



COMP4317: XML & Database

Tutorial 7: Xpath & DAG

Week 8

Thang Bui @ CSE.UNSW



DAG file into unranked Tree

- For each line in the text file
 - Get id, tag, list of (multiplicity) children
 - Make a node (id, tag)
 - Put to a hashtable (id, node)
 - Parse the list of children
 - Find in hashtable the child with child id
 - addChild(child)



DAG file into unranked Tree

- Multiplicity
 - Option 1: list of nodes with multiplicity
 - Option 2: list of duplicated nodes



Travel on DAG

- Option 2: Same as unranked Tree
- Option 1: repeat the number of multiplicity at each child



Next sibling on DAG

- Do it on-the-fly
- When checking for a child, we know:
 - The parent
 - The child number (on list of children)
 - The multiplicity (option 1)
- Want to move to the next sibling?
 - Ask the parent for the my child number
 - Retrieve the node from the next child number from the parent



Next sibling on DAG

- If a next-sibling filter fail?
 - The next of that next-sibling may help
 - Wait until no next-sibling success => totally fail



Streaming Xpath on DAG? Top-down Evaluation

- Remember DAG nodes with partial matched expression
- OR
- Remember expression with DAG nodes

= > Just a few numbers!



Example 1

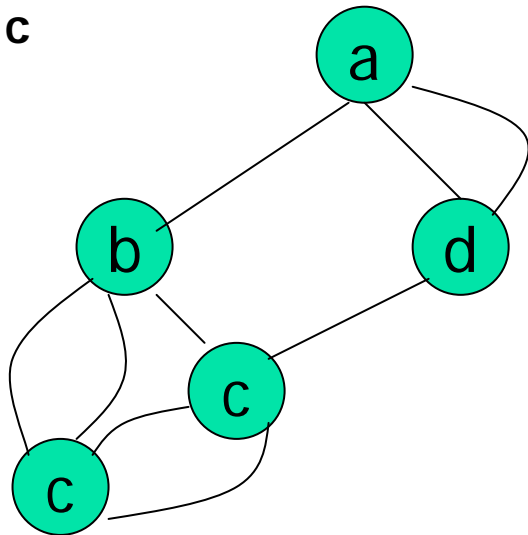
5:a[3,4:2] ex: //*/*///c

4:d[2]

3:b[1:2,2]

2:c[1:2]

1:c





Example 1

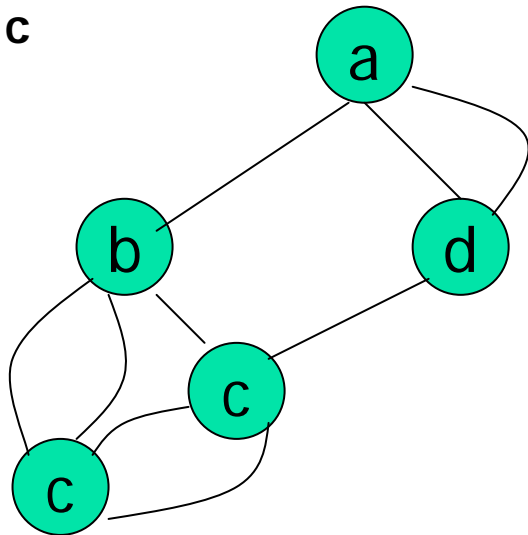
→ 5:a[3,4:2] ex: //*/*/c

4:d[2]

3:b[1:2,2]

2:c[1:2]

1:c



Example 1

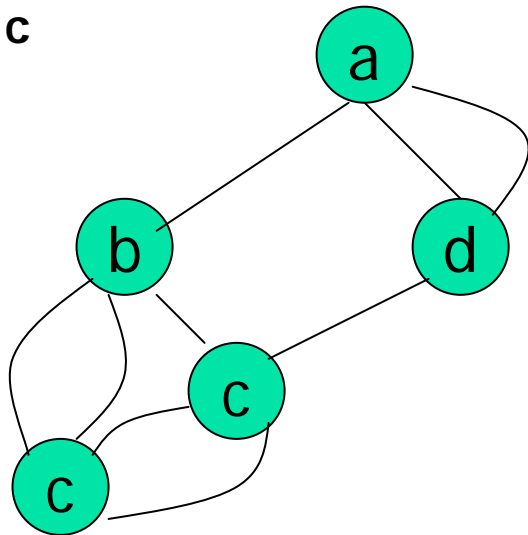
→ 5:a[3,4:2] ex: //*/*//c

4:d[2]

3:b[1:2,2]

2:c[1:2]

1:c



Partial matched at 0 =>

idx: 1 => child id: 3,4

Example 1

5:a[3,4:2] ex: //*/*///c

DAG nodes

→ 4:d[2]

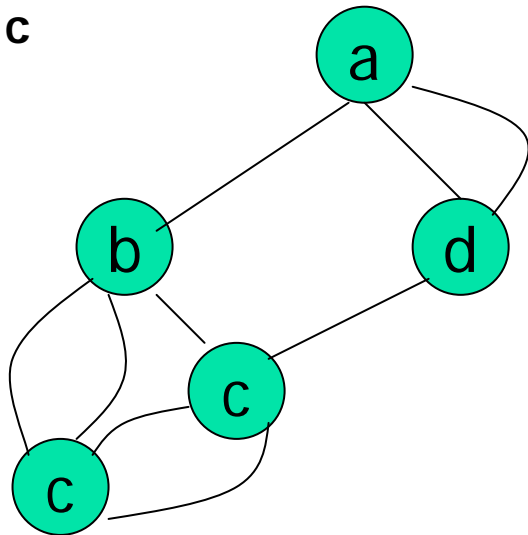
<id:3,idx:1>

3:b[1:2,2]

