DEPTH-FIRST SEARCH

- ALGORITHM DESCRIPTION
- ALGORITHM EXAMPLE
- ALGORITHM COMPLEXITY
- ALGORITHM APPLICATIONS
- PRACTICE: MAXIMUM STREAM PROBLEM

Author: prof. Yevhenii Borodavka

ALGORITHM DESCRIPTION

Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph.

A standard DFS implementation puts each vertex of the graph into one of two categories:

- Visited
- Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

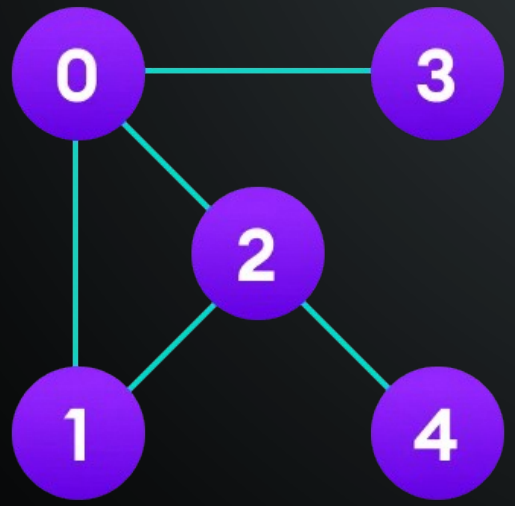
ALGORITHM DESCRIPTION

The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
4. Keep repeating steps 2 and 3 until the stack is empty.

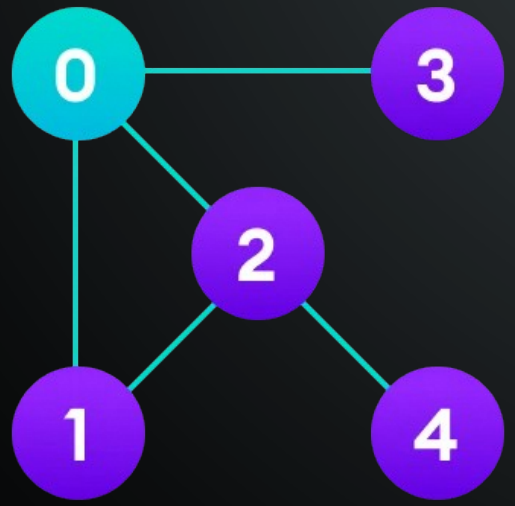
ALGORITHM EXAMPLE

Let's see how the Depth First Search algorithm works with an example. We use an undirected graph with 5 vertices.



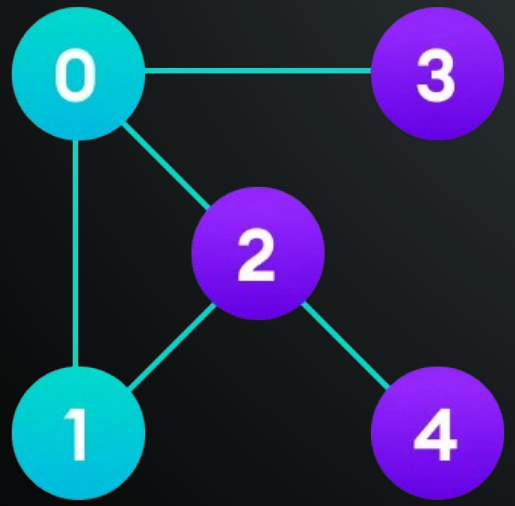
ALGORITHM EXAMPLE

We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.



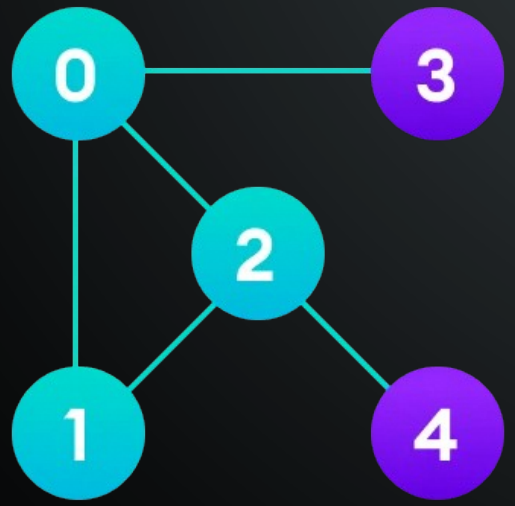
ALGORITHM EXAMPLE

Next, we visit the element at the top of stack i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.



