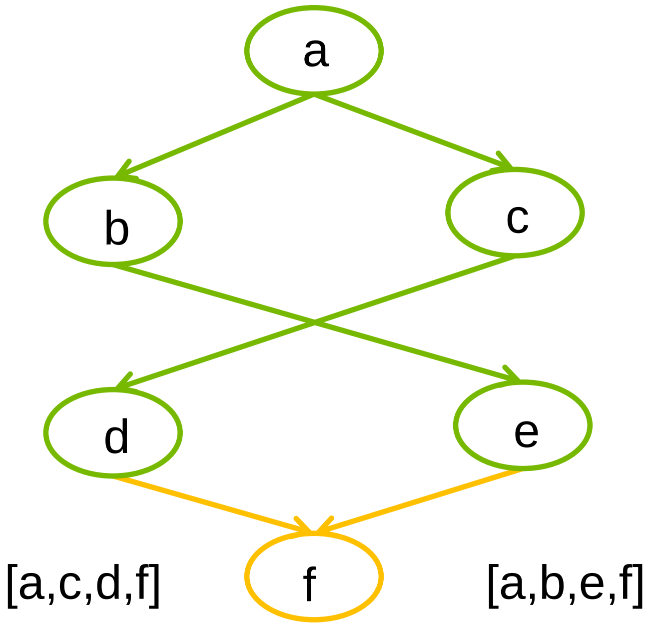
This phase is done

OPTIMIZATIONS

Path Pruning



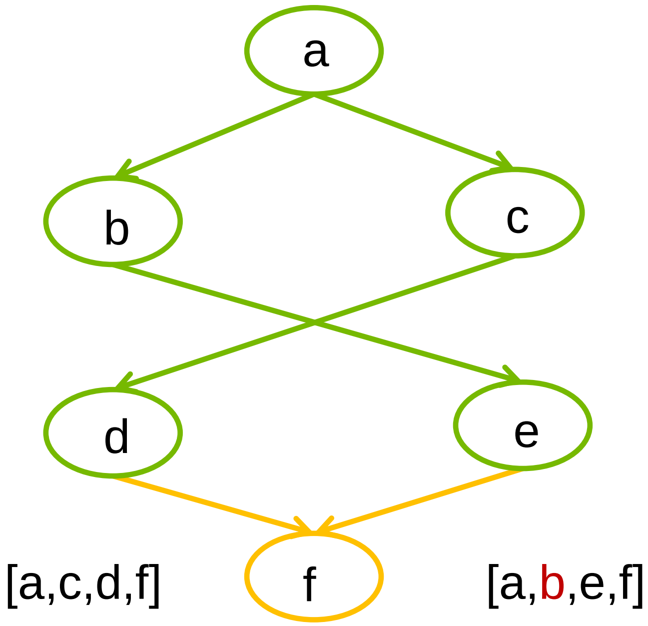
Path Pruning



When two paths reach the same node

- ✓ There exists a parent “a” where the path split [a,b,…] and [a,c,…]

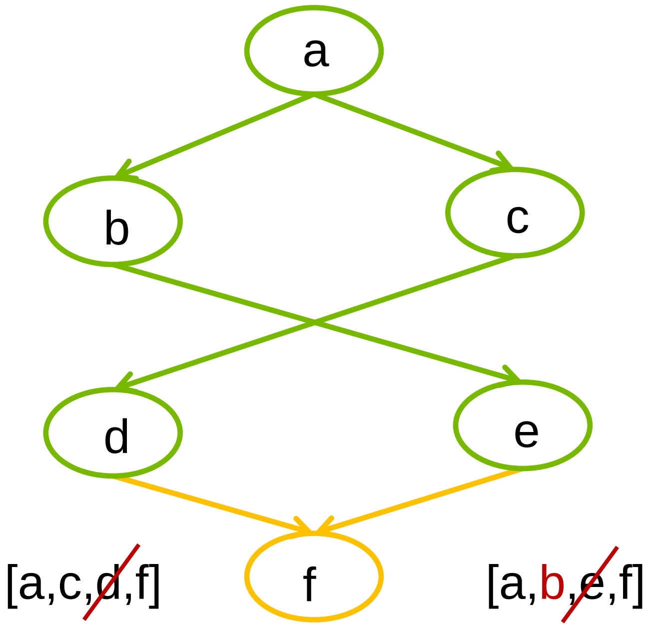
Path Pruning



When two paths reach the same node

- ✓ There exists a parent “a” where the path split [a,b,…] and [a,c,…]
- ✓ It is the comparison between “b” and “c” that allows us to distinguish between paths

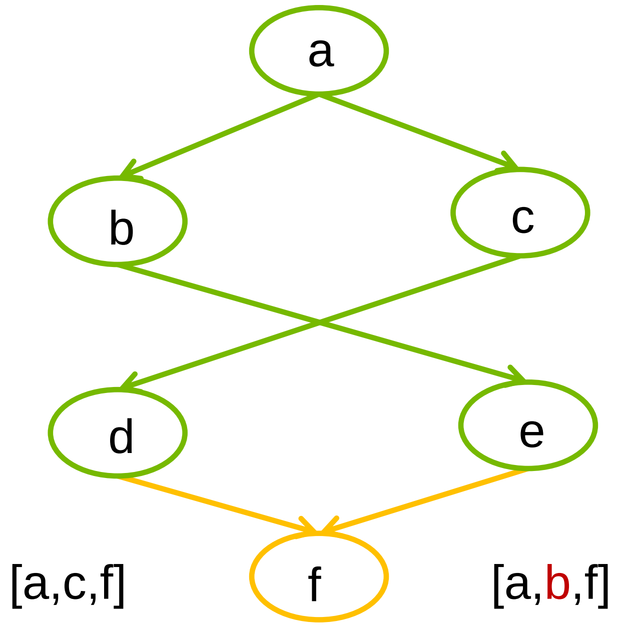
Path Pruning



When two paths reach the same node

- ✓ There exists a parent "a" where the path split [a,b,…] and [a,c,…]
- ✓ It is the comparison between "b" and "c" that allows us to distinguish between paths
- ✓ Parent node with a single edge will never be a decision point

