

Алгоритмы на графах

Лекция 3. Поиск в глубину.

Адигеев Михаил Георгиевич

2024

План лекции

1. Обход в глубину.
2. Применение обхода в глубину для решения задач на графах.
 - ✓ Обнаружение циклов
 - ✓ Определение двудольности графа

Обход в глубину

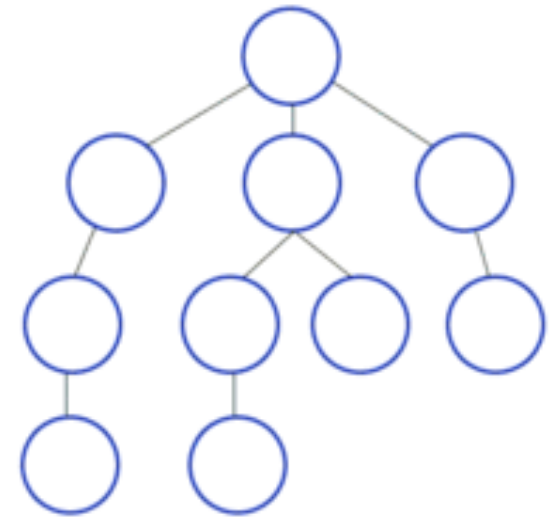
Общие принципы обхода

- Мы должны *обойти* все вершины графа в *определённом* порядке и *обработать* их.
- В процессе обхода для каждой вершины мы в какой-то момент посещаем её в первый раз, потом можем несколько раз возвращаться в неё, и в какой-то момент (обычно при первом или при последнем посещении) мы её обрабатываем.
- Для того чтобы корректно обработать вершину (а также инцидентные ей рёбра и смежные вершины), мы должны отслеживать *статус* вершины:
 - 1) Непосещённая
 - 2) Посещённая
 - 3) Обработанная.

Обход в глубину

Принцип обхода в глубину:

посетив вершину v ,
последовательно
выполняем *обход в
глубину* для каждого ещё
не посещённого соседа.



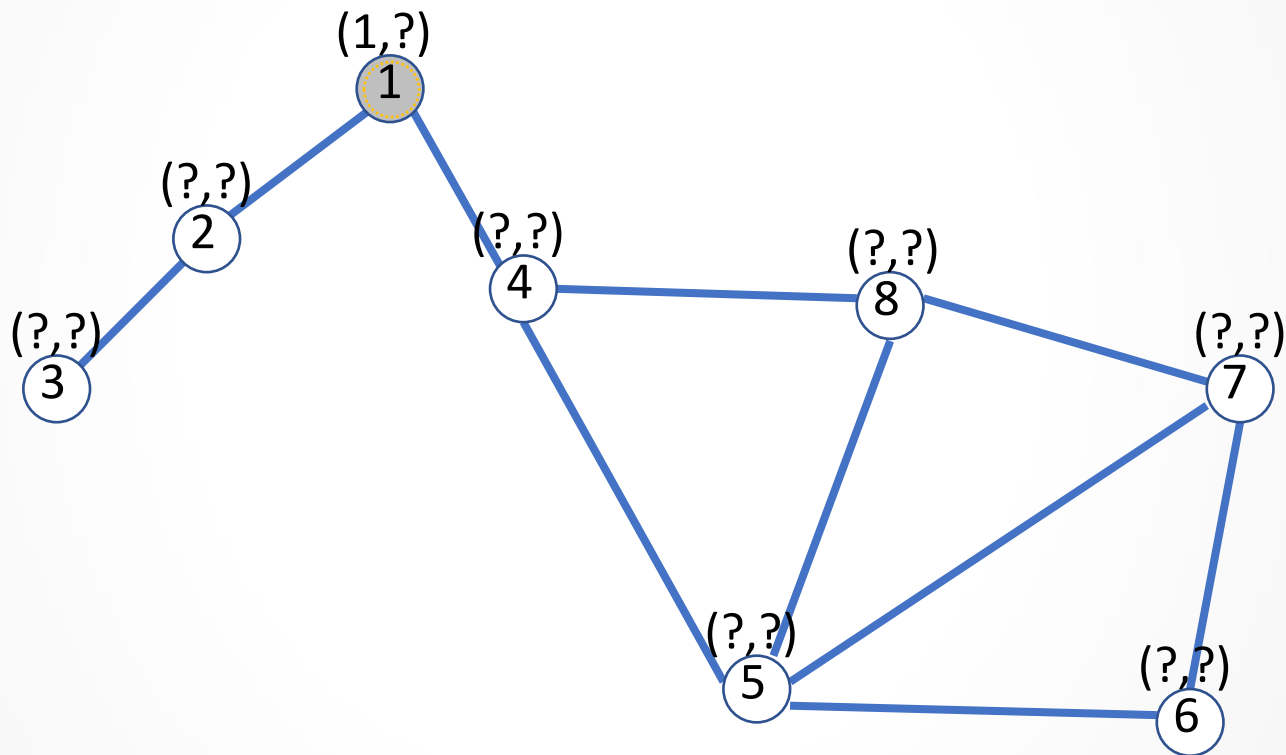
https://en.wikipedia.org/wiki/Depth-first_search