ALGORITHM EXAMPLE

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.



0	1	2		

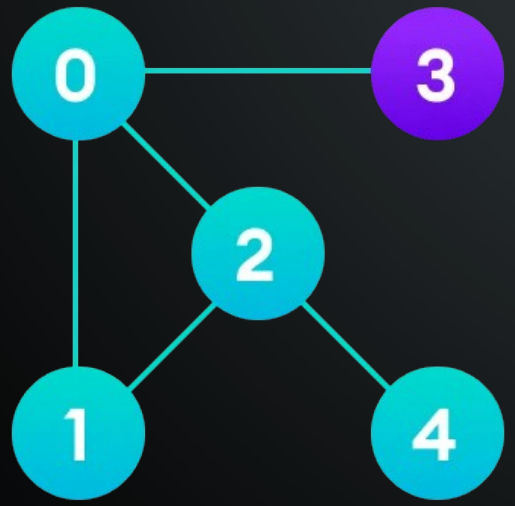
Visited

4	3			

Stack

ALGORITHM EXAMPLE

Vertex 4 has no unvisited adjacent vertex, so we have nothing to add to the top of the stack.



0	1	2	4	

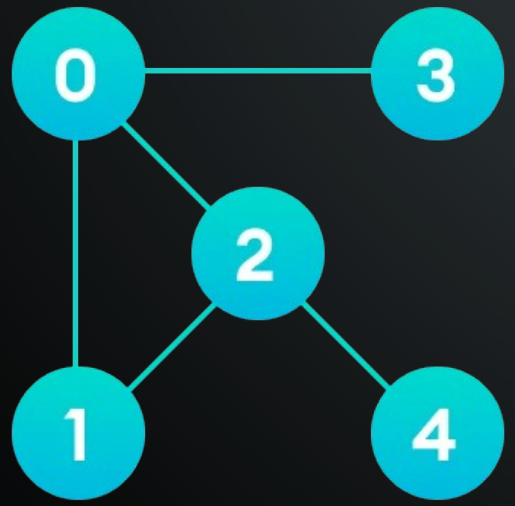
Visited

3				

Stack

ALGORITHM EXAMPLE

After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.



0	1	2	4	3

Visited

Stack

ALGORITHM COMPLEXITY

The time complexity of the DFS algorithm is represented in the form of **O**(**V** + **E**), where **V** is the number of nodes and **E** is the number of edges.

The space complexity of the algorithm is **O**(**V**).

The algorithm pseudocode:

```
DFS(G, u)
    u.visited = true
    for each v ∈ G.Adj[u]
        if v.visited == false
            DFS(G,v)
```

ALGORITHM APPLICATIONS

1. For finding the path
2. To test if the graph is bipartite
3. For finding the strongly connected components of a graph
4. For detecting cycles in a graph
5. Topological sorting
6. Solving puzzles with only one solution, such as mazes

PRACTICE: MAXIMUM STREAM PROBLEM

Problem. You have **N** network nodes (1<=**N**<=100) and **K** connections (1<=**K**<=1000) between the nodes with transfer speed **S** (1<=**S**<=1000). You need to compute the maximum transfer speed between nodes **A** and **B**.

Task. Create a program using C/C++/Python to solve this problem.

Input. **K**+1 strings: the first one contains 4 numbers **N**, **K**, **A**, and **B** divided by spaces; each of the next **K** strings contains 3 numbers – indexes of connected nodes and transfer speed divided by spaces.