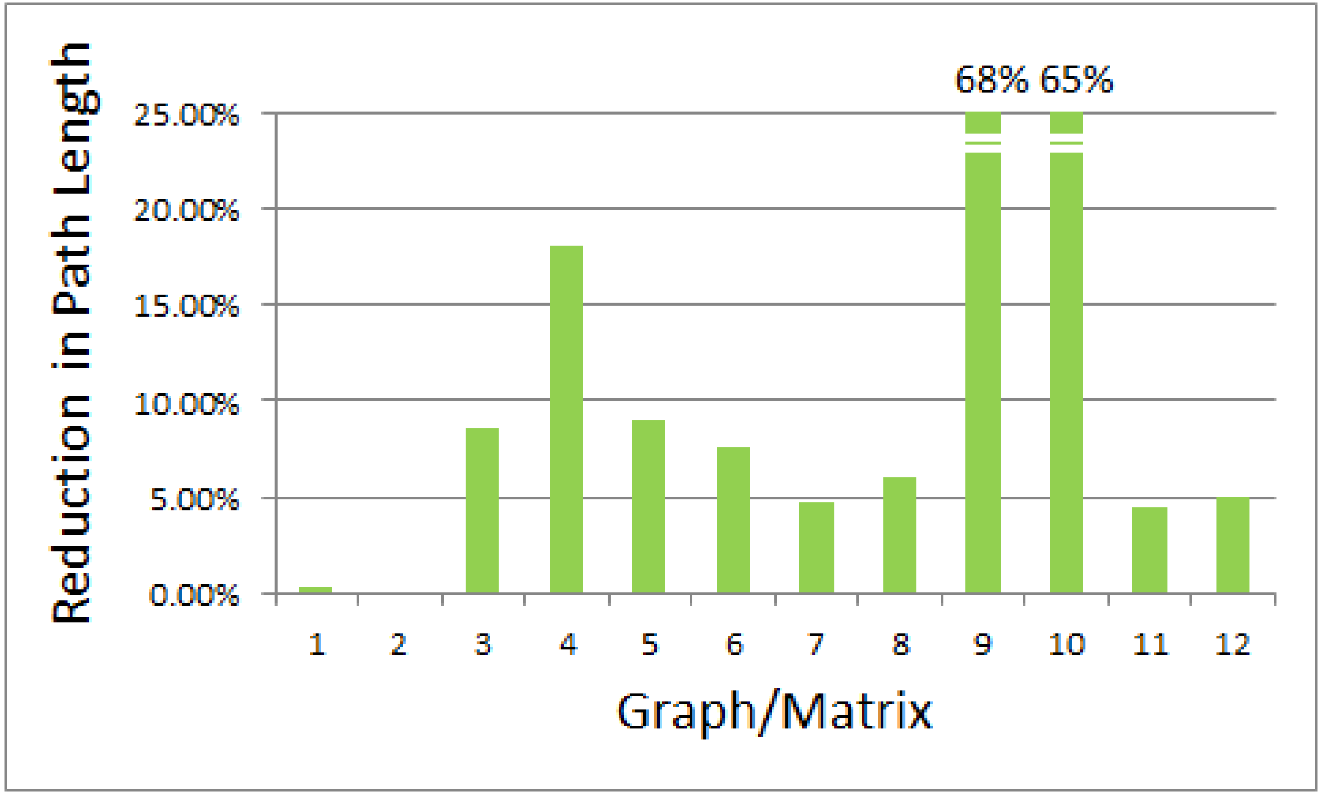
Path Pruning



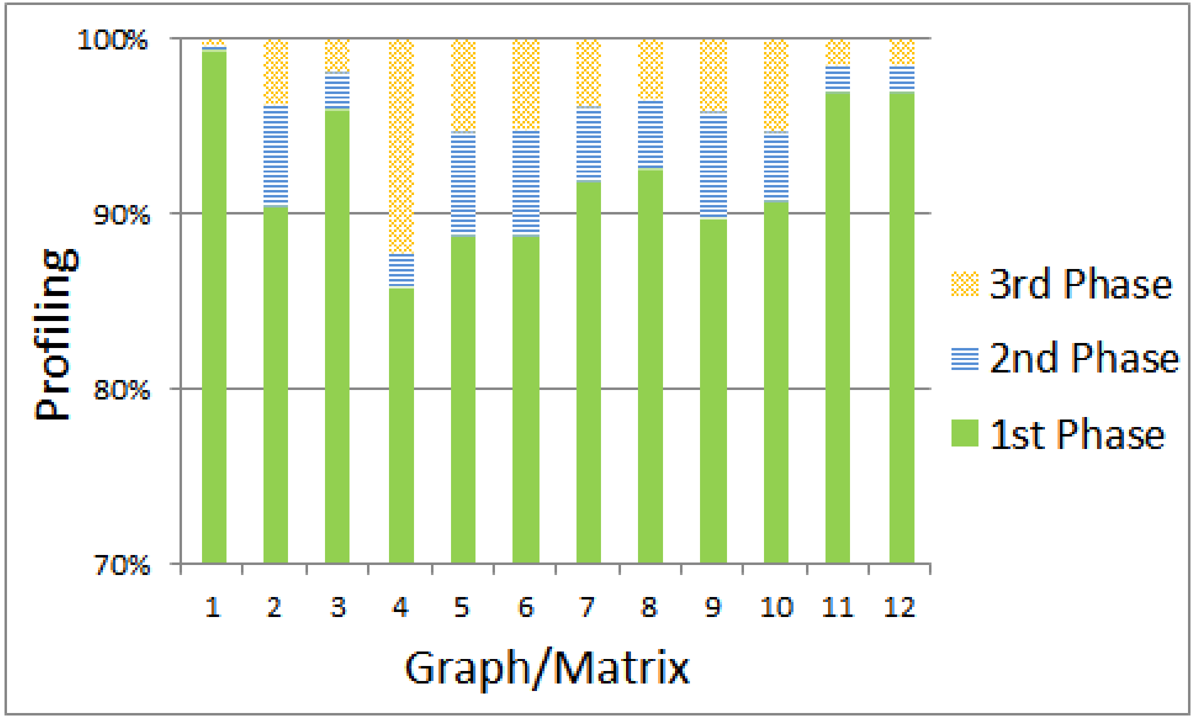
When two paths reach the same node

- ✓ There exists a parent “a” where the path split [a,b,…] and [a,c,…]
- ✓ It is the comparison between “b” and “c” that allows us to distinguish between paths
- ✓ Parent node with a single edge will never be a decision point
- ✓ No need to store nodes with such parents

