Topological Sort
(C++)

#csspree Online

Topological Sort

- Ordering tasks (objects) based on requirement

Topological Sort

- Ordering tasks (objects) based on requirement
- Task and prerequisite relation is represented as directed graph

Topological Sort

- Ordering tasks (objects) based on requirement
- Task and prerequisite relation is represented as directed graph
- A directed edge from U to V means U has to be done before V

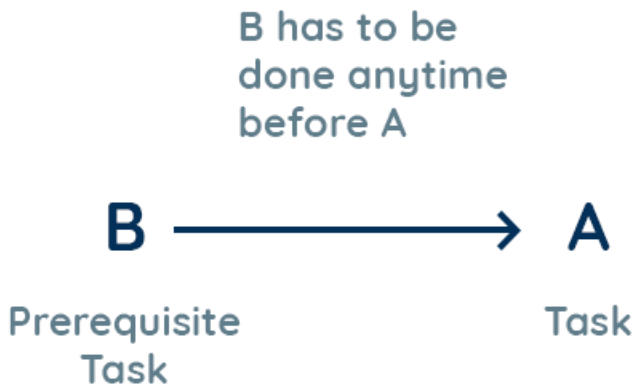
Topological Sort

- Ordering tasks (objects) based on requirement
- Task and prerequisite relation is represented as directed graph
- A directed edge from U to V means U has to be done before V
- The graph must be acyclic in order to have valid sorting, otherwise no solution is possible

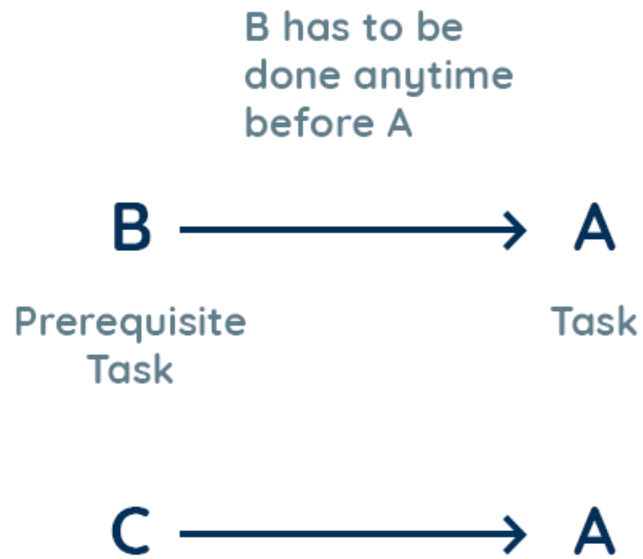
Graph Representation for Topological Sort

Graph Representation for Topological Sort

Graph Representation for Topological Sort



Graph Representation for Topological Sort



Graph Representation for Topological Sort

B has to be done anytime before A

Possible orders:

Graph Representation for Topological Sort

B has to be done anytime before A

Possible orders: A, B, C

Graph Representation for Topological Sort

B has to be done anytime before A

Possible orders: A, B, C ×

Graph Representation for Topological Sort

B has to be done anytime before A

Possible orders: A, B, C ×

B, A, C

Graph Representation for Topological Sort

B has to be done anytime before A

Prerequisite Task

Task

Possible orders: A, B, C ×

B, A, C ×

Graph Representation for Topological Sort

B has to be done anytime before A

Possible orders:
A, B, C ×
B, A, C ×
B, C, A

Graph Representation for Topological Sort

B has to be done anytime before A

Possible orders:
A, B, C ×
B, A, C ×
B, C, A ✓

Graph Representation for Topological Sort

B has to be done anytime before A

Prerequisite Task

Task

Possible orders:

A, B, C ✗

B, A, C ✗

B, C, A ✓

C, B, A

Graph Representation for Topological Sort

B has to be done anytime before A

Possible orders:

A, B, C ✗

B, A, C ✗

B, C, A ✓

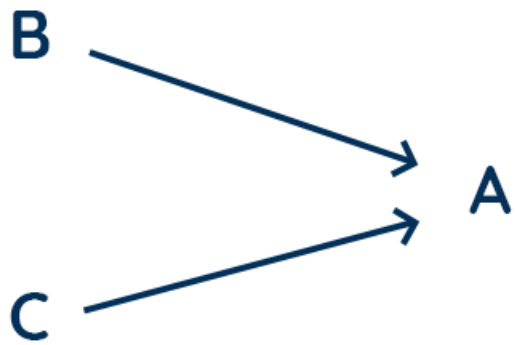
C, B, A ✓

Graph Representation for Topological Sort

B has to be done anytime before A

Prerequisite Task

Task



Possible orders:

A, B, C ✗

B, A, C ✗

B, C, A ✓

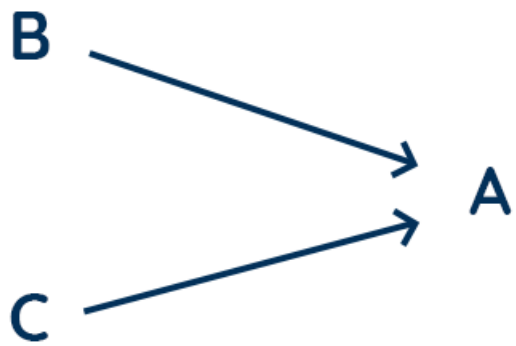
C, B, A ✓

Graph Representation for Topological Sort

B has to be done anytime before A

Prerequisite Task

Task



B and C have to be done anytime before A

Possible orders:

