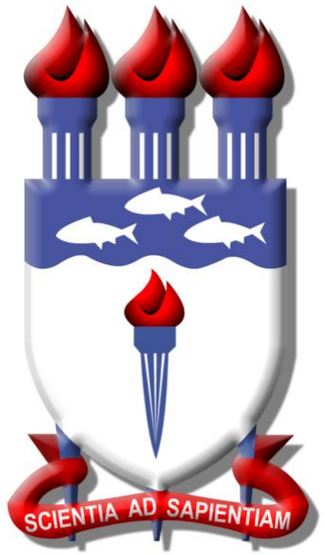
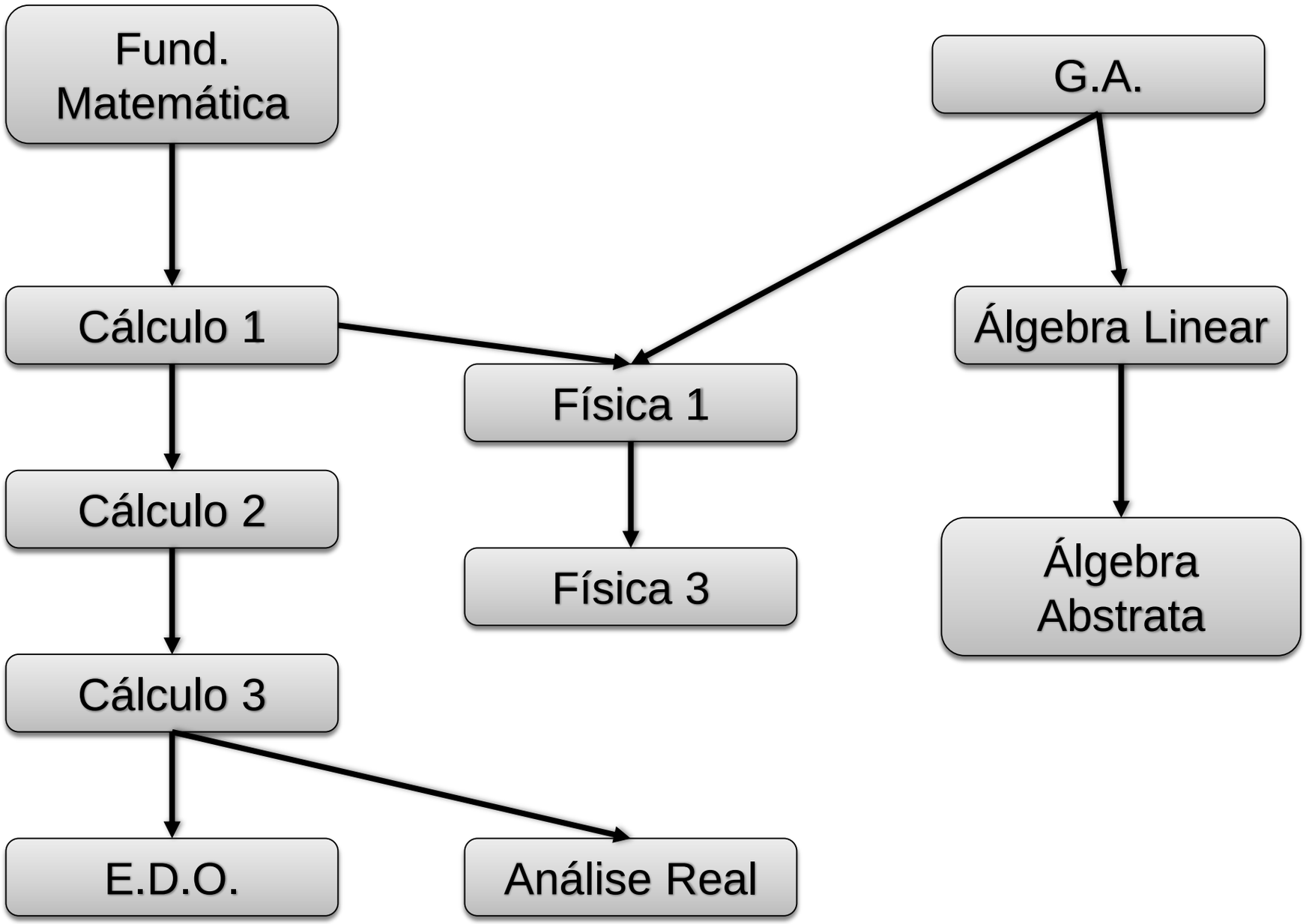