Path Pruning

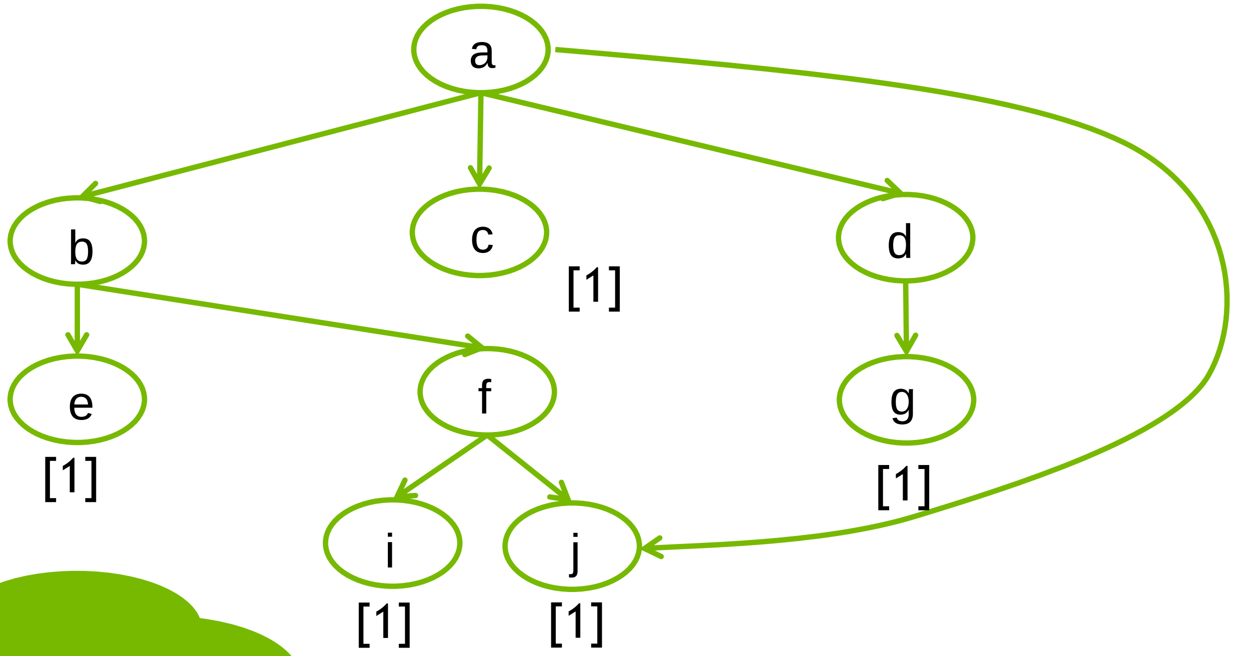


Phase Composition



SSSP-BASED VARIANT

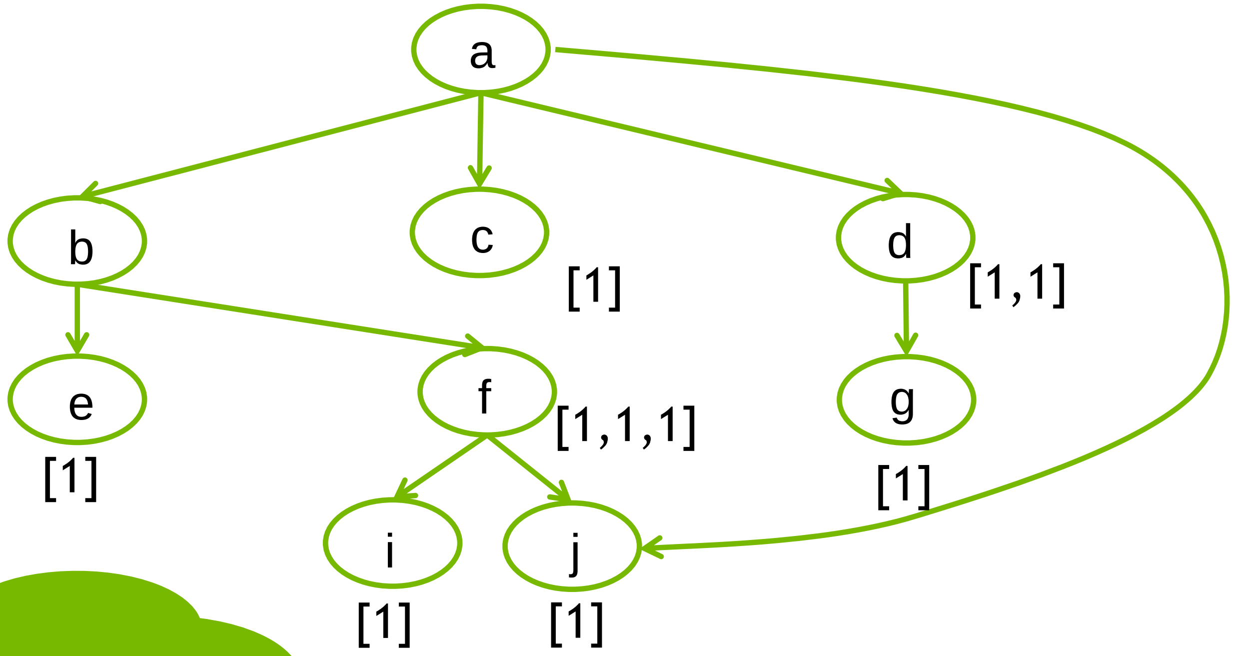
SSSP-based (for DAGs)



Run the algorithm for Directed Trees, but
- ✓ Propagate # of nodes to all the parents
- ✓ Start prefix sum with 1 (instead of 0)

Phase 1: Bottom-Up Traversal

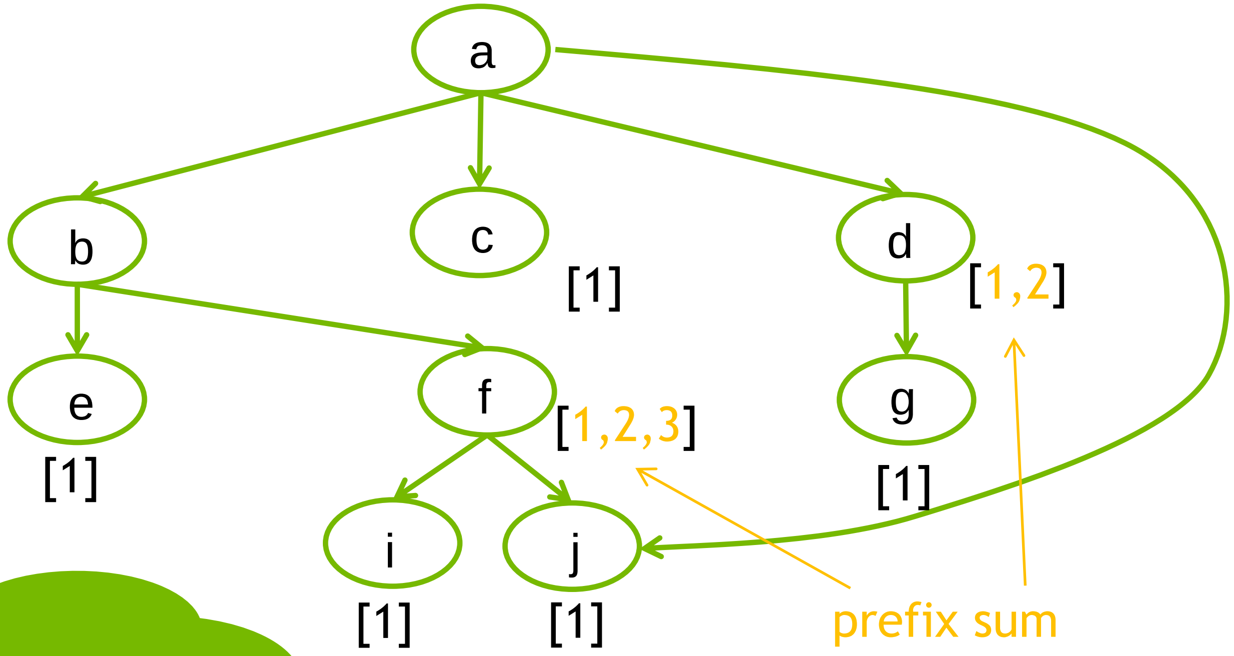
SSSP-based (for DAGs)



Run the algorithm for Directed Trees, but
- ✓ Propagate # of nodes to all the parents
- ✓ Start prefix sum with 1 (instead of 0)

Phase 1: Bottom-Up Traversal

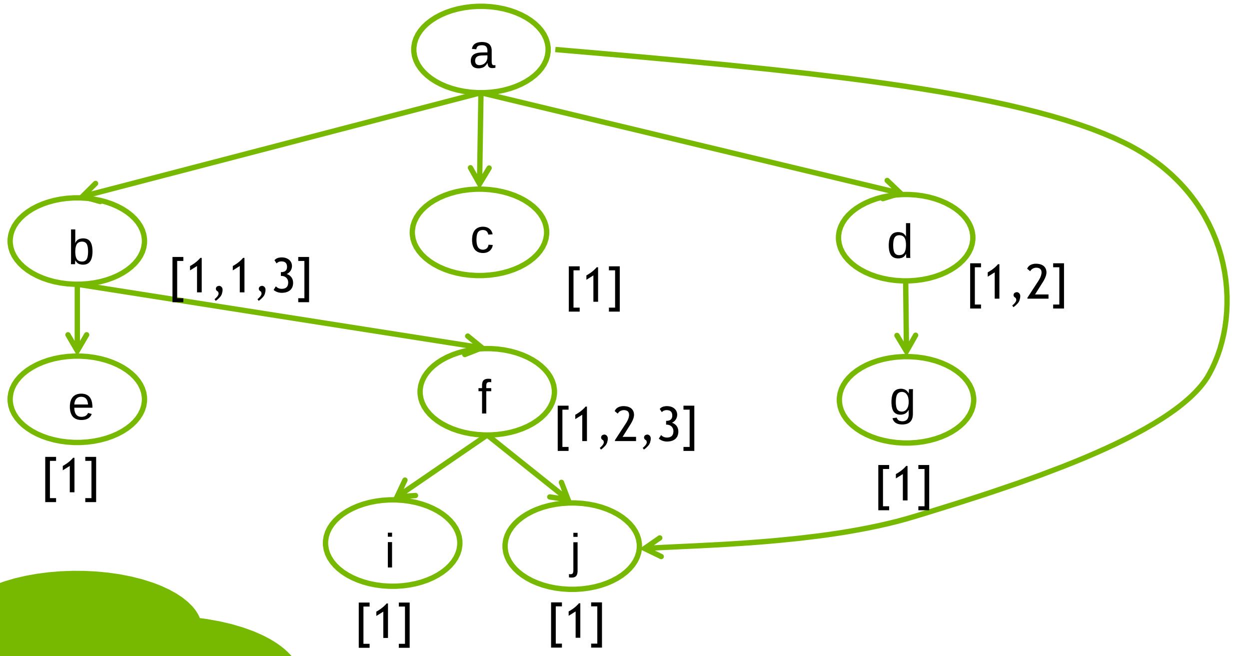
SSSP-based (for DAGs)



