Graph theory

Graph Traversal

Z.GUELLIL

Graph Traversal

- We call exploration/traversal of a graph any deterministic procedure which allows us to choose, from the visited vertices, the next vertex to visit.
- The problem is to determine an order on the visits of the vertices.
- There are two traversals of a graph, breadth-first traversal and depth-first traversal .

Breadth-first search (BFS) 1/2

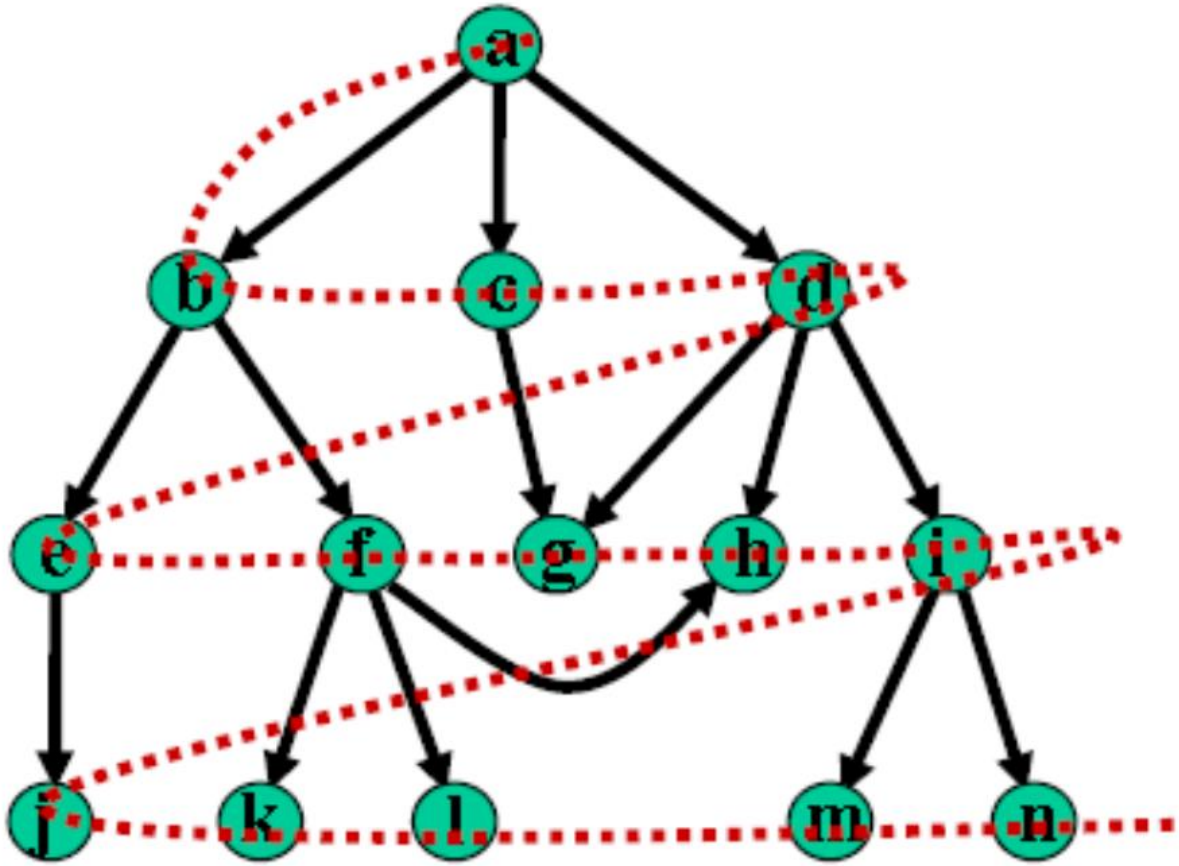
- We use a queue (FIFO).

1. Enqueue the starting vertex.
2. Visit the neighbors of the vertex at the front of the queue.
3. Enqueue each neighbor (assigning a discovery number as each is found)
    1. if it is not already in the queue and has not already been processed.
4. Dequeue (i.e., remove the vertex at the front of the queue).
5. Repeat from step 2 as long as possible, i.e., until the queue is empty.

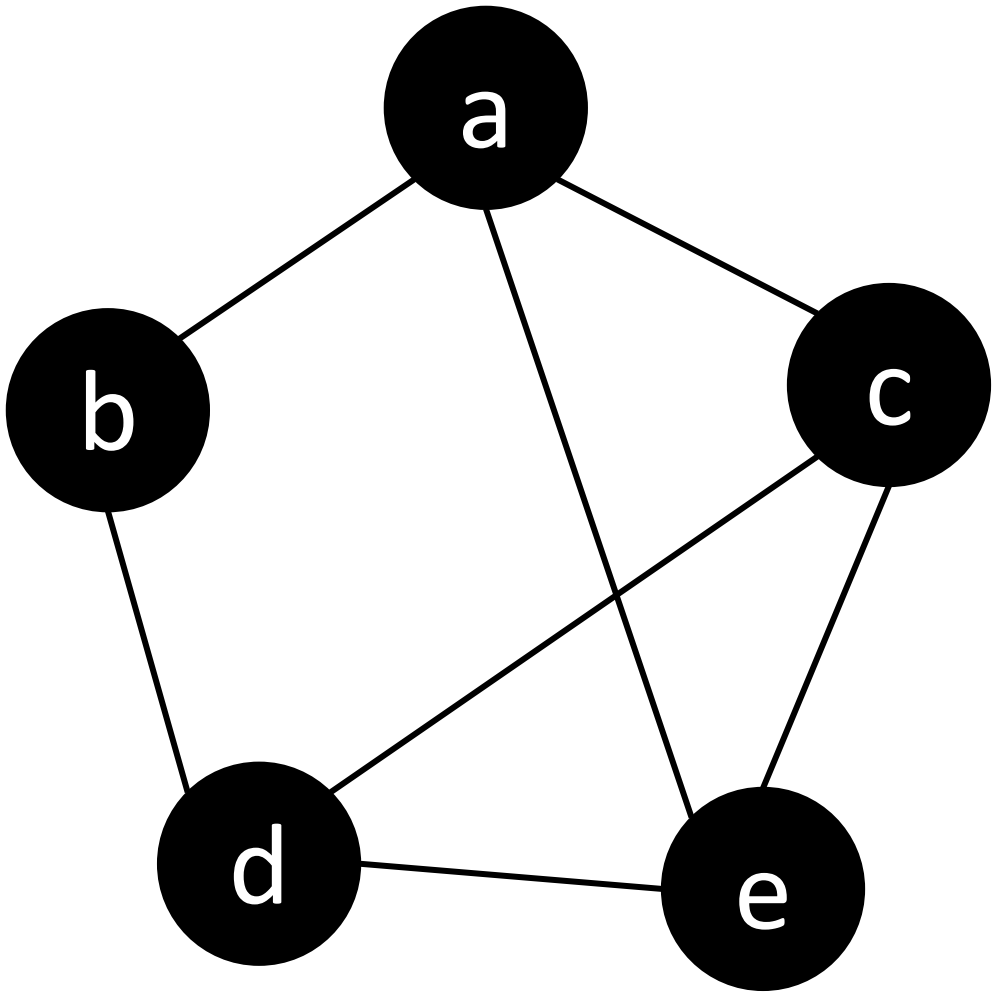
- It is essential to record which vertices have already been explored by the algorithm to avoid revisiting any previously explored vertices. For example, marking vertices can achieve this.

Breadth-first search (BFS) 2/2

```
BreadthFirstSearch(Vertex s) {
    Queue f = CreateQueue()
    f.enqueue(s)
    mark(s)
    WHILE (NOT f.isEmpty()) DO
        s = f.dequeue()
        Display(s)
        FOR EACH edge (s, v) DO
            IF NotMarked(v) THEN
                f.enqueue(v)
                mark(v)
            END IF
        End For
    END WHILE
}
```

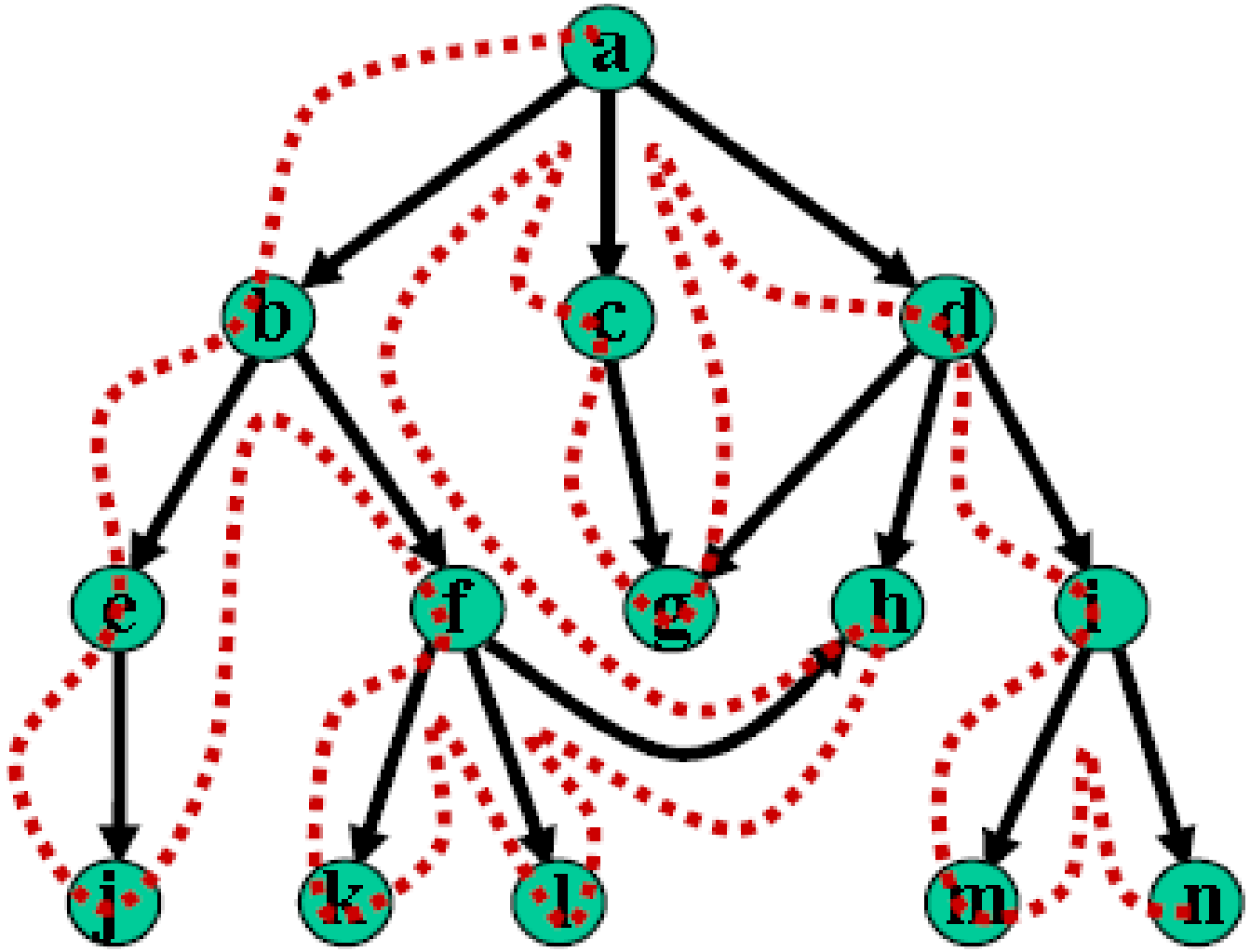


Example:



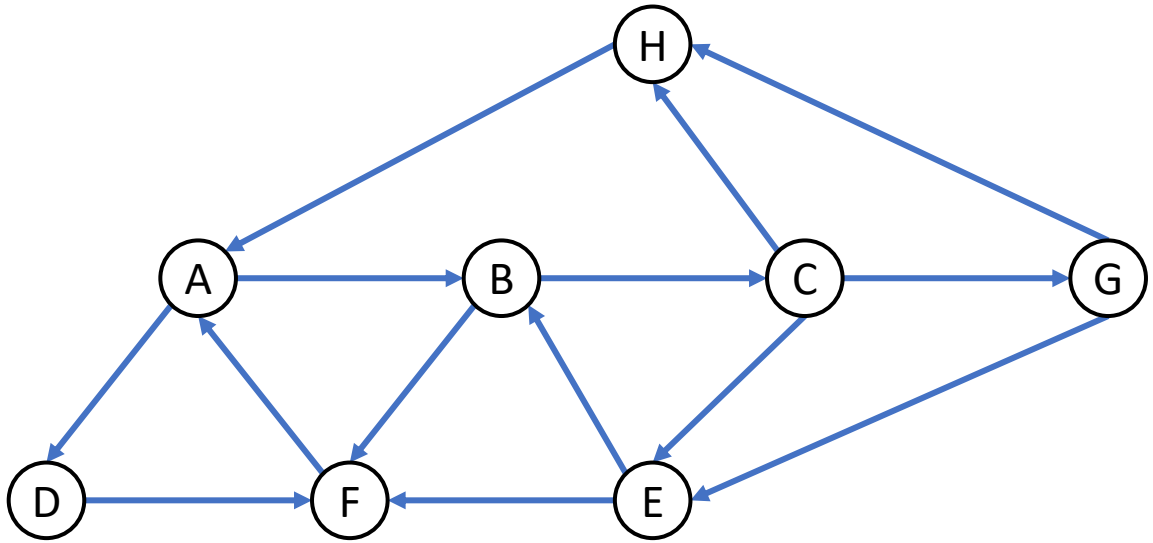
Depth-first search (DFS)

1. Push the starting vertex onto the stack.
2. **If** the vertex on top of the stack has neighbors that are not in the stack **and** have not already been visited **Then**
    1. Select one of these unvisited neighbors and push it onto the stack.
3. **Else**, pop the top vertex from the stack.
4. **Repeat** from step 2 until the stack is empty.


a
b
c
d
e
f
g
h
i
j
k
l
m
n

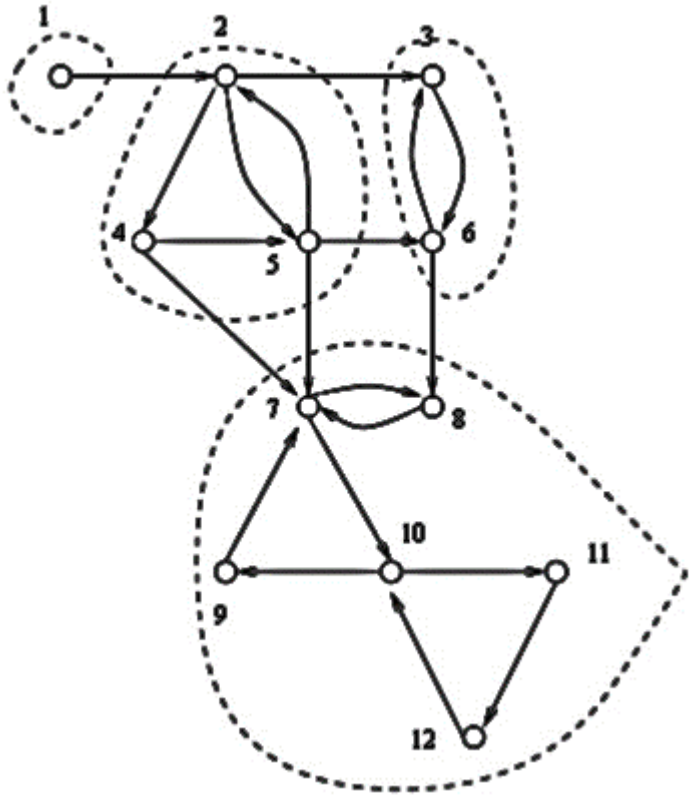
Exercise 1

- Perform a BFS starting from vertex A, visiting the vertices in lexicographical order.
- Perform a DFS starting from vertex A, visiting the vertices in lexicographical order.