A, B, C ✗

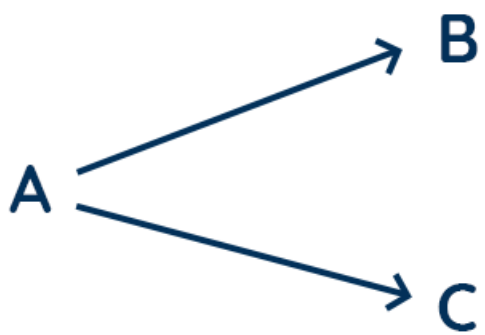
B, A, C ✗

B, C, A ✓

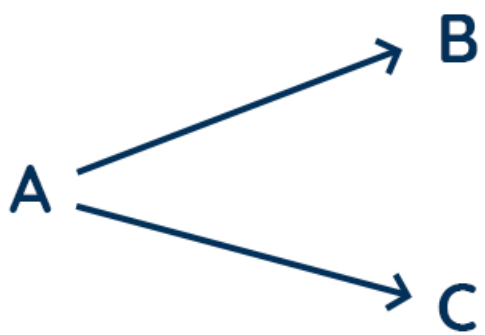
C, B, A ✓

Valid Graph for Topological Sort

Valid Graph for Topological Sort

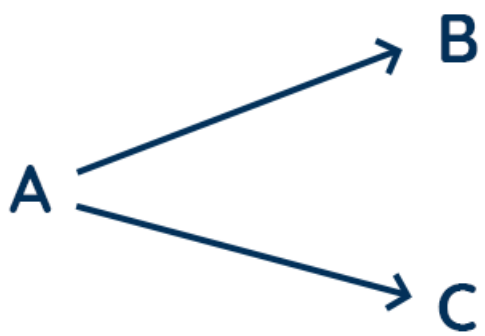


Valid Graph for Topological Sort



Directed Acyclic Graph (DAG)

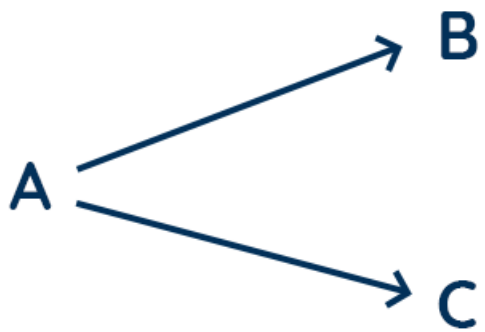
Valid Graph for Topological Sort



Directed Acyclic Graph (DAG)

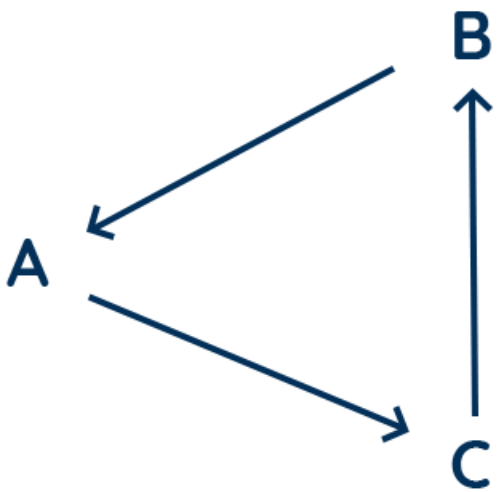
Valid ✓

Valid Graph for Topological Sort

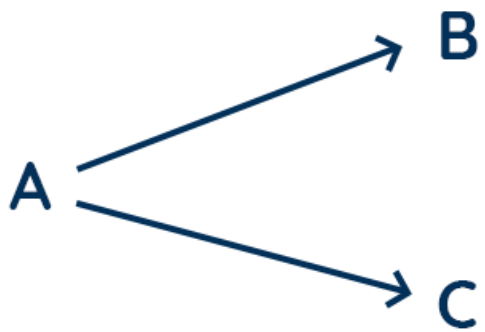


Directed Acyclic Graph (DAG)

Valid ✓

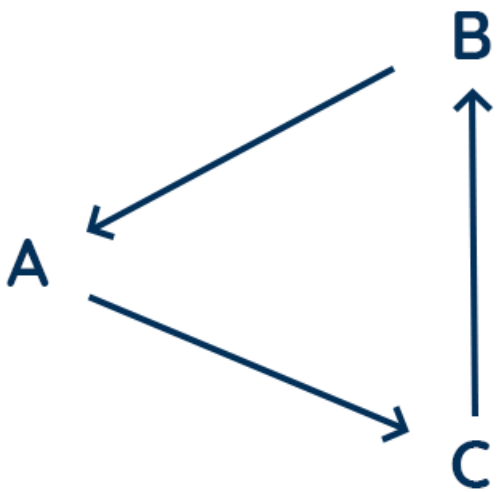


Valid Graph for Topological Sort



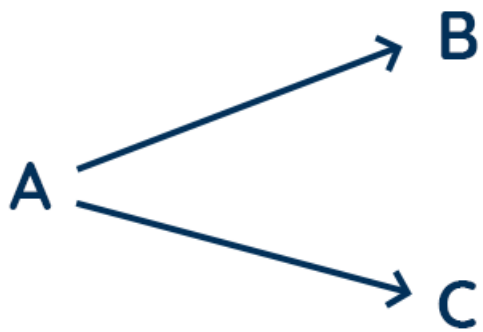
Directed Acyclic Graph (DAG)

Valid ✓



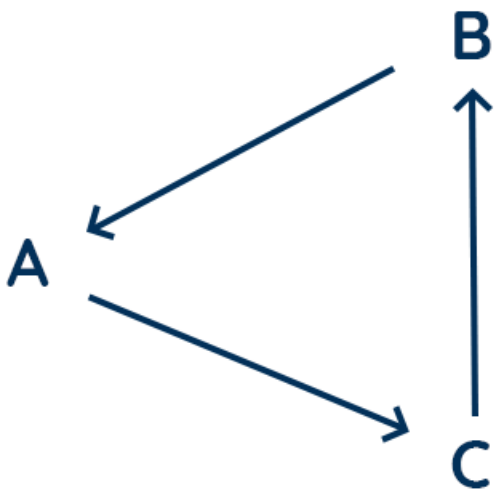
Cyclic Graph

Valid Graph for Topological Sort



Directed Acyclic Graph (DAG)