Для того чтобы запоминать порядок посещения вершин нам понадобится структура данных «Стек» (stack) – явный или неявный.

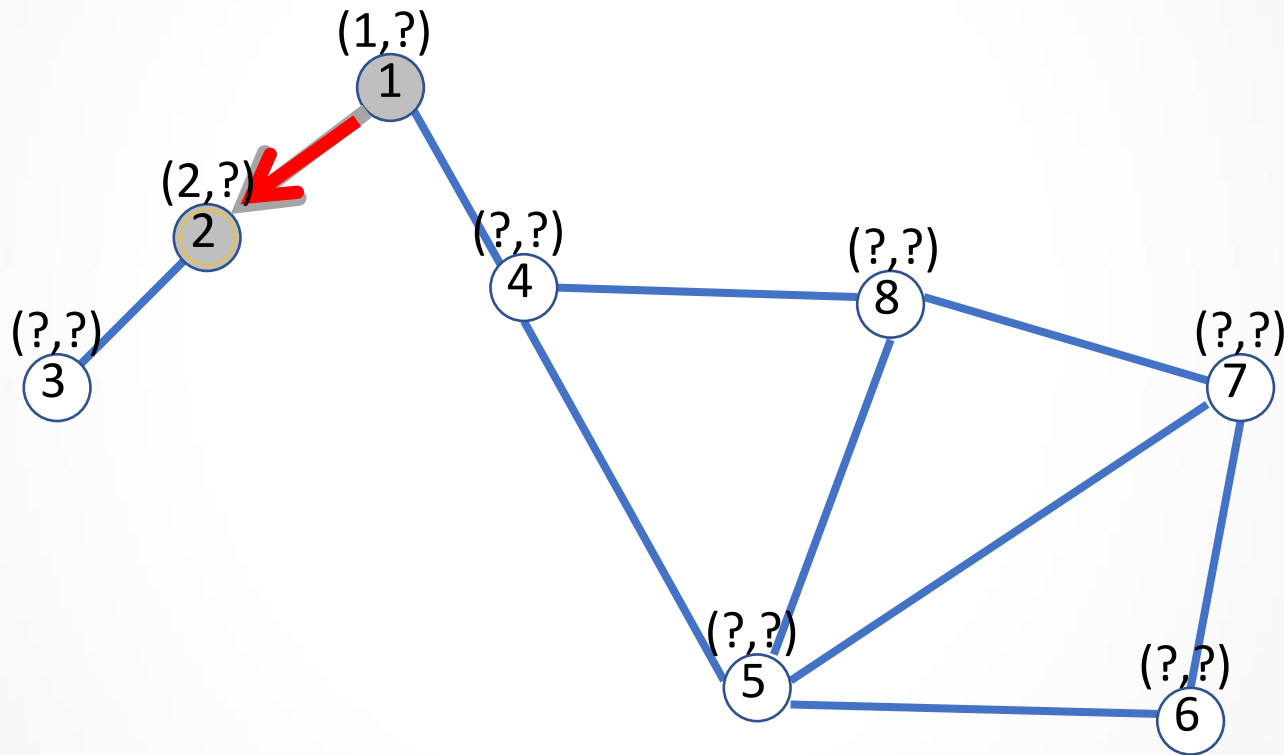
Обход в глубину

- При обходе в глубину будем для каждой вершины фиксировать время входа (in) и время выхода (out).
- 'Время' – это номер шага при выполнении обхода (сквозной счётчик).
- Эти показатели понадобятся для решения некоторых задач на графах с помощью обхода в глубину.