Run the algorithm for Directed Trees, but
- ✓ Propagate # of nodes to all the parents
- ✓ Start prefix sum with 1 (instead of 0)

Phase 1: Bottom-Up Traversal

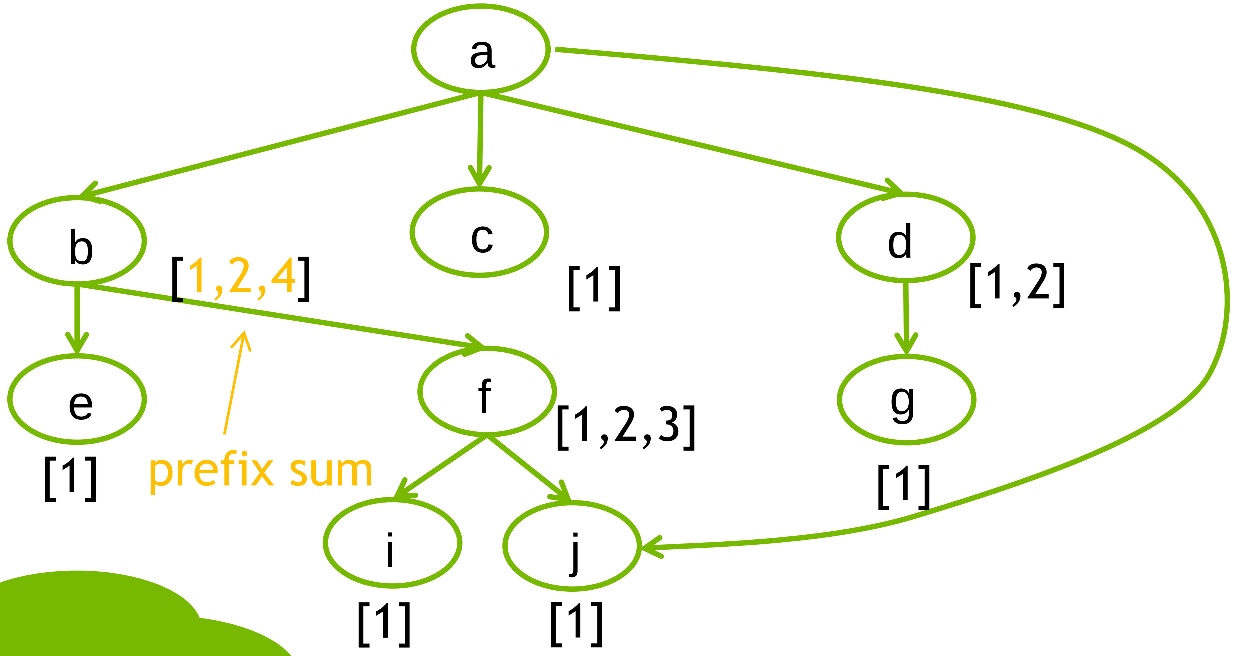
SSSP-based (for DAGs)



Run the algorithm for Directed Trees, but
- ✓ Propagate # of nodes to all the parents
- ✓ Start prefix sum with 1 (instead of 0)

Phase 1: Bottom-Up Traversal

SSSP-based (for DAGs)

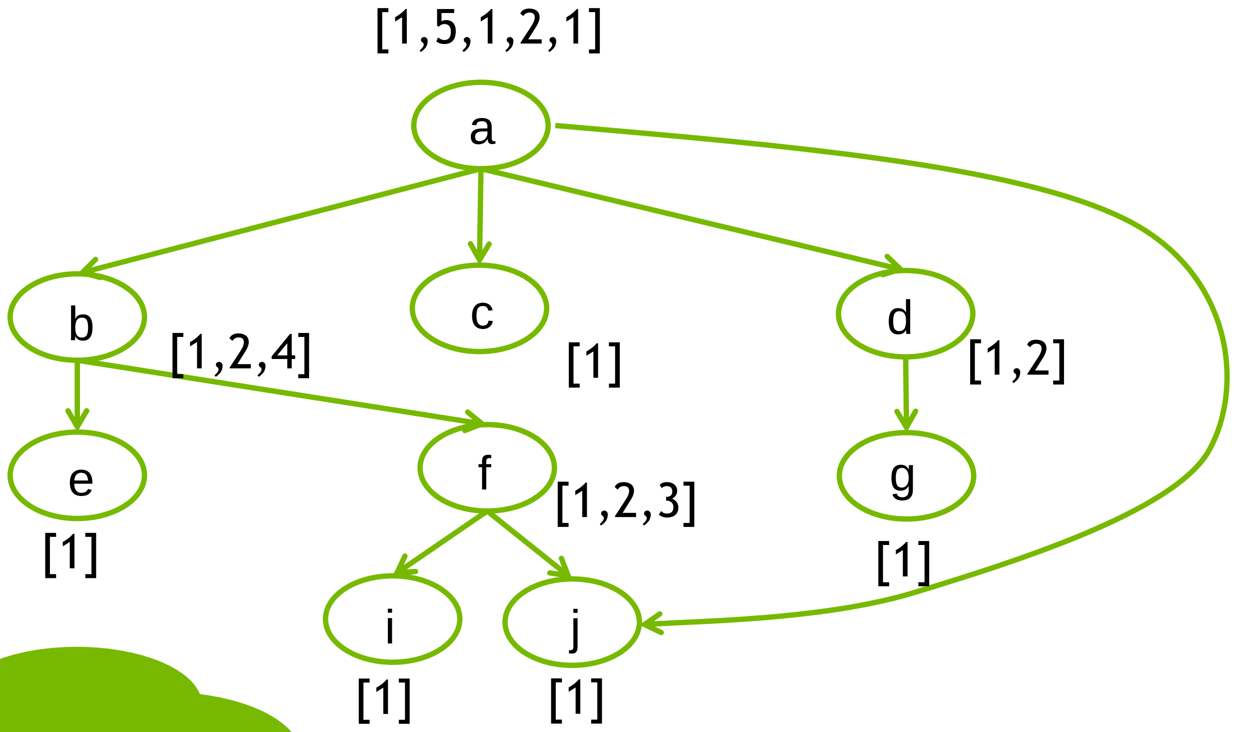


Run the algorithm for Directed Trees, but

- ✓ Propagate # of nodes to all the parents
- ✓ Start prefix sum with 1 (instead of 0)

Phase 1: Bottom-Up Traversal

SSSP-based (for DAGs)