Component strongly connectd

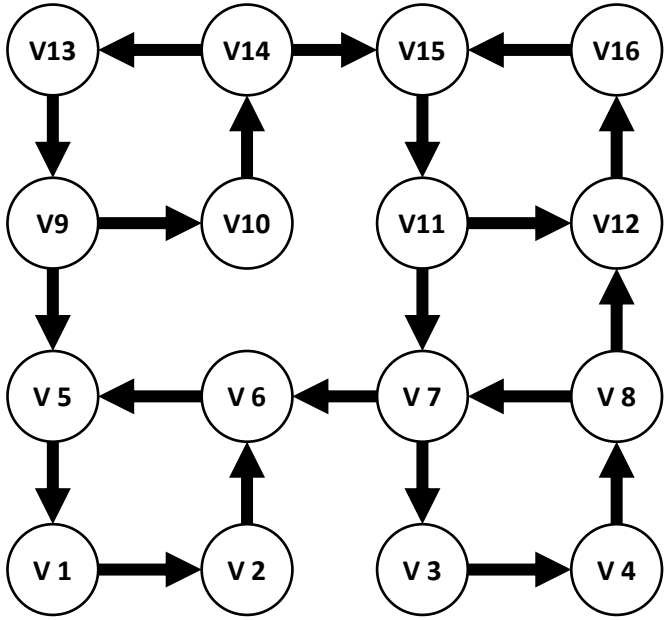
- A strongly connected component in a directed graph G = (V, E) is a maximal set of vertices such that for every pair of vertices u and v in the component, vertices u and v are reachable from each other.



Kosaraju Algorithm

- Based on the Depth-First Search (DFS) algorithm,
  1. initialize all vertices as unvisited.
  2. Perform an initial DFS traversal of the graph starting from an arbitrary vertex v, and record the order of vertex completions in a stack.
  3. Reverse all edges (or find the transposition of the graph).
  4. Mark all vertices as unvisited in the reversed graph.
  5. Perform a DFS traversal of the reversed graph starting from the same vertex order recorded in the stack.
- The trees produced by the second traversal represent the strongly connected components.