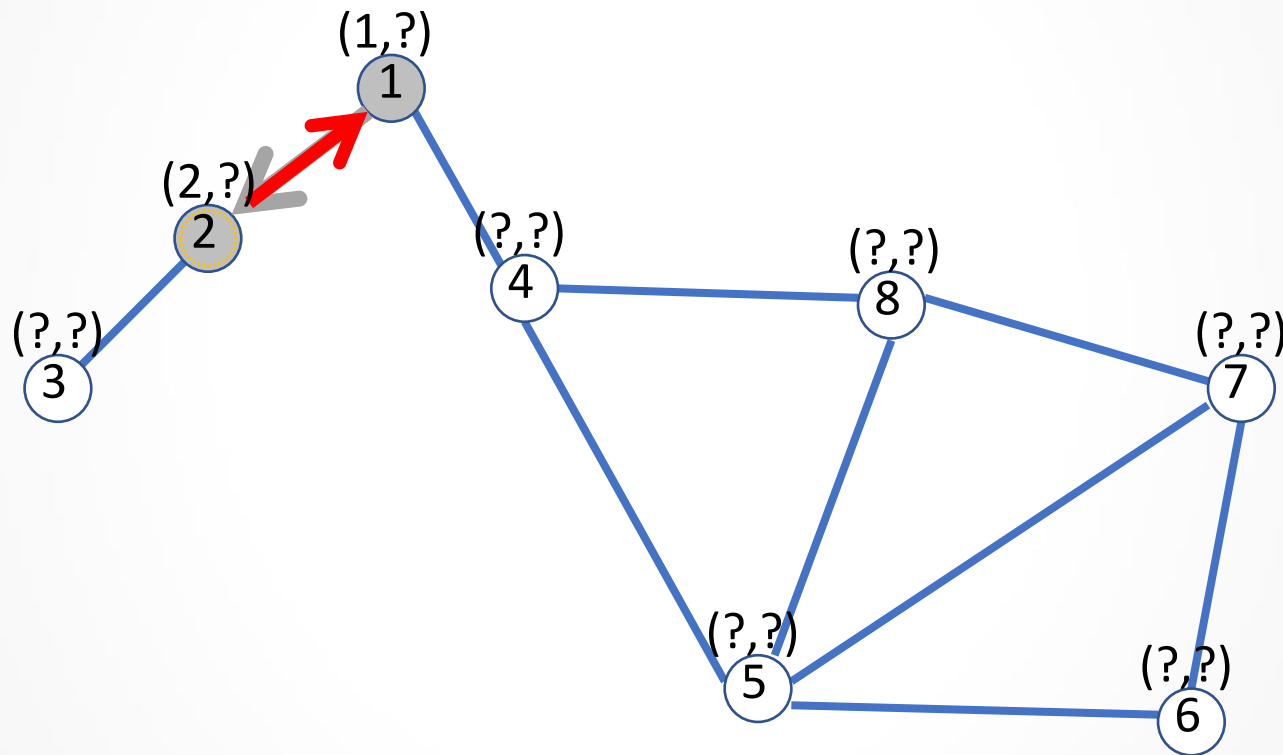
Обход в глубину: пример