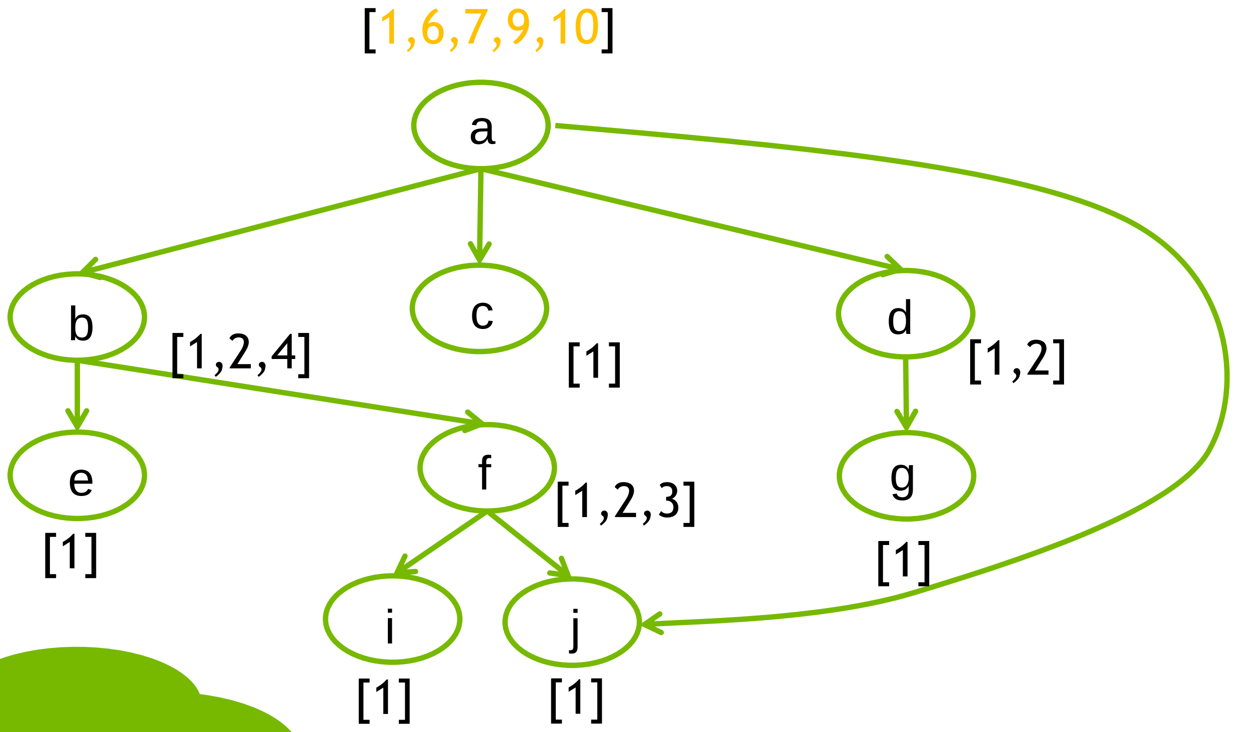


Run the algorithm for Directed Trees, but

- ✓ Propagate # of nodes to all the parents
- ✓ Start prefix sum with 1 (instead of 0)

Phase 1: Bottom-Up Traversal

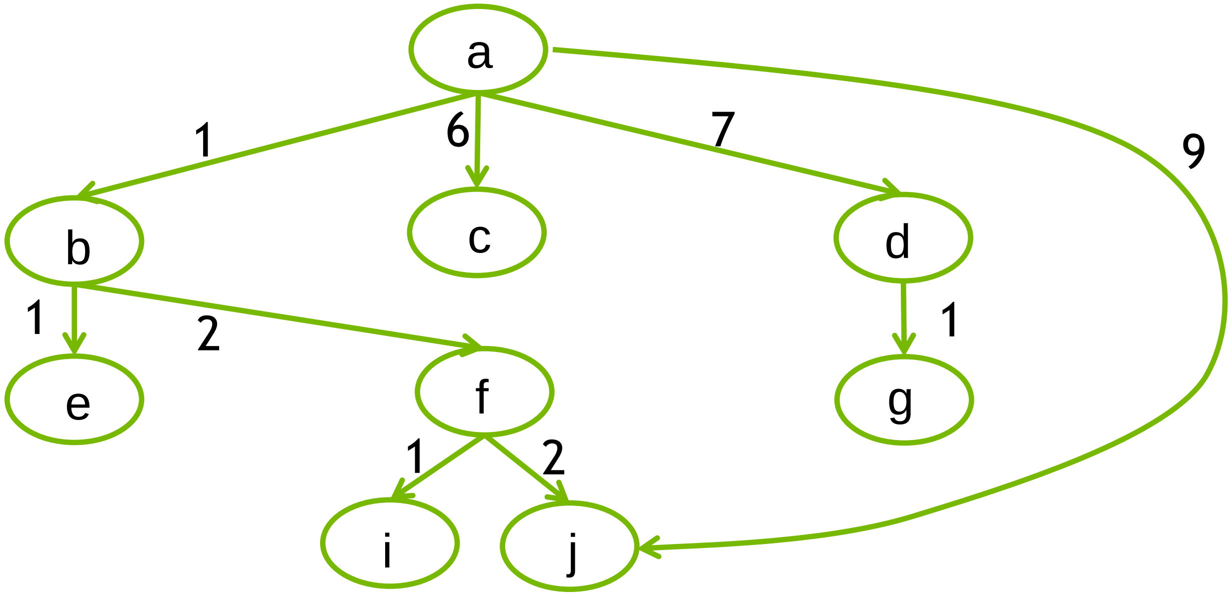
SSSP-based (for DAGs)



Run the algorithm for Directed Trees, but
- ✓ Propagate # of nodes to all the parents
- ✓ Start prefix sum with 1 (instead of 0)

Phase 1: Bottom-Up Traversal

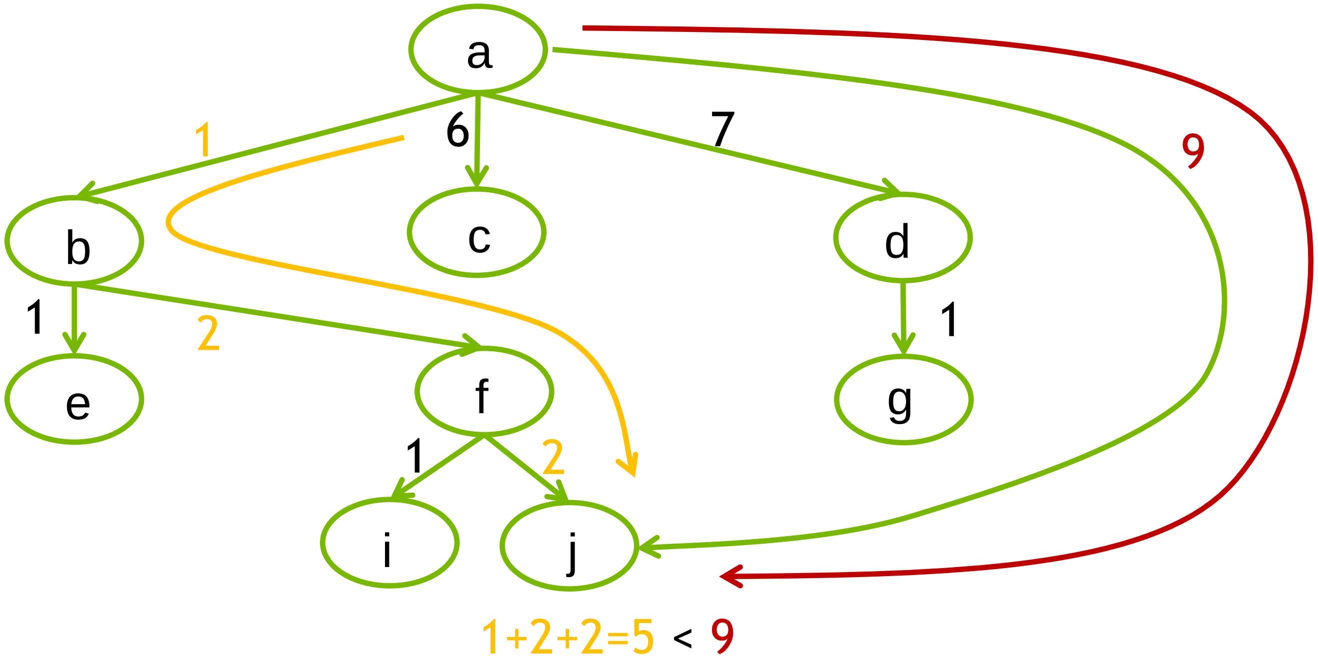
SSSP-based (for DAGs)