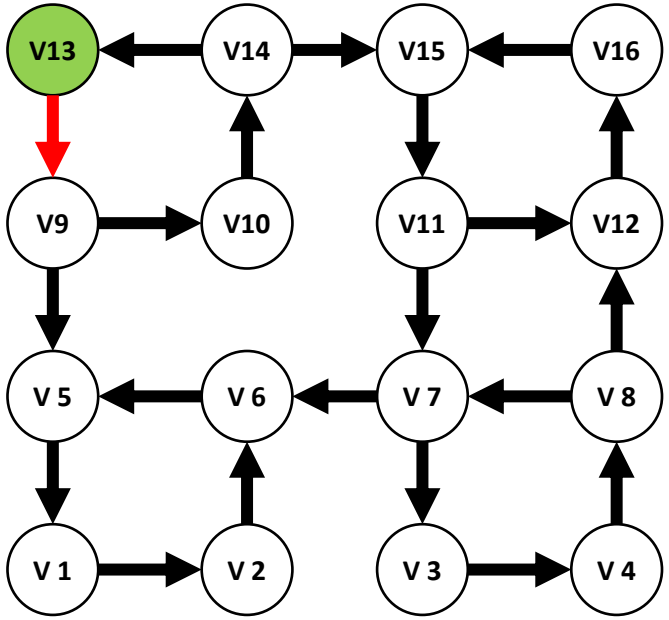
DFS starting from vertex V13



DFS Stack

Course stack

DFS starting from vertex V13

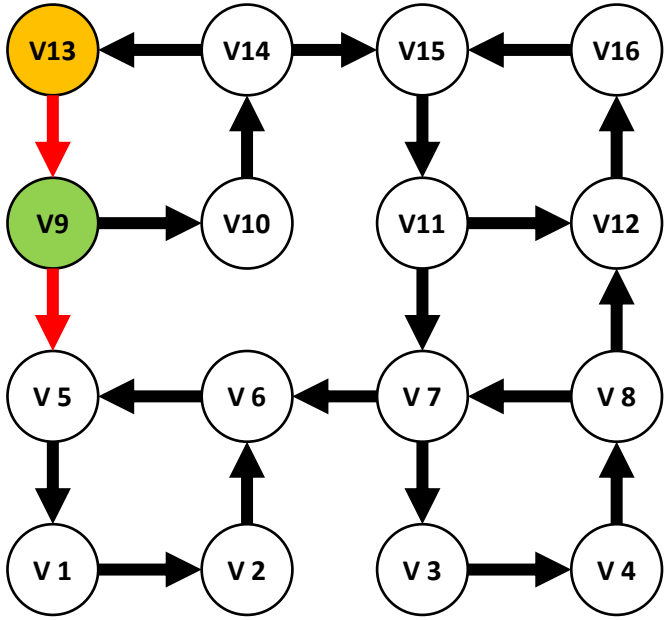


DFS Stack

V13

Course stack

DFS starting from vertex V13



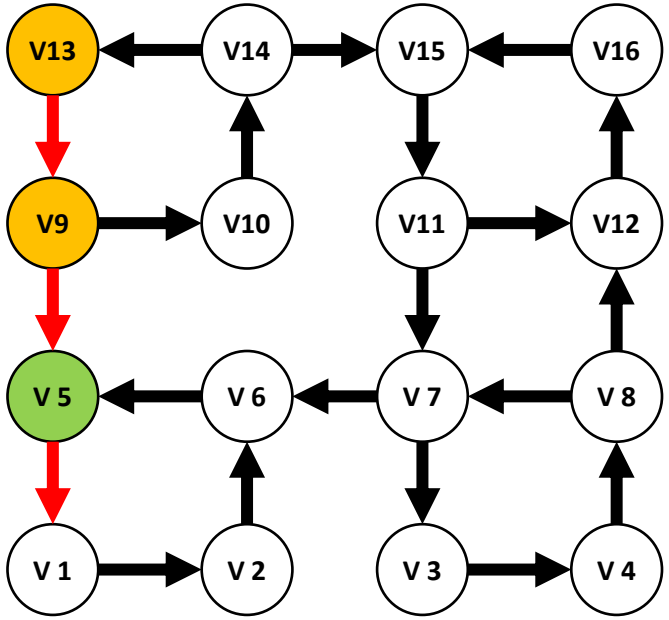
DFS Stack

V9

V13

Course stack

DFS starting from vertex V13

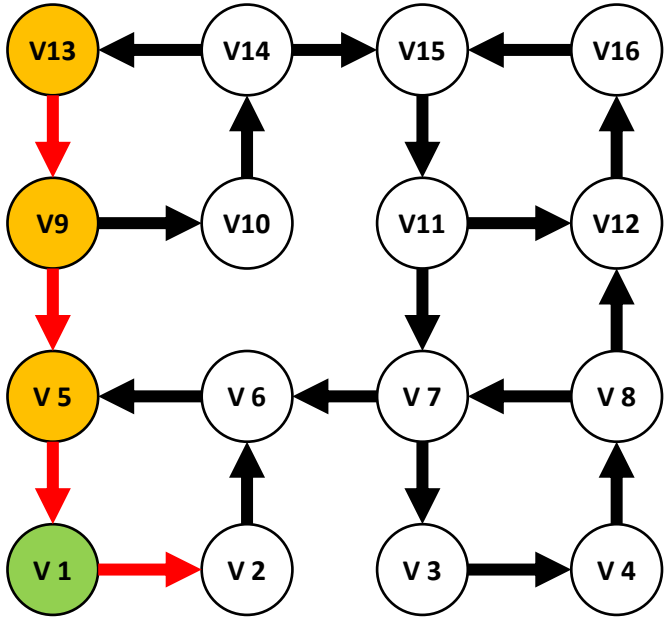


DFS Stack

V5
V9
V13

Course stack

DFS starting from vertex V13

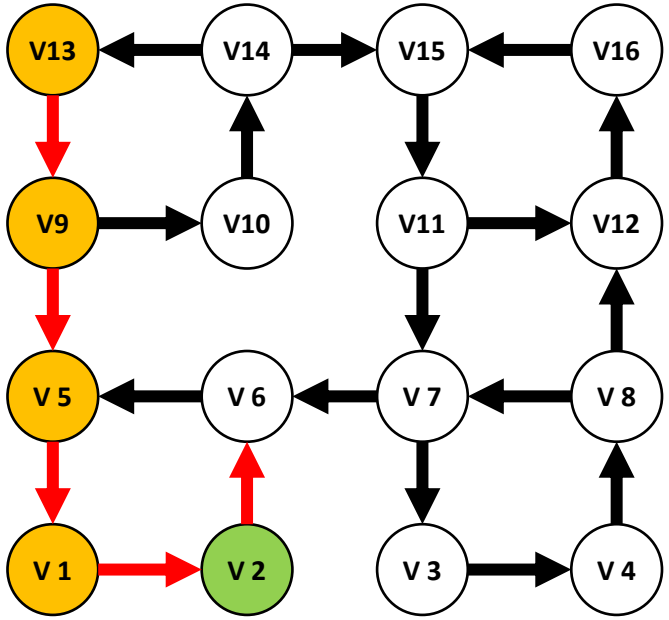


DFS Stack

V1
V5
V9
V13

Course stack

DFS starting from vertex V13

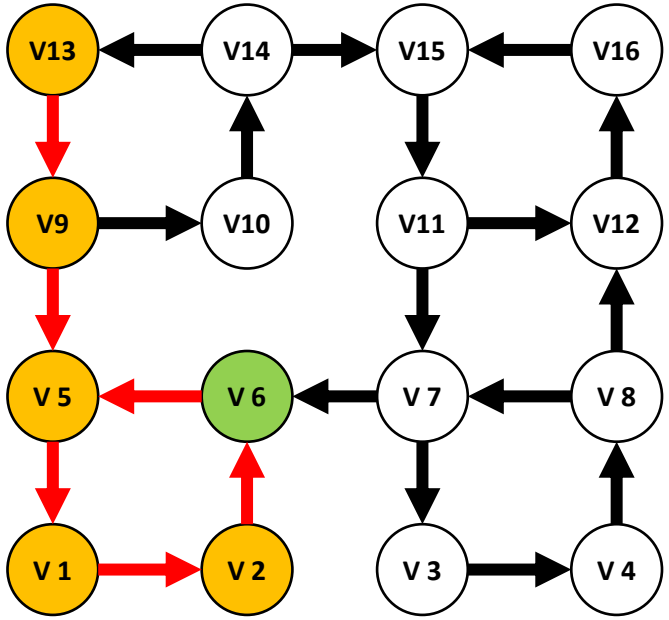


DFS Stack

V2
V1
V5
V9
V13

Course stack

DFS starting from vertex V13

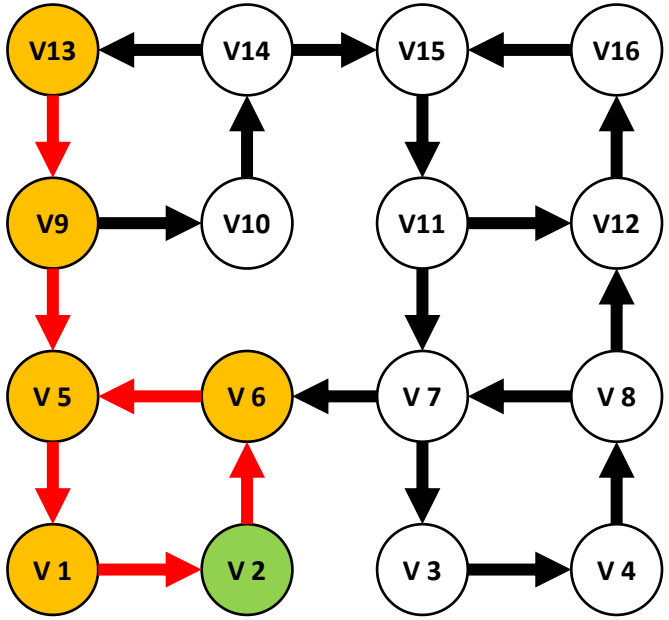


DFS Stack

V6
V2
V1
V5
V9
V13

Course stack

DFS starting from vertex V13



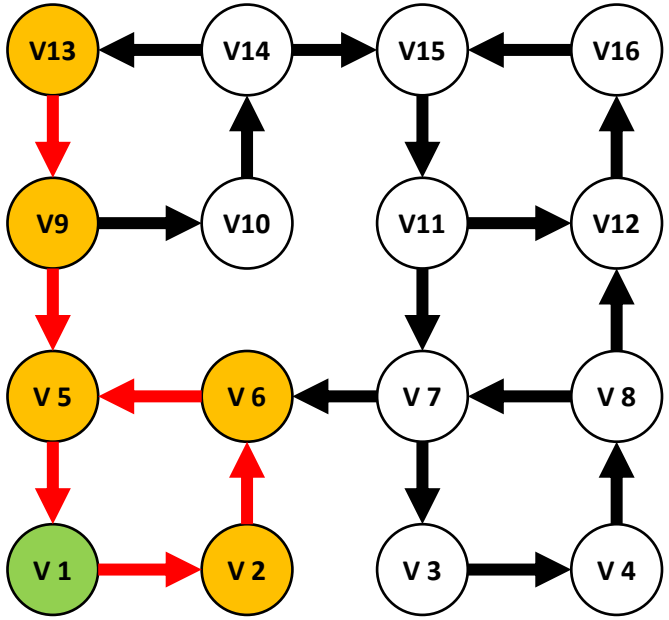
DFS Stack

V2
V1
V5
V9
V13

Course stack

V6

DFS starting from vertex V13



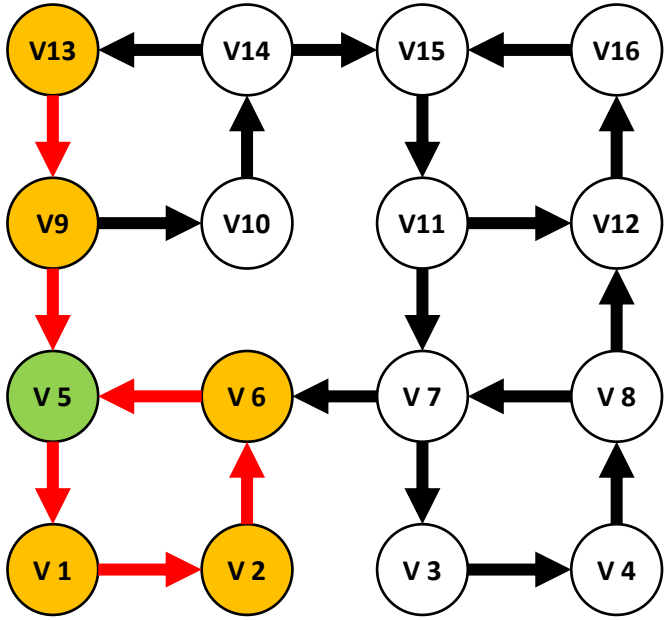
DFS Stack

V1
V5
V9
V13

Course stack

V2
V6

DFS starting from vertex V13



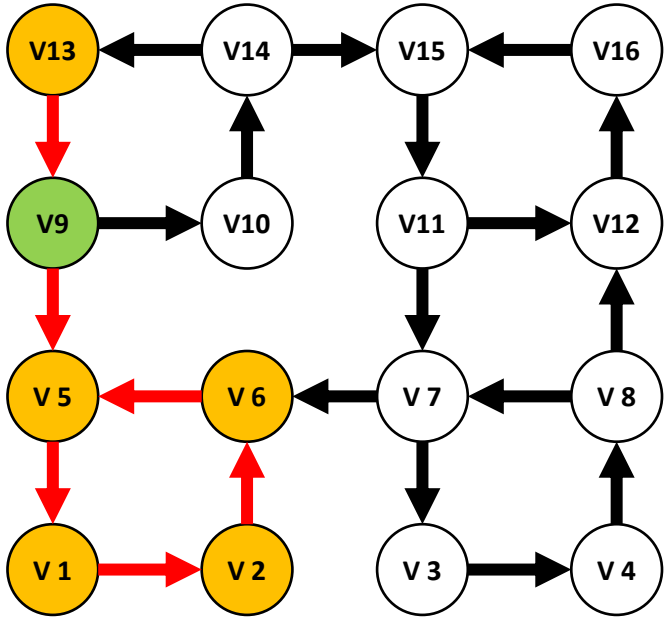
DFS Stack

V5
V9
V13

Course stack

V1
V2
V6

DFS starting from vertex V13



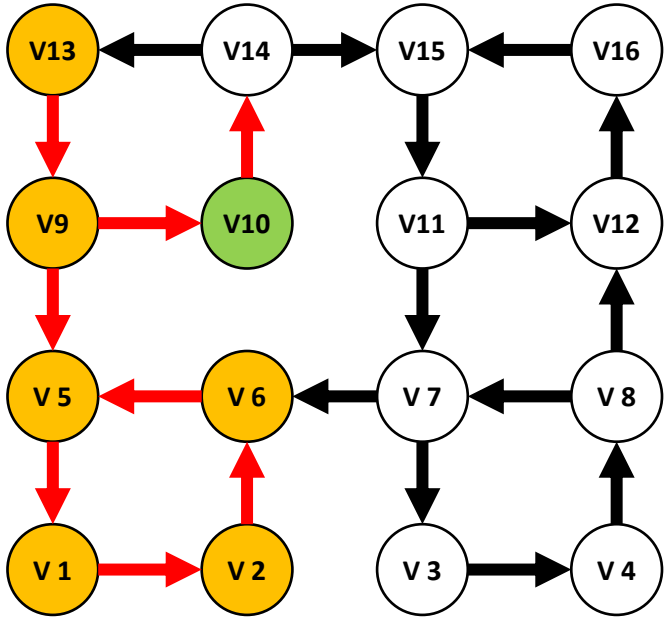
DFS Stack

V9
V13

Course stack

V5
V1
V2
V6

DFS starting from vertex V13



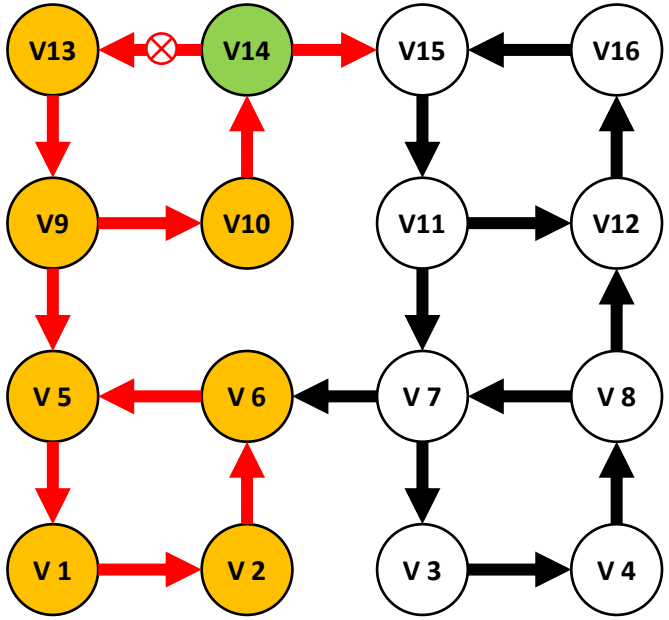
DFS Stack

V10
V9
V13

Course stack

V5
V1
V2
V6

DFS starting from vertex V13