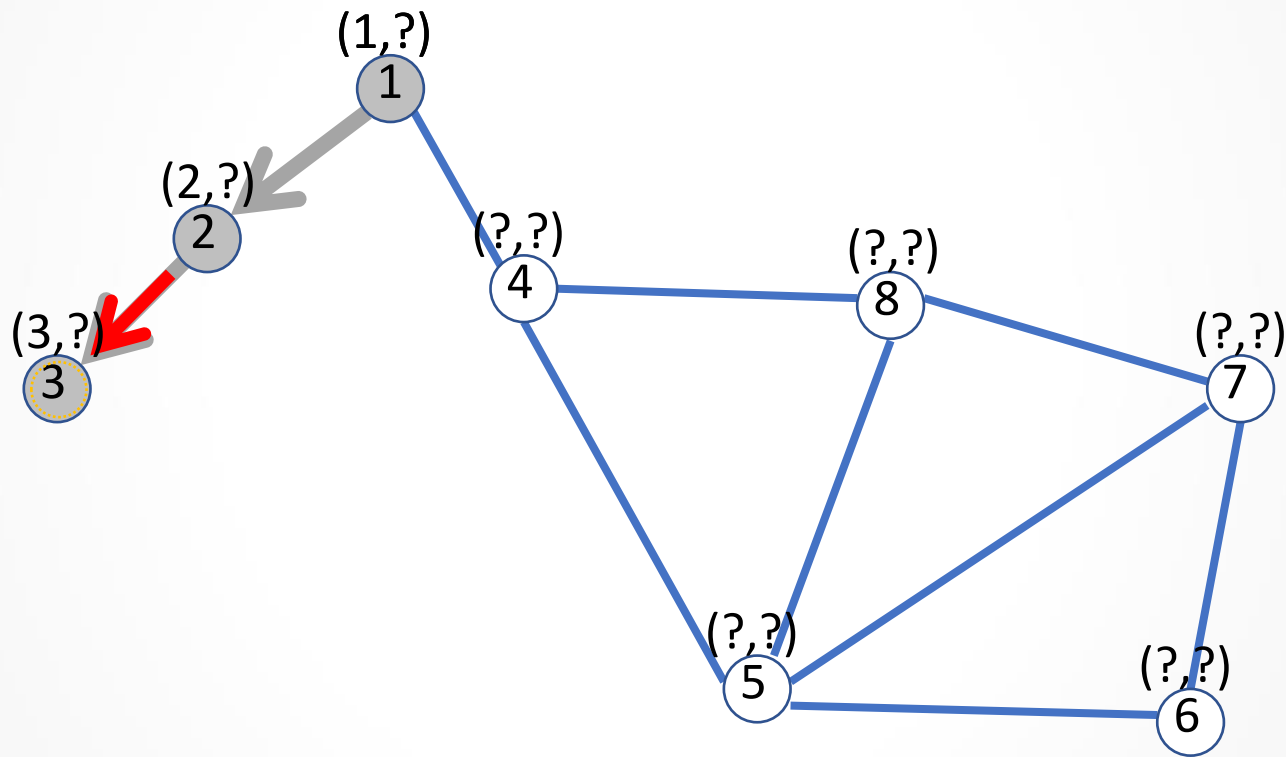
Обход в глубину: пример



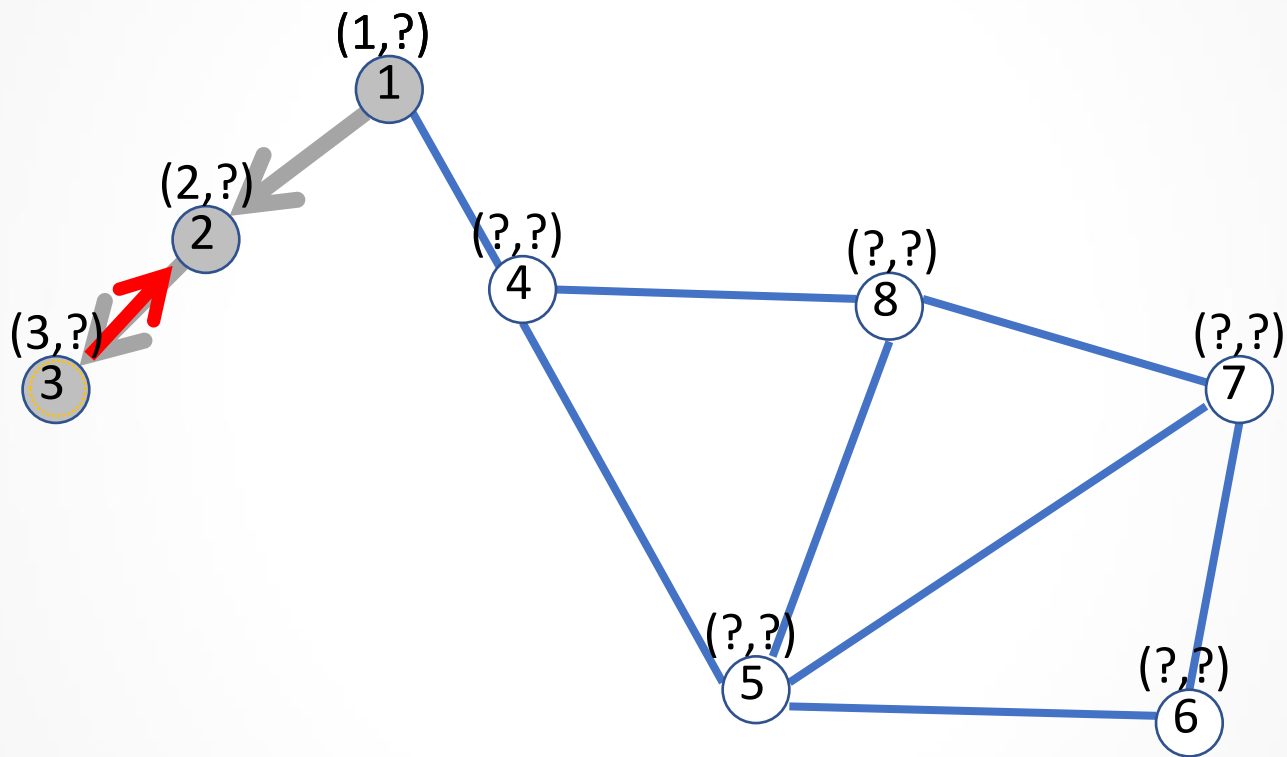