Topological Sort

John Victor – Mateus Pereira – Waddinsohn Franklin

https://github.com/ProjetoP2Huffman

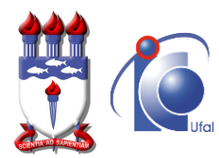
Pré-Requisitos de Disciplinas



E se quisermos saber uma ordem geral das dependências?

Quais disciplinas preciso cursar antes de cursar uma disciplina x?

Topological Sort



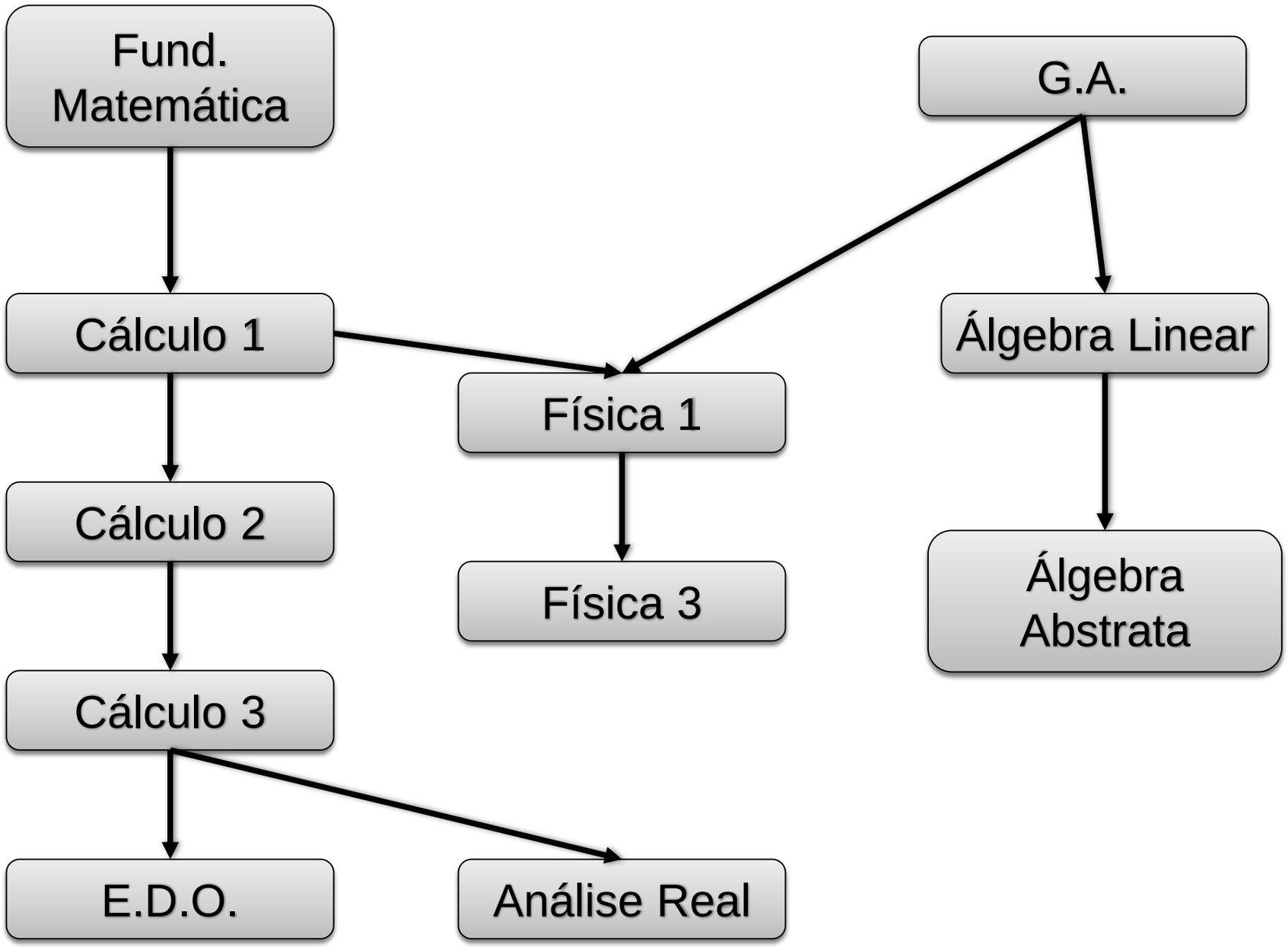
Topological Sort

- O Topological Sort é um método de ordenação linear aplicada em DAG’s.

- Ordena os nós do DAG de maneira que os nós pais vêm antes de todos os seus nós descendentes.

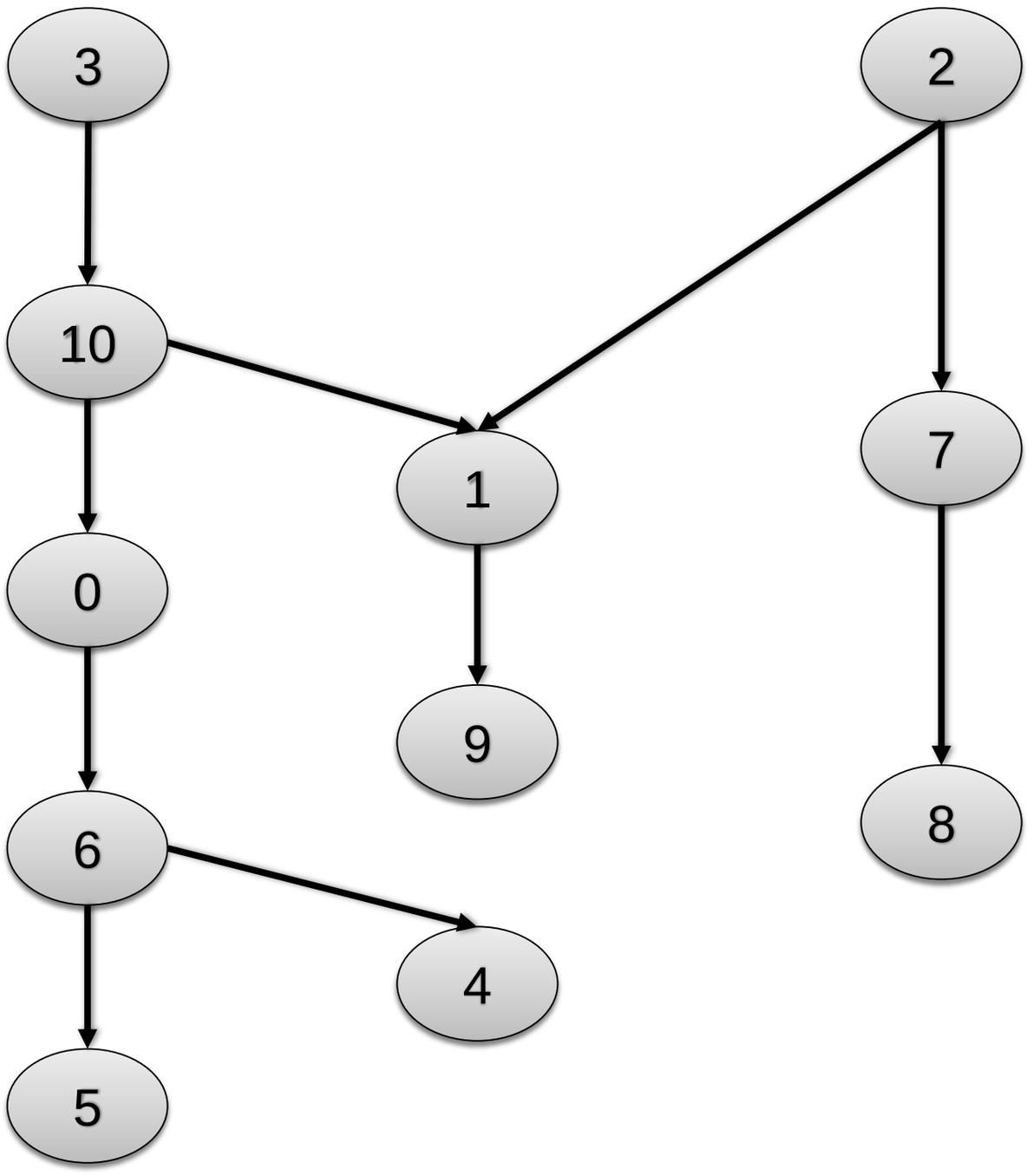
- Usaremos o TS baseado em dfs.