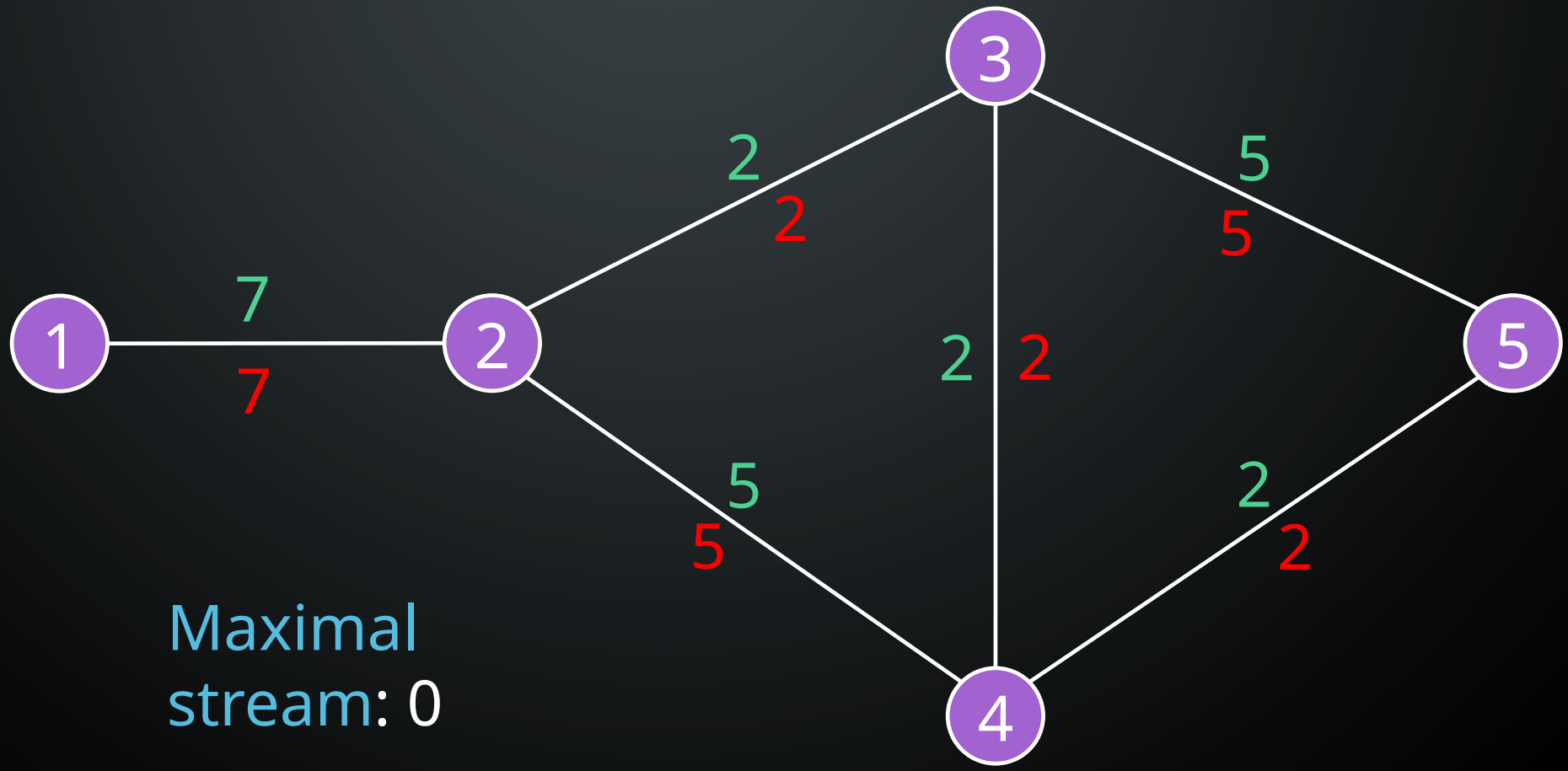
Output. Maximal transfer speed between the node **A** and the node **B**.

PRACTICE: MAXIMUM STREAM PROBLEM

Example. We have **N**=5 and **K**=6, **A**=1 and **B**=5.