Assign # of nodes as the edge weight

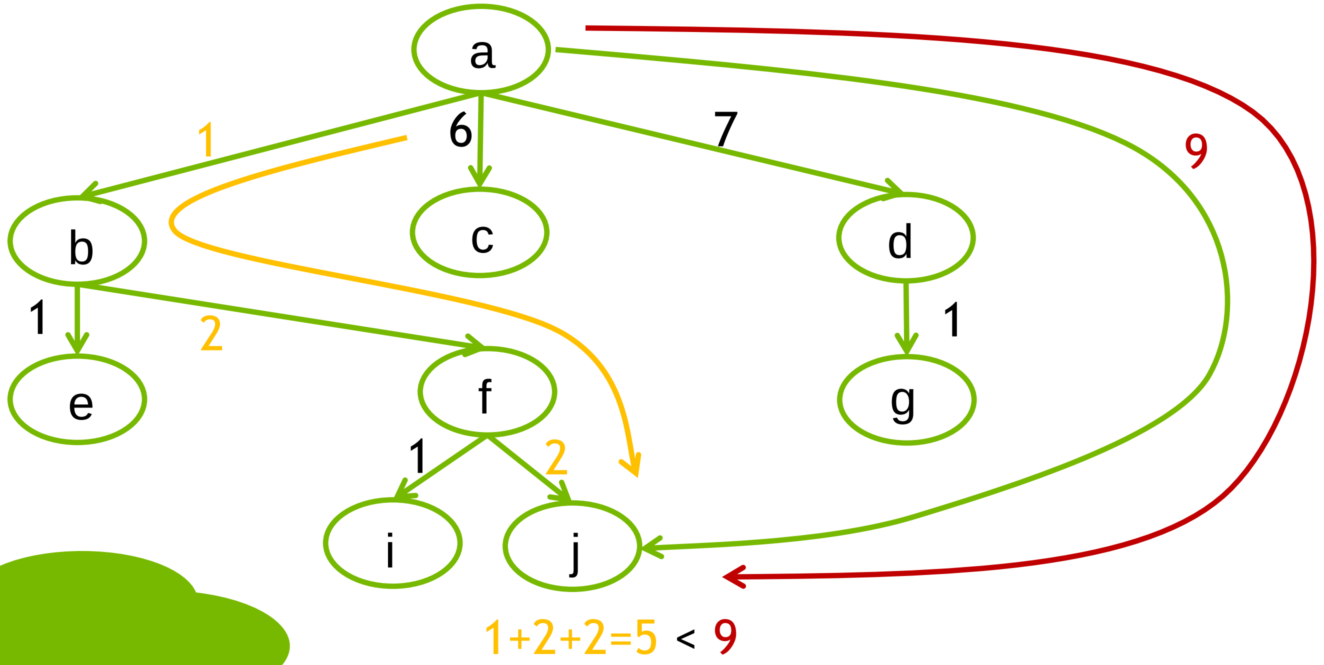
This phase is done, next phase is about to start …

SSSP-based (for DAGs)



Phase 2: Top-down traversal

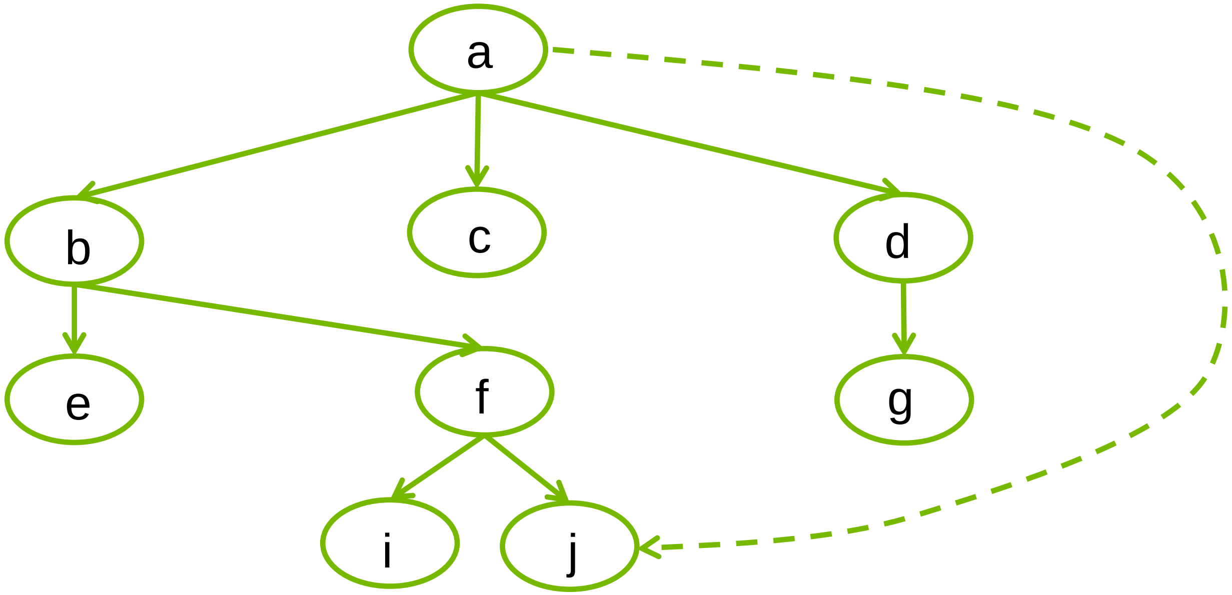
SSSP-based (for DAGs)



Shortest Path is the DFS path

Phase 2: Top-down traversal

SSSP-based (for DAGs)

