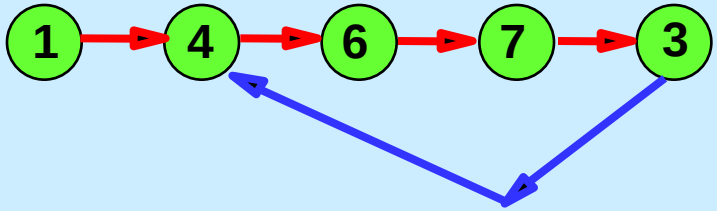
Network Optimization

Topological Ordering

Preliminary to Topological Sorting

LEMMA. If each node has at least one arc going out, then the first inadmissible arc of a depth first search determines a directed cycle.



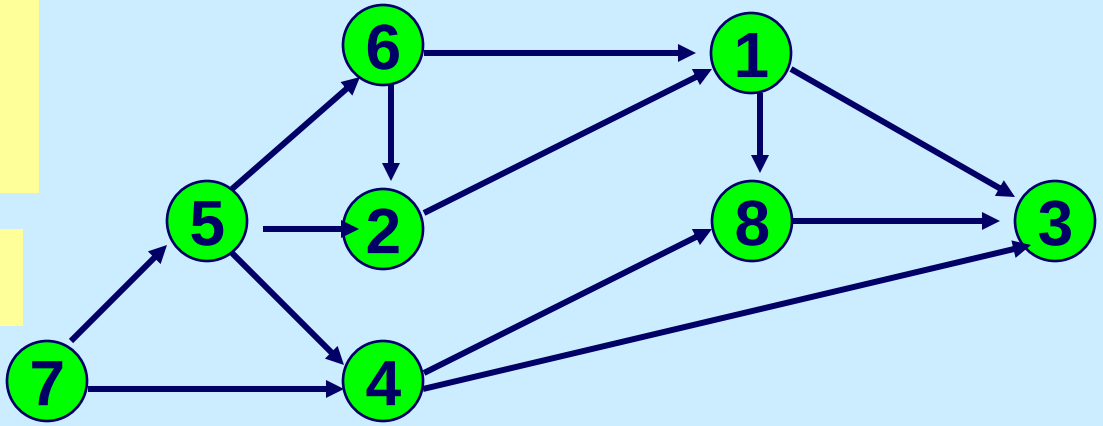
COROLLARY 1. If G has no directed cycle, then there is a node in G with no arcs going. And there is at least one node in G with no arcs coming in.

COROLLARY 2. If G has no directed cycle, then one can relabel the nodes so that for each arc (i,j), i < j.

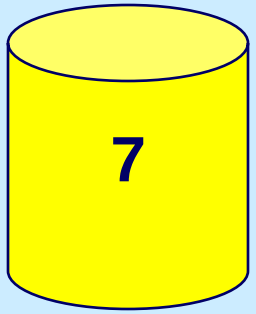
Initialization

Determine the indegree d_i of each node i.

LIST = {i : d_i = 0}



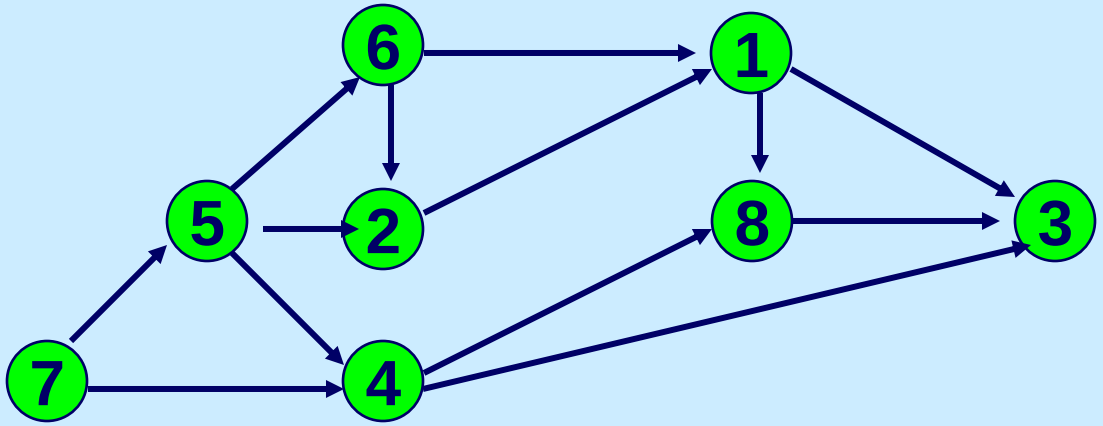
Node	1	2	3	4	5	6	7	8
Indegree	2	2	3	2	1	1	0	2