Input:

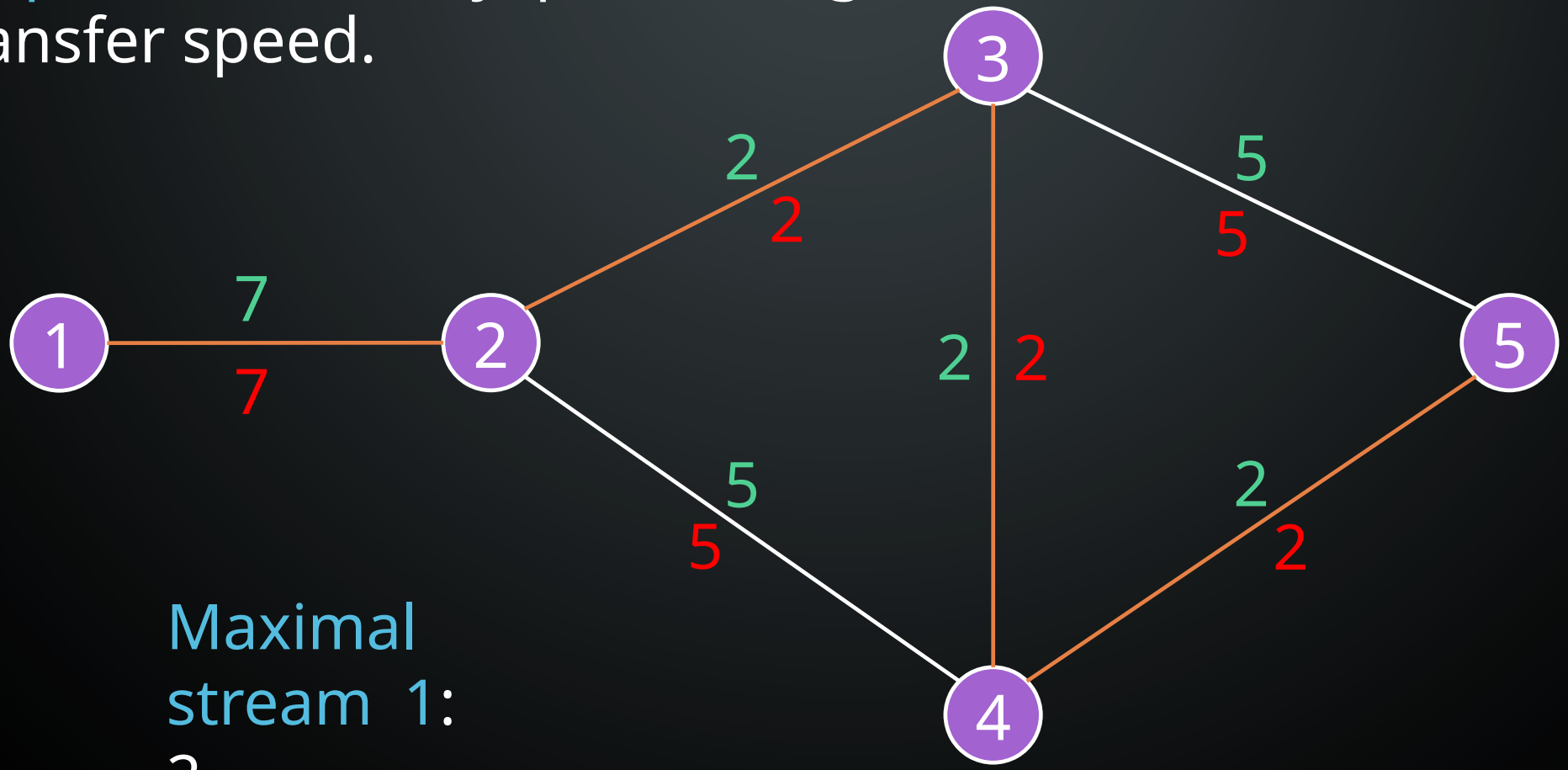
```
5 6 1 5
1 2 7
2 3 2
2 4 5
3 4 2
3 5 5
4 5 2
```