<id:4,idx:1>

2:c[1:2]

1:c



Partial matched at 0 =>

idx: 1 => child id: 2

Example 1

5:a[3,4:2] ex: //*/*///c DAG nodes

→ 4:d[2]

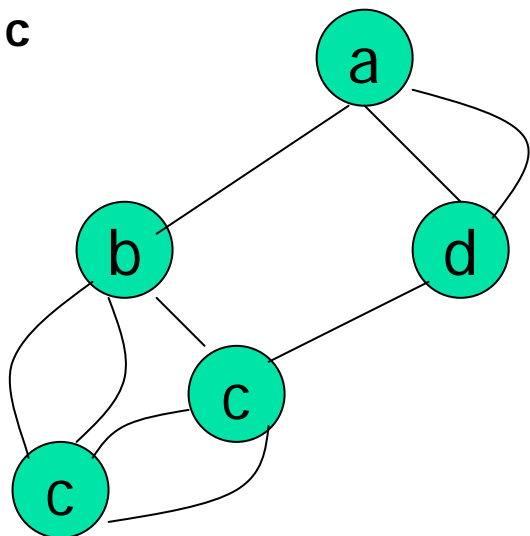
<id:3,idx:1>

3:b[1:2,2]

<id:4,idx:1>

2:c[1:2]

1:c



Partial matched at 0 =>

idx: 1 => child id: 2

“Previous” partial matched (id:4)

idx: 2 => child id: 2

Example 1

5:a[3,4:2] ex: //*/*///c

DAG nodes

→ 4:d[2]

<id:3,idx:1>

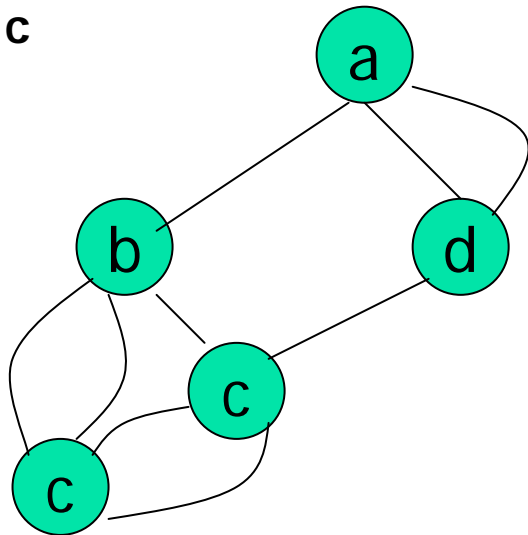
3:b[1:2,2]

<id:4,idx:1>

2:c[1:2]

<id:2,idx:1,2>

1:c



Example 1

5:a[3,4:2] ex: //*/*///c

DAG nodes

4:d[2]

<id:3,idx:1>

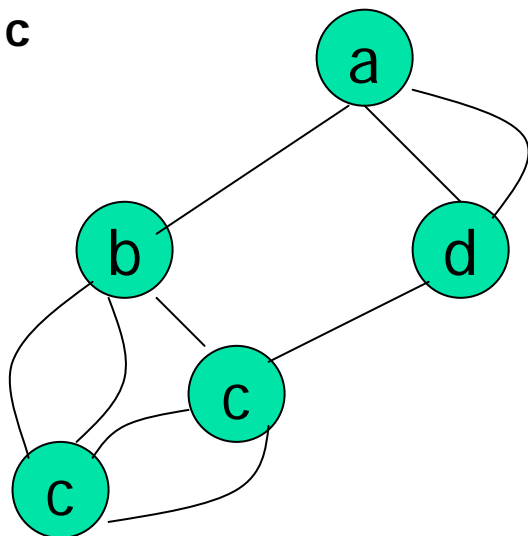
→ 3:b[1:2,2]

<id:4,idx:1>

2:c[1:2]

<id:2,idx:1,2>

1:c



Partial matched at 0 =>

idx: 1 => child id: 1,2

“Previous” partial matched (id:3)

idx: 2 => child id: 1,2

Example 1

5:a[3,4:2] ex: //*/*///c

4:d[2]

→ 3:b[1:2,2]

2:c[1:2]

1:c

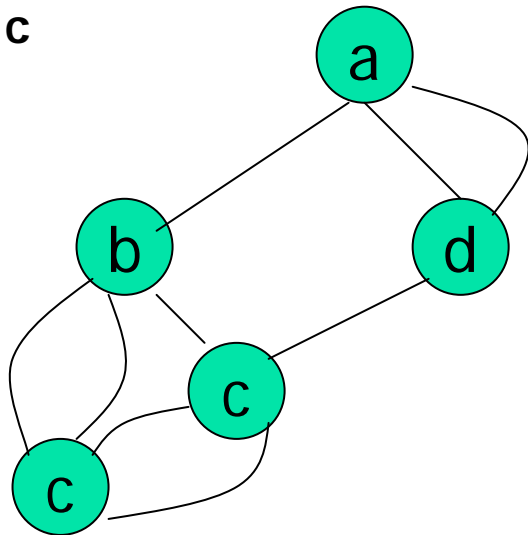
DAG nodes

<id:3,idx:1>

<id:4,idx:1>

<id:2,idx:1,2>

<id:1,idx:1,2>



Example 1

5:a[3,4:2] ex: //*/*///c

4:d[2]

3:b[1:2,2]

→ 2:c[1:2]