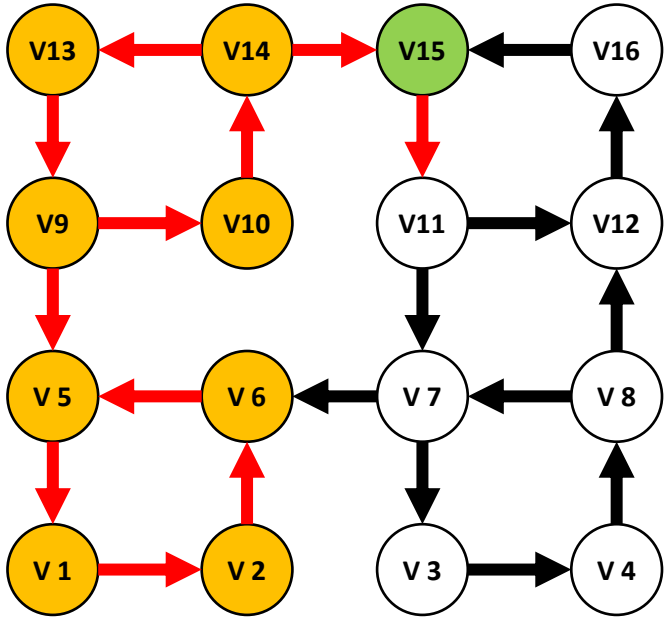
DFS Stack

V14
V10
V9
V13

Course stack

V5
V1
V2
V6

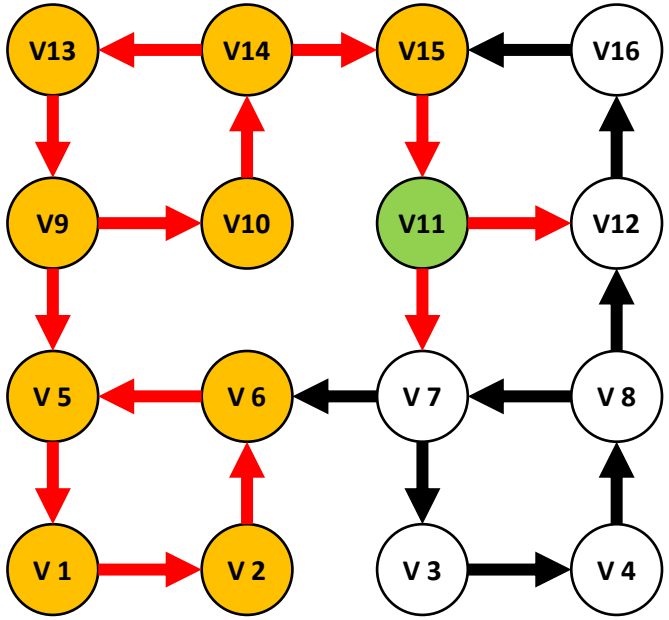
DFS starting from vertex V13



DFS Stack
V15
V14
V10
V9
V13

Course stack
V5
V1
V2
V6

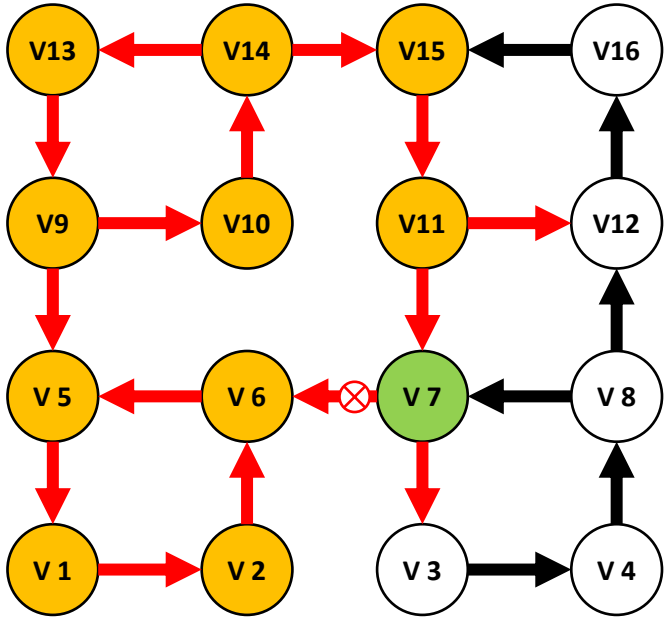
DFS starting from vertex V13



DFS Stack
V11
V15
V14
V10
V9
V13

Course stack
V5
V1
V2
V6

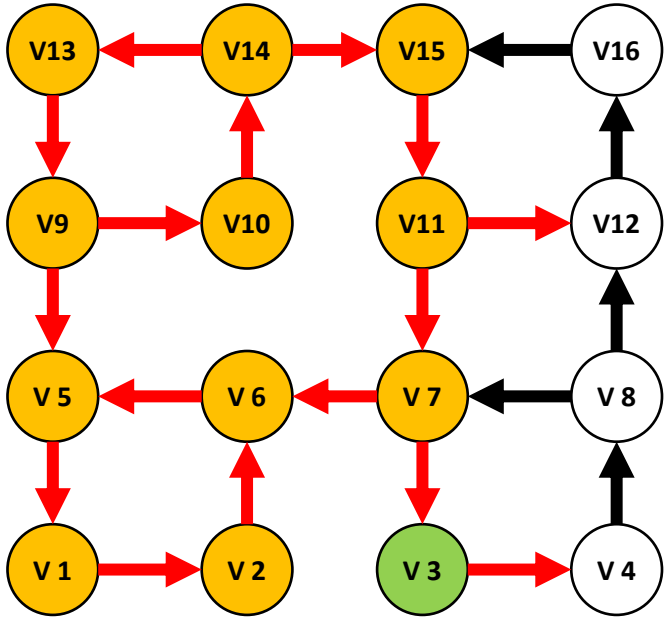
DFS starting from vertex V13



DFS Stack
V7
V11
V15
V14
V10
V9
V13

Course stack
V5
V1
V2
V6

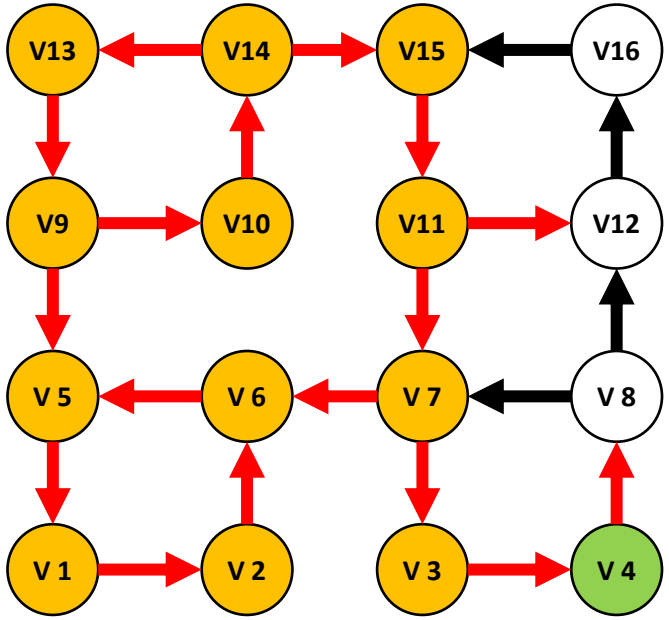
DFS starting from vertex V13



DFS Stack
V 3
V7
V11
V15
V14
V10
V9
V13

Course stack
V5
V1
V2
V6

DFS starting from vertex V13



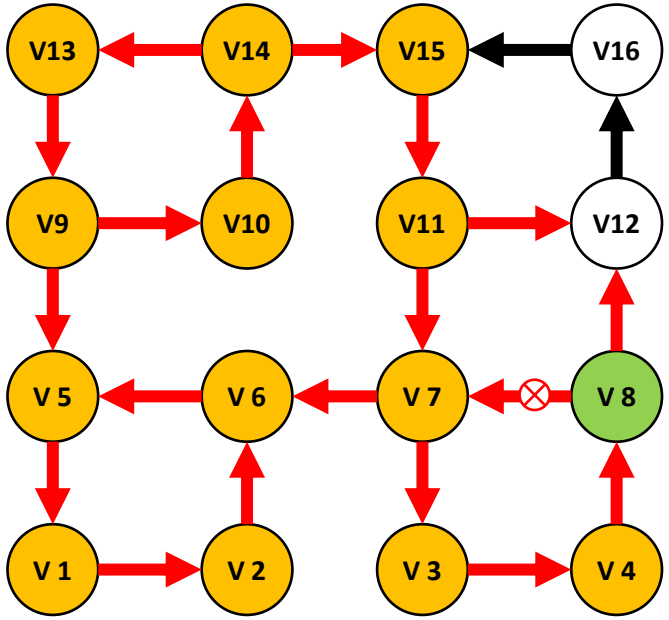
DFS Stack

V4
V3
V7
V11
V15
V14
V10
V9
V13

Course stack

V5
V1
V2
V6

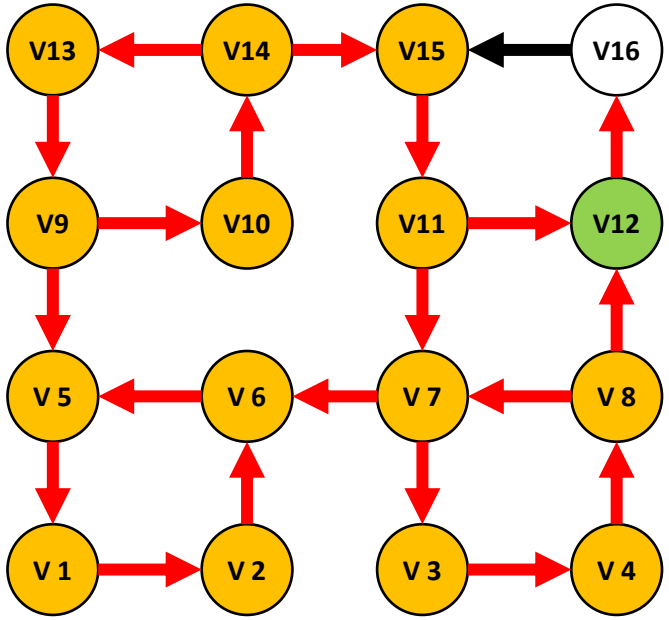
DFS starting from vertex V13



DFS Stack
V8
V4
V3
V7
V11
V15
V14
V10
V9
V13

Course stack
V5
V1
V2
V6

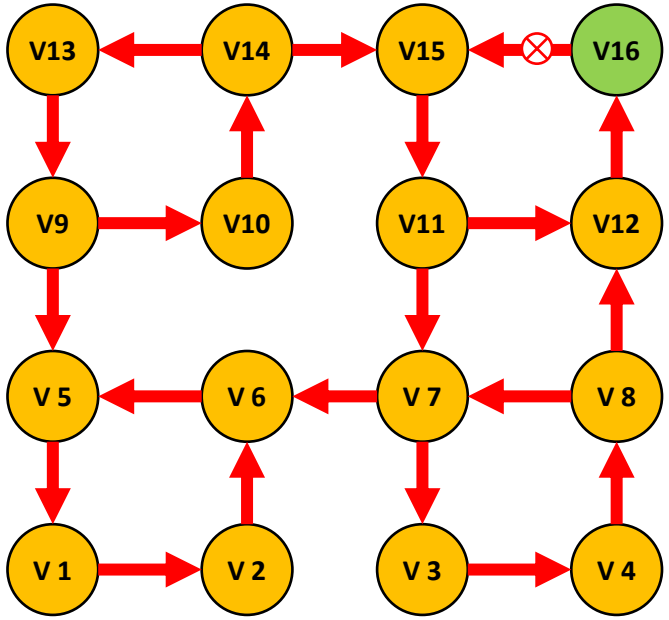
DFS starting from vertex V13



DFS Stack
V12
V8
V4
V3
V7
V11
V15
V14
V10
V9
V13

Course stack
V5
V1
V2
V6

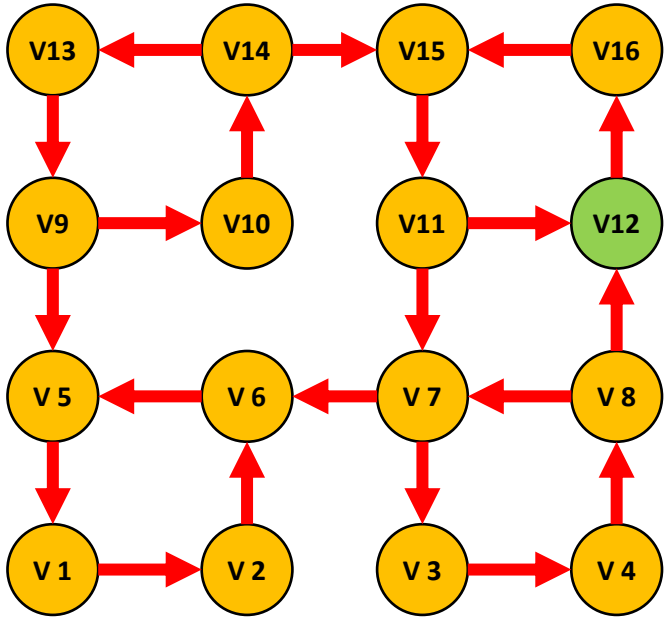
DFS starting from vertex V13



DFS Stack
V16
V12
V8
V4
V3
V7
V11
V15
V14
V10
V9
V13

Course stack
V5
V1
V2
V6

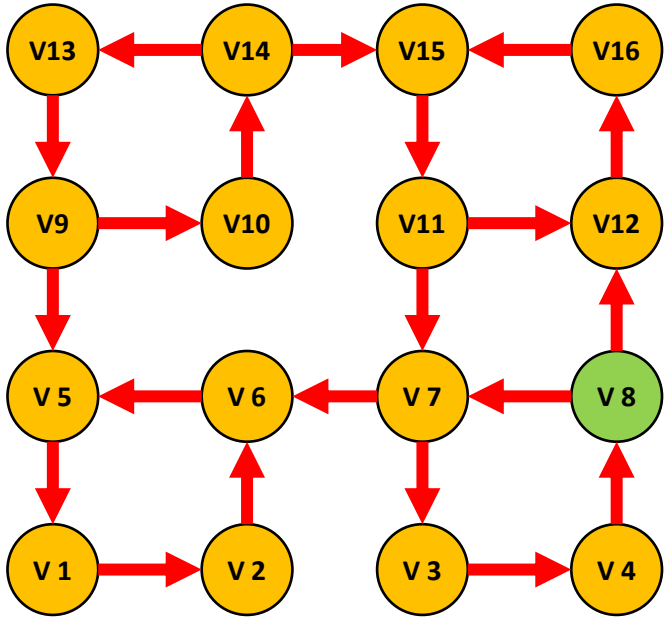
DFS starting from vertex V13



DFS Stack
V12
V8
V4
V3
V7
V11
V15
V14
V10
V9
V13

Course stack
V16
V5
V1
V2
V6