Maximal stream: 0

PRACTICE: MAXIMUM STREAM PROBLEM

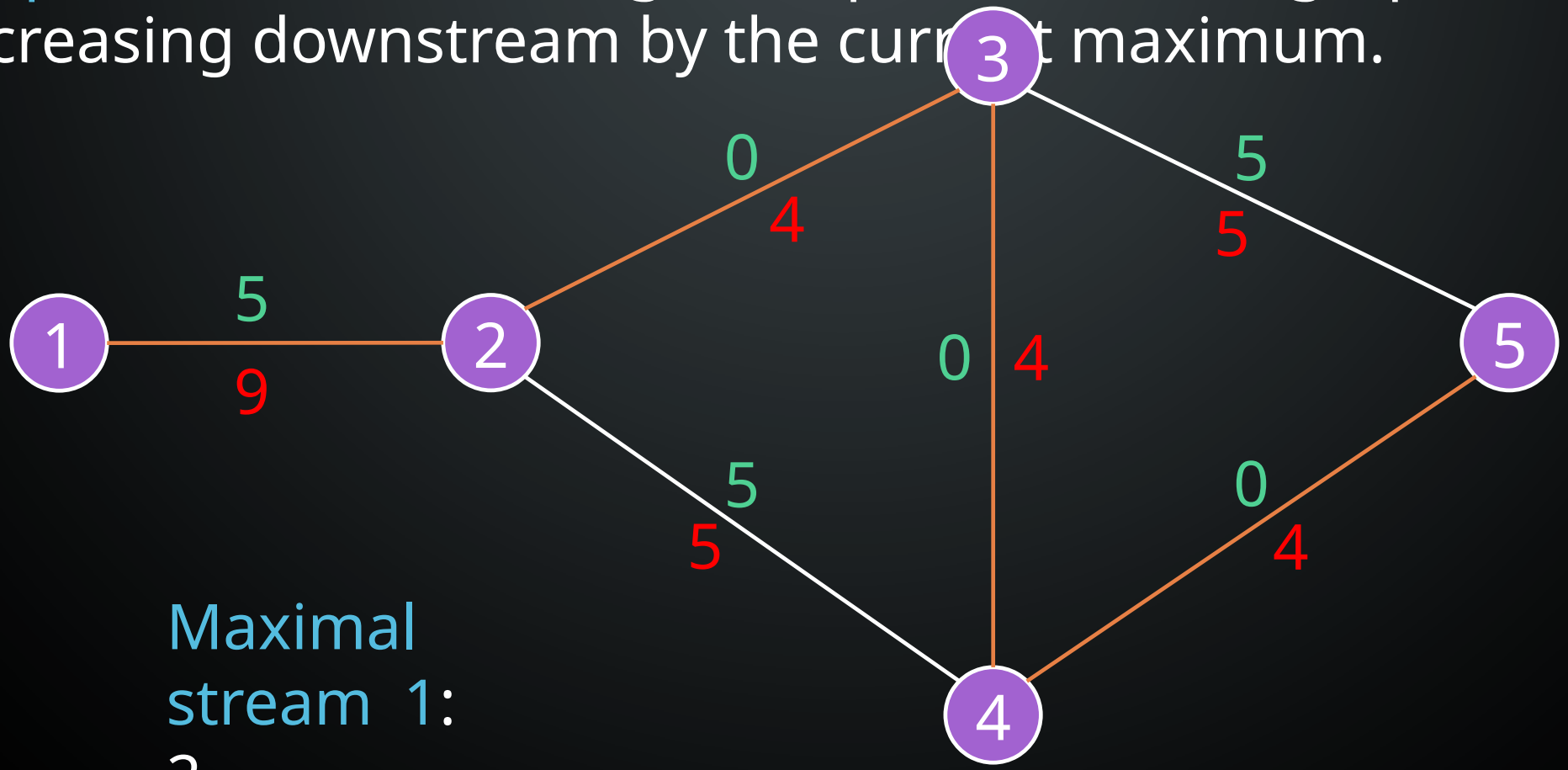
Step 1. Found any path using DFS and count the maximal transfer speed.