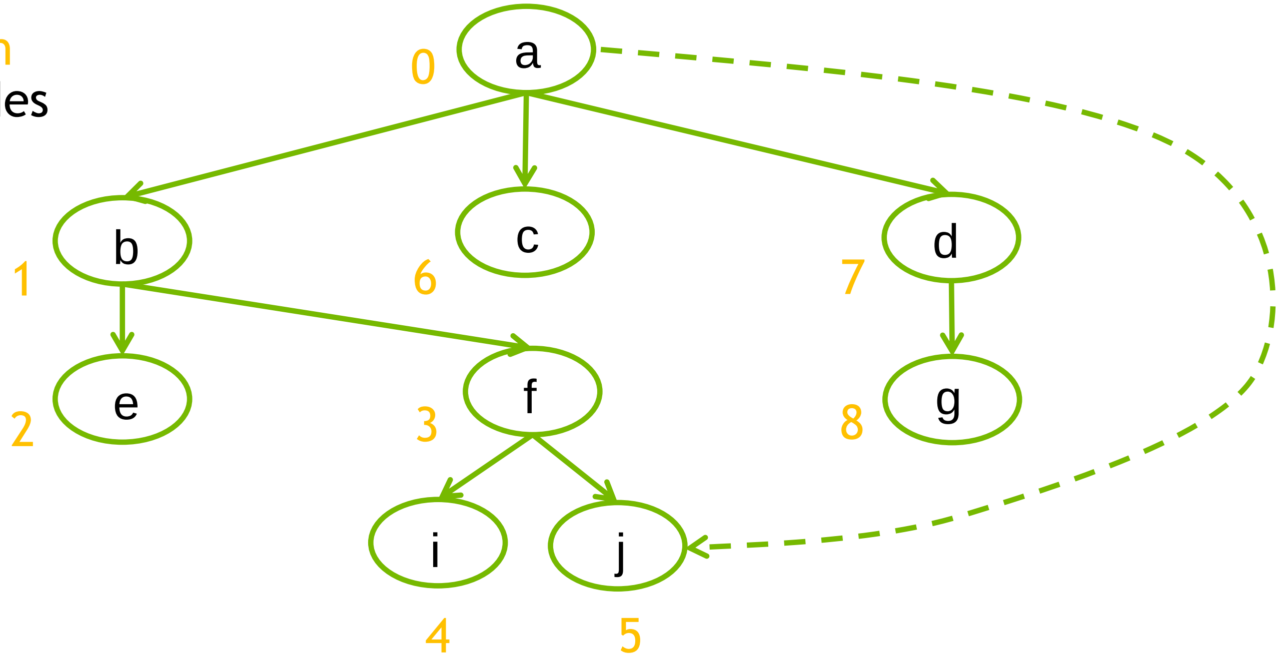


Phase 2: This phase is done

OPTIMIZATIONS

Discovery time

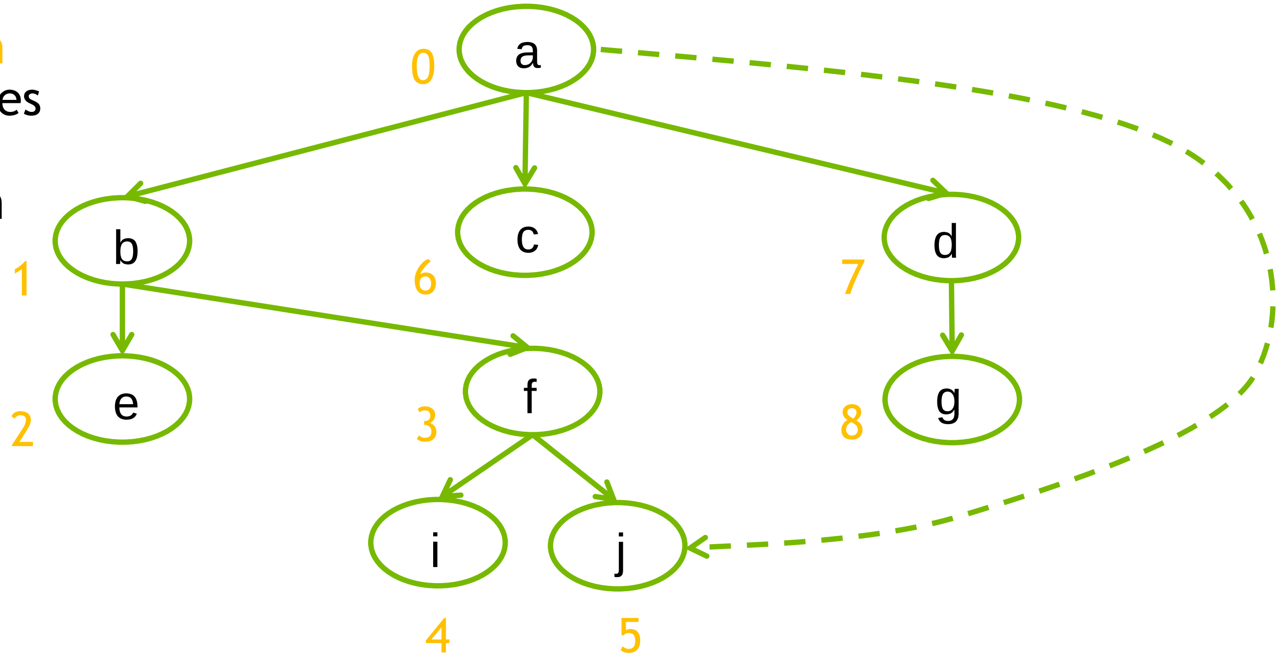
✓ The length of shortest path defines an ordering of nodes



Phase 3a: Sorting

Discovery time

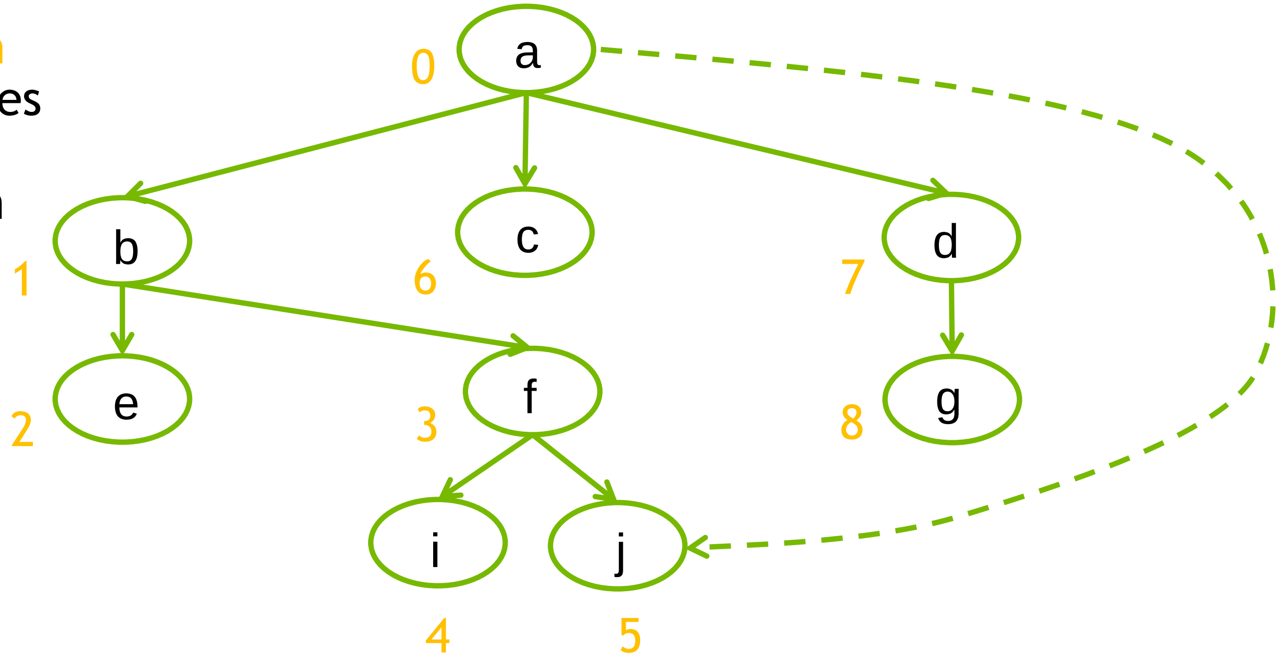
- ✓ The length of shortest path defines an ordering of nodes
- ✓ We can sort them to obtain discovery time