1:c

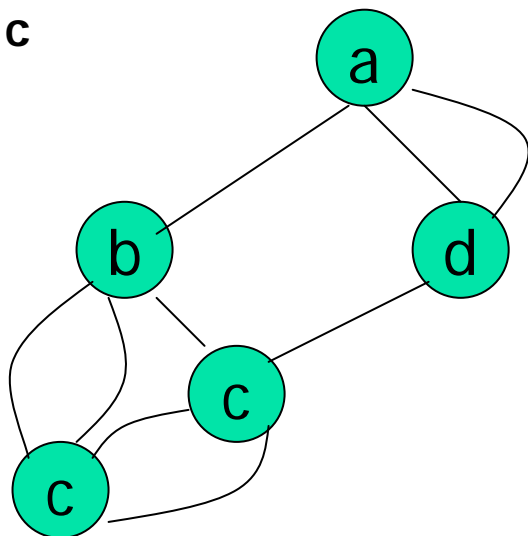
DAG nodes

<id:3,idx:1>

<id:4,idx:1>

<id:2,idx:1,2>

<id:1,idx:1,2>



Partial matched at 0 =>

idx: 1 => child id: 1

“Previous” partial matched (id:2)

idx: 2 => child id: 1

idx: 3 => full matching: output

Example 1

5:a[3,4:2] ex: //*/*///c

4:d[2]

3:b[1:2,2]

→ 2:c[1:2]

1:c

DAG nodes

<id:3,idx:1>

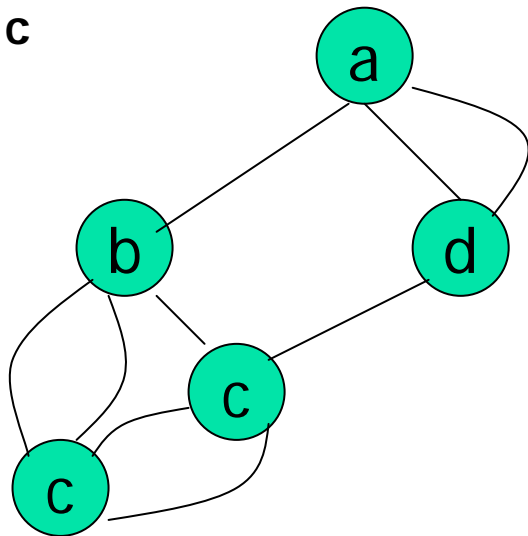
<id:4,idx:1>

<id:2,idx:1,2>

<id:1,idx:1,2>

Output List:

2:c[1:2]



Example 1

5:a[3,4:2] ex: //*/*///c

4:d[2]

3:b[1:2,2]

2:c[1:2]

→ 1:c

DAG nodes

<id:3,idx:1>

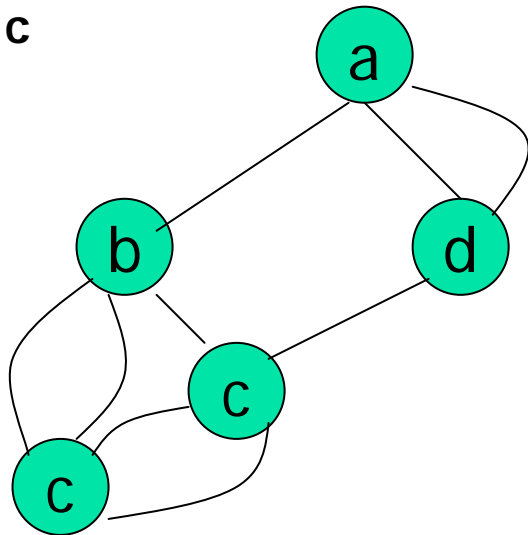
<id:4,idx:1>

<id:2,idx:1,2>

<id:1,idx:1,2>

Output List:

2:c[1:2]



Partial matched at 0 =>

idx: 1 => child id: NO

“Previous” partial matched (id:1)

idx: 2 => child id: NO

idx: 3 => full matching: output

Example 1

5:a[3,4:2] ex: //*/*///c

4:d[2]

3:b[1:2,2]

2:c[1:2]

→ 1:c

DAG nodes

<id:3,idx:1>

<id:4,idx:1>

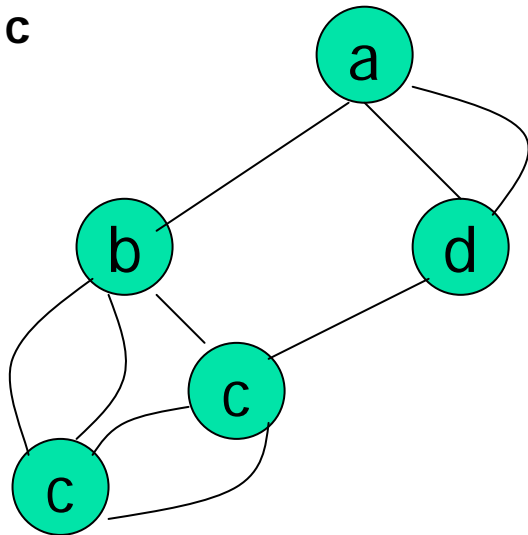
<id:2,idx:1,2>

<id:1,idx:1,2>

Output List:

2:c[1:2]