Topological Sort



0 – Cálculo 2
1 – Física 1
2 – G.A.
3 – Fund. Matemática
4 – Análise Real
5 – E.D.O.
6 – Cálculo 3
7 – Álgebra Linear
8 – Álgebra Abstrata
9 – Física 3
10 – Cálculo 1

Topological Sort



Definições

- **Nó**: elemento que contém uma informação e pode ou não ter relação com outros nós do grafo.
- **Aresta**: representa a relação entre dois nós. Liga um nó a outro(os).
- **Caminho**: conjunto de arestas a serem percorridas para partindo de um vértice v chegar em um outro vértice u.
- **Grafo Acíclico**: grafo onde partindo de um vértice v qualquer, não existe nenhum caminho que passe atinja v novamente.
- **Digrafo ou Grafo Dirigido**: grafo cujas arestas têm orientação.

Implementação

TAD

Node

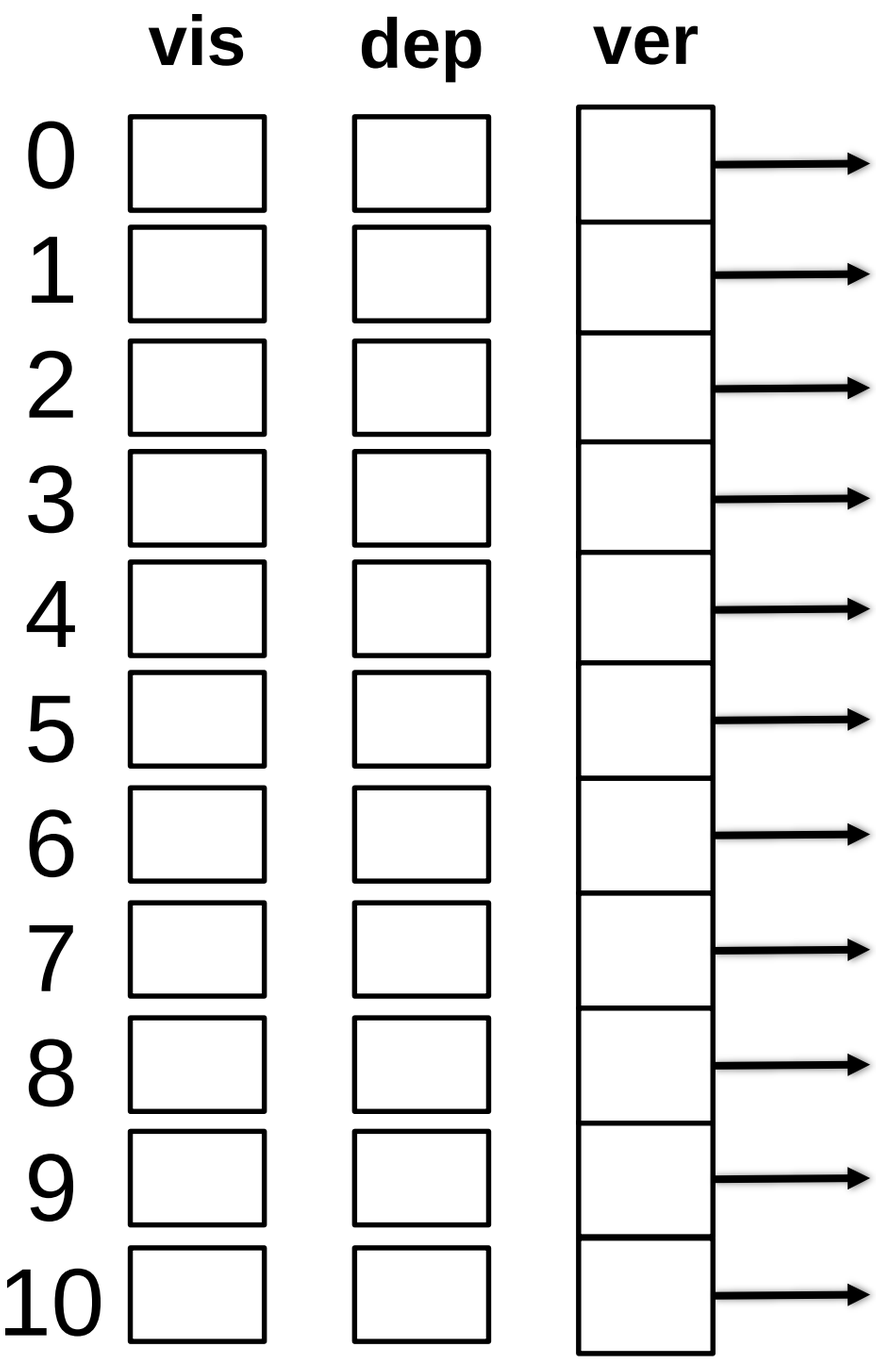
```
struct node {

    int item;
    node* next;
};
```

Stack

```
struct stack {

    int nodes_number;
    node *head;
};
```

Graph

```
struct graph {

    int *depending;
    int *visited;
    node **vertex;
};
```