LIST

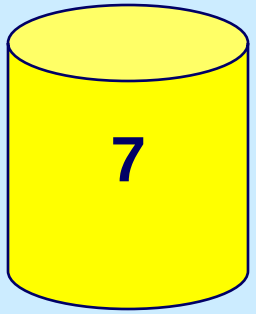
Initialization

"Next" will be the label of nodes in the topological order.



next 1

Node	1	2	3	4	5	6	7	8
Indegree	2	2	3	2	1	1	0	2



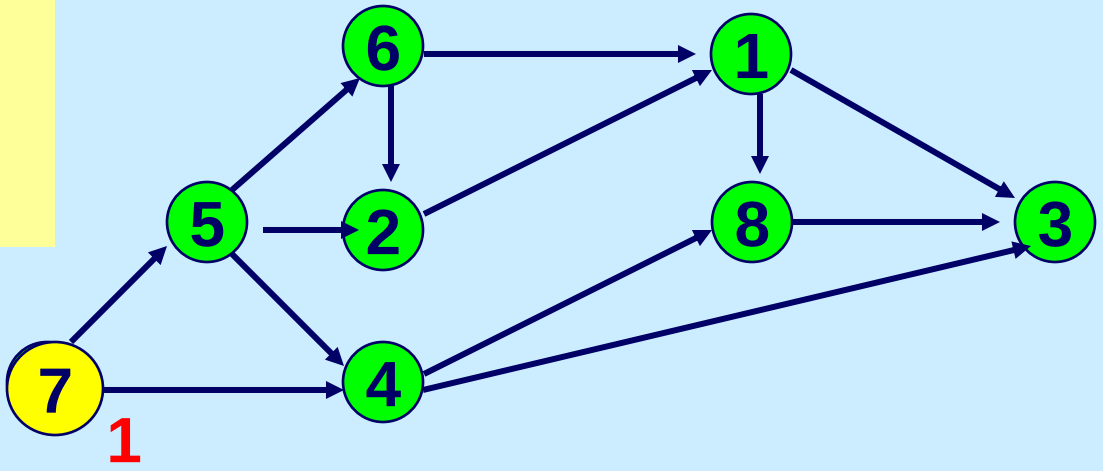
LIST

Select a node from LIST

Select Node 7.

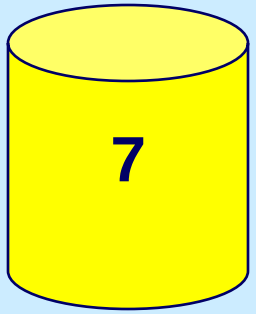
Order(7) := 1

Delete node 7.



next 1

Node	1	2	3	4	5	6	7	8
Indegree	2	2	3	2	1	1	0	2



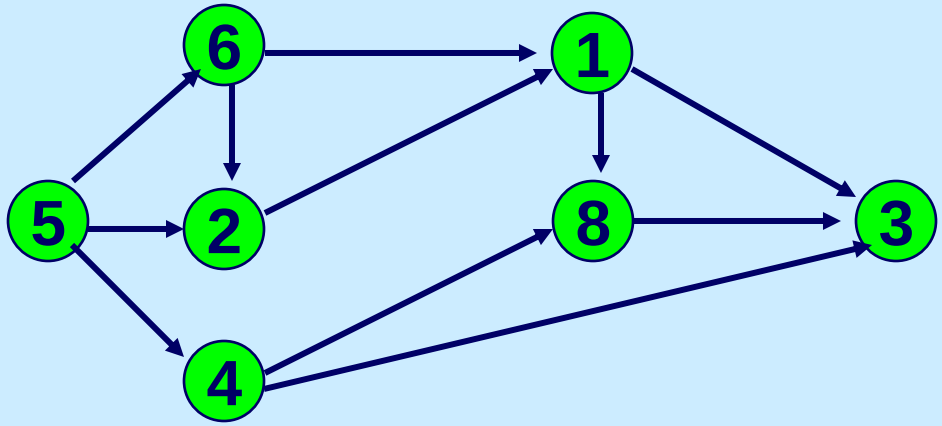
LIST

Updates

update “next”