1:c



Example 2

5:a[3,4:2] ex: /*//c

DAG nodes

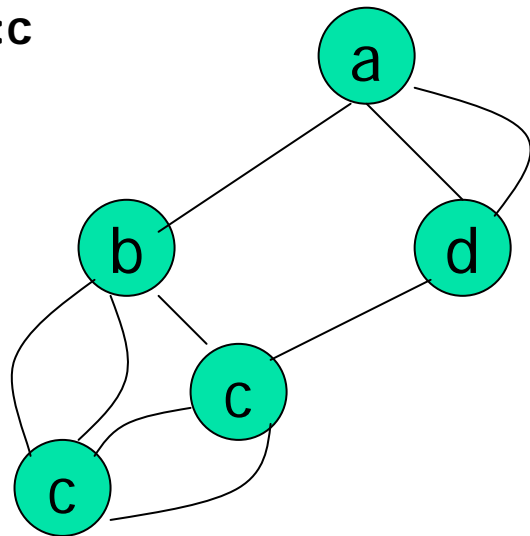
Output List:

4:d[2]

3:b[1:2,2]

2:c[1:2]

1:c



Example 2

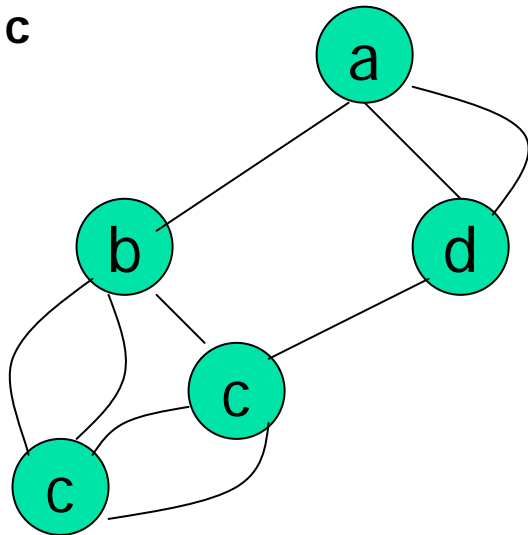
→ 5:a[3,4:2] ex: /*//c
4:d[2]
3:b[1:2,2]
2:c[1:2]
1:c

DAG nodes

<id:3,idx:1>

<id:4,idx:1>

Output List:



Partial matched at 0 =>

idx: 1 => child id: 3,4

Example 2

5:a[3,4:2] ex: /*//c

→ 4:d[2]

3:b[1:2,2]

2:c[1:2]

1:c

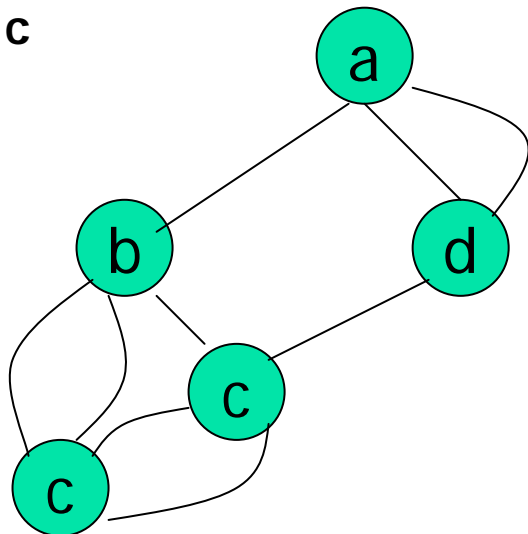
DAG nodes

<id:3,idx:1>

<id:4,idx:1>

<id:2,idx:1>

Output List:



“Previous” partial matched (id:4)

NO match => keep idx for //

idx: 1 => child id: 2

Example 2

5:a[3,4:2] ex: /*//c

4:d[2]

→ 3:b[1:2,2]

2:c[1:2]

1:c

DAG nodes

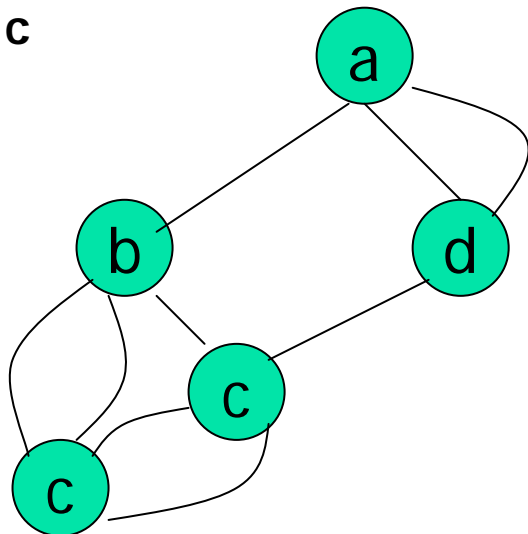
<id:3,idx:1>

<id:4,idx:1>

<id:2,idx:1>

<id:1,idx:1>

Output List:



“Previous” partial matched (id:3)

NO match => keep idx for //

idx: 1 => child id: 1,2

Example 2

5:a[3,4:2] ex: /*//c

4:d[2]

3:b[1:2,2]

→ 2:c[1:2]

1:c

DAG nodes

<id:3,idx:1>

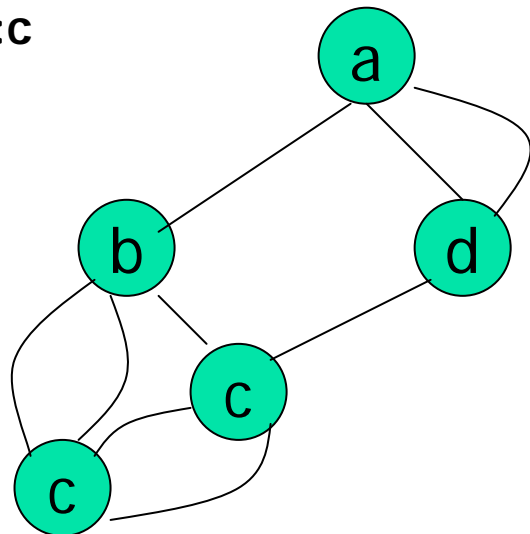
<id:4,idx:1>

<id:2,idx:1>

<id:1,idx:1>

Output List:

2:c[1:2]



“Previous” partial matched (id:2)

idx: 2 => full matching: output

idx: 1 => child id: 1 (for //)

Example 2

5:a[3,4:2] ex: /*//c

4:d[2]

3:b[1:2,2]

2:c[1:2]

→ 1:c

DAG nodes

<id:3,idx:1>

<id:4,idx:1>

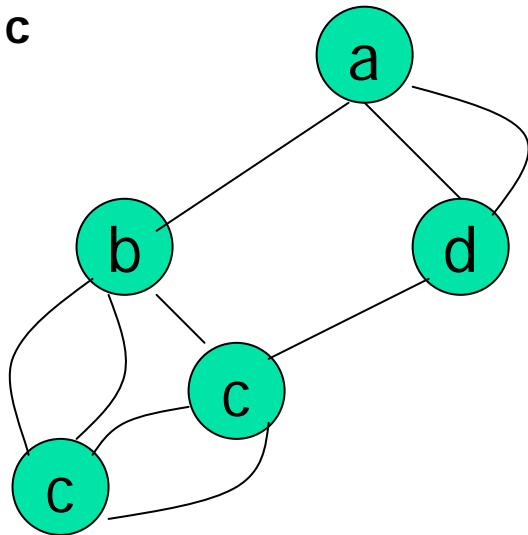
<id:2,idx:1>

<id:1,idx:1>

Output List:

2:c[1:2]

1:c



“Previous” partial matched (id:1)

idx: 2 => full matching: output

idx: 1 => child id: NO

Reduce the memory

5:a[3,4:2] ex: /*//c

4:d[2]

3:b[1:2,2]

2:c[1:2]

→ 1:c

DAG nodes

~~<id:3,idx:1>~~

~~<id:4,idx:1>~~

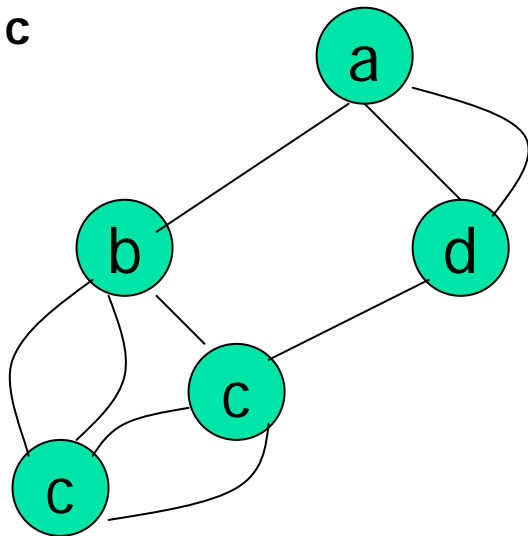
~~<id:2,idx:1>~~

<id:1,idx:1>

Output List:

2:c[1:2]

1:c





Streaming Xpath on DAG? Bottom-up Evaluation

- Reverse Expression
- Remember DAG nodes with partial matched filters
- OR
- Remember exp with DAG nodes



Example

1:c

F1: c\b\\.

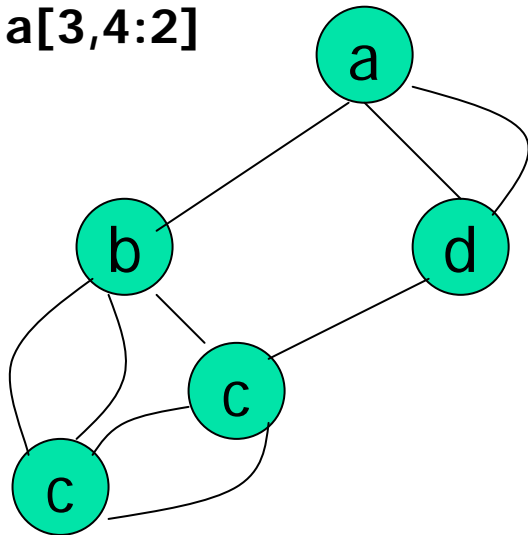
2:c[1:2]

F2: c\c\b\\.

3:b[1:2,2]

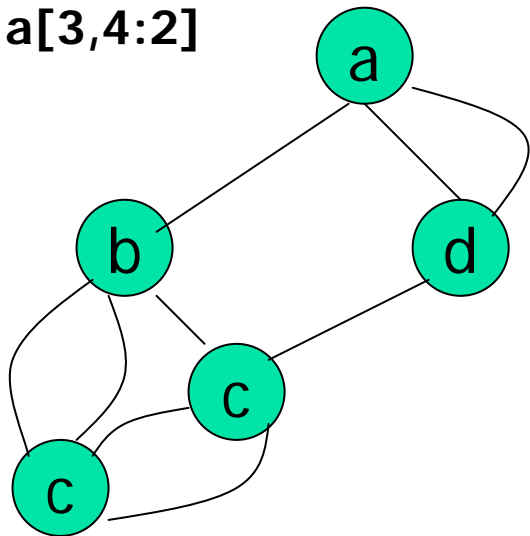
4:d[2]

5:a[3,4:2]