Обход в глубину: пример



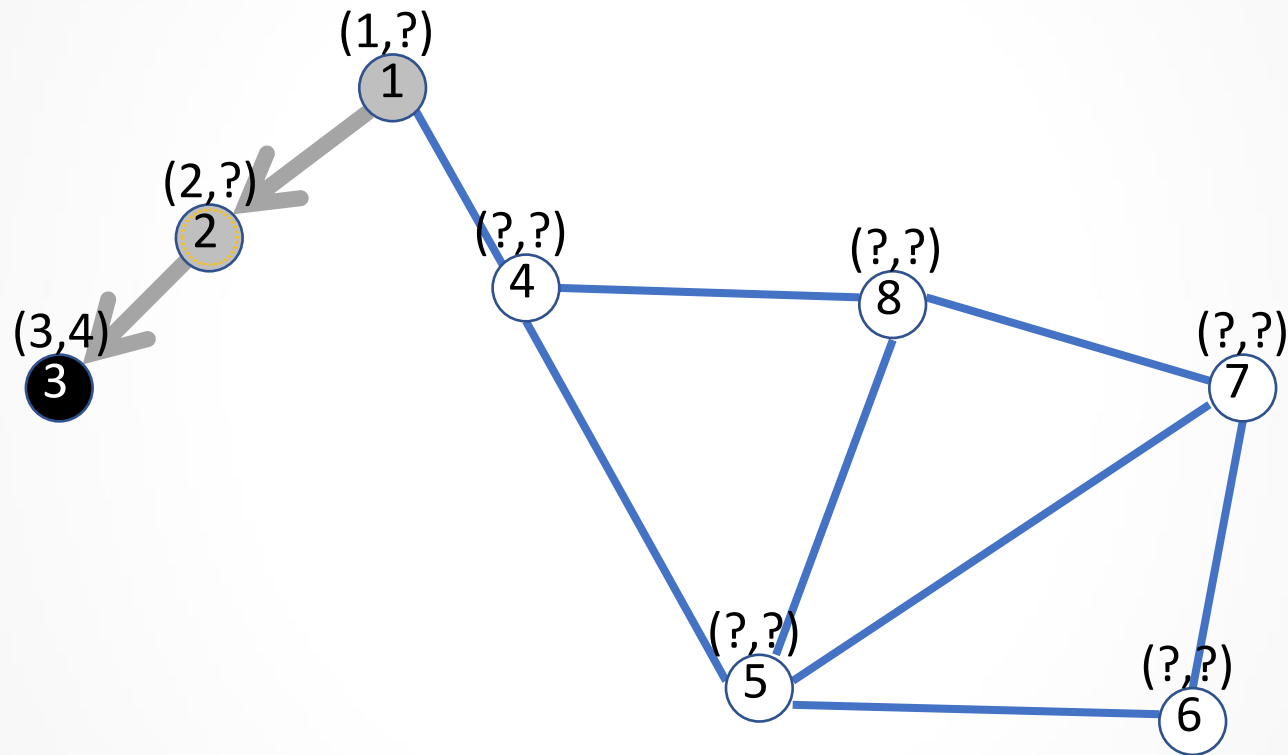
