Problem Discussion

Topological Sorting (SPOJ) (C++)

+ Cycle finding demonstration

#csspree Online

Topological Sorting

Link: https://www.spoj.com/problems/TOPOSORT/

Topological Sorting

Link: https://www.spoj.com/problems/TOPOSORT/

What this problem asks for:

Topological Sorting

Link: https://www.spoj.com/problems/TOPOSORT/

What this problem asks for:

1. Topologically Sort given relations

Topological Sorting

Link: https://www.spoj.com/problems/TOPOSORT/

What this problem asks for:

1. Topologically Sort given relations
2. Indicate if sorting is not possible (graph has cycle)

Topological Sorting

Link: https://www.spoj.com/problems/TOPOSORT/

What this problem asks for:

1. Topologically Sort given relations
2. Indicate if sorting is not possible (graph has cycle)
3. If multiple sorting possible, print lexicographically smallest one

Cycle Finding

Cycle Finding

In directed graph, cycle exists, when successor of a node is also ancestor of it.

Cycle Finding

In directed graph, cycle exists, when successor of a node is also ancestor of it.

To detect cycle,

Cycle Finding

In directed graph, cycle exists, when successor of a node is also ancestor of it.

To detect cycle,

1. We will use 3 colors (markings) for nodes (as opposed to just 0/1 or visited/unvisited)

Cycle Finding

In directed graph, cycle exists, when successor of a node is also ancestor of it.

To detect cycle,

1. We will use 3 colors (markings) for nodes (as opposed to just 0/1 or visited/unvisited)
2. WHITE (0) will indicate the node is unvisited

Cycle Finding

In directed graph, cycle exists, when successor of a node is also ancestor of it.

To detect cycle,

1. We will use 3 colors (markings) for nodes (as opposed to just 0/1 or visited/unvisited)
2. WHITE (0) will indicate the node is unvisited
   GRAY (1) will indicate the node is visited, but all its children are not explored yet

Cycle Finding

In directed graph, cycle exists, when successor of a node is also ancestor of it.

To detect cycle,

1. We will use 3 colors (markings) for nodes (as opposed to just 0/1 or visited/unvisited)
2. WHITE (0) will indicate the node is unvisited
   GRAY (1) will indicate the node is visited, but all its children are not explored yet
   BLACK (2) will indicate this node and its successors all are visited.

Cycle Finding

In directed graph, cycle exists, when successor of a node is also ancestor of it.

To detect cycle,

1. We will use 3 colors (markings) for nodes (as opposed to just 0/1 or visited/unvisited)
2. WHITE (0) will indicate the node is unvisited
   GRAY (1) will indicate the node is visited, but all its children are not explored yet
   BLACK (2) will indicate this node and its successors all are visited.
3. If we see, a node adjacent to current node is GRAY, that means, current node is its successor and also ancestor!

Cycle Finding

In directed graph, cycle exists, when successor of a node is also ancestor of it.

To detect cycle,

1. We will use 3 colors (markings) for nodes (as opposed to just 0/1 or visited/unvisited)
2. WHITE (0) will indicate the node is unvisited
   GRAY (1) will indicate the node is visited, but all its children are not explored yet
   BLACK (2) will indicate this node and its successors all are visited.
3. If we see, a node adjacent to current node is GRAY, that means, current node is its successor and also ancestor!

   Since GRAY means right now it's children are being explored.

Cycle Finding

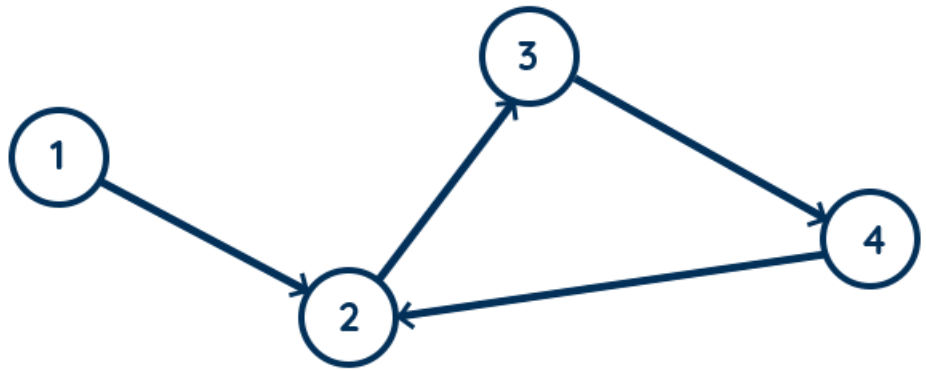
In directed graph, cycle exists, when successor of a node is also ancestor of it.