Valid ✓

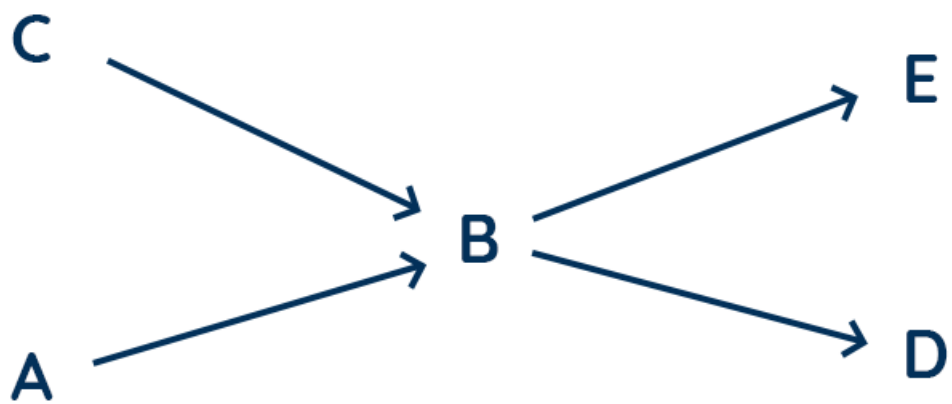


Cyclic Graph

Invalid ×

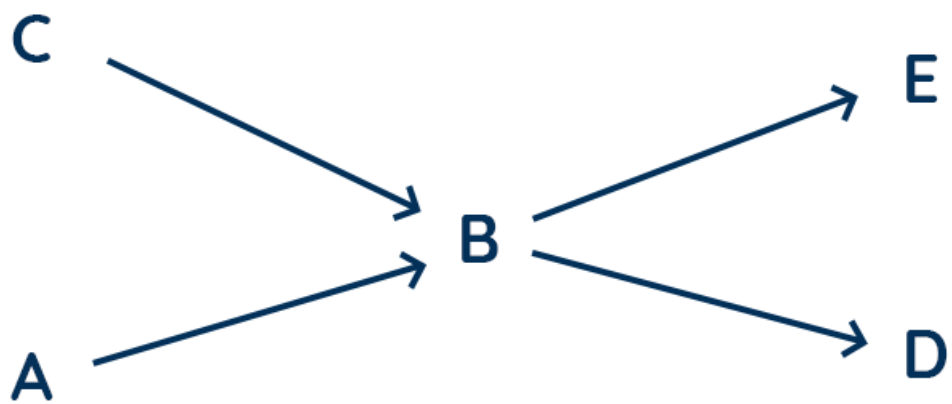
Implementation using DFS

Implementation using DFS



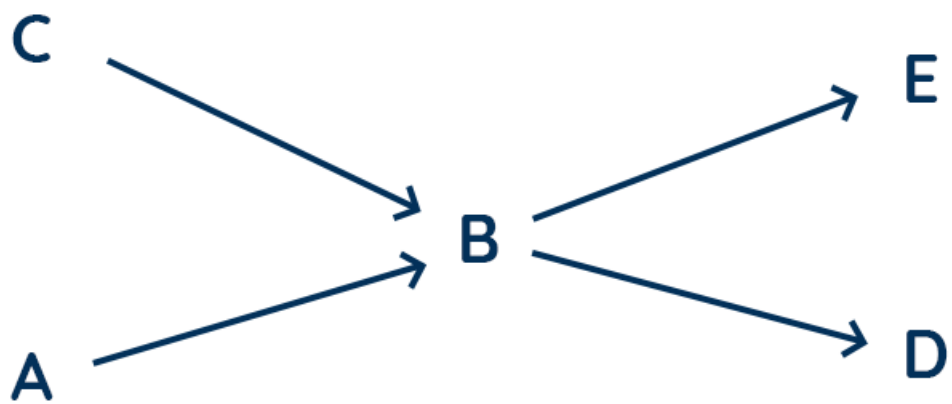
Implementation using DFS

- Explore the graph using Depth First Search (DFS)



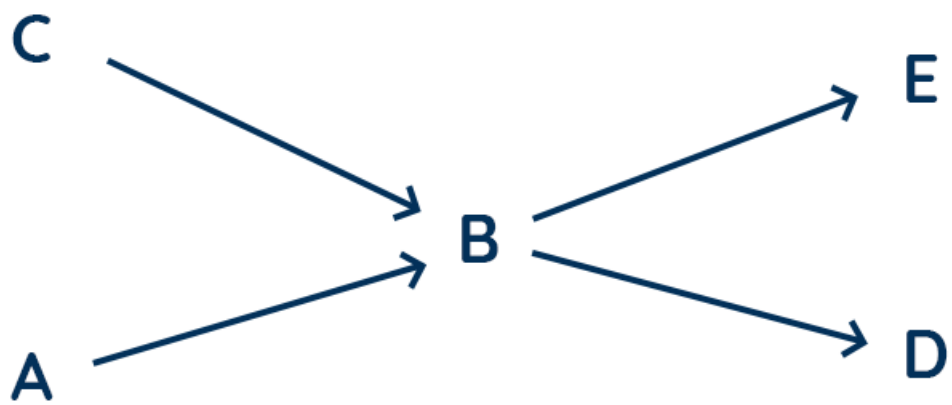
Implementation using DFS

- Explore the graph using Depth First Search (DFS)
- When a node has no unvisited child, push it into the result list



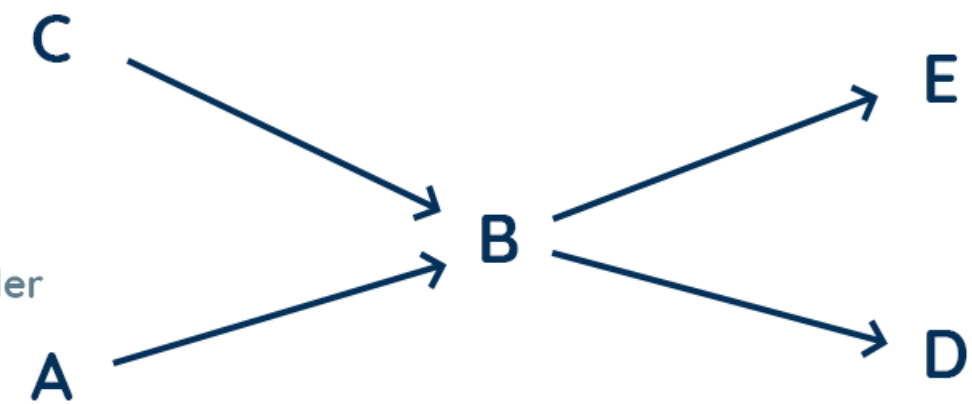
Implementation using DFS

- Explore the graph using Depth First Search (DFS)
- When a node has no unvisited child, push it into the result list
- After every node is visited by DFS, reverse the result list