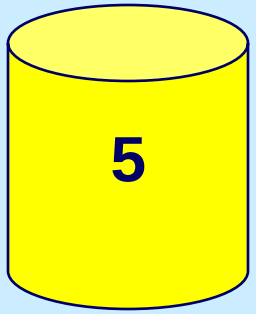
update indegrees

update LIST



next 2

Node	1	2	3	4	5	6	8
Indegree	2	2	3	1	0	1	2



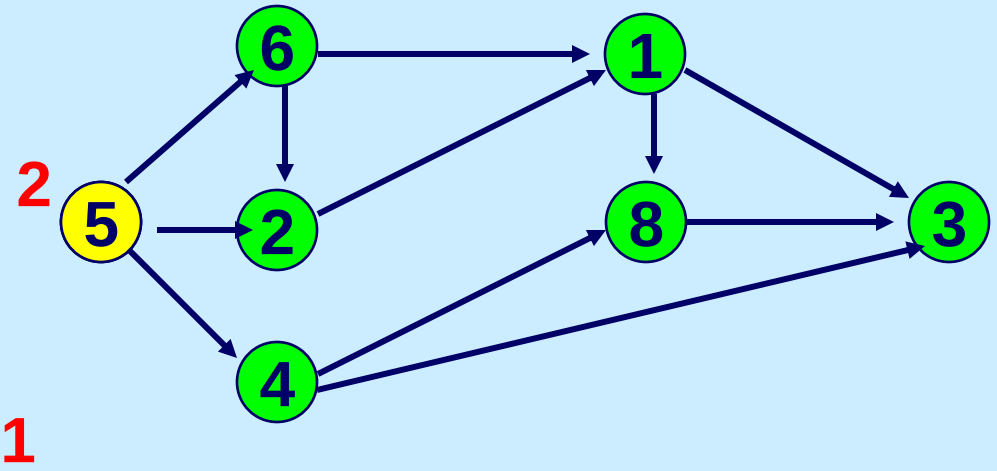
LIST

Select node 5

Select Node 5.

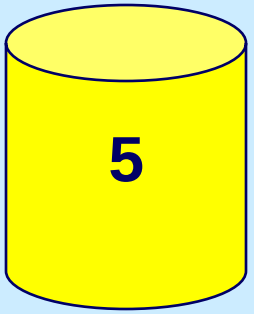
Order(5) := 2

Delete node 5.



next 2

Node	1	2	3	4	5	6	8
Indegree	2	2	3	1	0	1	2



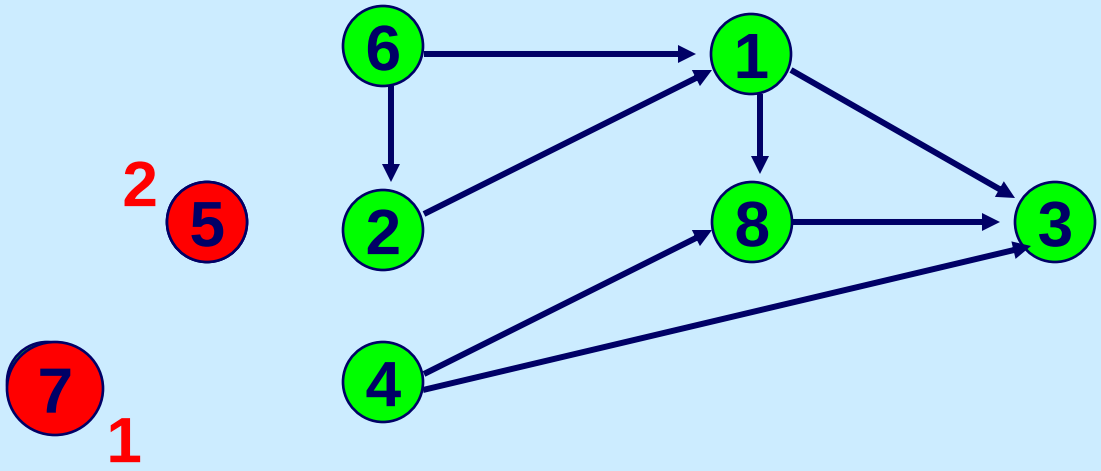
LIST

Updates

update “next”