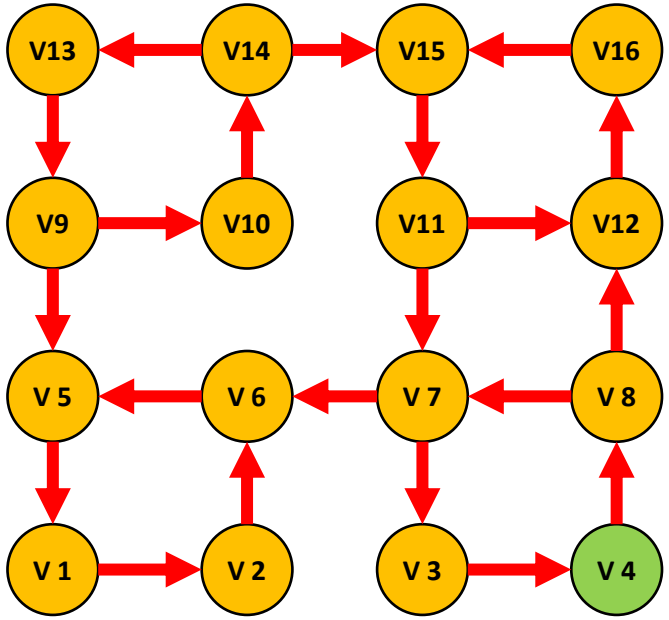
DFS starting from vertex V13



DFS Stack
V8
V4
V3
V7
V11
V15
V14
V10
V9
V13

Course stack
V12
V16
V5
V1
V2
V6

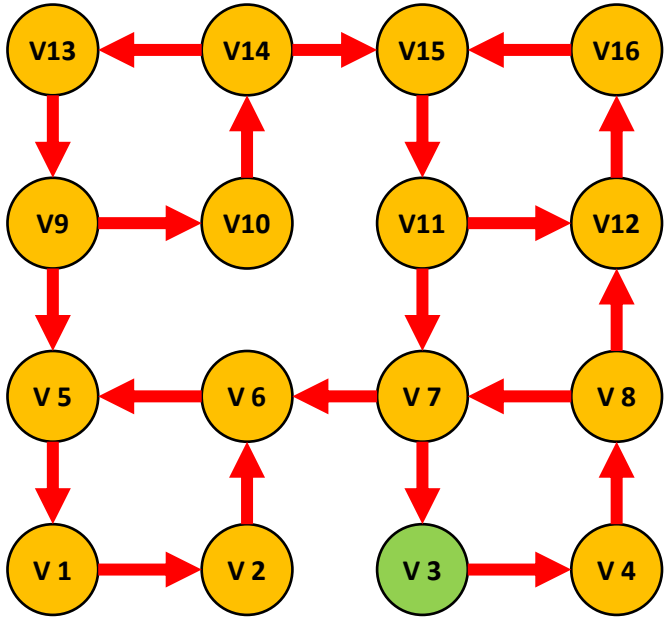
DFS starting from vertex V13



DFS Stack
V4
V3
V7
V11
V15
V14
V10
V9
V13

Course stack
V8
V12
V16
V5
V1
V2
V6

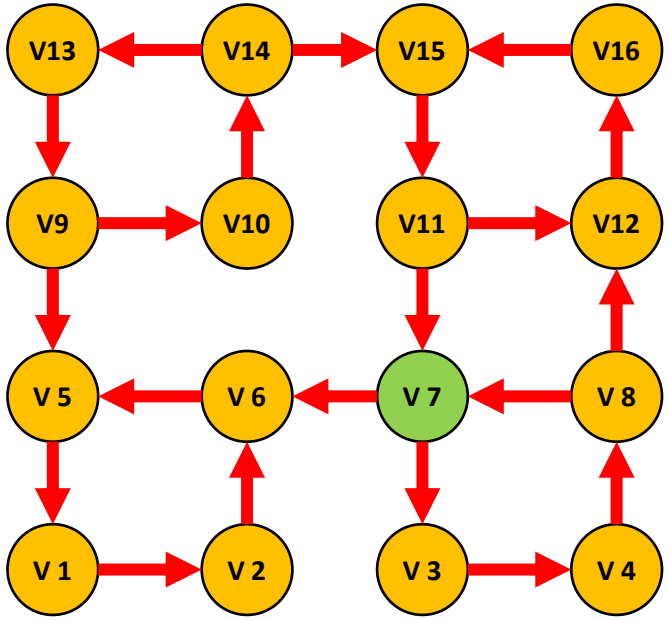
DFS starting from vertex V13



DFS Stack
V3
V7
V11
V15
V14
V10
V9
V13

Course stack
V4
V8
V12
V16
V5
V1
V2
V6

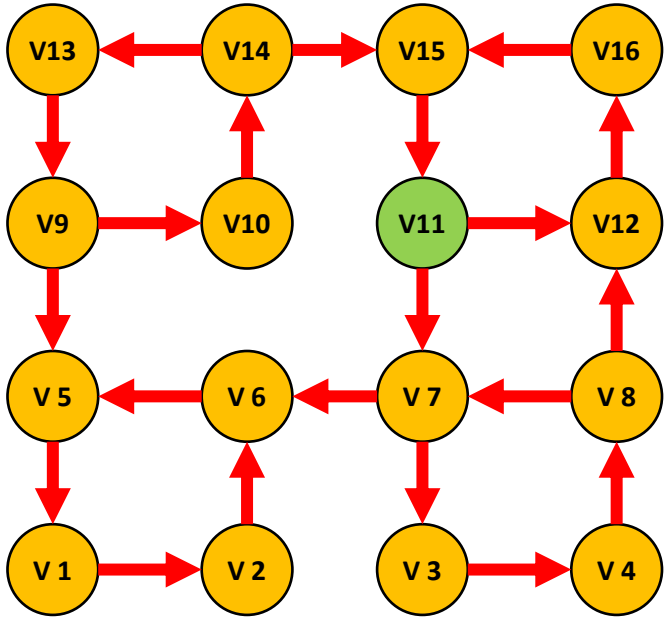
DFS starting from vertex V13



DFS Stack
V7
V11
V15
V14
V10
V9
V13

Course stack
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS starting from vertex V13



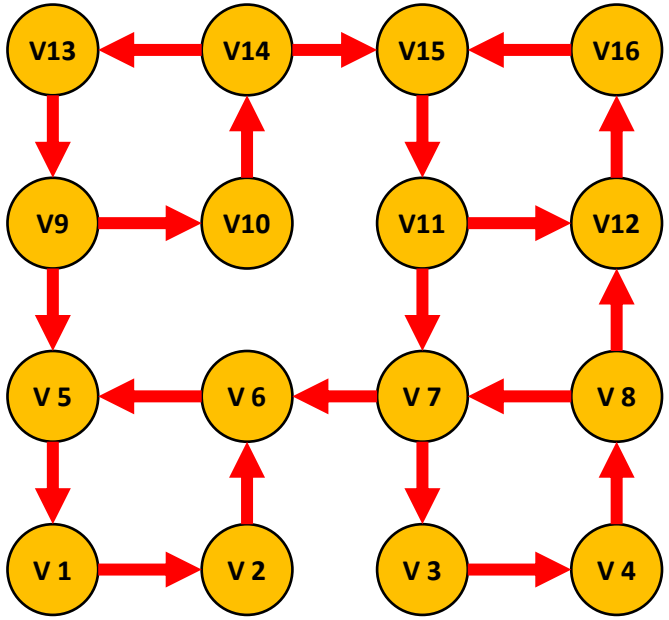
DFS Stack
V11
V15
V14
V10
V9
V13

Course stack
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

We continue to unstack the Vertex V11, V15, V14, V10, V9, V13 from the DFS stack

And we stack them in the course pile,

DFS starting from vertex V13



DFS Stack

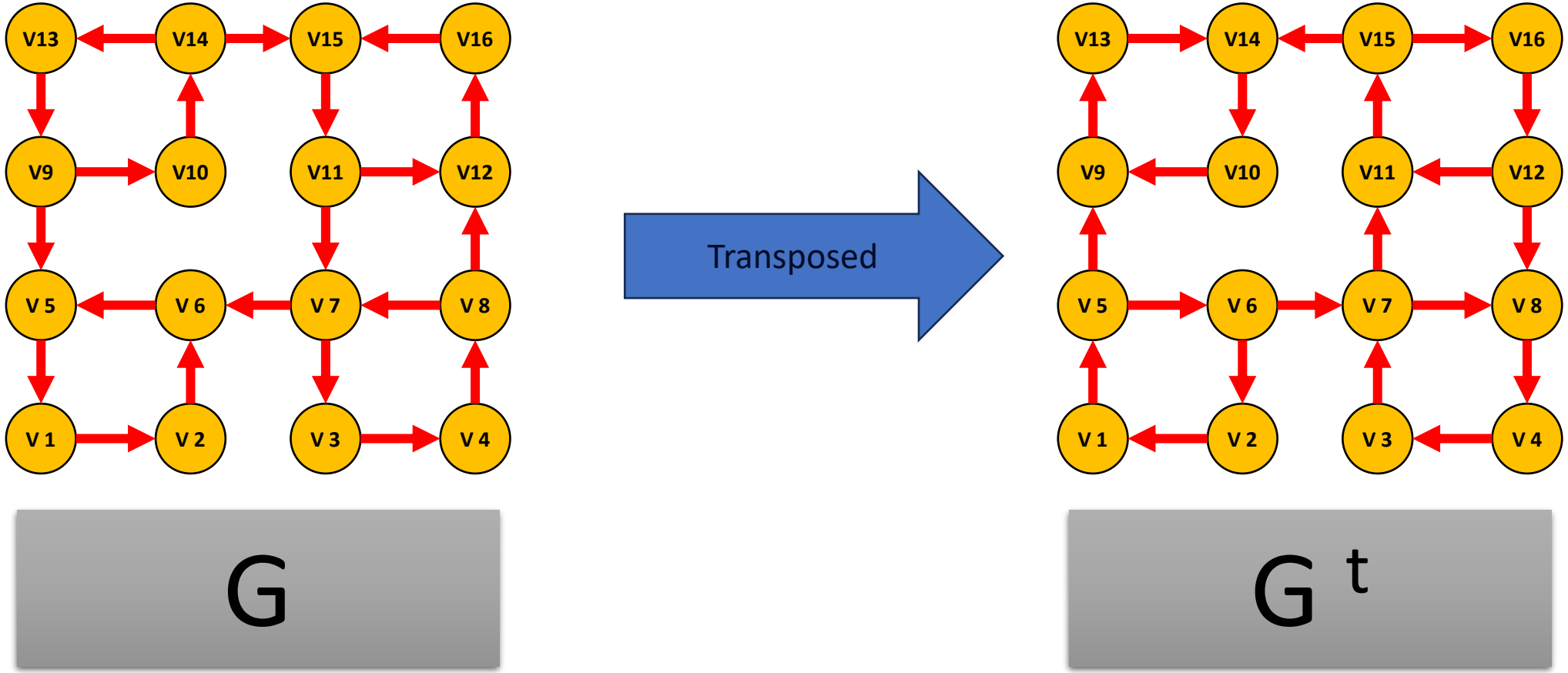
Course stack

V13
V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

We continue to unstack the Vertex V11, V15, V14, V10, V9, V13 from the DFS stack

And we stack them in the course pile,

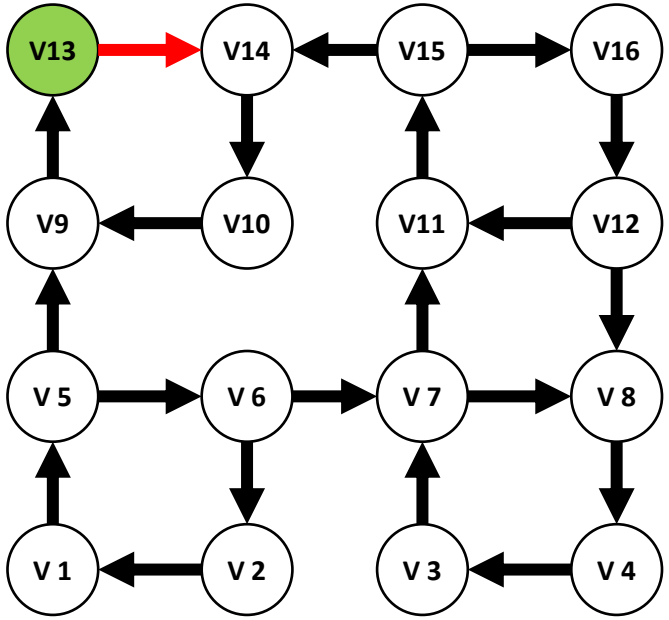
End of the first course


V13
V14
V15
V16
V9
V10
V11
V12
V 5
V 6
V 7
V 8
V 1
V 2
V 3
V 4
Transposed
V13
V14
V15
V16
V9
V10
V11
V12
V 5
V 6
V 7
V 8
V 1
V 2
V 3
V 4