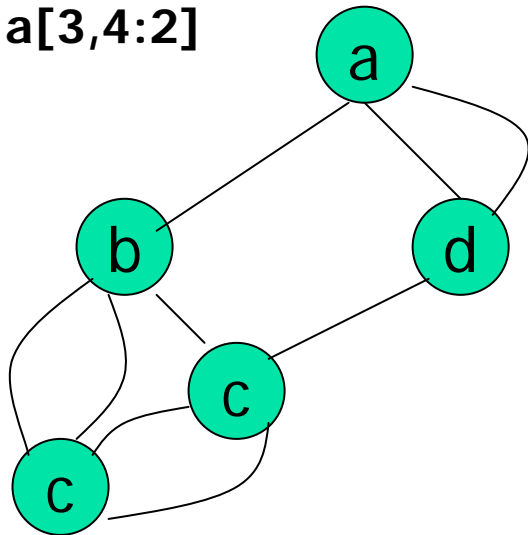
Example

- 1:c F1: c\b\\.
- 2:c[1:2] F2: c\c\b\.
- 3:b[1:2,2]
- 4:d[2]
- 5:a[3,4:2]



Example

- 1:c F1: c\b\\.
- 2:c[1:2] F2: c\c\b\\.
- 3:b[1:2,2]
- 4:d[2]
- 5:a[3,4:2]



Partial matched:

F1: 0

F2: 0

Example



1:c

F1: c\b\\.

DAG nodes

Output:

2:c[1:2]

F2: c\c\b\\.

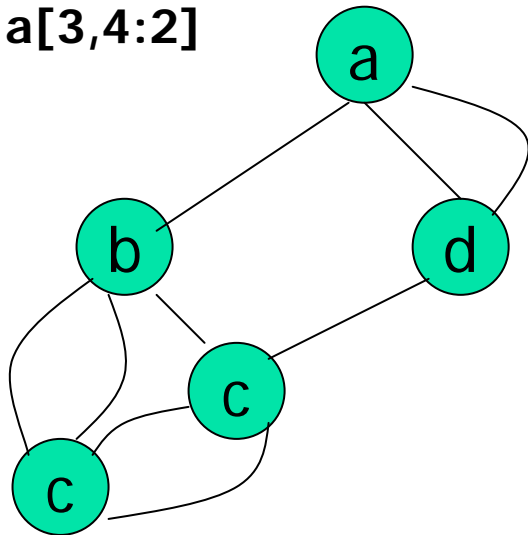
p1 <id:1,F1:0,F2:0>

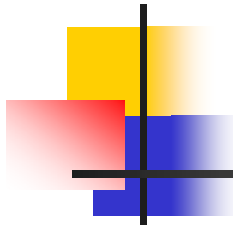
1:c

3:b[1:2,2]

4:d[2]

5:a[3,4:2]





Example

1:c

F1: c\b\\.

DAG nodes

Output:



2:c[1:2]

F2: c\c\b\\.

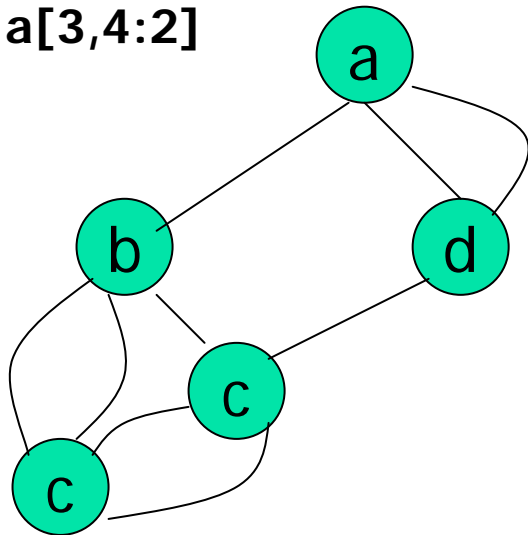
p1 <id:1,F1:0,F2:0>

1:c

3:b[1:2,2]

4:d[2]

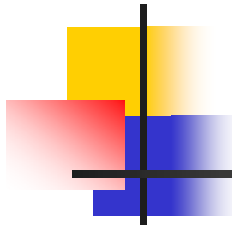
5:a[3,4:2]



Partial matched:

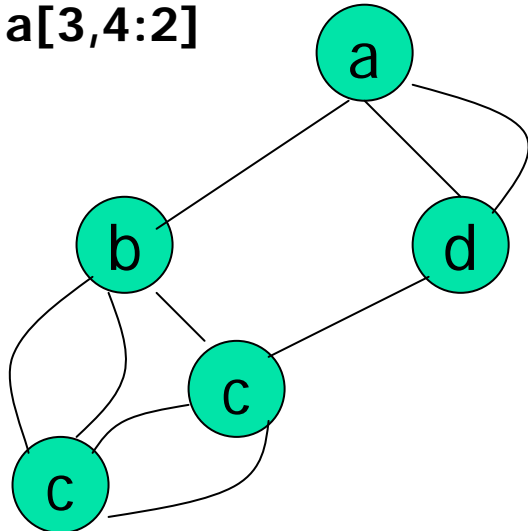
F1: 0

F2: 0



Example

| | | | |
|------------|-------------|---------------------|----------------|
| 1:c | F1: c\b\\. | DAG nodes | Output: |
| → 2:c[1:2] | F2: c\c\b\. | p1 <id:1,F1:0,F2:0> | 1:c |
| 3:b[1:2,2] | | | |
| 4:d[2] | | | |
| 5:a[3,4:2] | | | |

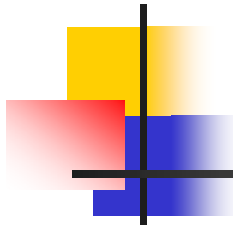


Partial matched:

F1: 0

F2: 0

F2: 1 (from its children id:1)



Example

1:c

F1: c\b\\.