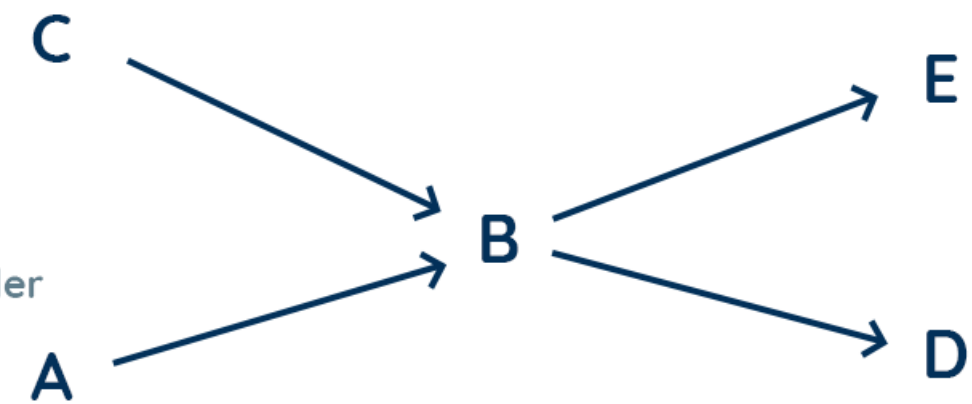
Implementation using DFS

- Explore the graph using Depth First Search (DFS)
- When a node has no unvisited child, push it into the result list
- After every node is visited by DFS, reverse the result list
- Now the order present in the list is a Topologically Sorted order



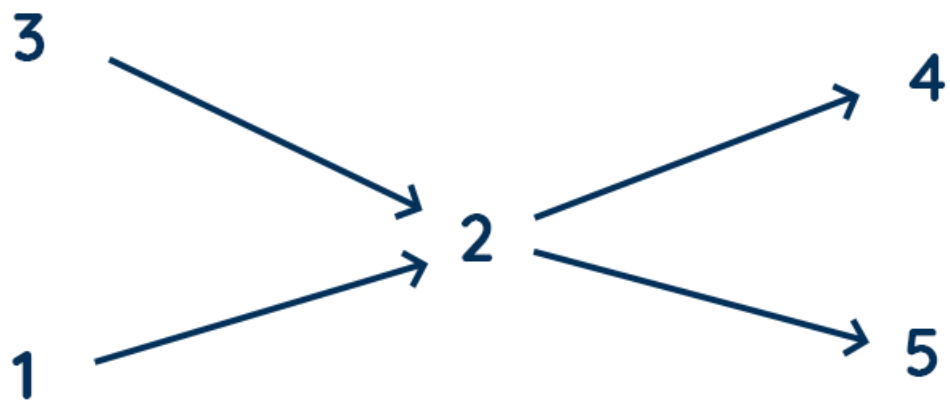
Implementation using DFS

- Explore the graph using Depth First Search (DFS)
- When a node has no unvisited child, push it into the result list
- After every node is visited by DFS, reverse the result list
- Now the order present in the list is a Topologically Sorted order
- There can be more than one correct ordering