G

G^t

DFS transposed from G^t

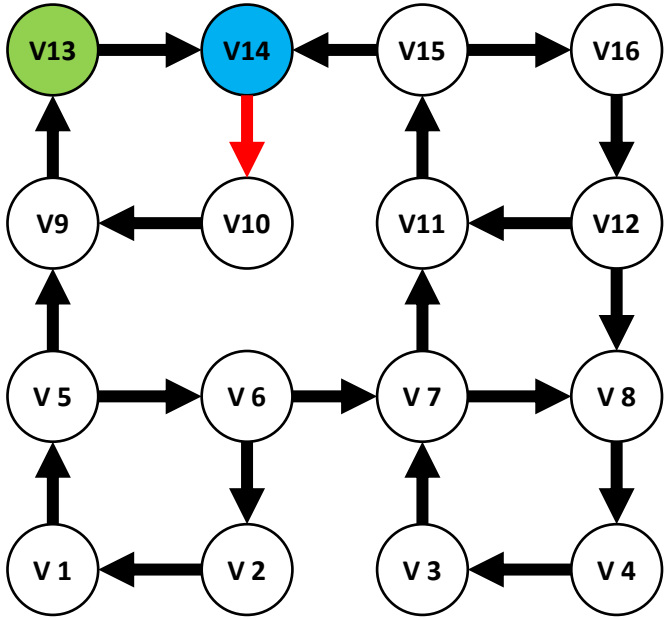


V13

DFS Stack
V13

Course stack
V13
V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t

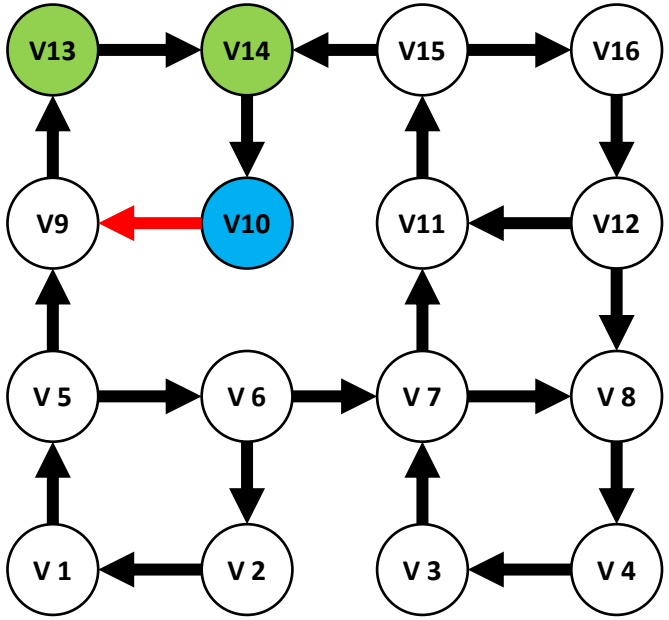


V13,V14

DFS Stack
V14
V13

Course stack
V13
V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t

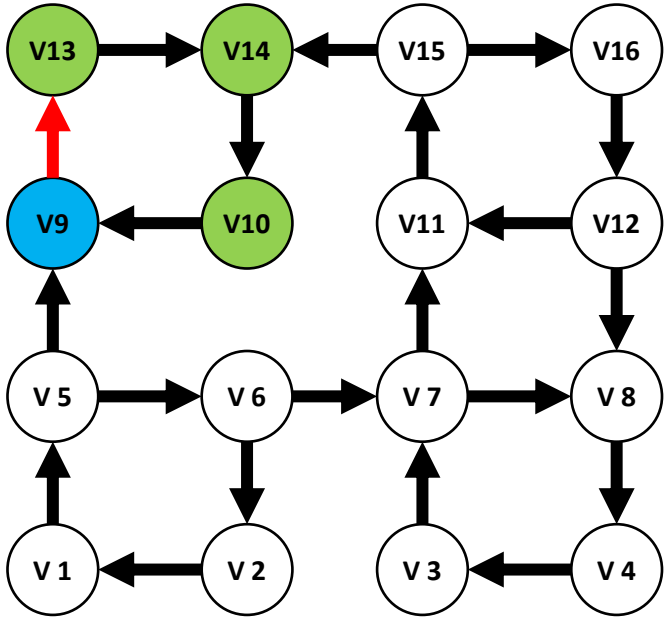


V13,V14,V10

DFS Stack
V10
V14
V13

Course stack
V13
V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t

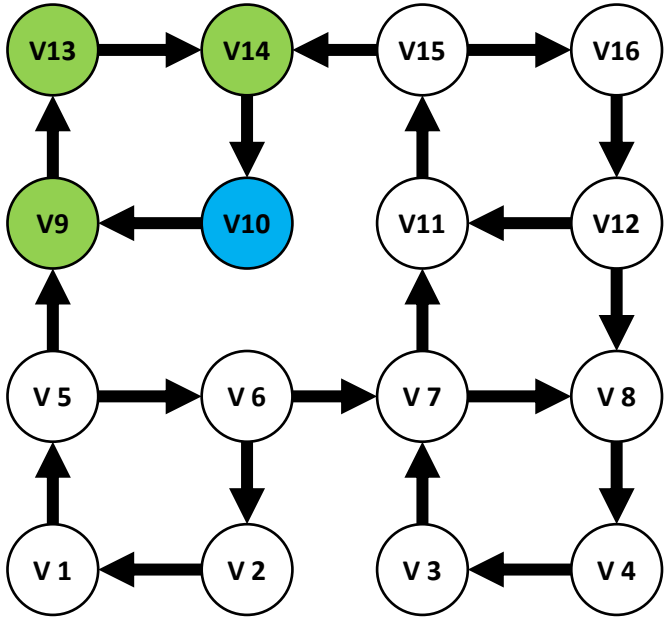


V13,V14,V10,V9

DFS Stack
V9
V10
V14
V13

Course stack
V13
V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}

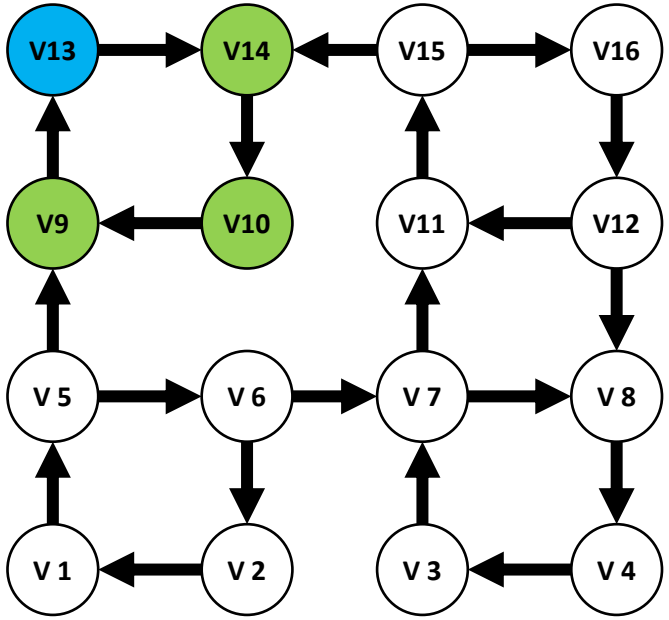


V13,V14,V10,V9

DFS Stack
V10
V14
V13

Course stack
V13
V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t

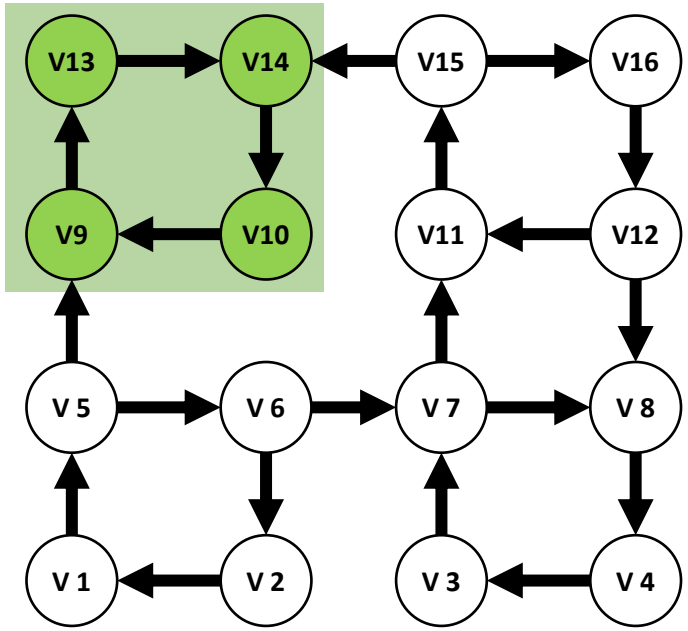


V13,V14,V10,V9

DFS Stack
V13

Course stack
V13
V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



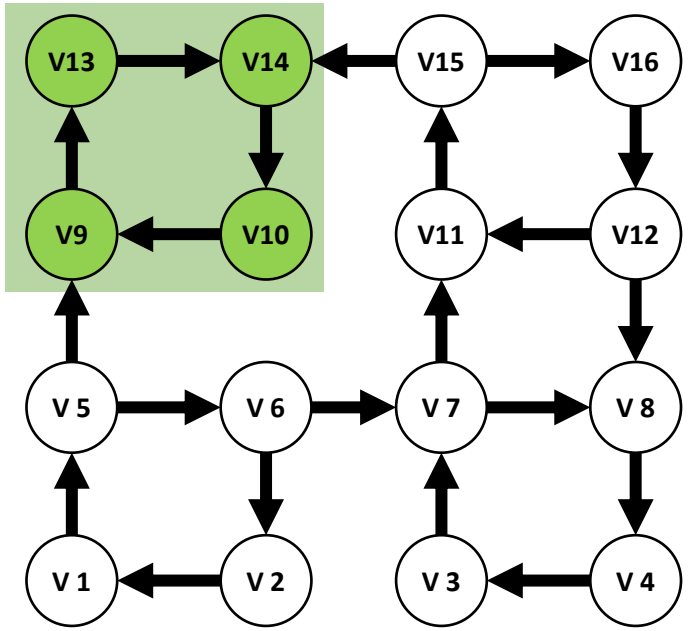
V13,V14,V10,V9

DFS Stack

Course stack

V13
V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



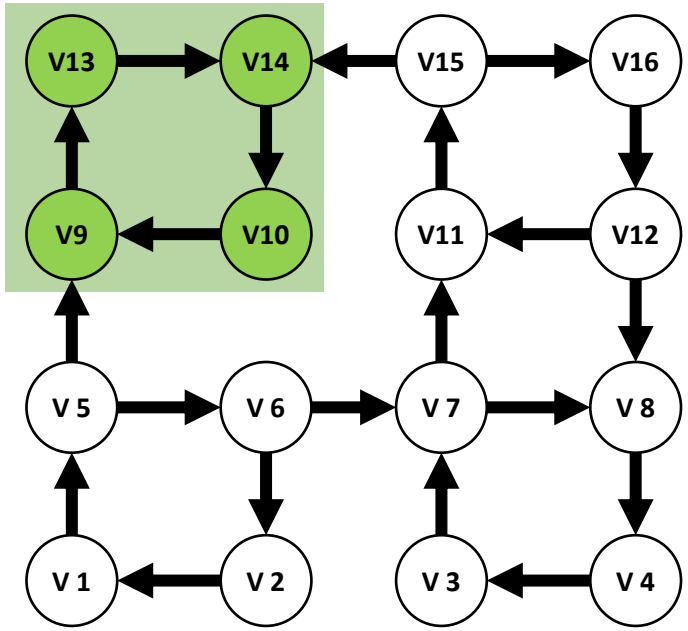
V13,V14,V10,V9

DFS Stack

Course stack

V9
V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



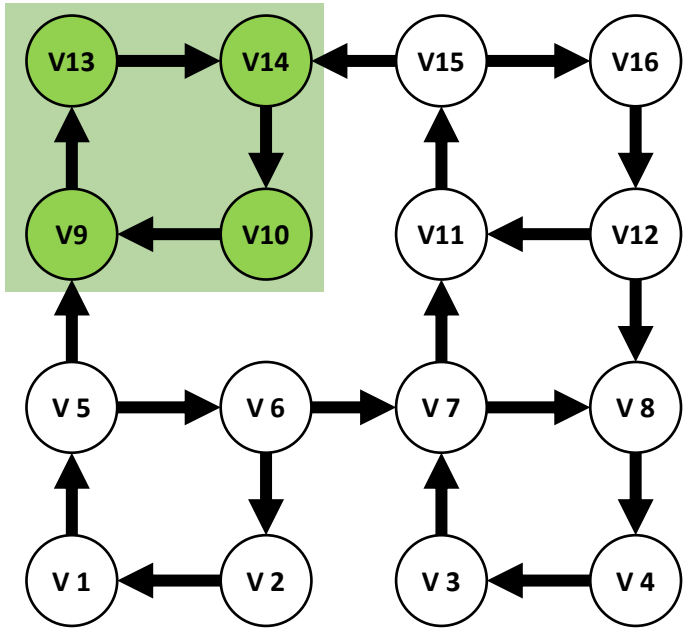
V13,V14,V10,V9

DFS Stack

Course stack

V10
V14
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



DFS Stack

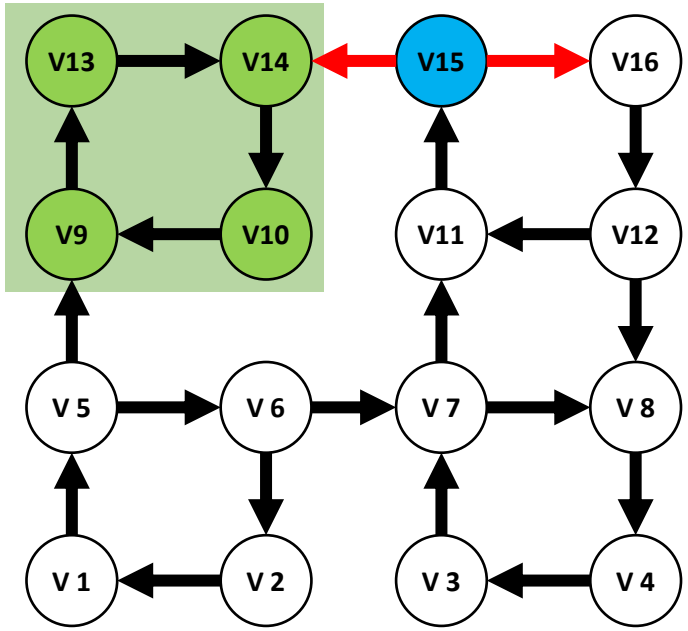
Course stack

V13,V14,V10,V9

V15

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15

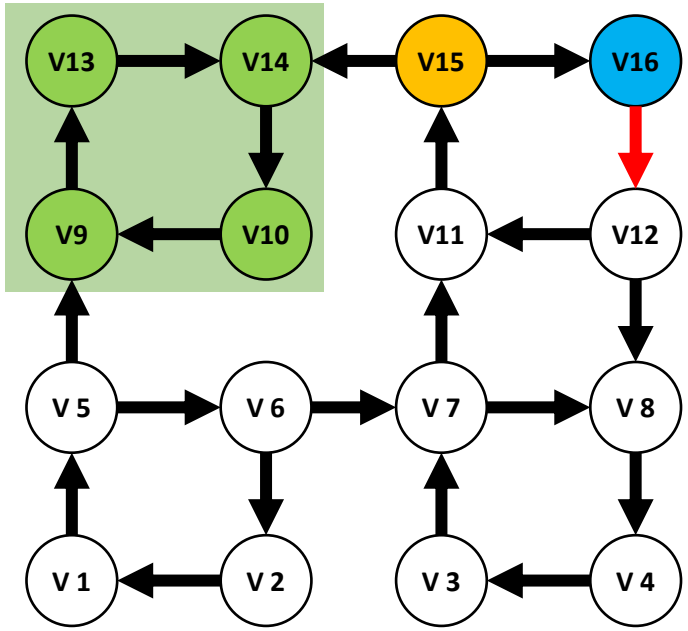
DFS Stack

V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



V13,V14,V10,V9

V15,V16

DFS Stack

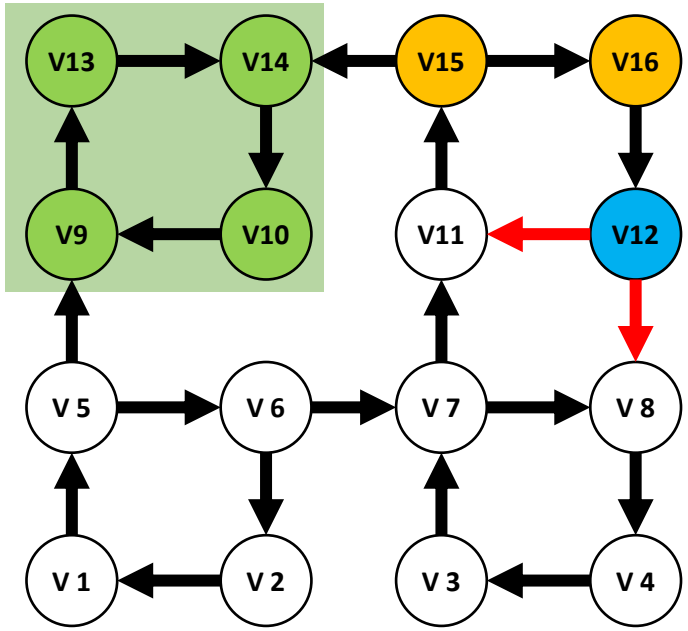
V16

V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



V13,V14,V10,V9

V15,V16,V12

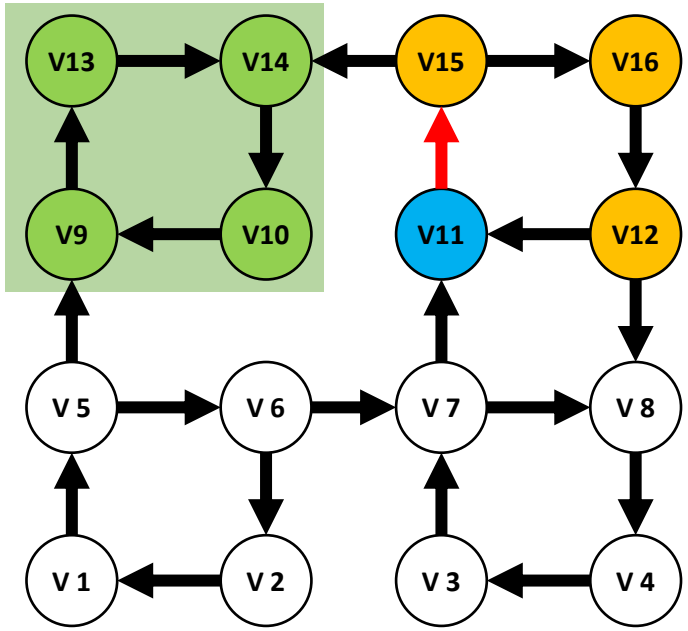
DFS Stack

V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



V13,V14,V10,V9

V15,V16,V12,V11

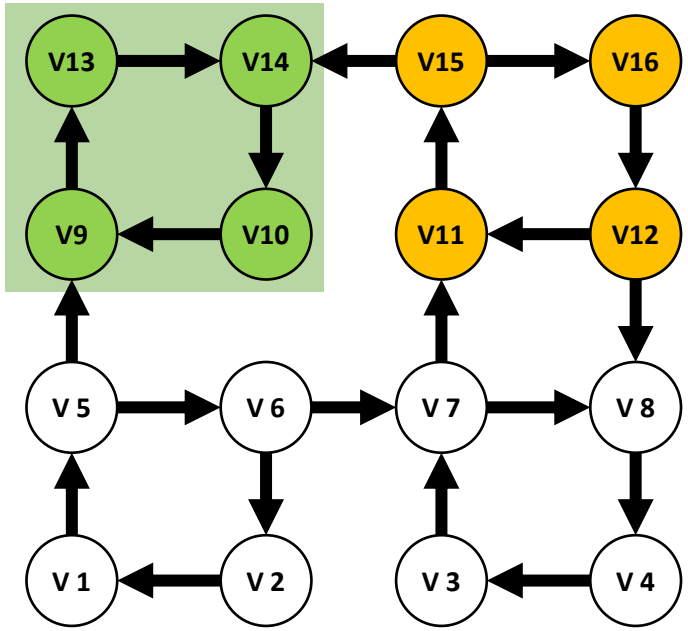
DFS Stack

V11
V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11

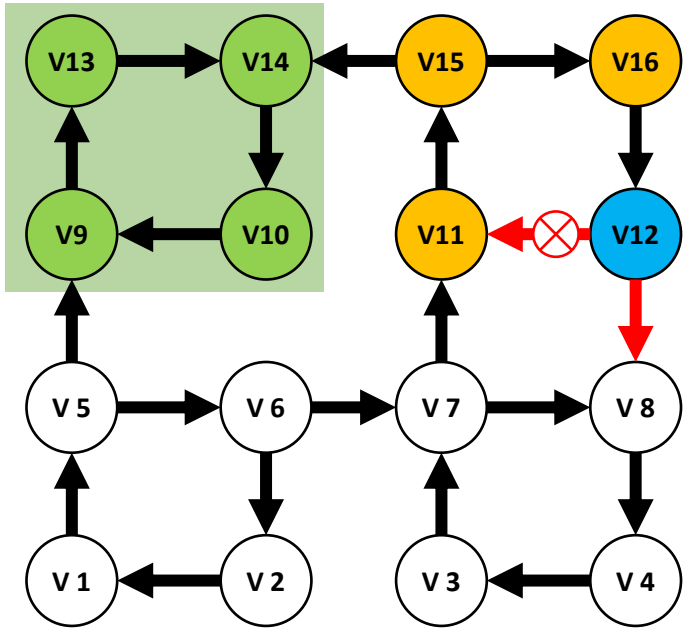
DFS Stack

V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



V13,V14,V10,V9

V15,V16,V12,V11

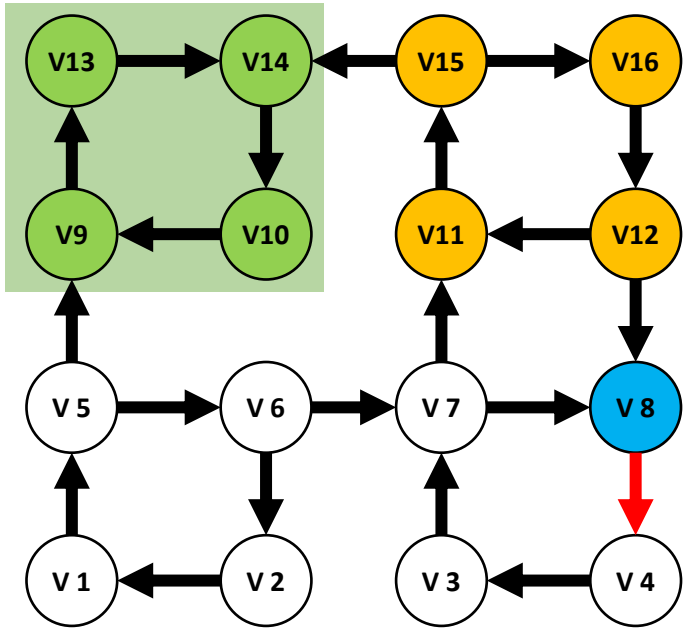
DFS Stack

V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11,V8

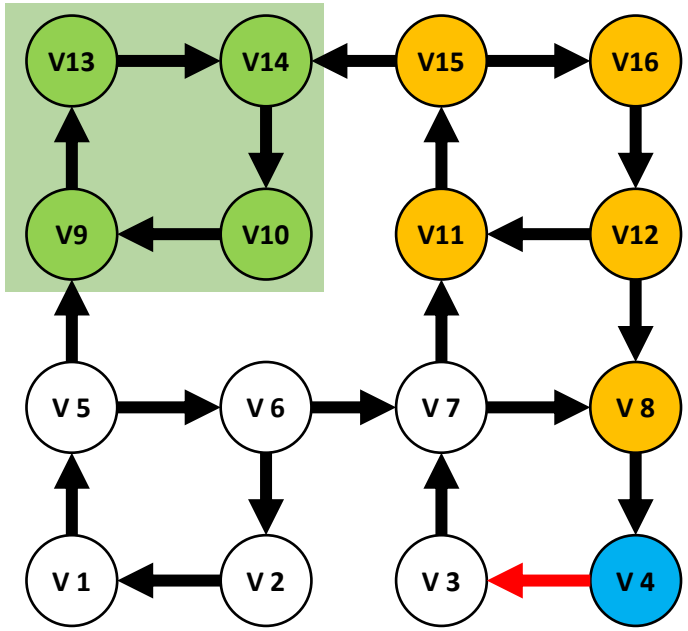
DFS Stack

V8
V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t



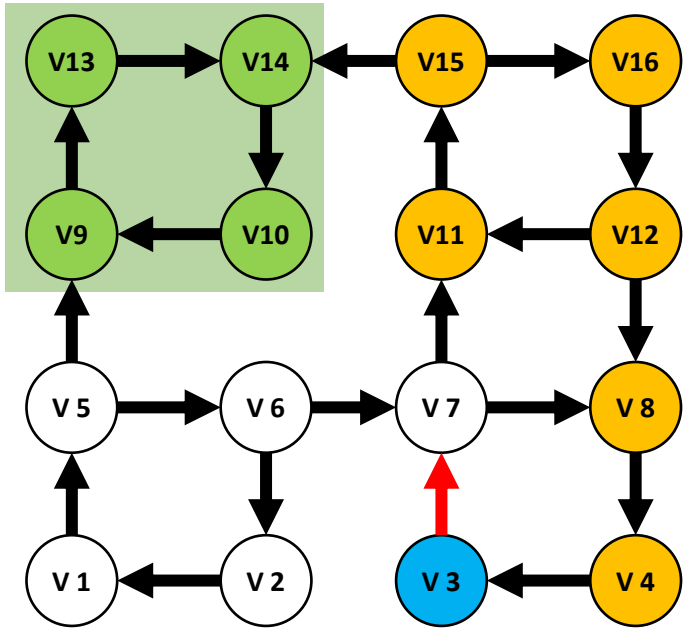
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4

DFS Stack
V4
V8
V12
V16
V15

Course stack
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



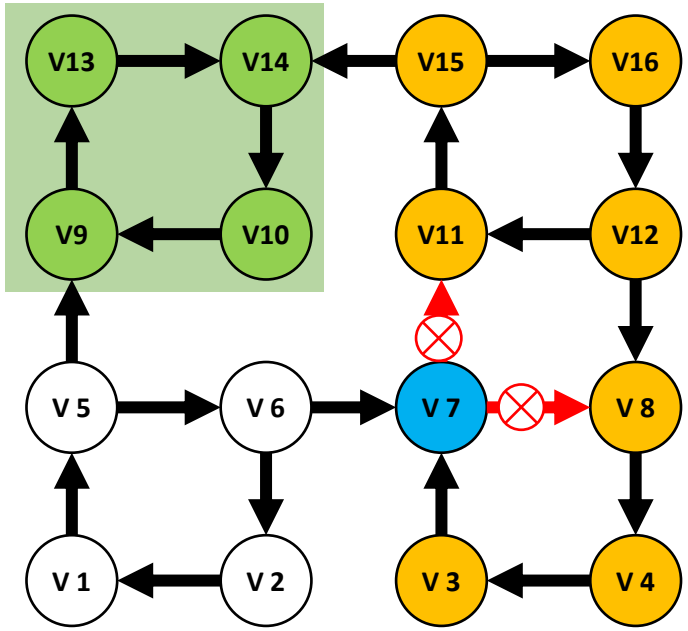
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3

DFS Stack
V3
V4
V8
V12
V16
V15

Course stack
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



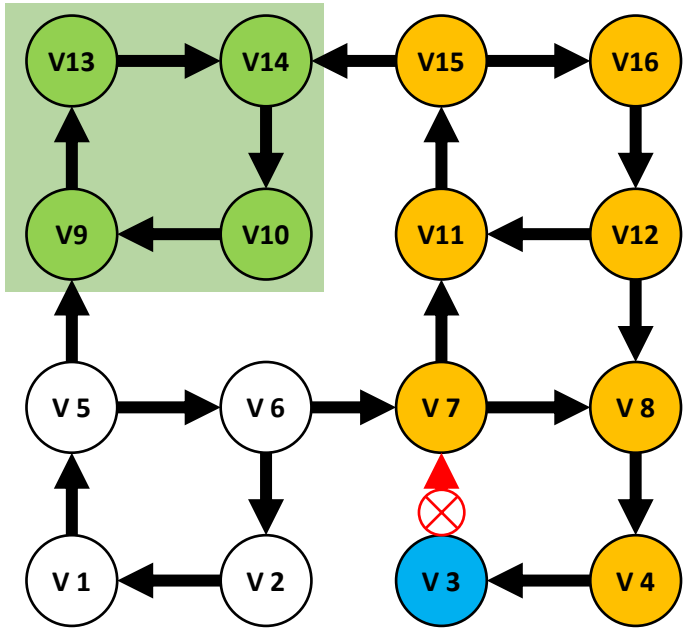
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