Maximal stream 1:
2

PRACTICE: MAXIMUM STREAM PROBLEM

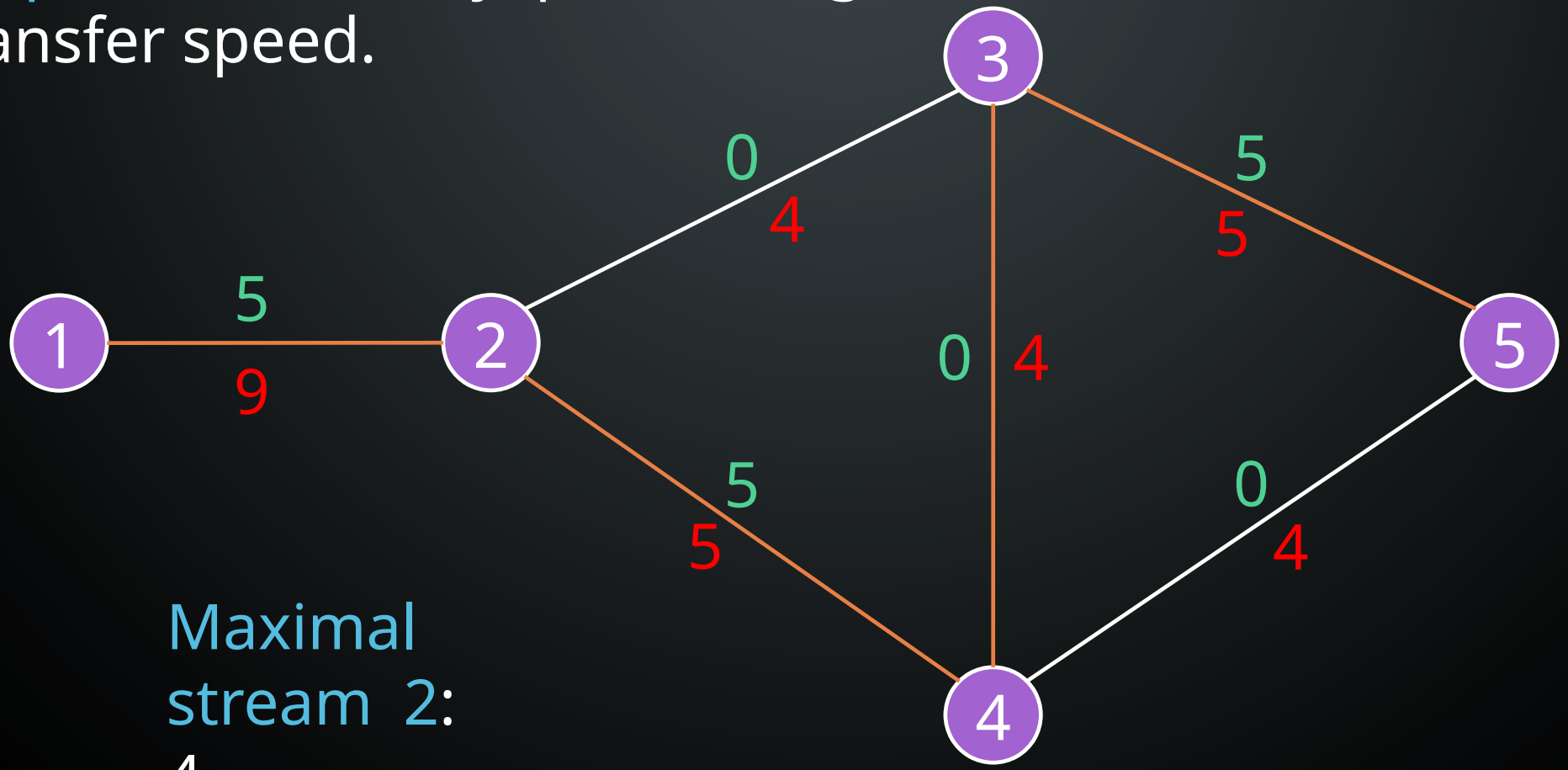
Step 2. Move back trough the path decreasing upstream and increasing downstream by the current maximum.