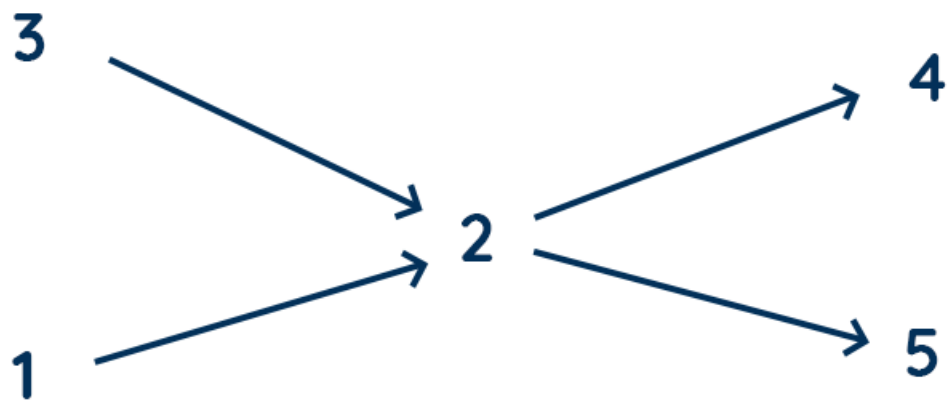


Simulation of DFS

Simulation of DFS

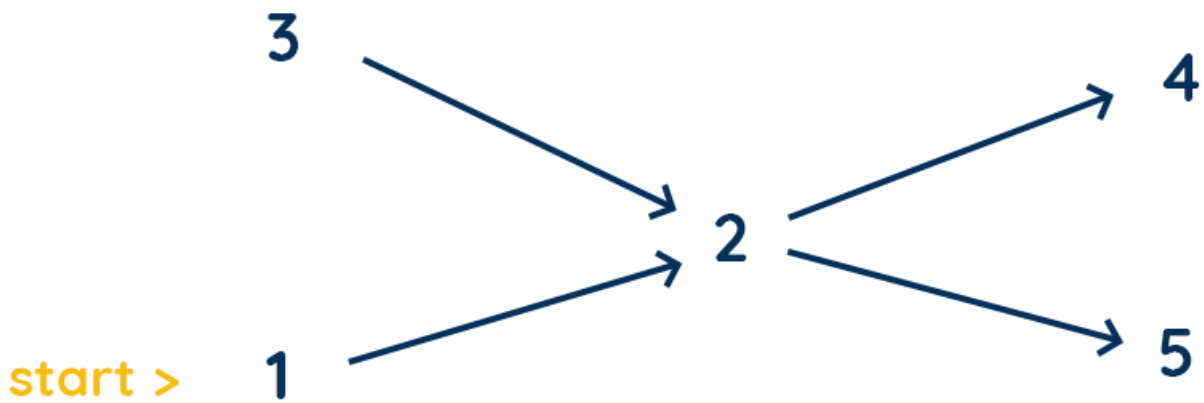


Simulation of DFS




```
vector < int > result;
```

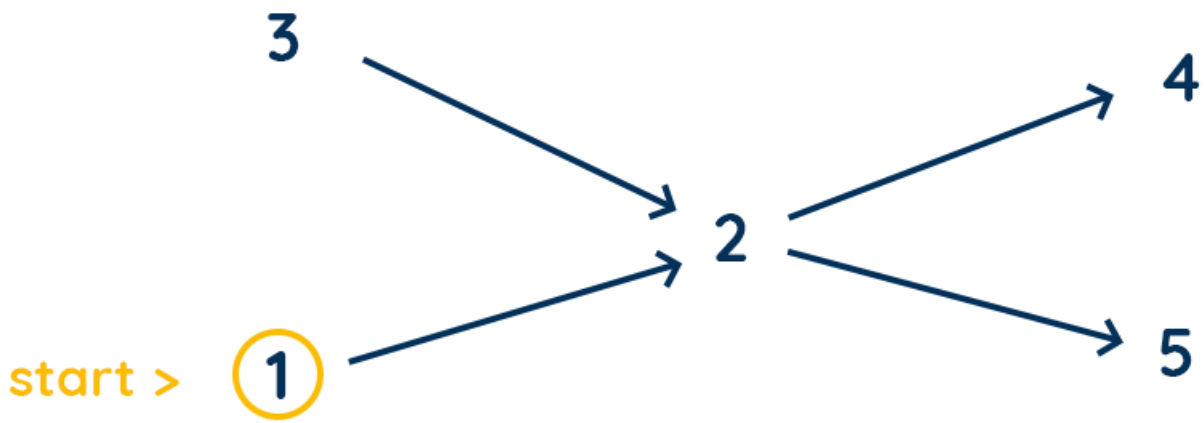
Simulation of DFS




```
vector < int > result;
```

result:

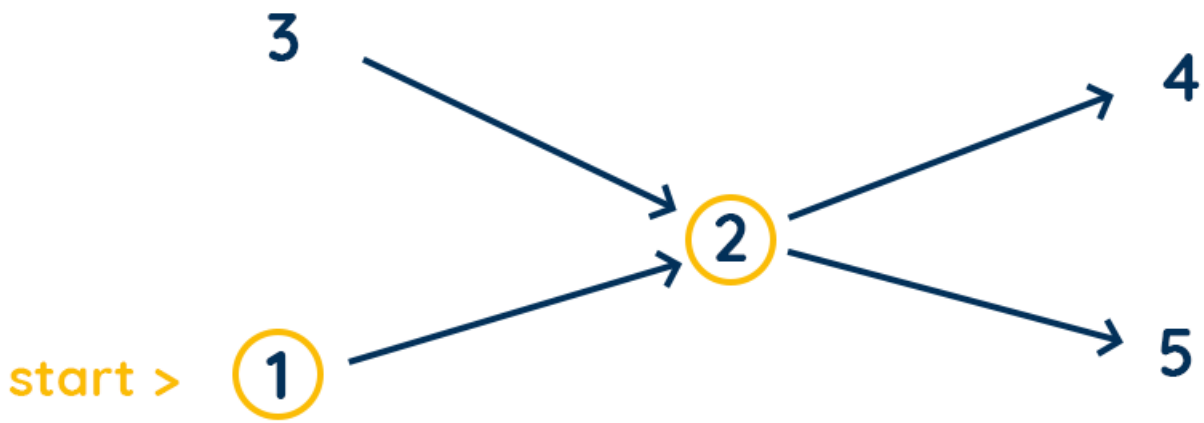
Simulation of DFS




```
vector < int > result;
```

result:

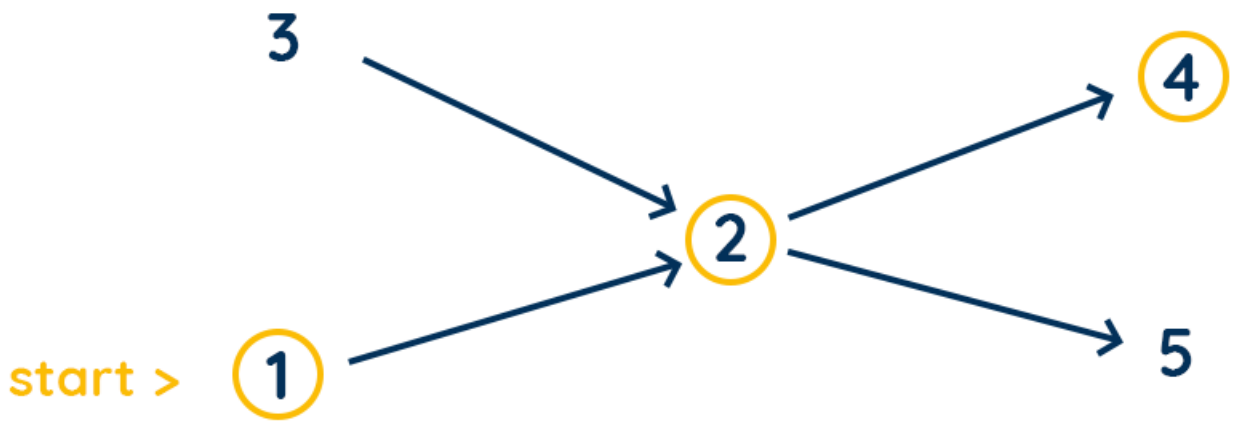
Simulation of DFS




```
vector < int > result;
```

result:

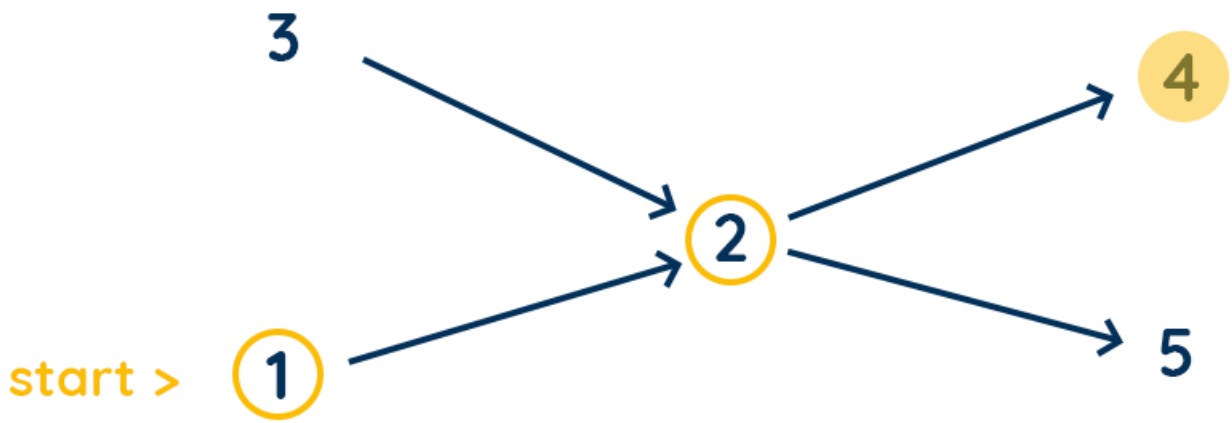
Simulation of DFS




```
vector < int > result;
```

result:

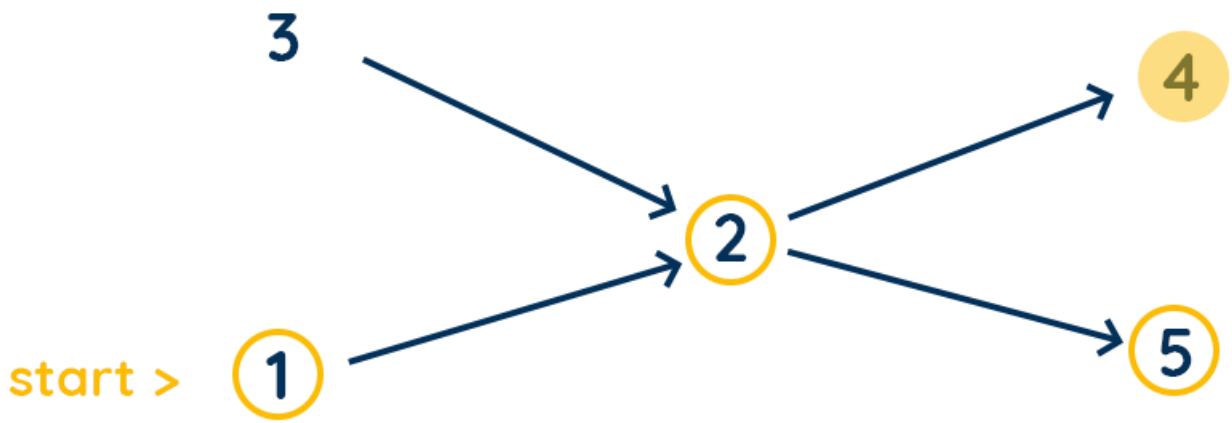
Simulation of DFS




```
vector < int > result;
```

result: 4

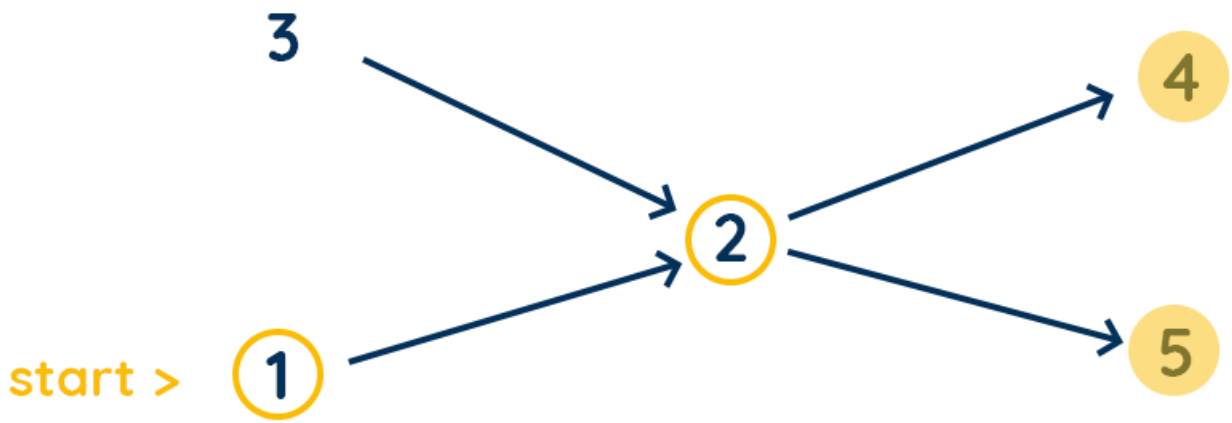
Simulation of DFS




```
vector < int > result;
```

result: 4

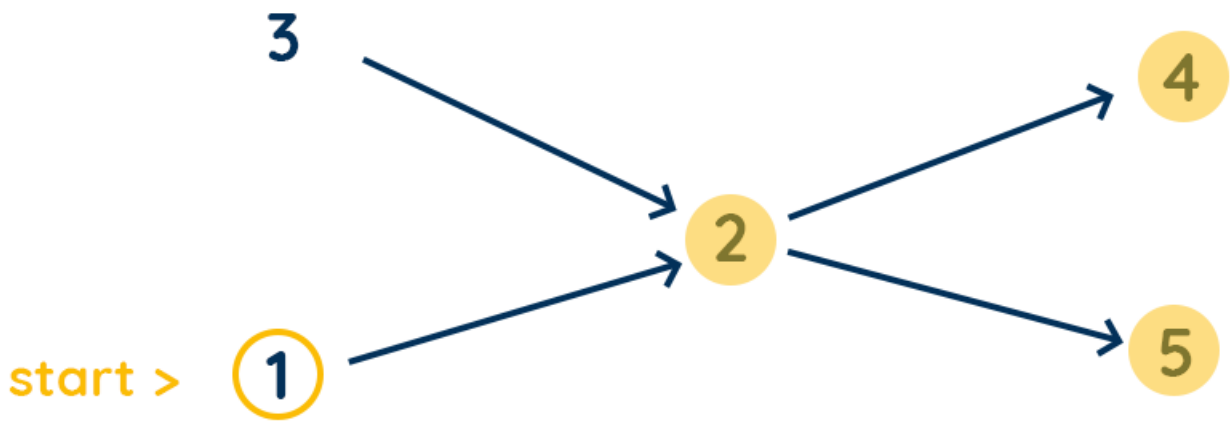
Simulation of DFS



vector < int > result;