update indegrees

update LIST

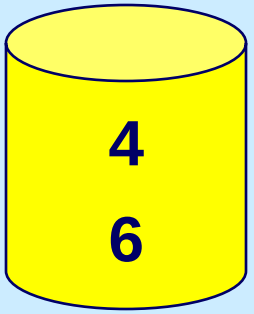


next 3

Node	1	2	3	4
Indegree	2	1	3	0

6
0

8
2



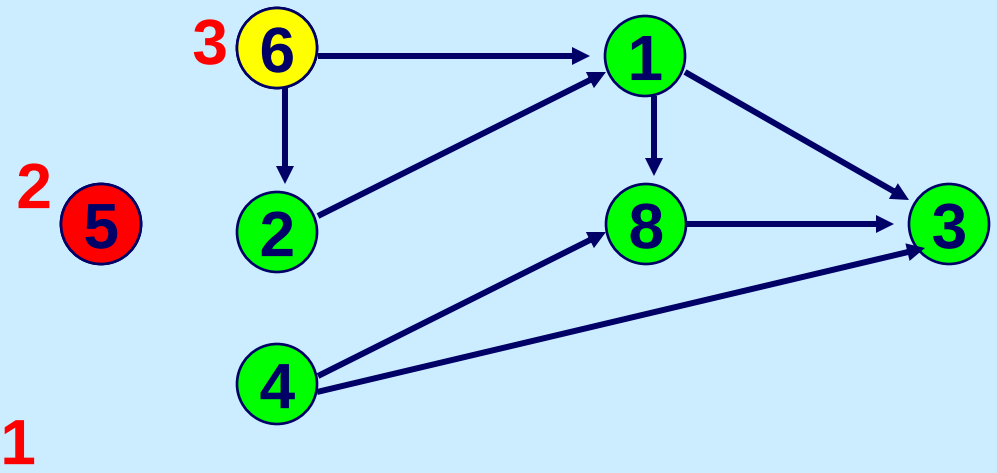
LIST

Select Node 6 (or 4)

Select Node 6.

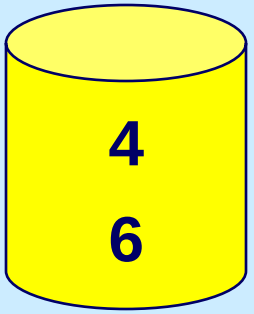
Order(6) := 3

Delete node 6.



next 3

Node	1	2	3	4	6	8
Indegree	2	1	3	0	0	2



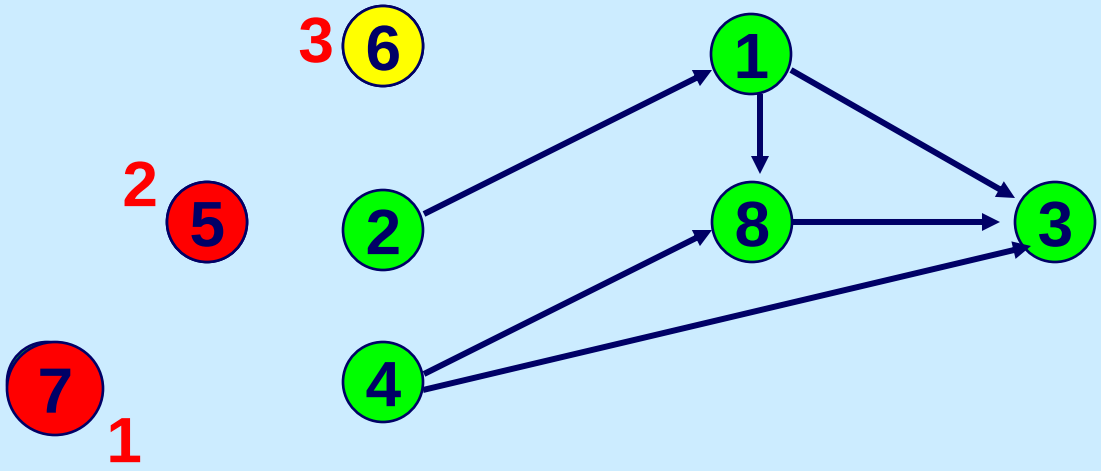
LIST

Updates

update “next”