DAG nodes

Output:



2:c[1:2]

F2: c\c\b\.

p1 <id:1,F1:0,F2:0>

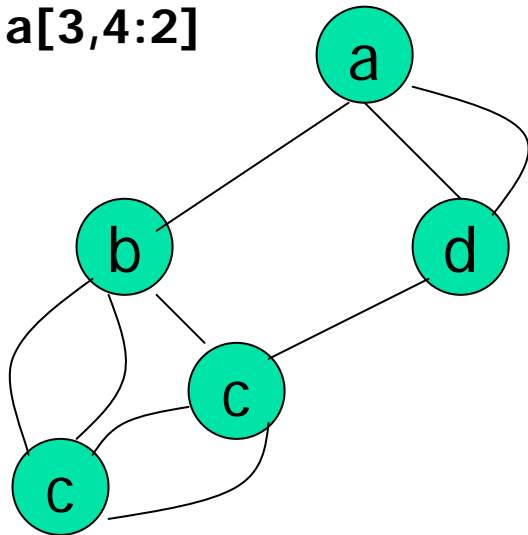
1:c

3:b[1:2,2]

p2 <id:2,F1:0,F2:0,F2:1> 2:c[1:2]

4:d[2]

5:a[3,4:2]



Example

1:c

F1: c\b\\.

DAG nodes

Output:

2:c[1:2]

F2: c\c\b\\.

p1 <id:1,F1:0,F2:0>

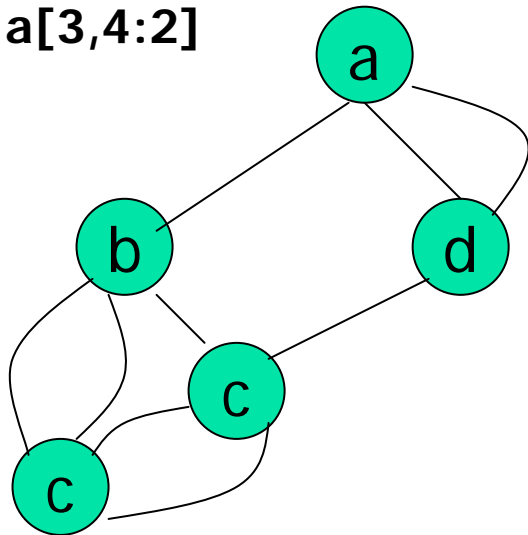
1:c

→ 3:b[1:2,2]

p2 <id:2,F1:0,F2:0,F2:1> 2:c[1:2]

4:d[2]

5:a[3,4:2]



Partial matched:

F1: 1 (from its children id:1)

F2: 2 (from its children id:2)

Example

1:c

F1: c\b\\.

DAG nodes

Output:

2:c[1:2]

F2: c\c\b\\.

p1 <id:1,F1:0,F2:0>

1:c

→ 3:b[1:2,2]

p2 <id:2,F1:0,F2:0,F2:1>

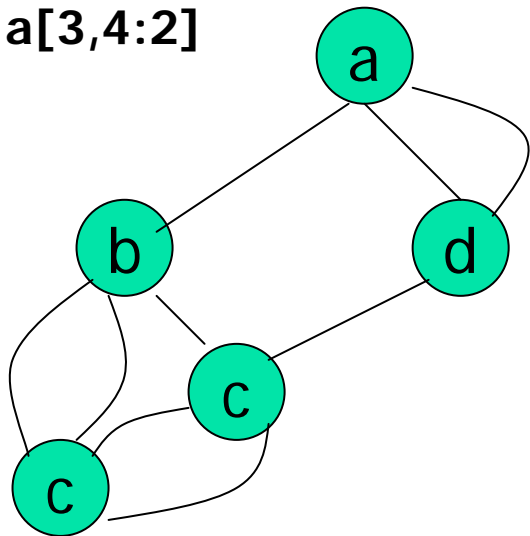
2:c[1:2]

4:d[2]

p3 <id:3,F1:1,F2:2>

3:b[1:2,2]

5:a[3,4:2]



Example

1:c

F1: c\b\\.

DAG nodes

Output:

2:c[1:2]

F2: c\c\b\\.

p1 <id:1,F1:0,F2:0>

1:c

3:b[1:2,2]

p2 <id:2,F1:0,F2:0,F2:1>

2:c[1:2]

→ 4:d[2]

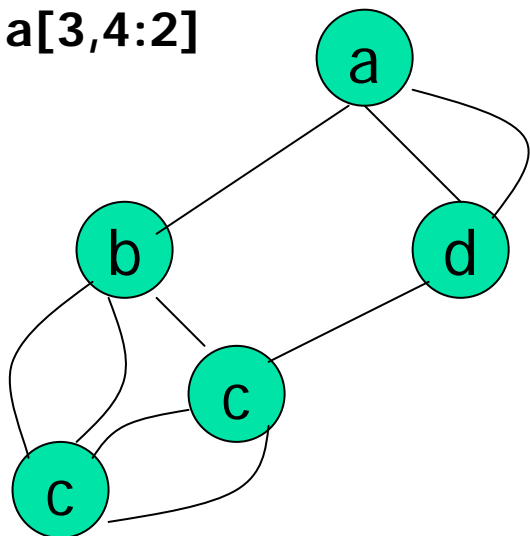
p3 <id:3,F1:1,F2:2>

3:b[1:2,2]

5:a[3,4:2]

p4 <id:4>

4:d[2]



Example

1:c

F1: c\b\\.