To detect cycle,

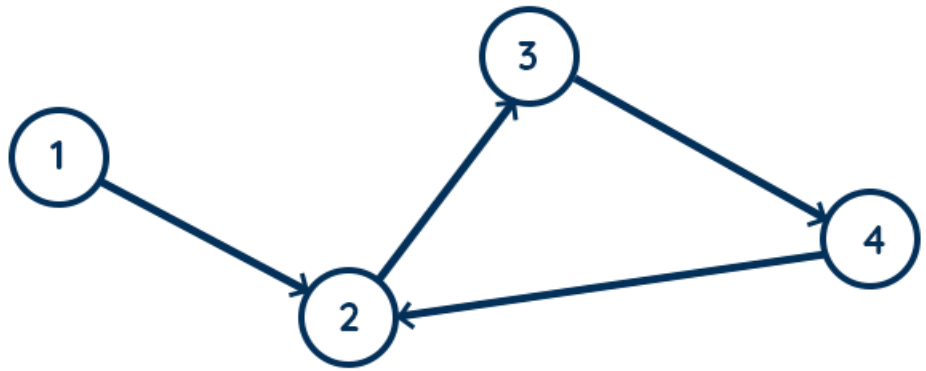
1. We will use 3 colors (markings) for nodes (as opposed to just 0/1 or visited/unvisited)
2. WHITE (0) will indicate the node is unvisited
   GRAY (1) will indicate the node is visited, but all its children are not explored yet
   BLACK (2) will indicate this node and its successors all are visited.
3. If we see, a node adjacent to current node is GRAY, that means, current node is its successor and also ancestor!

   Since GRAY means right now it's children are being explored. And an edge from child to ancestor means there is a cycle.