result: 4, 5

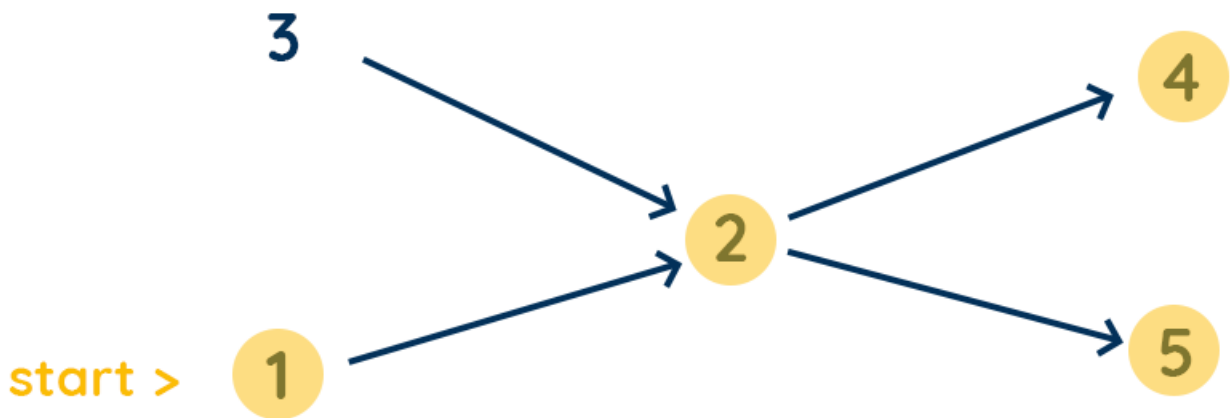
Simulation of DFS



vector < int > result;

result: 4, 5, 2

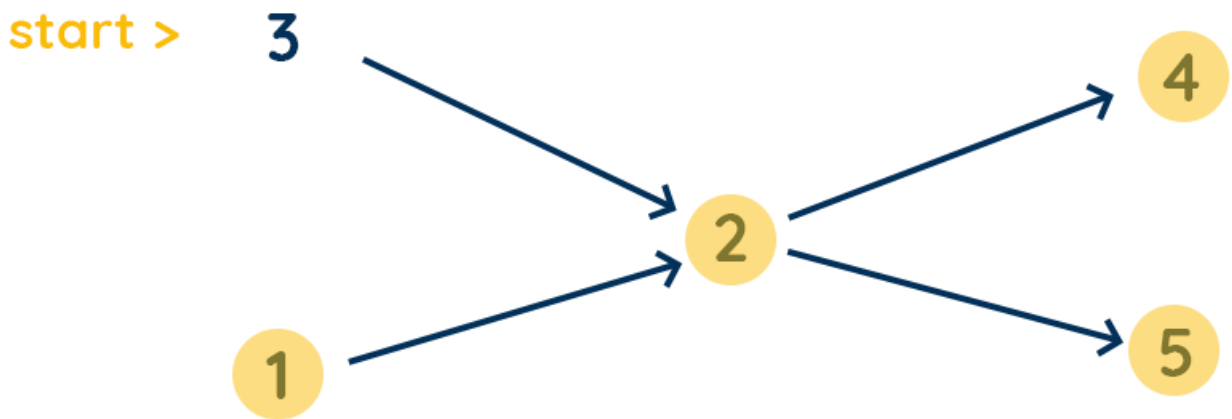
Simulation of DFS



vector < int > result;

result: 4, 5, 2, 1

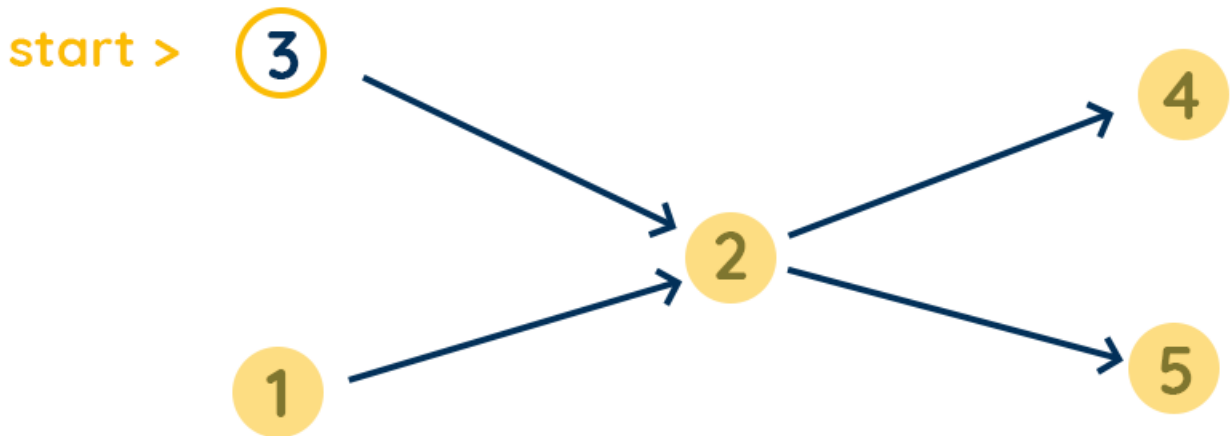
Simulation of DFS



vector < int > result;

result: 4, 5, 2, 1

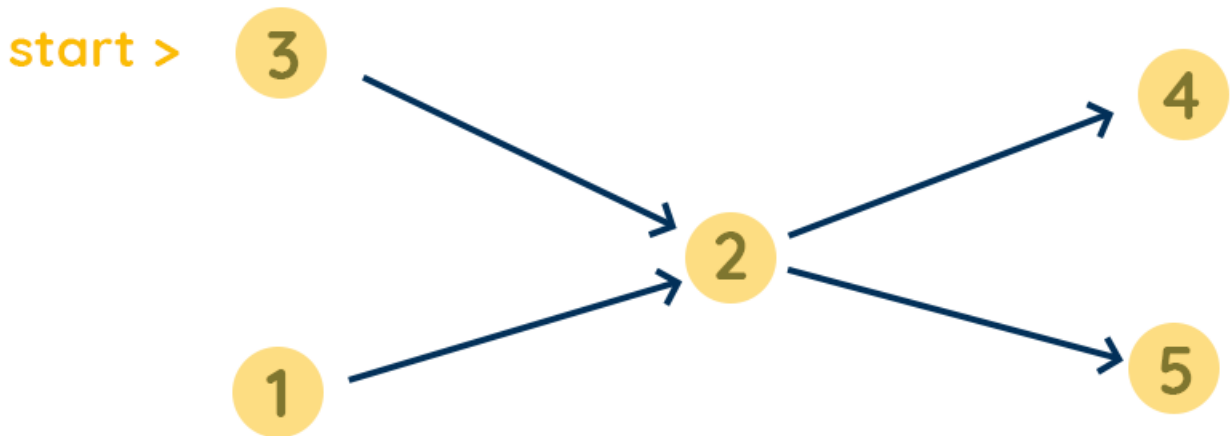
Simulation of DFS



vector < int > result;