Stack

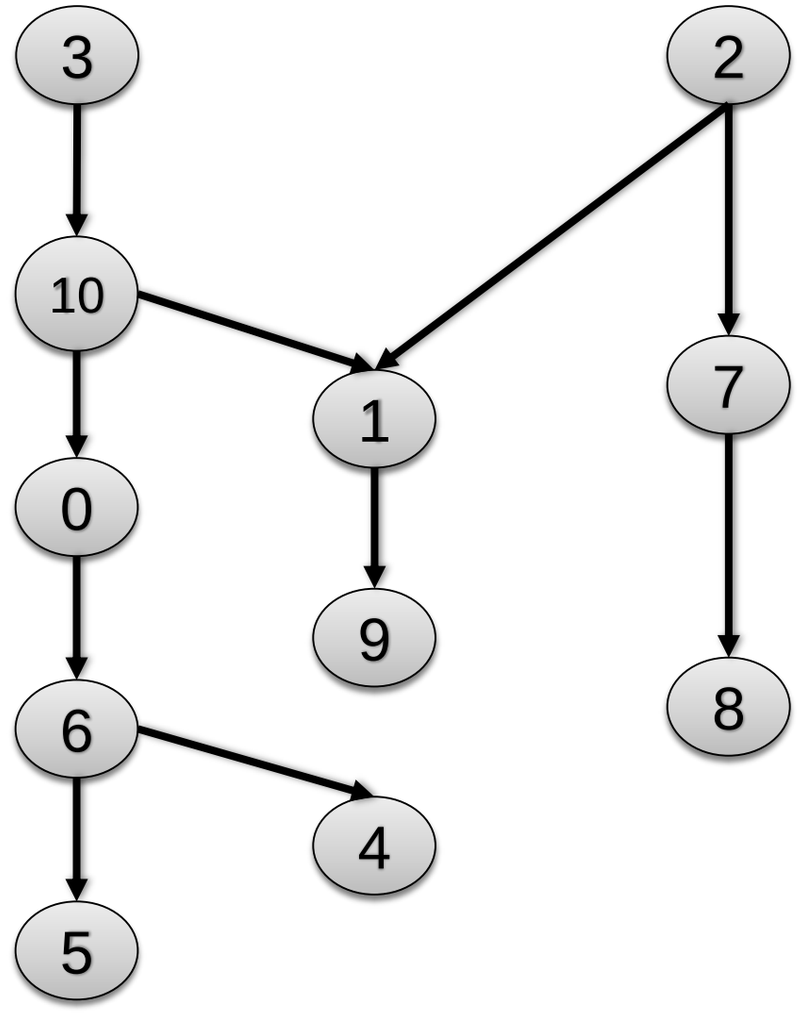
```c
stack* create_empty_stack();

void push(stack *stack, int item);

void print_stack(stack *stack);

int empty_stack(stack *stack);
```

Graph

```c
graph* create_empty_graph(int vertex_number);

void add_vertex(graph *graph, int vertex1, int vertex2);

void bfs(graph *graph, stack *stack, int initial_vertex);
```