Cycle Finding



Cycle Finding

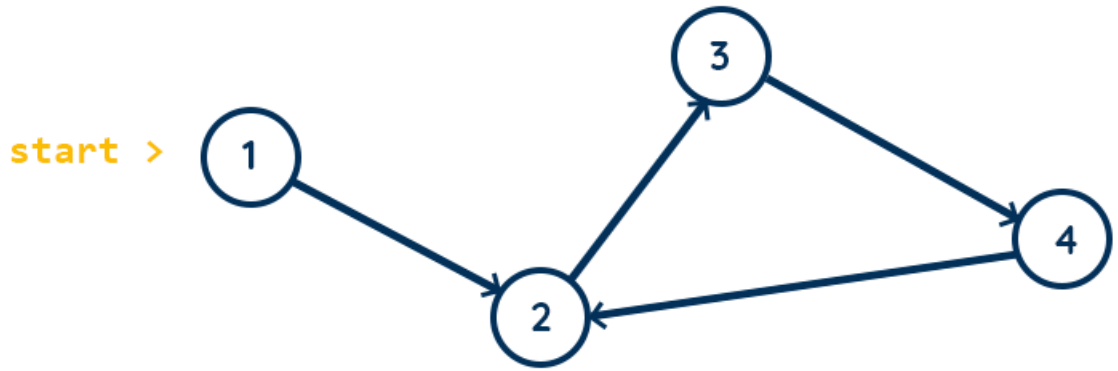


color[] =

0	0	0	0
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

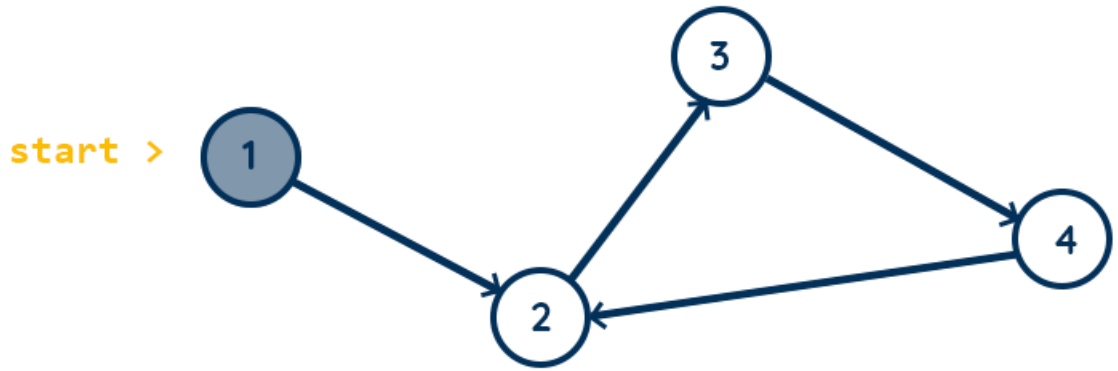


color[] =

0	0	0	0
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

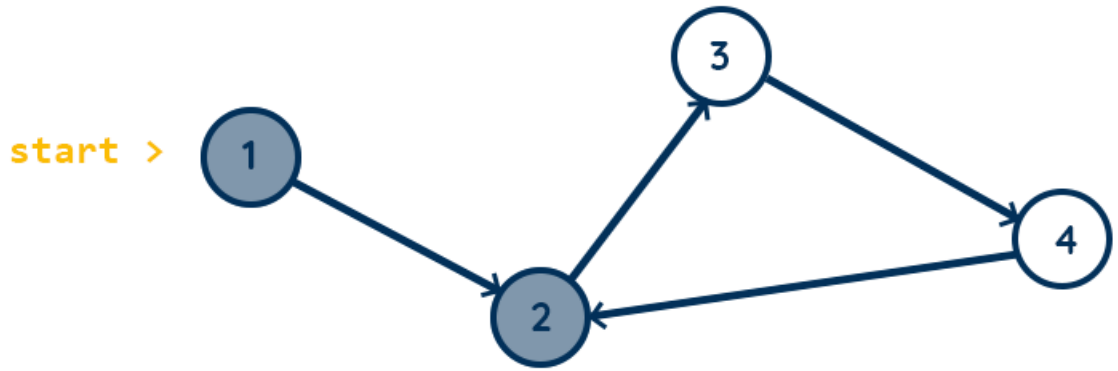


color[] =

1	0	0	0
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

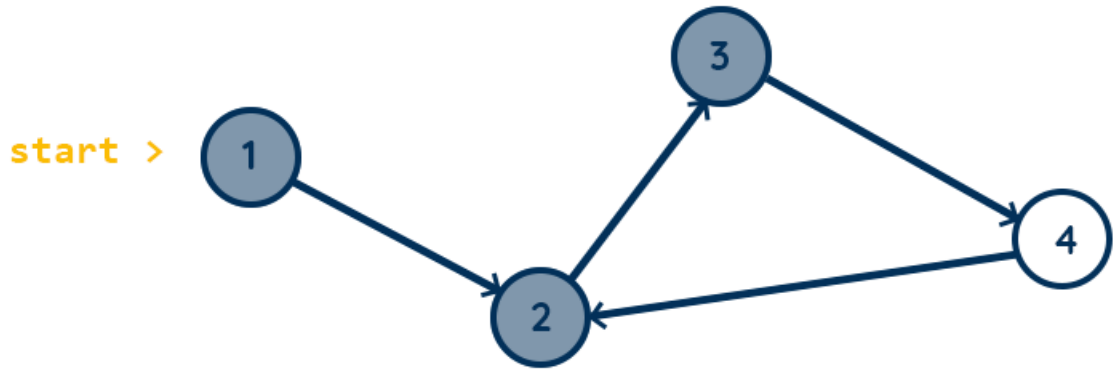