Phase 3a: Sorting

Discovery time

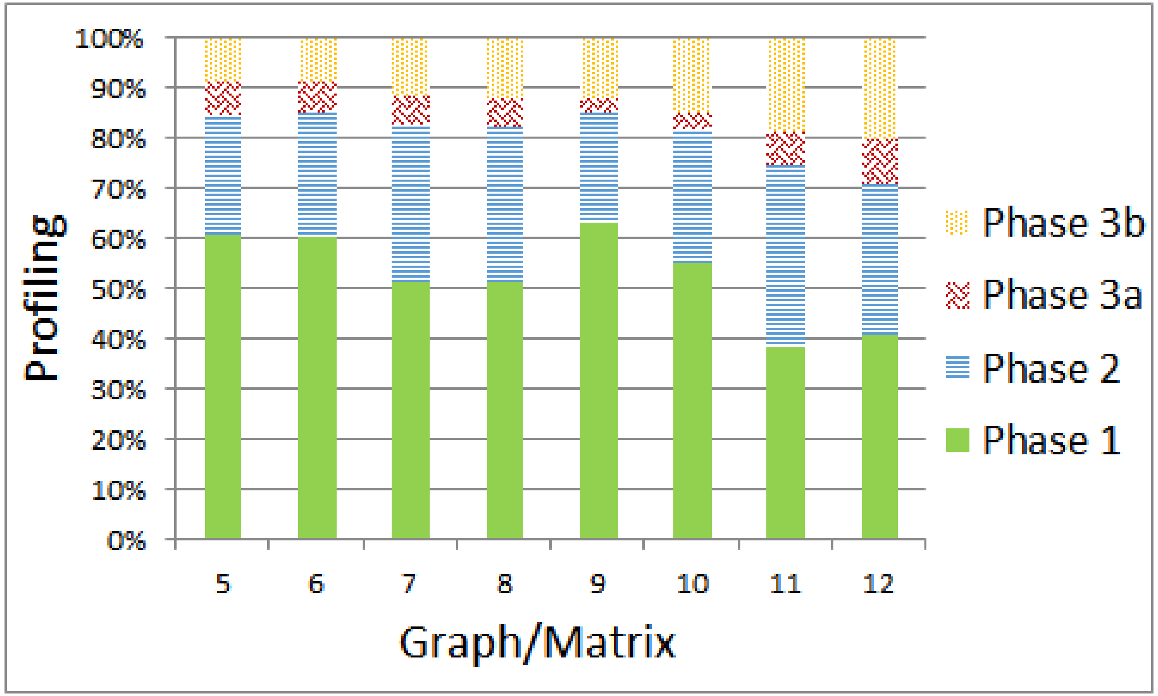
- ✓ The length of shortest path defines an ordering of nodes
- ✓ We can sort them to obtain discovery time



Discovery: a,b,e,f,i,j,c,d,g

Phase 3a: This phase is done
(Phase 3b will find the finish time)

Phase composition



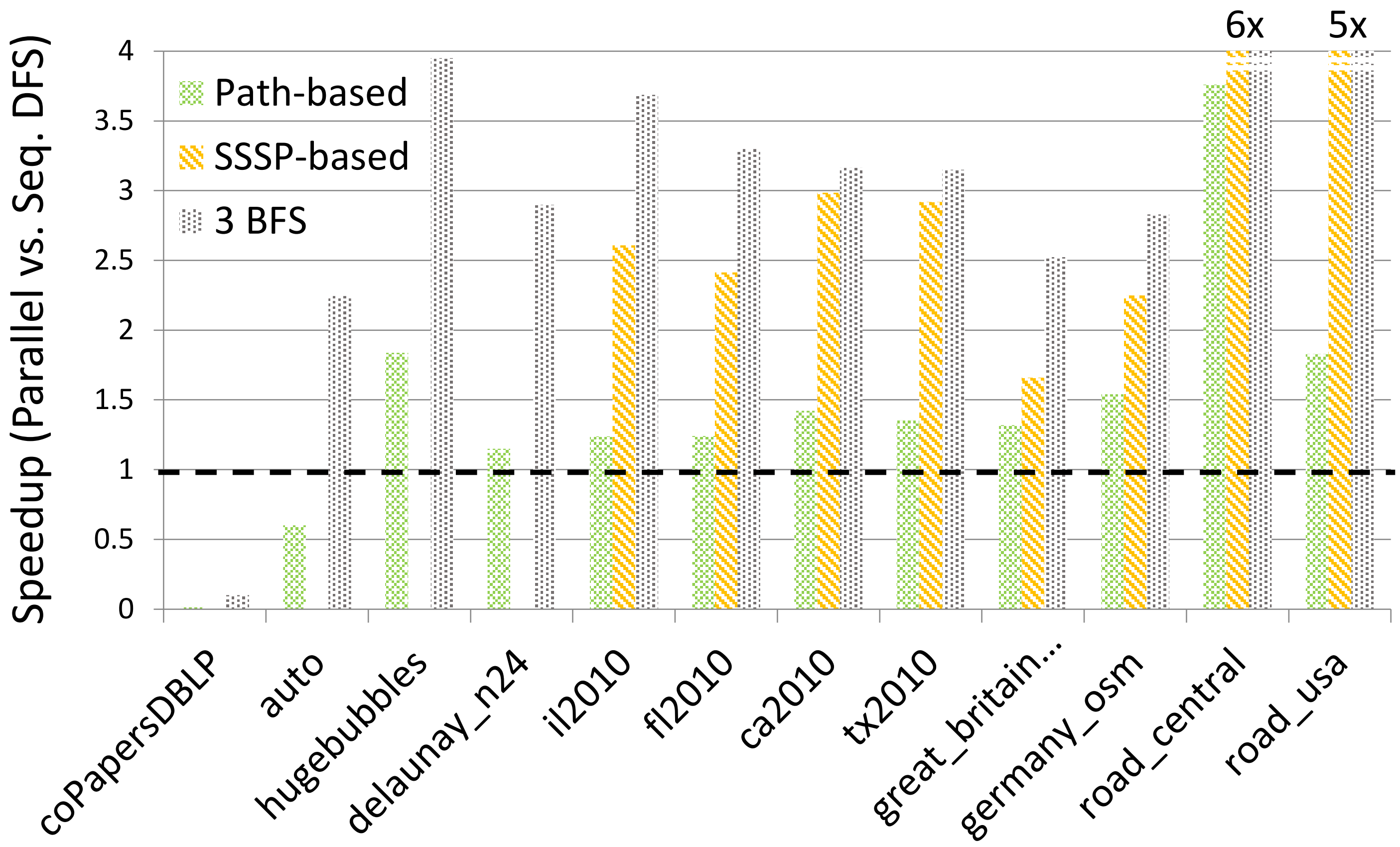
EXPERIMENTS

Data

#	Graph	n	m	Application
1	coPapersDBLP	540487	15251812	Citations
2	auto	448696	3350678	Numeric Sim.
3	hugebubbles-000…	18318144	30144175	Numeric Sim.
4	delaunay_n24	16777217	52556391	Random Tri.
5	il2010	451555	1166978	Census Data
6	fl2010	484482	1270757	Census Data
7	ca2010	710146	1880571	Census Data
8	tx2010	914232	2403504	Census Data
9	great-britain_osm	7733823	8523976	Road Network
10	germanu_osm	11548846	12793527	Road Network
11	road_central	14081817	21414269	Road Network
12	road_usa	23947348	35246600	Road Network

When necessary DAGs are created from general graphs by dropping back edges

Performance



Results obtained with Nvidia Pascal TitanX GPU, Intel Core i7-3930K @3.2GHz CPU, Ubuntu 14.04 LTS OS, CUDA Toolkit 8.0

CONCLUSIONS

Conclusions

- Parallel DFS for DAGs
  - ✓ Work-efficient O(m+n)
  - ✓ The algorithm takes O(z log n) steps, where z is the maximum depth of a node
- Performance
  - ✓ Depends highly on the connectivity/sparsity pattern
  - ✓ Can achieve up to 6x speedup (but slowdown possible)
- Details
  - ✓ M. Naumov, A. Vrielink and M. Garland, “Parallel Depth-First Search for Directed Acyclic Graphs”, Technical Report, NVR-2017-001, 2017
    https://research.nvidia.com/publication/parallel-depth-first-search-directed-acyclic-graphs

Thank you

https://research.nvidia.com/publication/parallel-depth-first-search-directed-acyclic-graphs