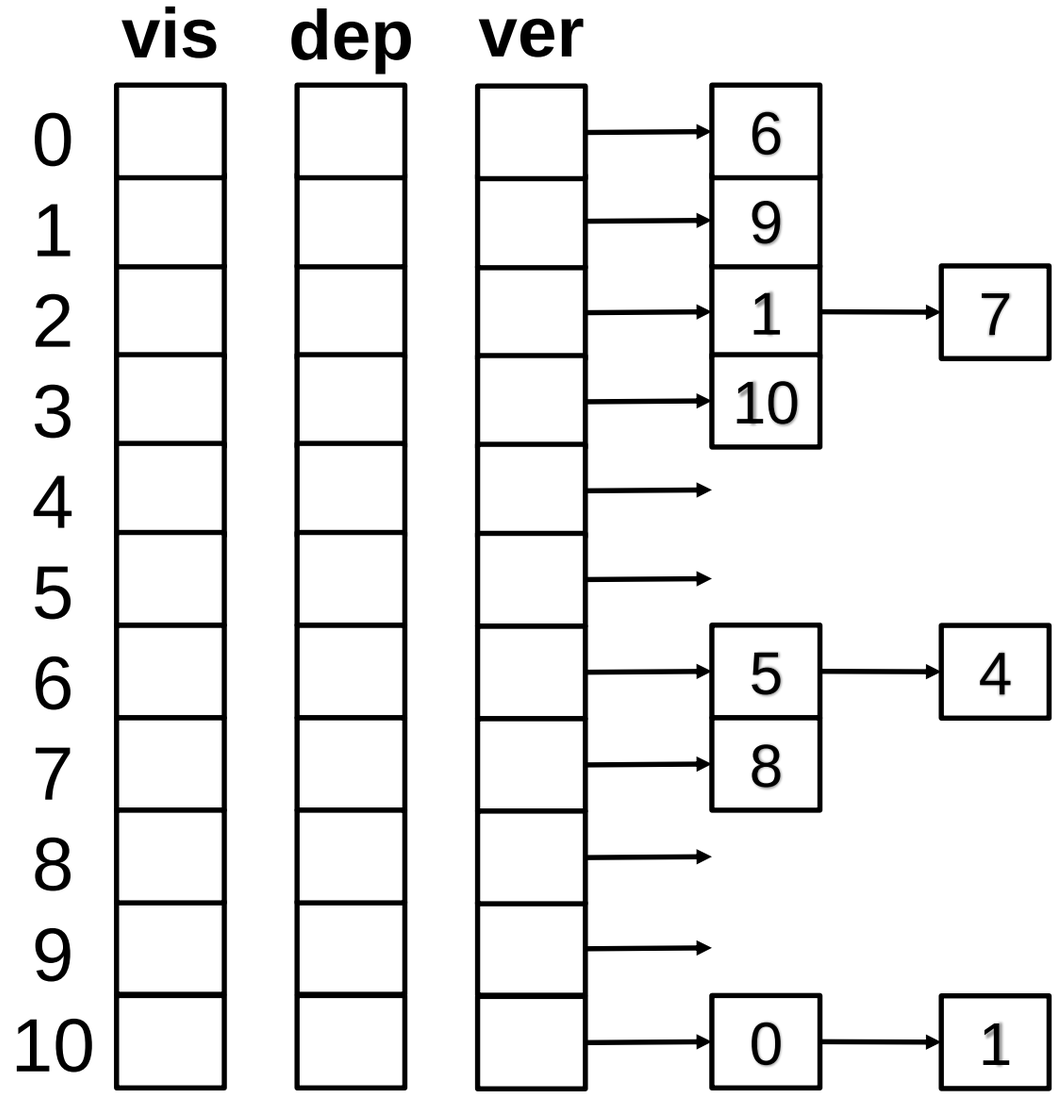

Main

```c
int main () {

    int vertex_number = 11;
    int i;

    graph *G = create_empty_graph(vertex_number);
    stack *S = create_empty_stack();

    add_vertex(G, 3, 10);
    add_vertex(G, 10, 0);
    add_vertex(G, 10, 1);
    add_vertex(G, 0, 6);
    add_vertex(G, 6, 5);
    add_vertex(G, 6, 4);
    add_vertex(G, 1, 9);
    add_vertex(G, 2, 1);
    add_vertex(G, 2, 7);
    add_vertex(G, 7, 8);

    for (i = 0 ; i < vertex_number ; i++) {

        if (G->depending[i] == 0) {

            bfs(G, S, i);
        }
    }

    print_stack(S);

    return 0;
}
```

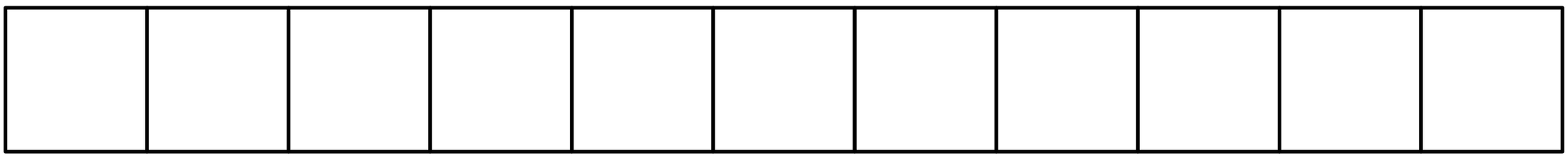


bfs()

```c
void bfs(graph *graph, stack *stack, int initial_vertex) {

    if (graph->visited[initial_vertex] == 0) {

        graph->visited[initial_vertex] = 1;
        node *current = graph->vertex[initial_vertex];

        while (current != NULL) {

            if (graph->visited[current->item] != 1) {

                bfs(graph, stack, current->item);
            }
            current = current->next;
        }

        push(stack, initial_vertex);
    }

}
```



Stack



Animação

Topological Sort

Solução

- Com a ordenação topológica, sabemos quais disciplinas devem ser cursadas antes de outras ou, por outro lado, dada uma determinada disciplina, quais são pré-requisitos para esta.

- Não é a única solução

- Só é única se o grafo é hamiltoniano.