update indegrees

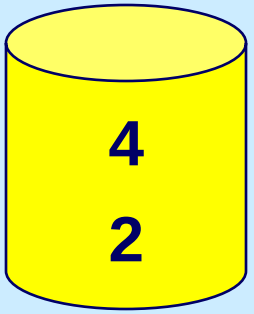
update LIST



next 4

Node	1	2	3	4
Indegree	1	0	3	0

8
2



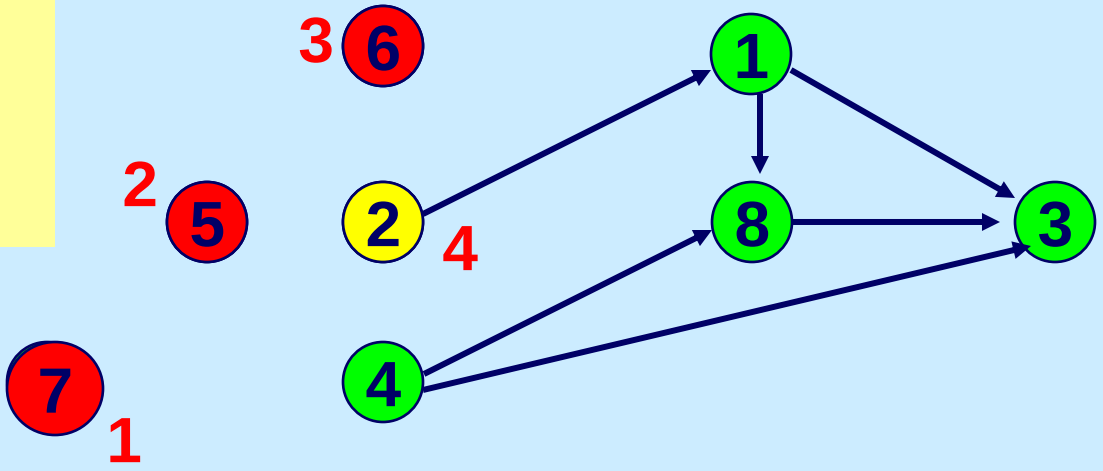
LIST

Select Node 2 (or 4)

Select Node 2.

Order(2) := 4

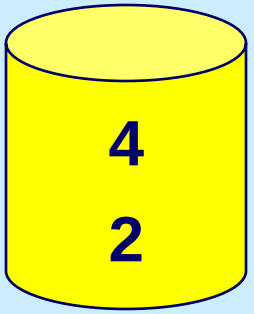
Delete node 2.



next 4

Node	1	2	3	4
Indegree	1	0	3	0

8
2



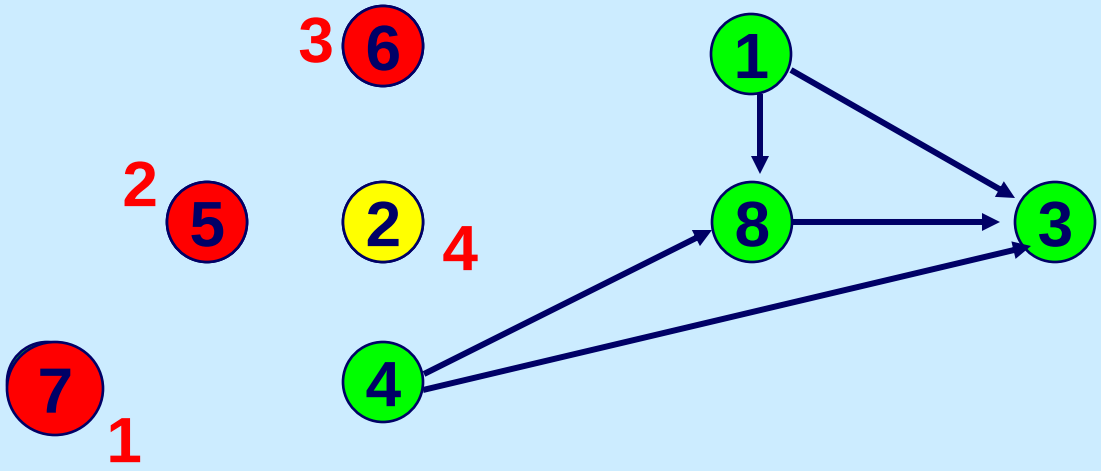
LIST

Updates

update “next”

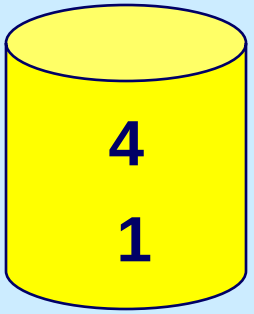
update indegrees

update LIST



next 5

Node	1	3	4	8
Indegree	0	3	0	2



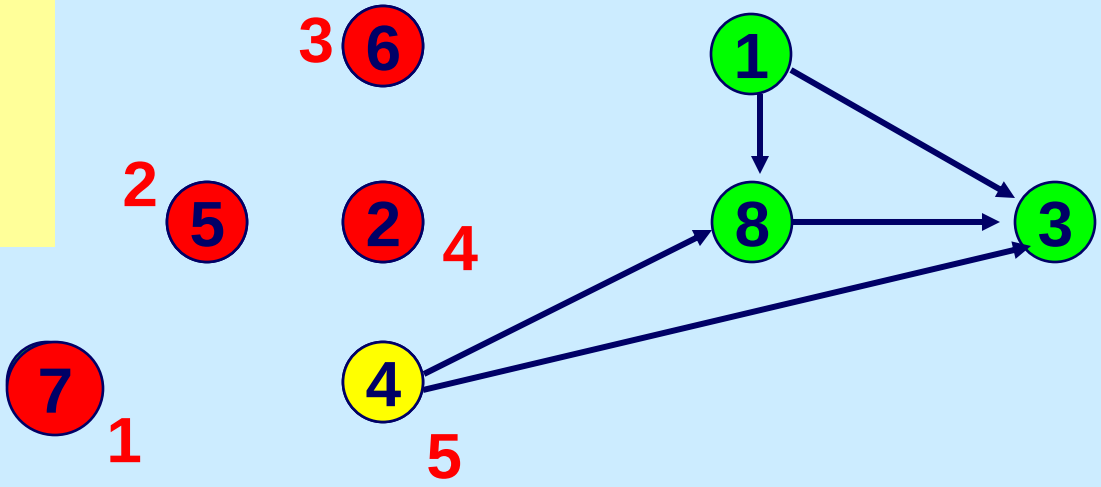
LIST

Select node 4 (or 1)