result: 4, 5, 2, 1

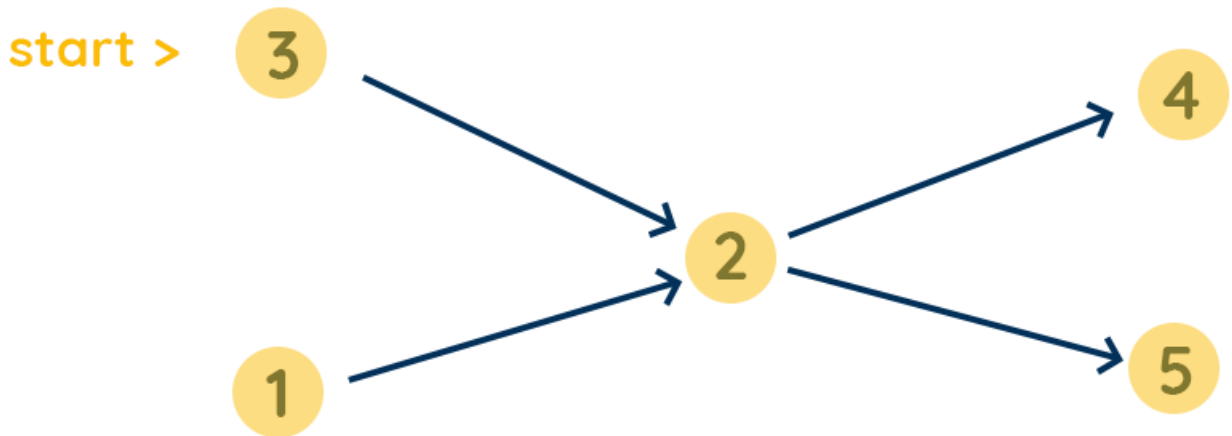
Simulation of DFS




```
vector < int > result;
```

result: 4, 5, 2, 1, 3

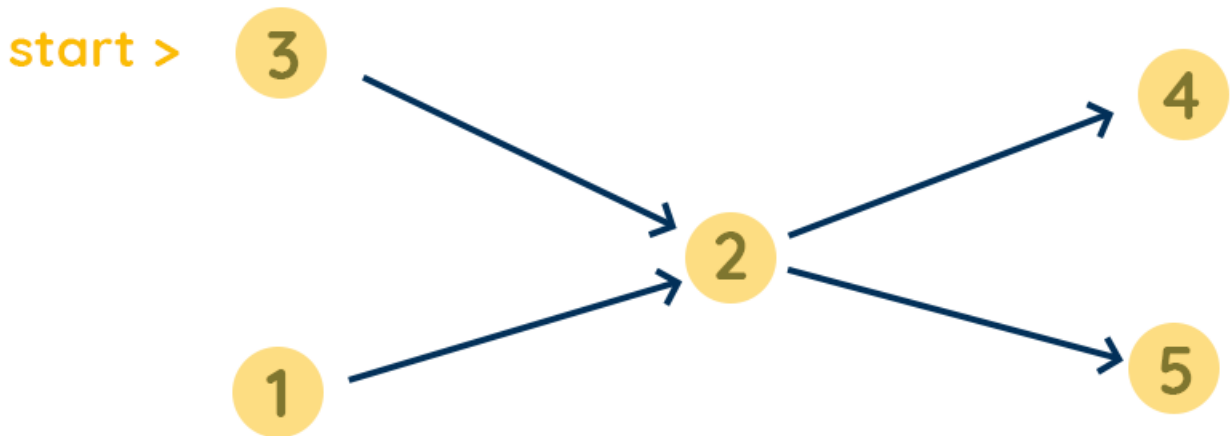
Simulation of DFS



vector < int > result;

result: 4, 5, 2, 1, 3 > reverse(result)

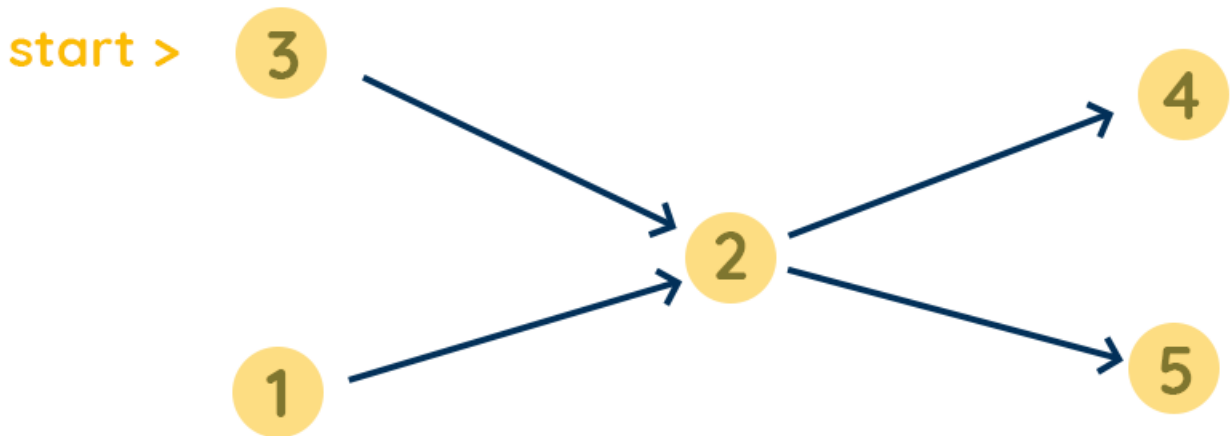
Simulation of DFS



vector < int > result;

result: 4, 5, 2, 1, 3 > reverse(result) result: 3, 1, 2, 5, 4

Simulation of DFS



vector < int > result;

result: 4, 5, 2, 1, 3 > reverse(result) result: 3, 1, 2, 5, 4

answer