color[] =

1	1	0	0
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

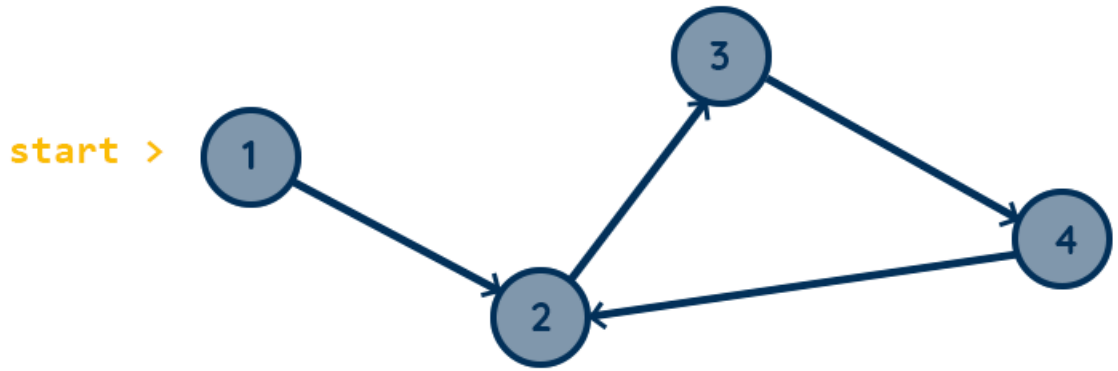


color[] =

1	1	1	0
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

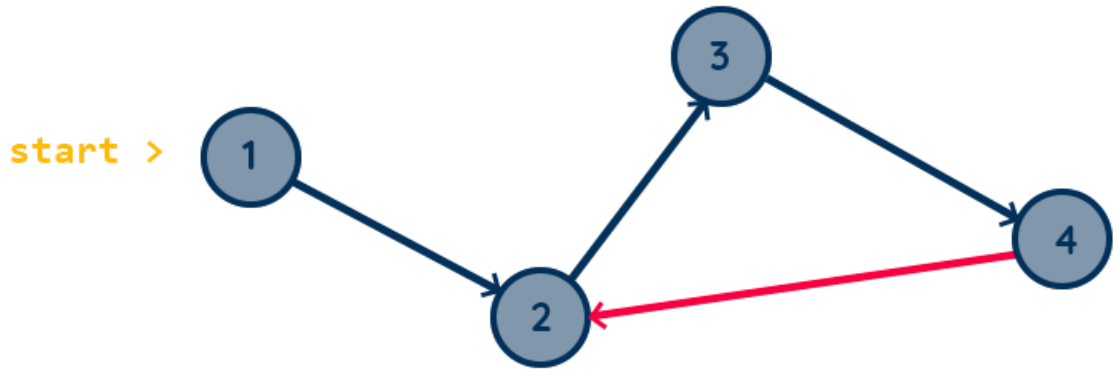


color[] =

1	1	1	1
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding



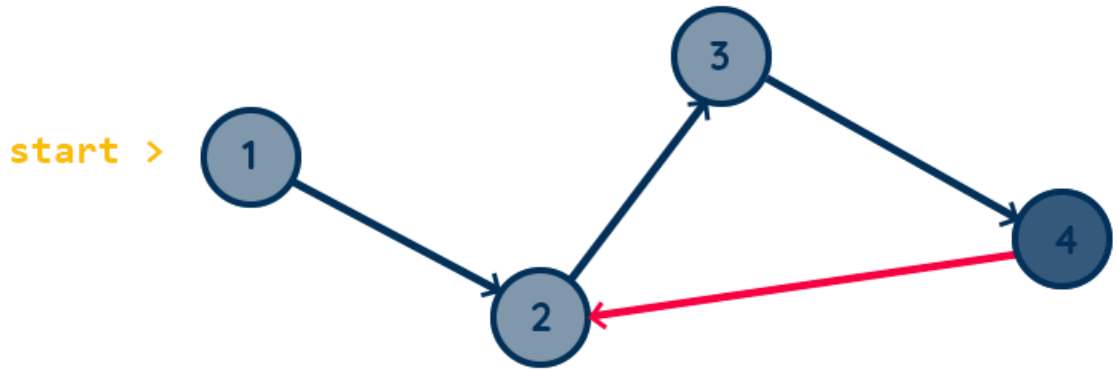
4 is 2's successor
2 is 4's successor!
Prerequisite conflict!!