Select Node 4.

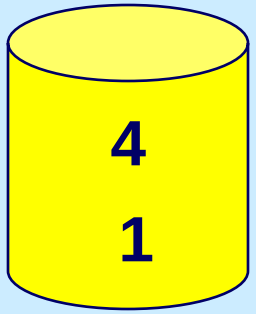
Order(4) := 5

Delete node 4.



next 5

Node	1	3	4	8
Indegree	0	3	0	2



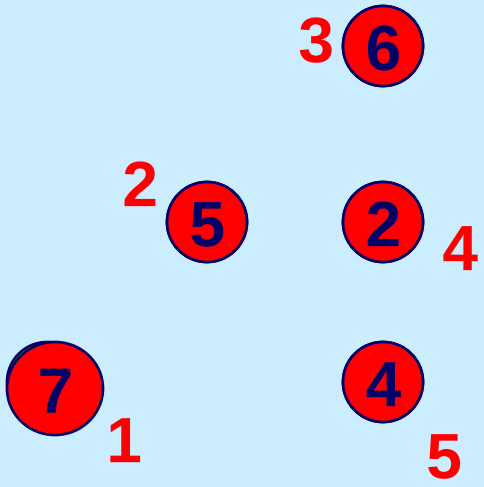
LIST

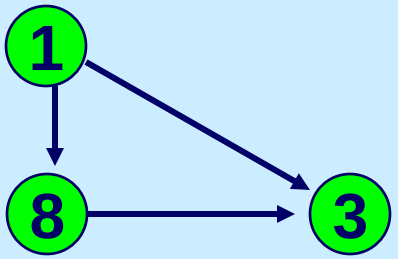
Updates

update “next”

update indegrees

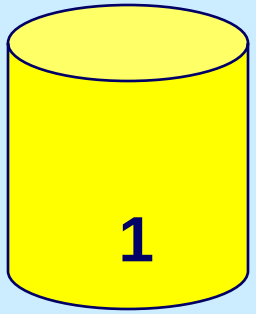
update LIST





next 6

Node	1	3	8
Indegree	0	2	1



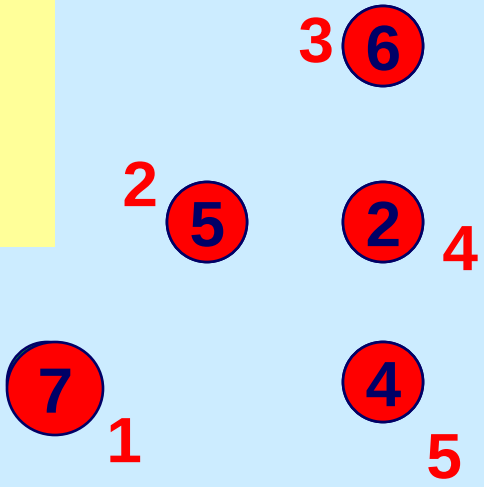
LIST

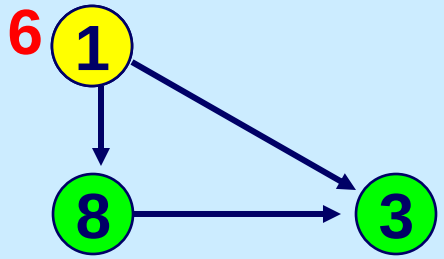
Select Node 1

Select Node 1.