Maximal stream 1:
2

PRACTICE: MAXIMAUM STREAM PROBLEM

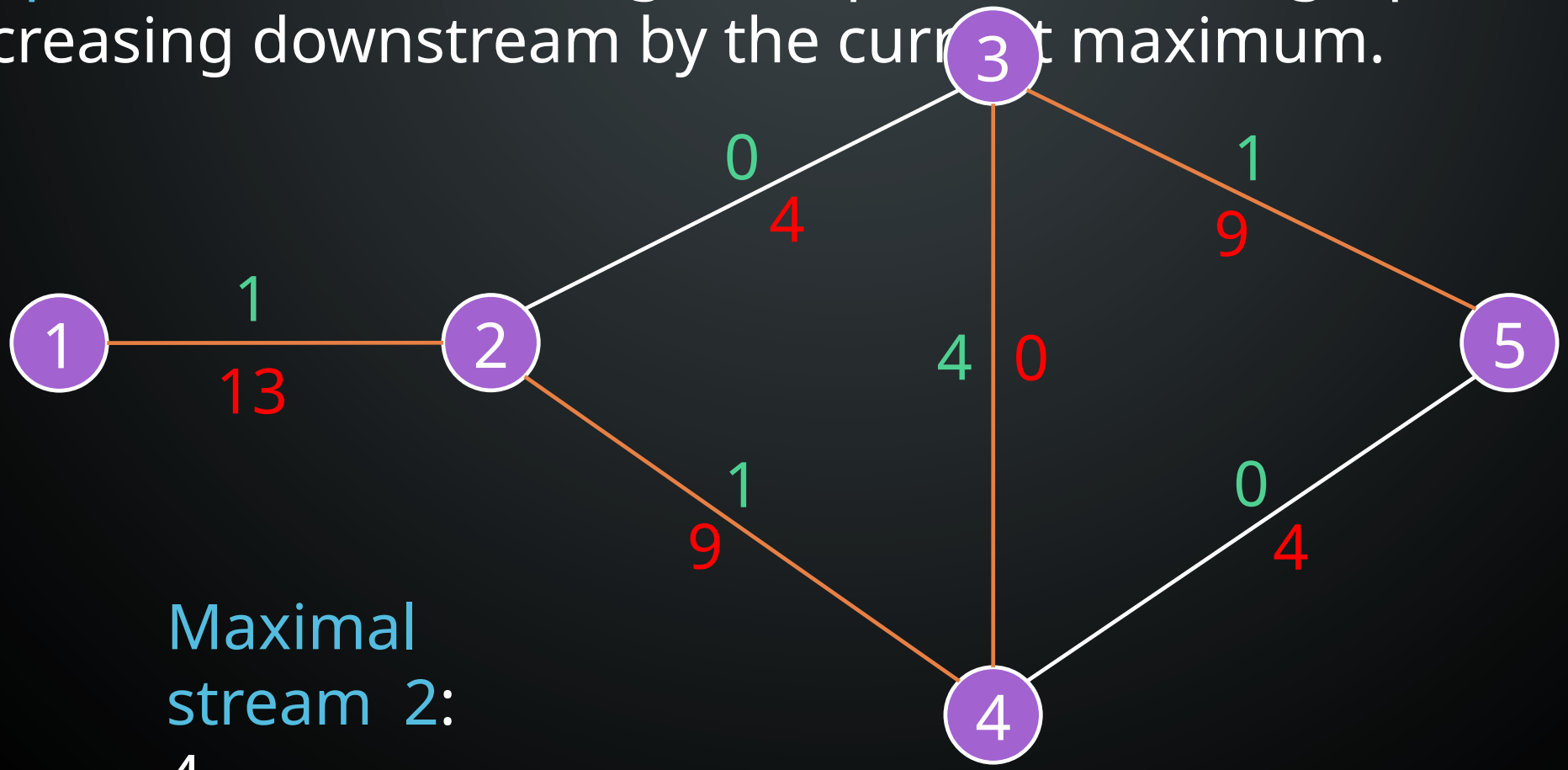
Step 3. Found any path using DFS and count the maximal transfer speed.