color[] =

1	1	1	1
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

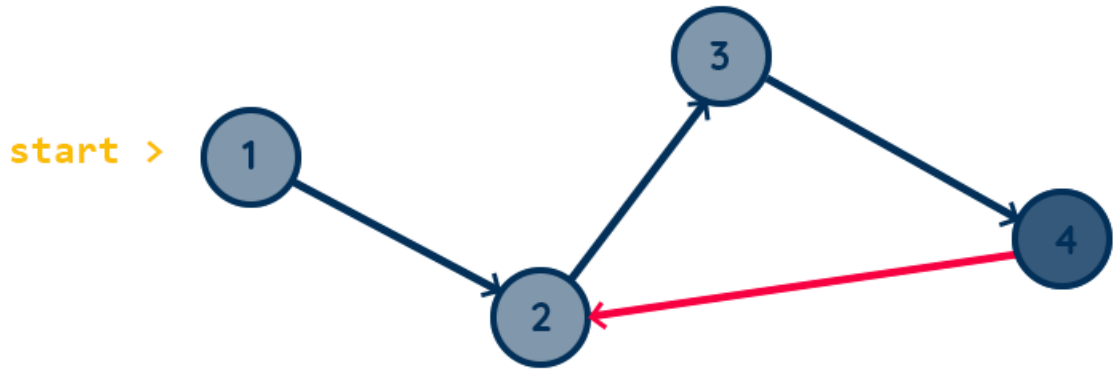


color[] =

1	1	1	2
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding



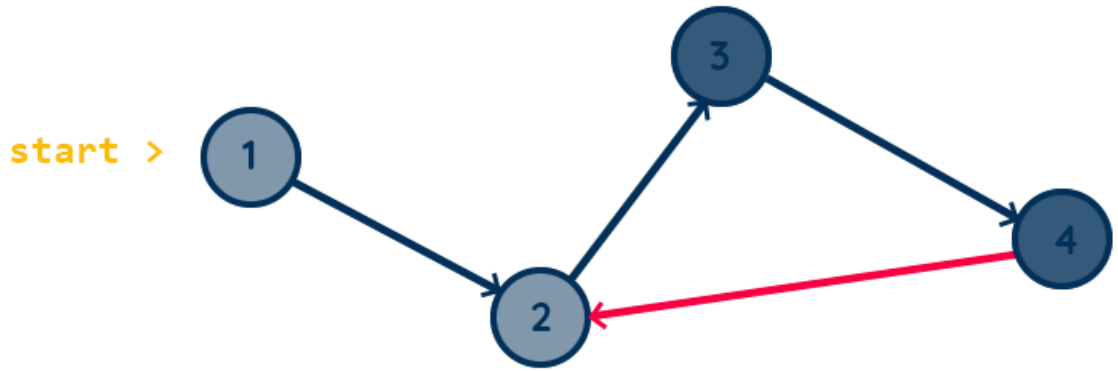
color[] =

1	1	1	2
1	2	3	4

2 means 4 and its children are all done

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

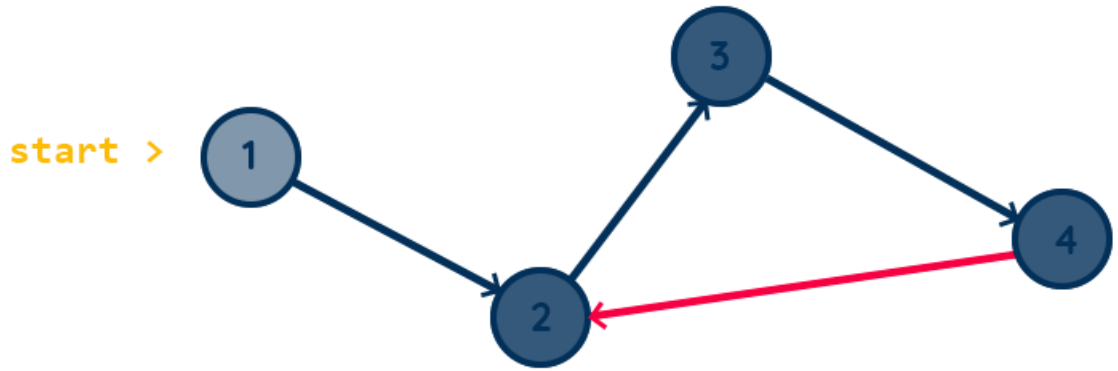


color[] =

1	1	2	2
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

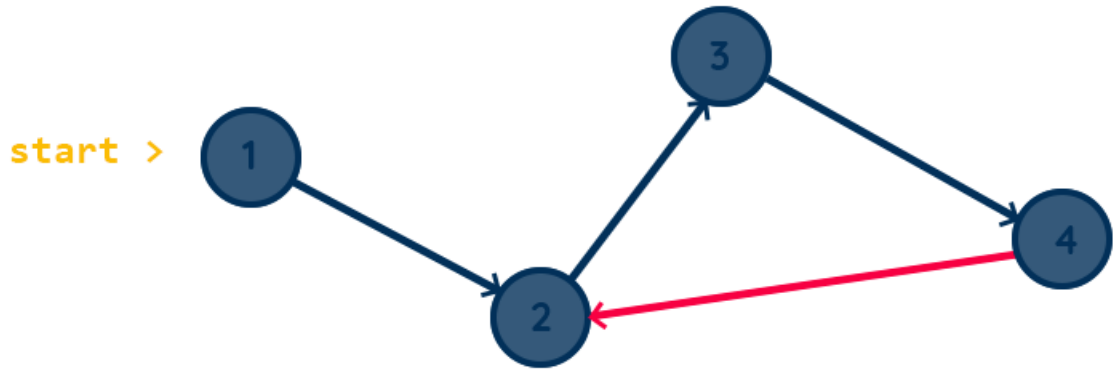