Order(1) := 6

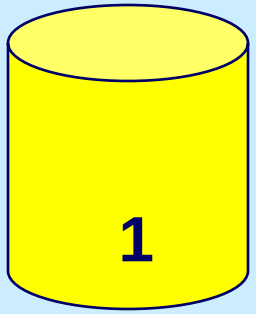
Delete node 1.





next | 6 |

Node	1	3	8
Indegree	0	2	1



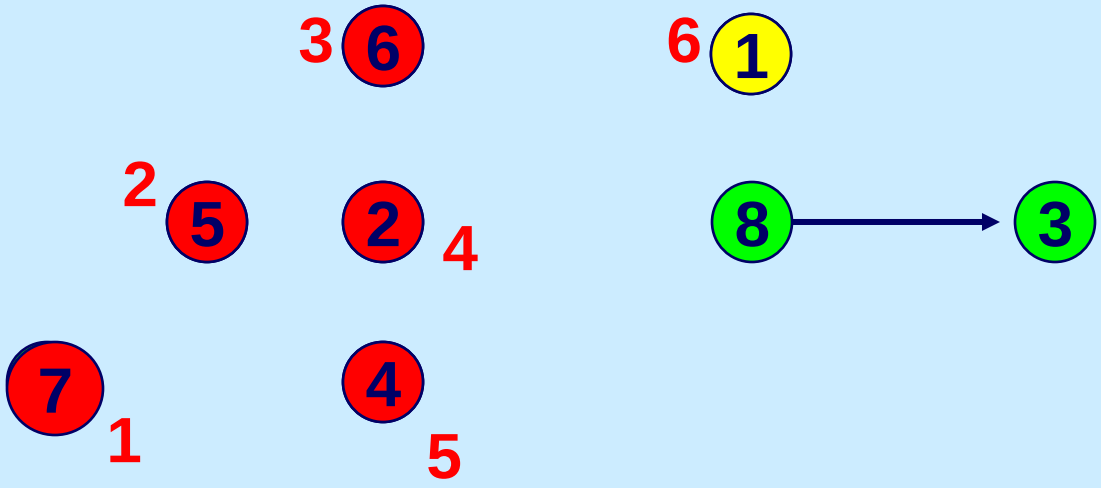
LIST

Updates

update “next”

update indegrees

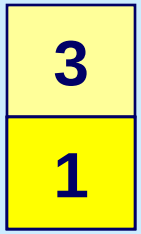
update LIST

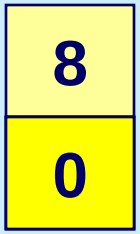


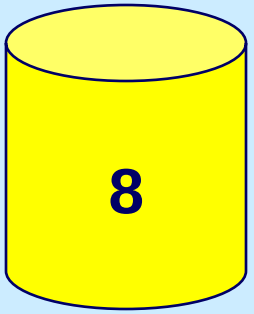
next 7

Node

Indegree







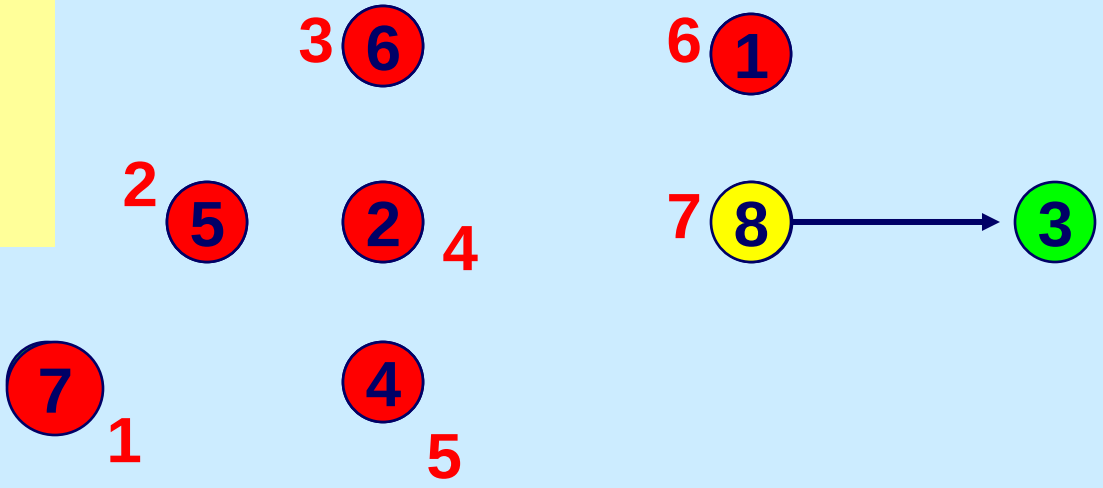
LIST

Select Node 8

Select Node 8.