DFS Stack
V7
V3
V4
V8
V12
V16
V15

Course stack
V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

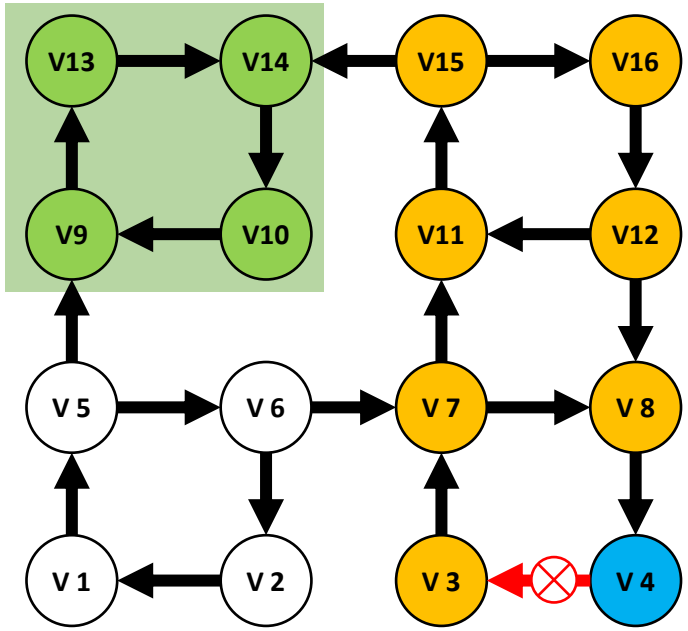
DFS Stack

V3
V4
V8
V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

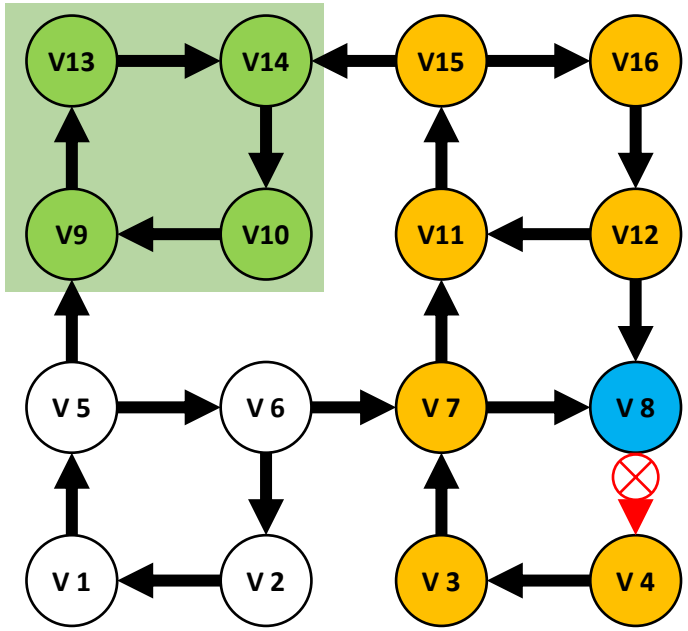
DFS Stack

V4
V8
V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

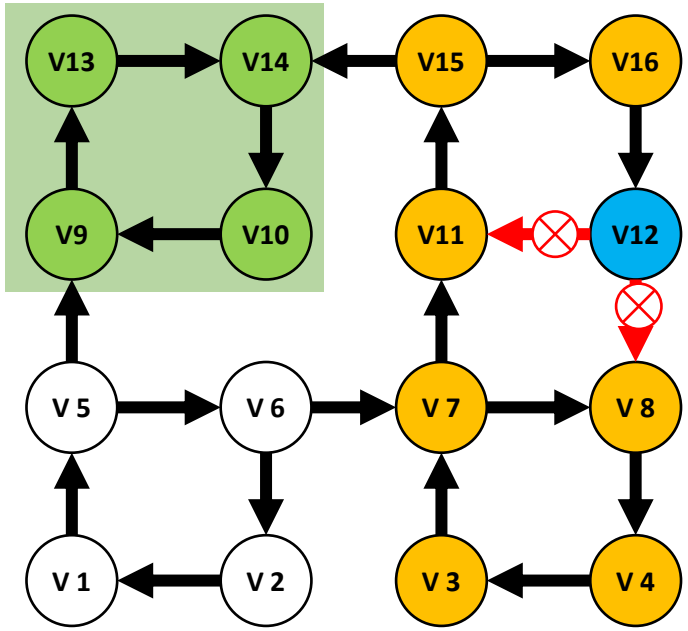
DFS Stack

V8
V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

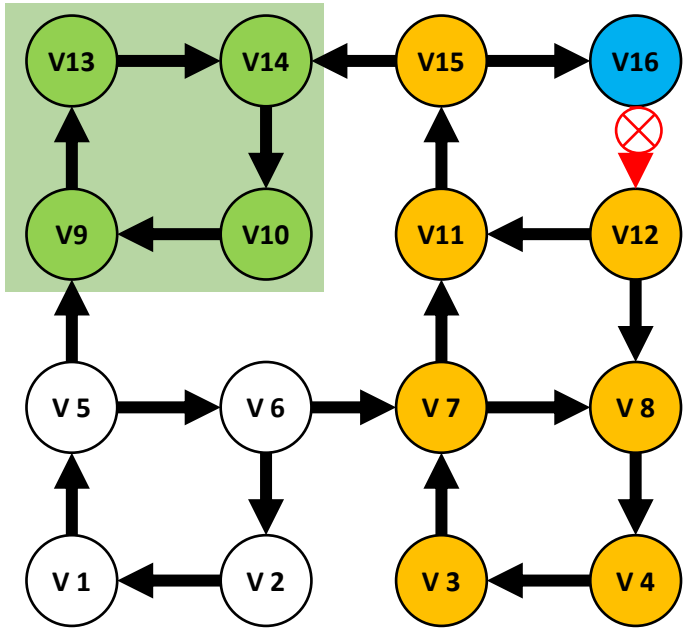
DFS Stack

V12
V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

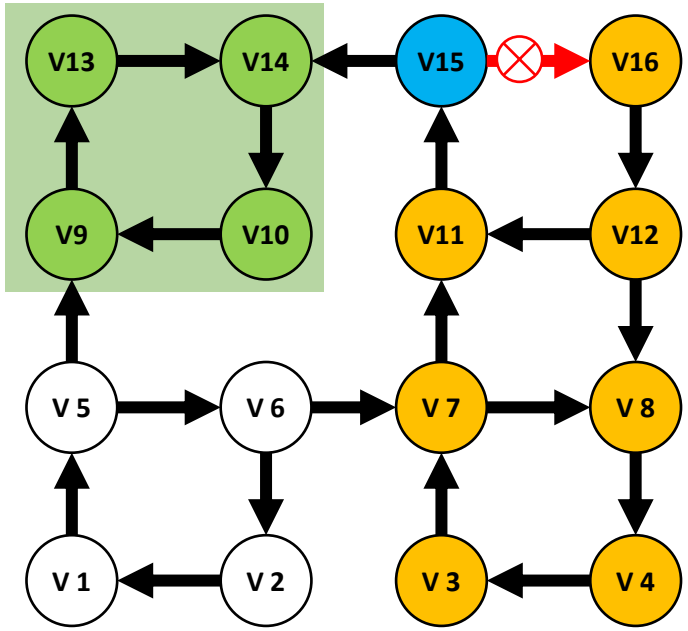
DFS Stack

V16
V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

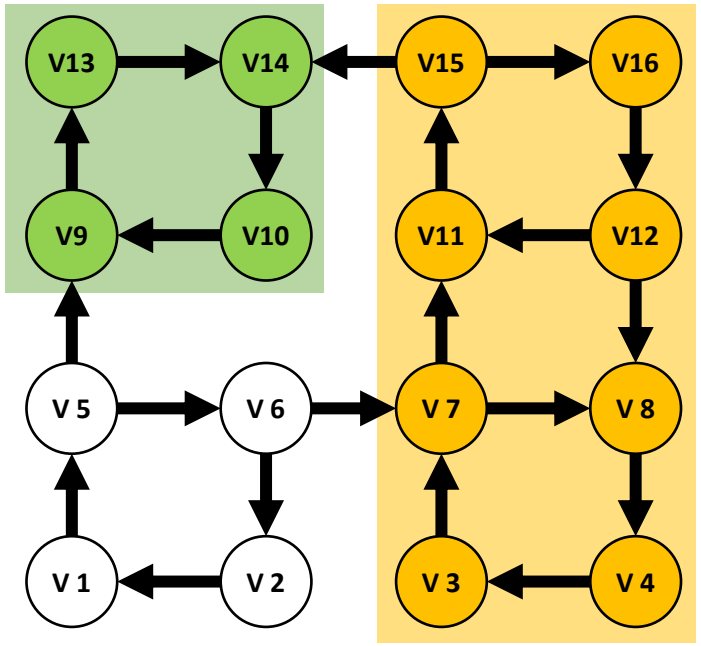
DFS Stack

V15

Course stack

V15
V11
V7
V3
V4
V8
V12
V16
V5
V1
V2
V6

DFS transposed from G^t

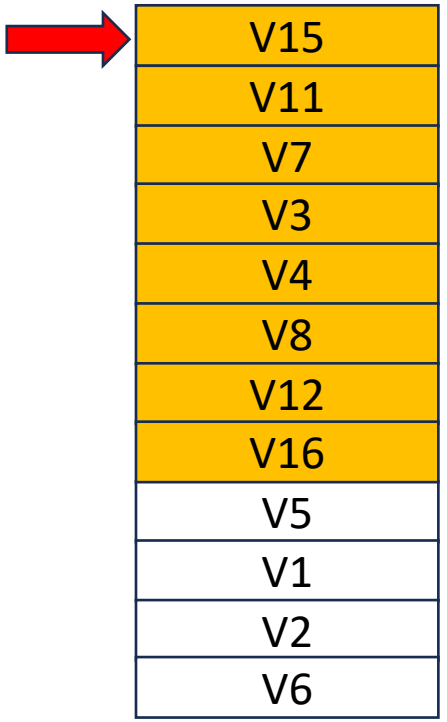


V13,V14,V10,V9

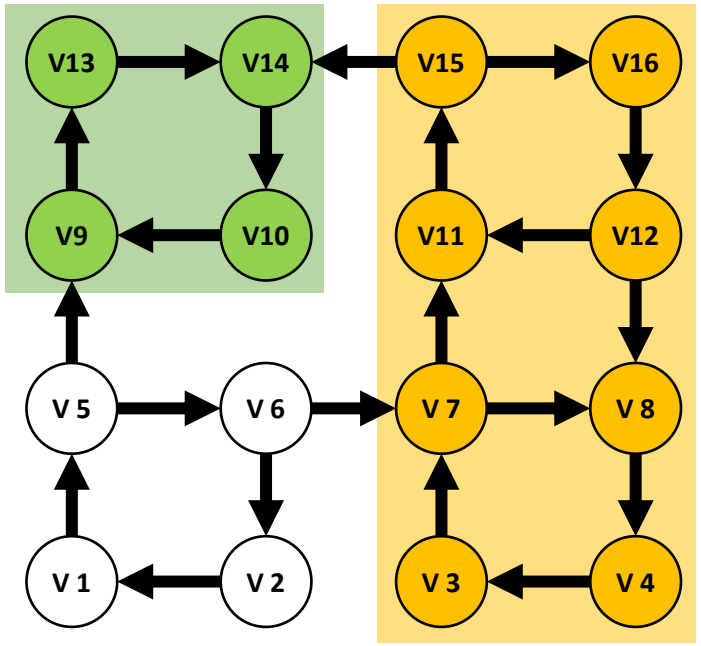
V15,V16,V12,V11,V8,V4,V3,V7

DFS Stack

Course stack



DFS transposed from G^t

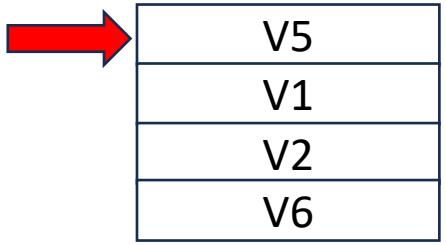


DFS Stack

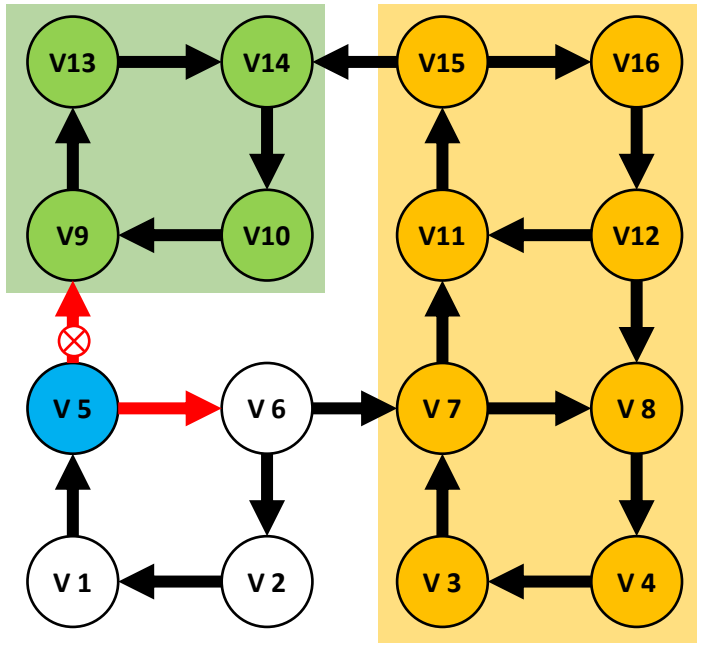
Course stack

V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7



DFS transposed from G^t



DFS Stack

V5

Course stack

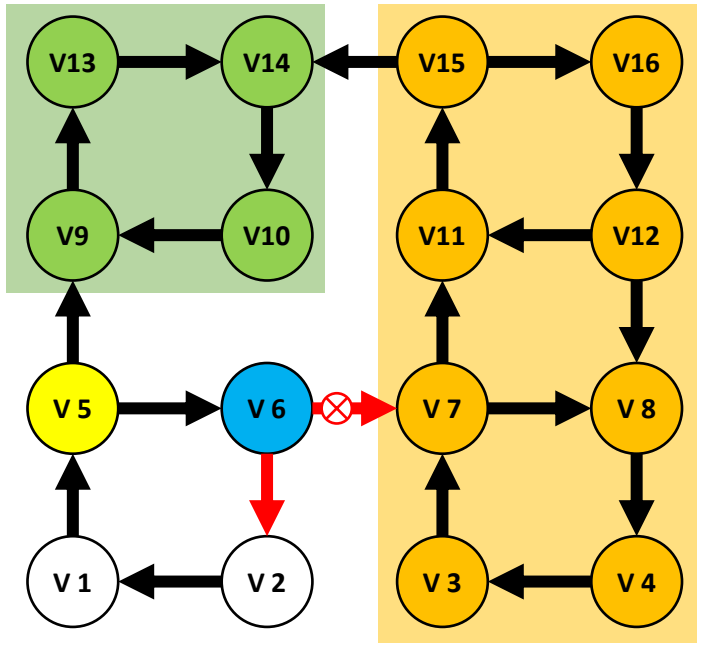
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5

V5
V1
V2
V6

DFS transposed from G ^{t}



DFS Stack

V6
V5

Course stack

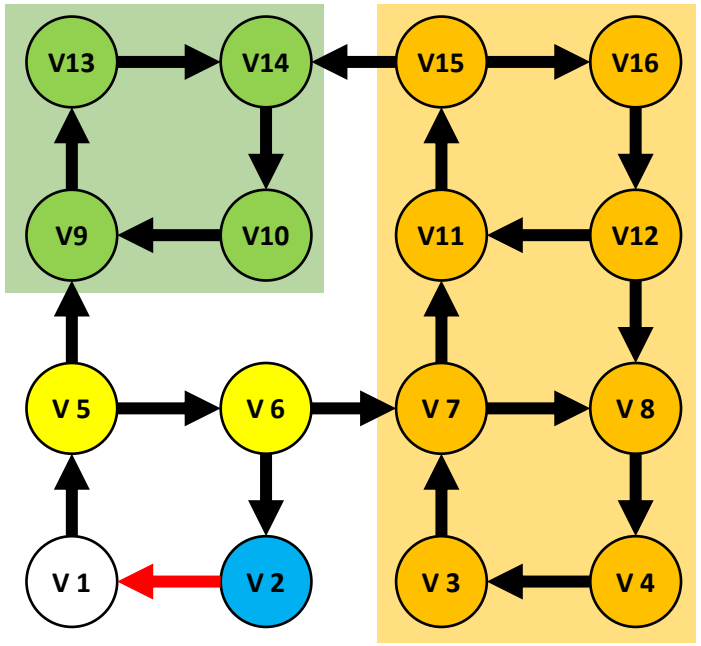
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6