DAG nodes

Output:

2:c[1:2]

F2: c\c\b\\.

p1 <id:1,F1:0,F2:0>

1:c

3:b[1:2,2]

p2 <id:2,F1:0,F2:0,F2:1>

2:c[1:2]

4:d[2]

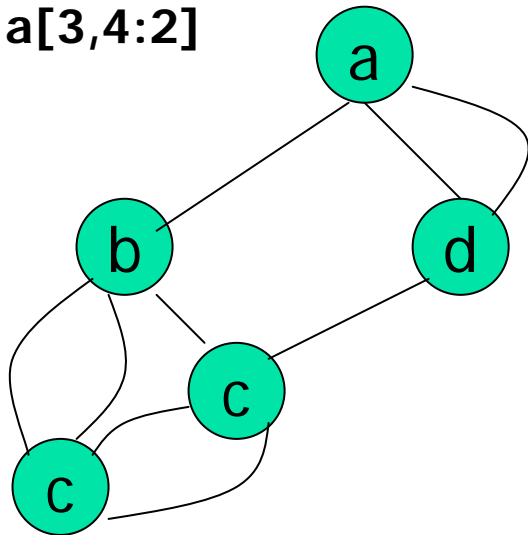
p3 <id:3,F1:1,F2:2>

3:b[1:2,2]

→ 5:a[3,4:2]

p4 <id:4>

4:d[2]



Full matched:

F1: (from its children id:3)

F2: (from its children id:3)

=> Output attributes

Example

1:c

F1: c\b\\.

2:c[1:2]

F2: c\c\b\\.

3:b[1:2,2]

4:d[2]

→ 5:a[3,4:2]

DAG nodes

p1 <id:1,F1:0,F2:0>

p2 <id:2,F1:0,F2:0,F2:1>

p3 <id:3,F1:1,F2:2>

p4 <id:4>

p5 <id:5,F1:1>

Output:

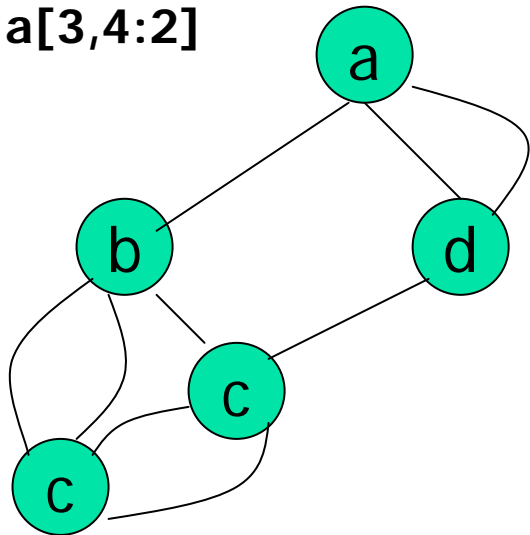
1:c

2:c[1:2]

3:b[1:2,2]

4:d[2]

5:a <e1=1> [3,4:2]



For its parent

Example

1:c

F1: c\b\\.

DAG nodes

Output:

2:c[1:2]

F2: c\\c\b\\.

p1 <id:1,F1:0,F2:0>

1:c

3:b[1:2,2]

p2 <id:2,F1:0,F2:0,F2:1>

2:c[1:2]

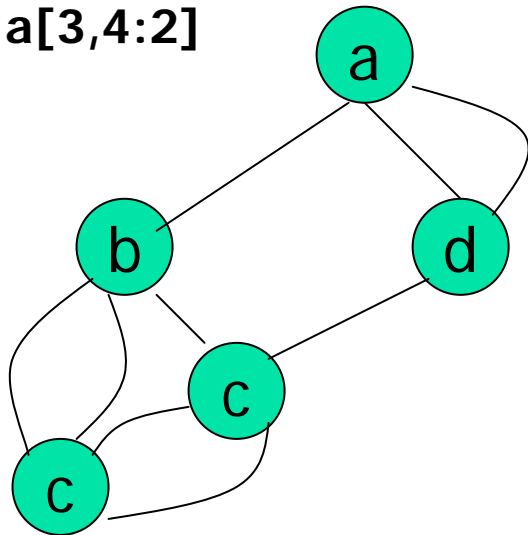
→ 4:d[2]

p3 <id:3,F1:1,F2:2>

3:b[1:2,2]

5:a[3,4:2]

p4 <id:4>



Not matched: =>

F1: gone

F2: 0 (from its children id:2)

Example

1:c

F1: c\b\\.

DAG nodes

Output:

2:c[1:2]

F2: c\\c\b\\.

p1 <id:1,F1:0,F2:0>

1:c

3:b[1:2,2]

p2 <id:2,F1:0,F2:0,F2:1>

2:c[1:2]

→ 4:d[2]

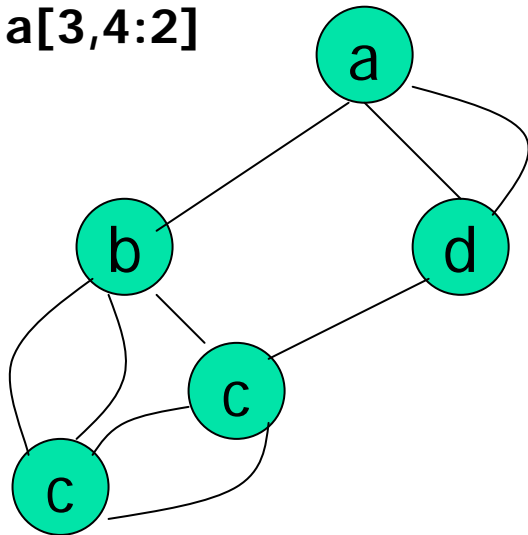
p3 <id:3,F1:1,F2:2>

3:b[1:2,2]

5:a[3,4:2]

p4 <id:4,F2:0>

4:d[2]



For its parent