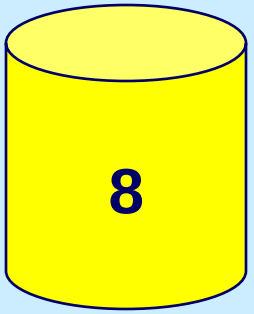
Order(8) := 7

Delete node 8.



next 7

Node	3	8
Indegree	1	0



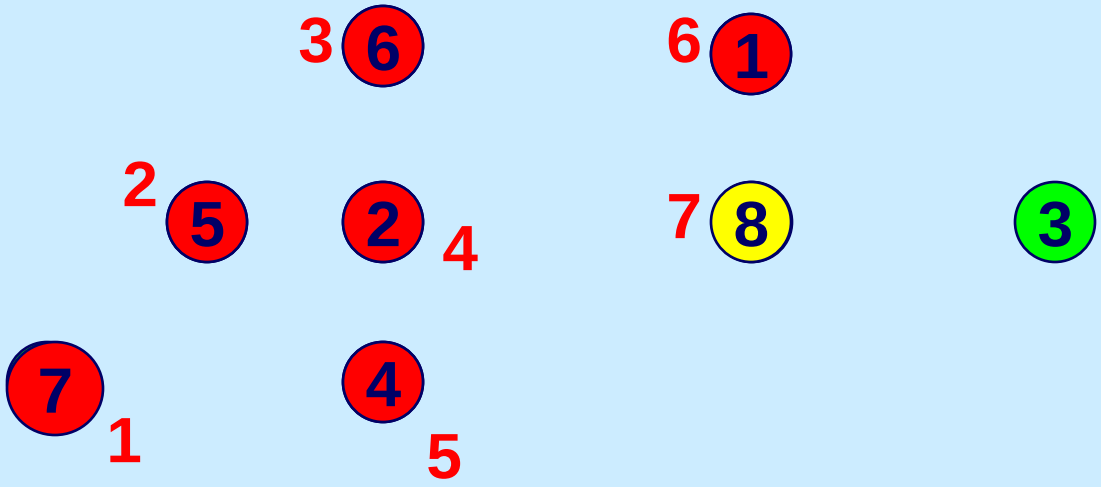
LIST

Updates

update “next”

update indegrees

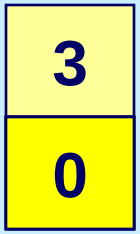
update LIST

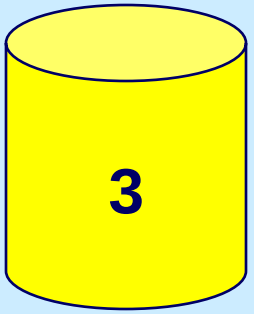


next 8

Node

Indegree





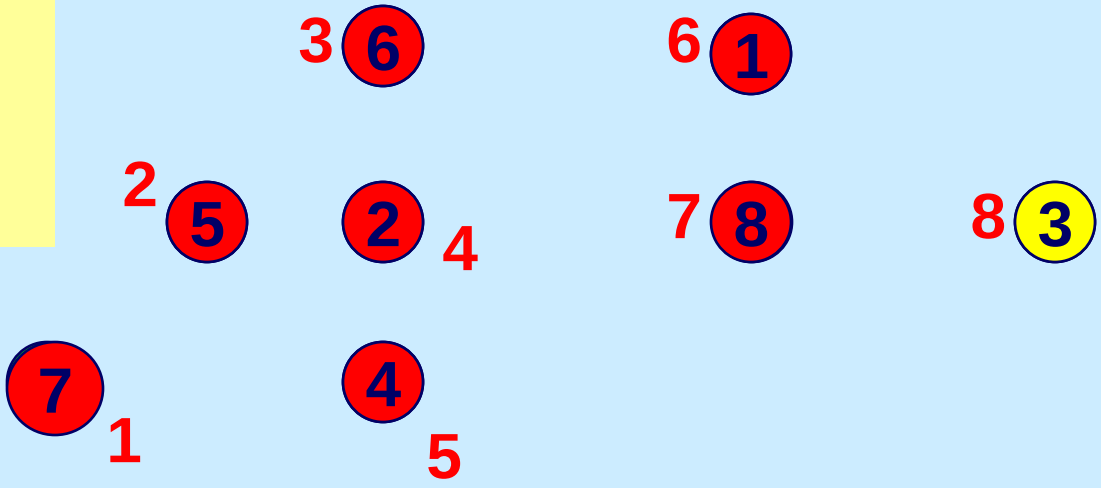
LIST

Select node 3

Select Node 3.

Order(3) := 8

Delete node 3.

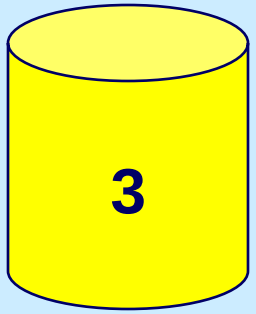


next 8

Node	3
Indegree	0

List is empty.

The algorithm terminates with a topological order of the nodes



LIST

15.082J / 6.855J / ESD.78J Network Optimization

Fall 2010