V5
V1
V2
V6

DFS transposed from G ^{t}



DFS Stack

V2
V6
V5

Course stack

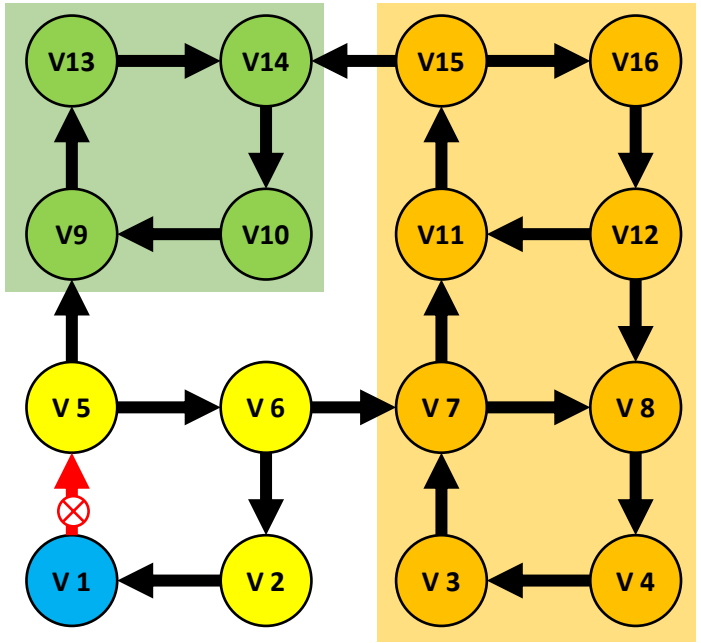
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2

V5
V1
V2
V6

DFS transposed from G ^{t}



DFS Stack
V1
V2
V6
V5

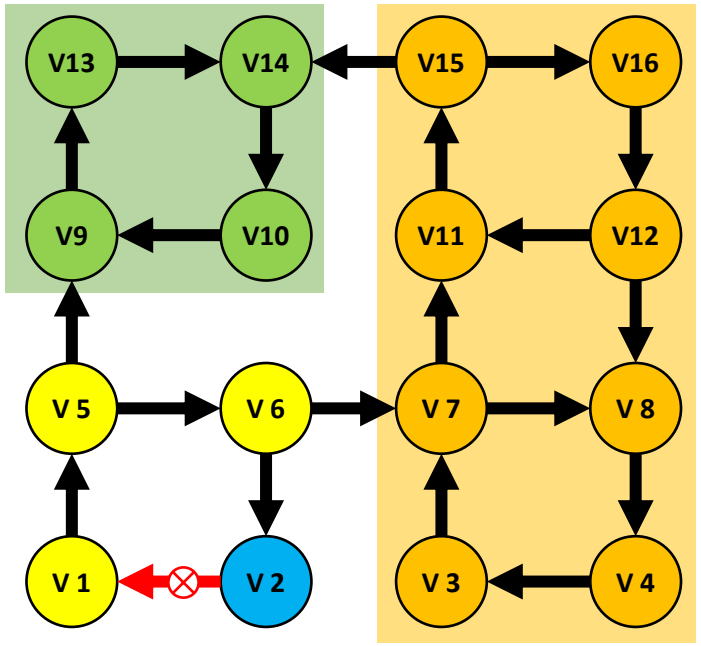
Course stack
V5
V1
V2
V6

V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1

DFS transposed from G ^{t}



V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1

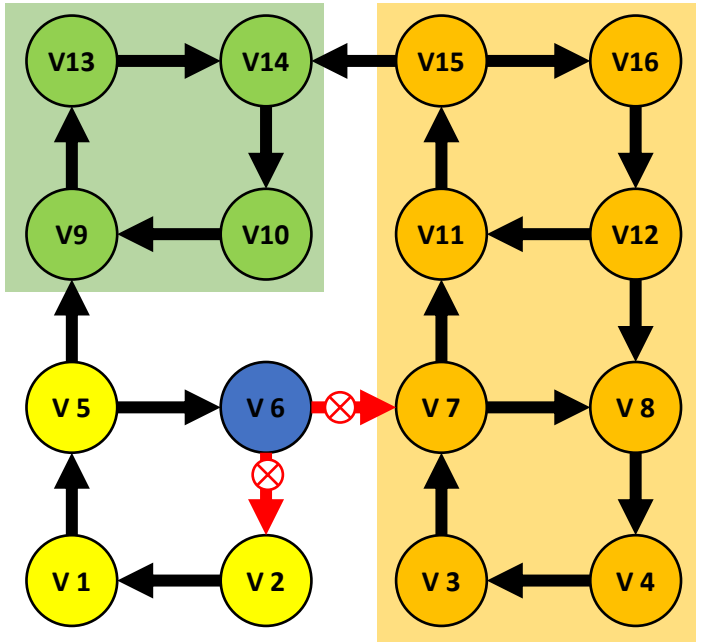
DFS Stack

V2
V6
V5

Course stack

V5
V1
V2
V6

DFS transposed from G^t



DFS Stack

V6
V5

Course stack

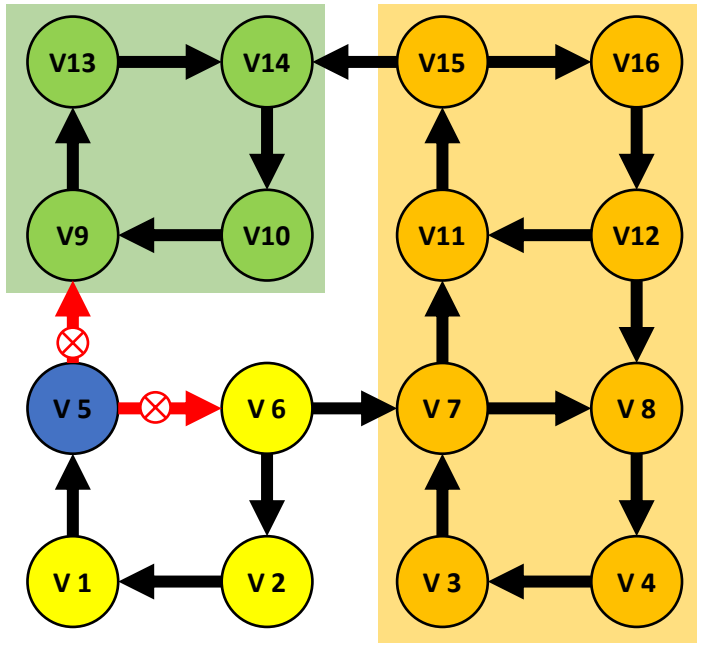
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1

V5
V1
V2
V6

DFS transposed from G ^{t}



DFS Stack

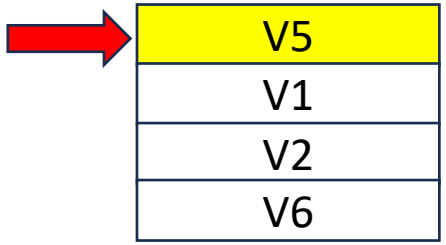
V5

Course stack

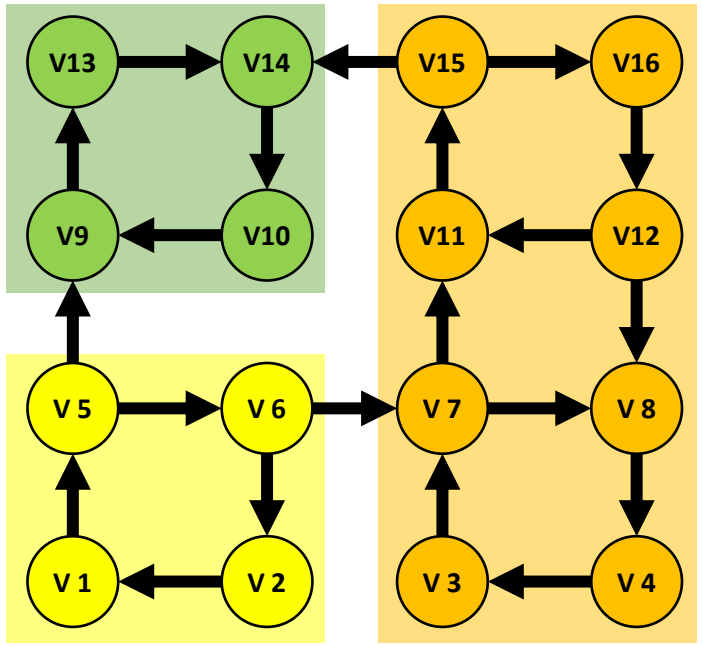
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1



DFS transposed from G ^{t}



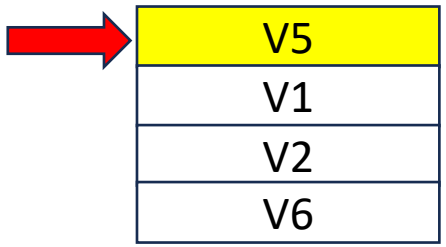
DFS Stack

Course stack

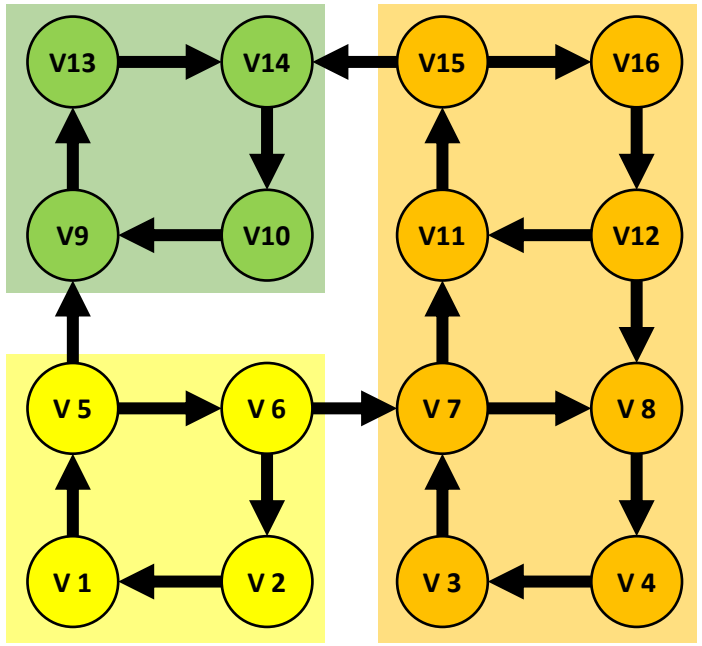
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1



DFS transposed from G ^{t}



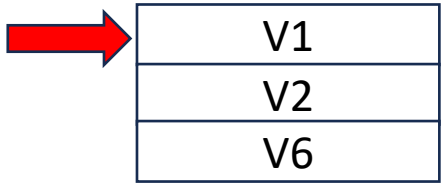
DFS Stack

Course stack

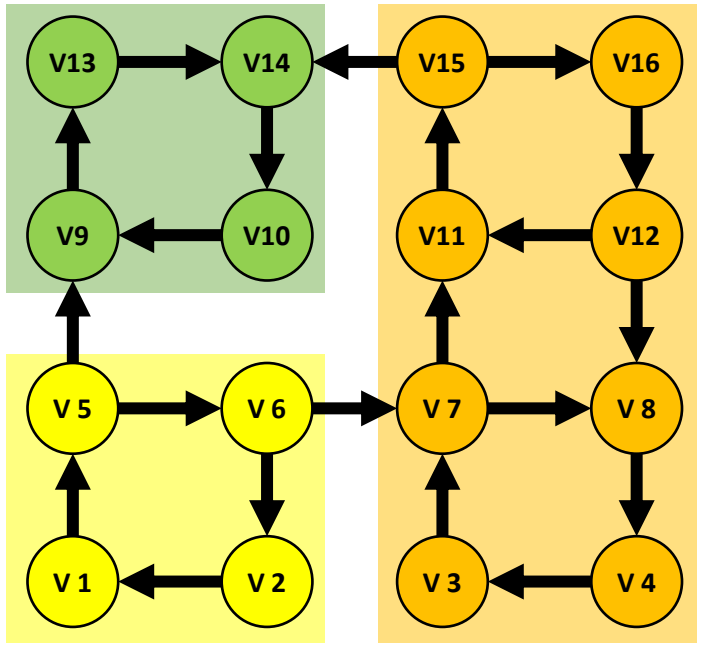
V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1



DFS transposed from G^t



DFS Stack

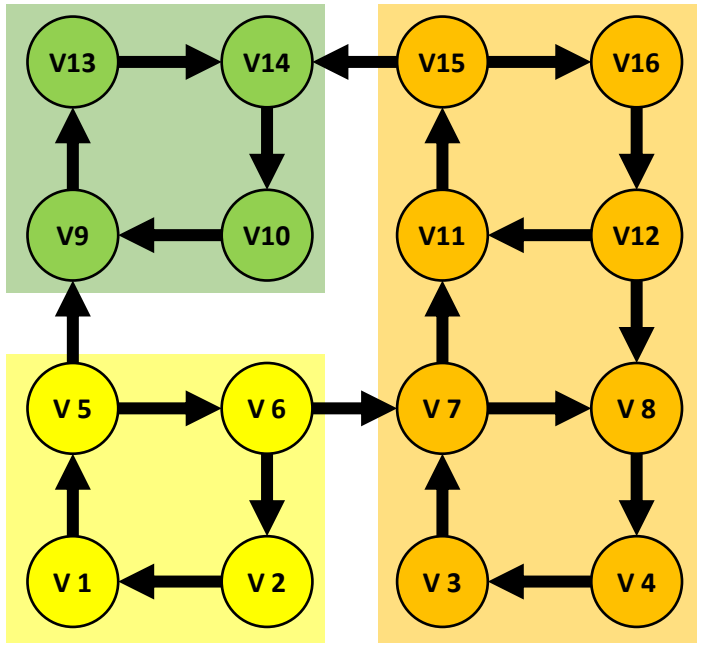
Course stack

V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1

DFS transposed from G^t



DFS Stack

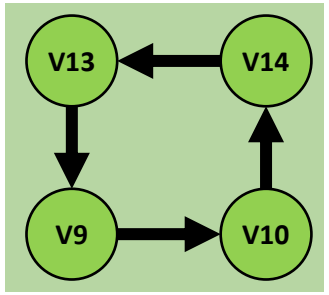
Course stack

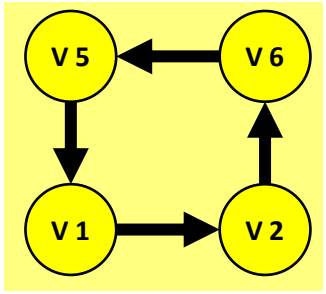
V13,V14,V10,V9

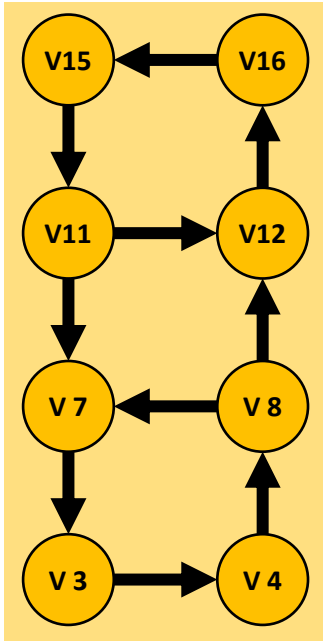
V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1

V6







4 Strongly connected component

V13,V14,V10,V9

V15,V16,V12,V11,V8,V4,V3,V7

V5,V6,V2,V1

Exercise 2

Redo the example from the previous slide by applying DFS first on G^T (the transposed graph) and then on G in the second step.

Exercise 3

Using Kosaraju's algorithm, find the strongly connected components of the following graphs.