color[] =

1	2	3	4
1	2	2	2

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding

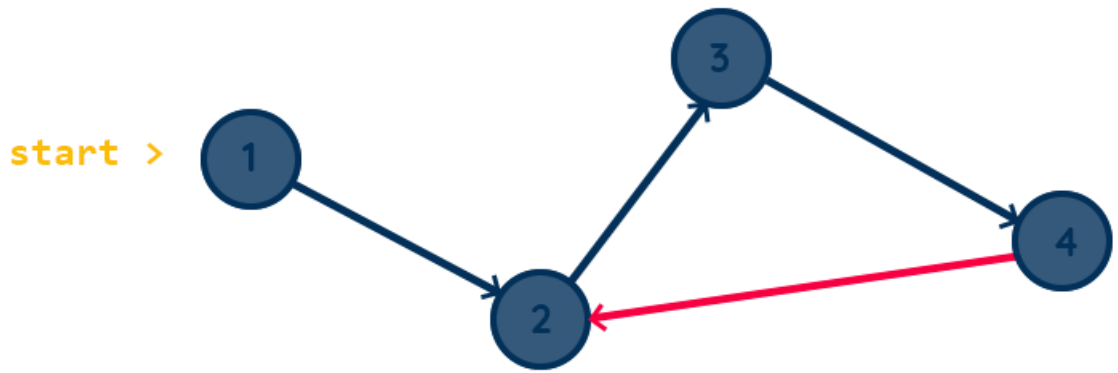


color[] =

2	2	2	2
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding



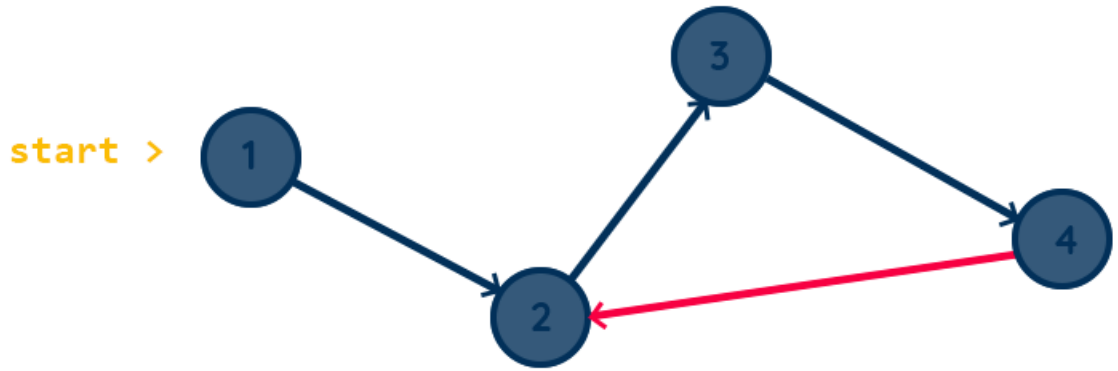
This graph has cycle!

color[] =

2	2	2	2
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK

Cycle Finding



This graph
has cycle!

Topological Sorting
is not possible

color[] =

2	2	2	2
1	2	3	4

0 = WHITE
1 = GRAY
2 = BLACK