Maximal stream 2:
4

PRACTICE: MAXIMUM STREAM PROBLEM

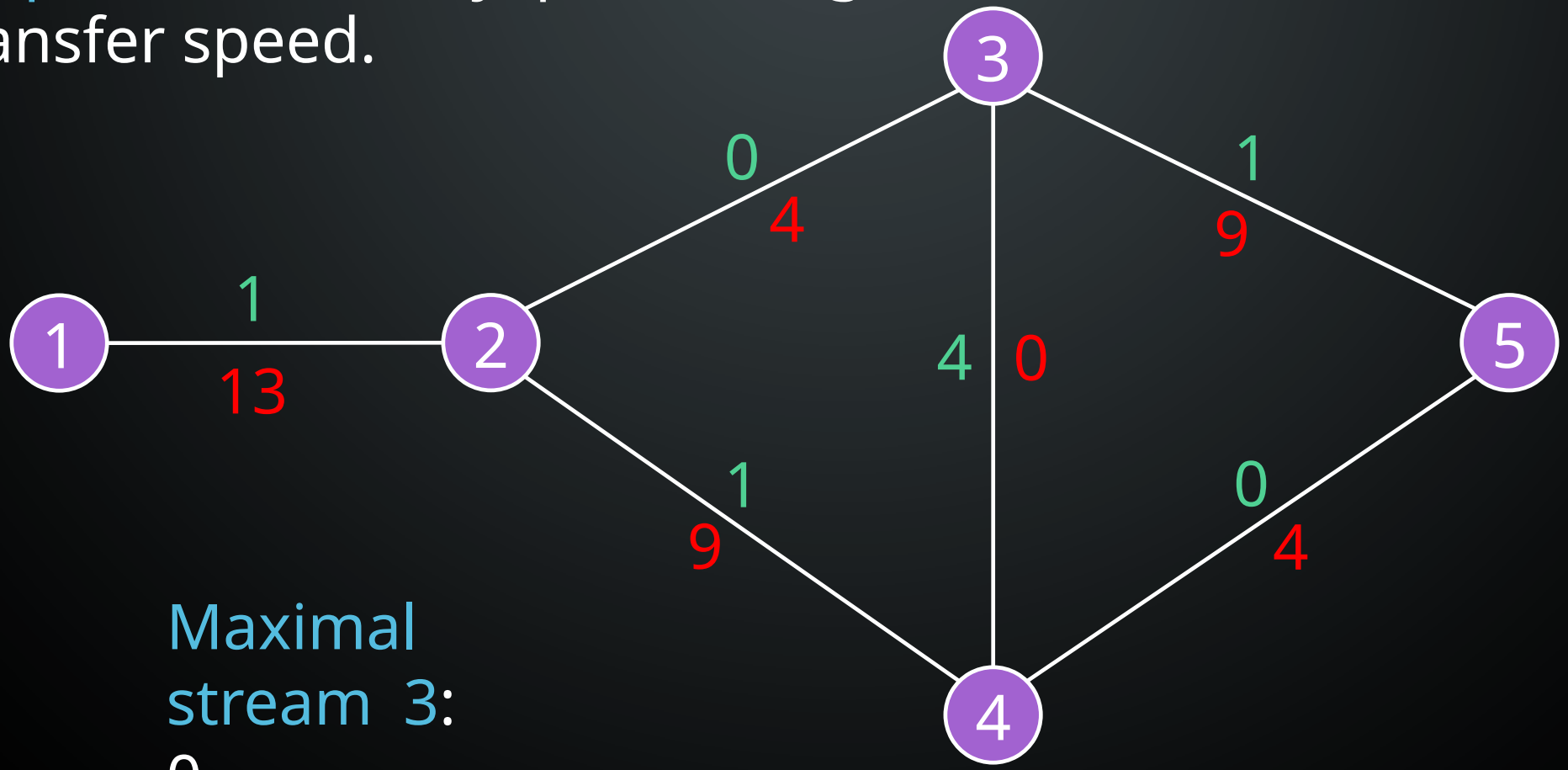
Step 4. Move back trough the path decreasing upstream and increasing downstream by the current maximum.



Maximal stream 2:
4

PRACTICE: MAXIMUM STREAM PROBLEM

Step 5. Found any path using DFS and count the maximal transfer speed.



Maximal stream 3:
0

PRACTICE: MAXIMUM STREAM PROBLEM

As we can’t find any path from the start node to the finish node the algorithm stops. The maximal stream will be sum of all maximal streams at each forward step. For this example it is **2+4=6**.

PRACTICE: MAXIMUM STREAM PROBLEM

Code example:

```
#include <iostream>
#define MAX 101
int N, K, A, B, R, P[MAX], V[MAX], M[MAX][MAX];
int main()
{
    cin >> N >> K >> A >> B;
    int f, s, w;
    for(int i = 0; i < K; i++)
    {
        cin >> f >> s >> w;
        M[f][s] = w;
        M[s][f] = w;
    }
    // Your code here
    cout << R;
    return 0;
}
```

THANK YOU!