Обход в глубину: пример



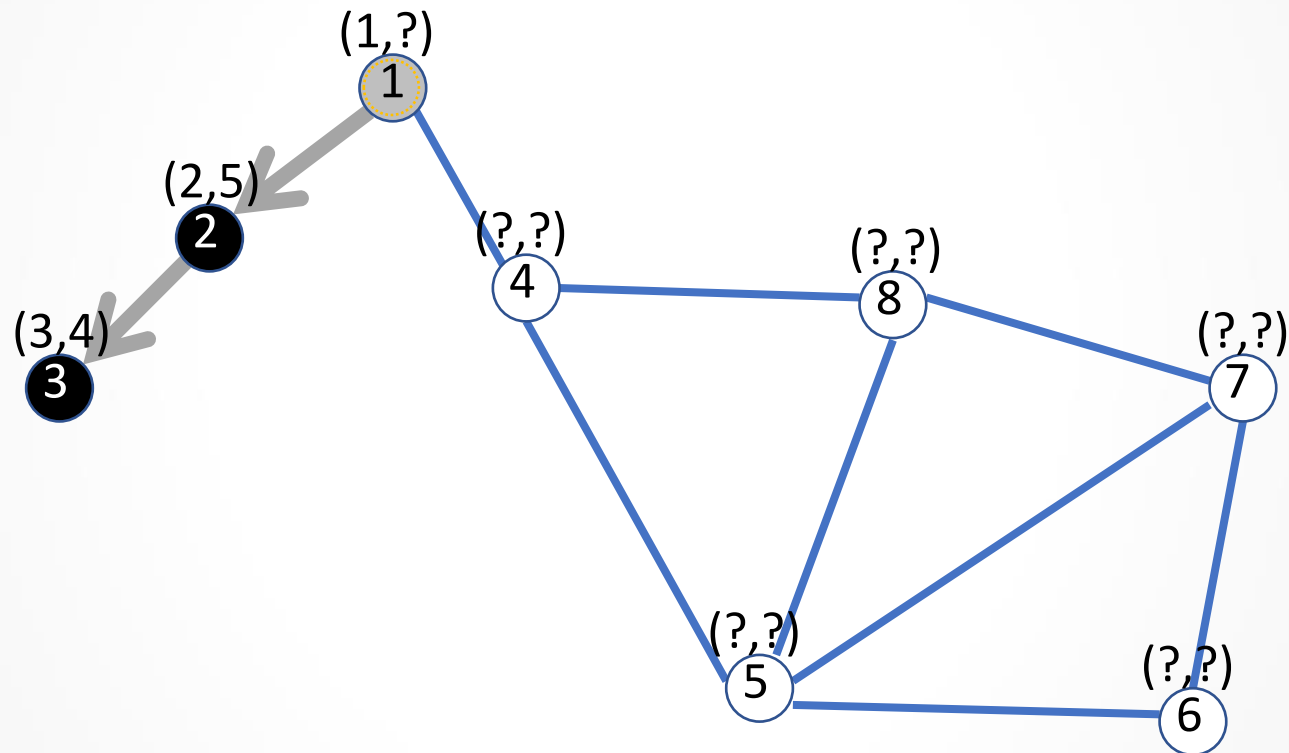
Обход в глубину: пример



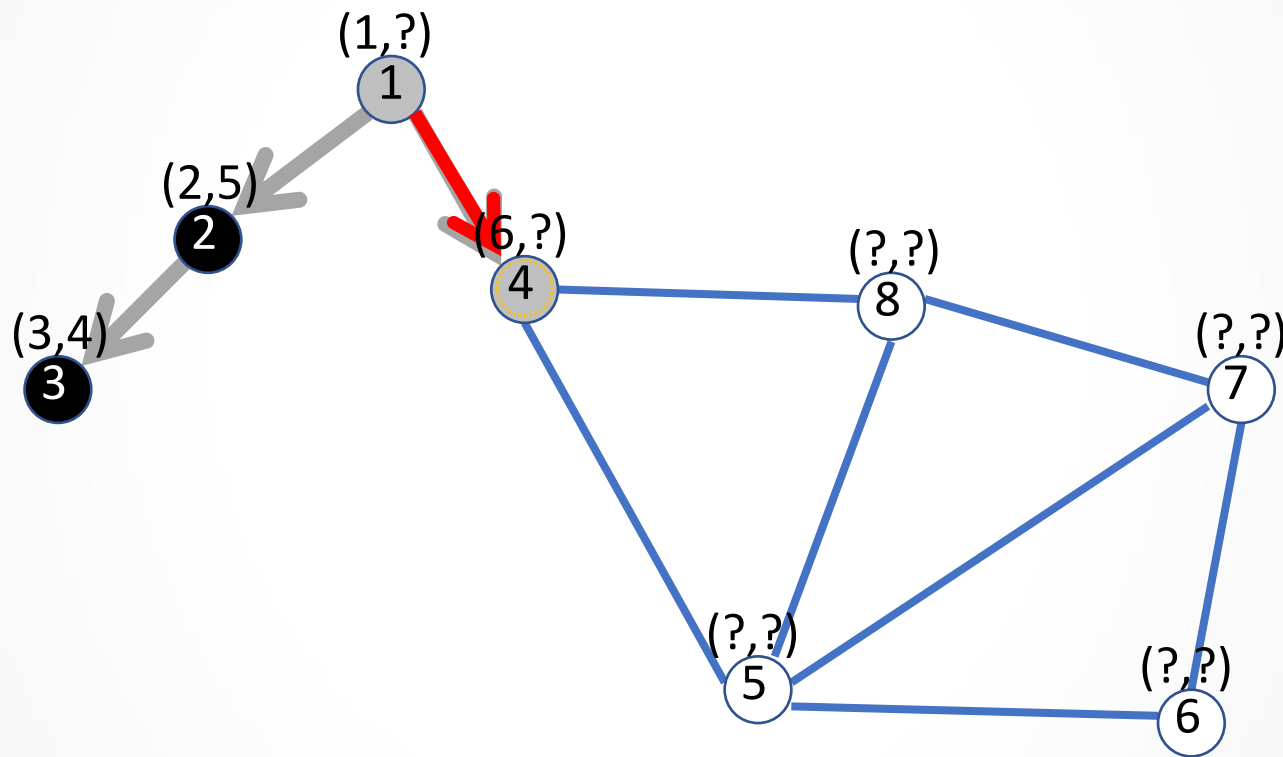
Обход в глубину: пример



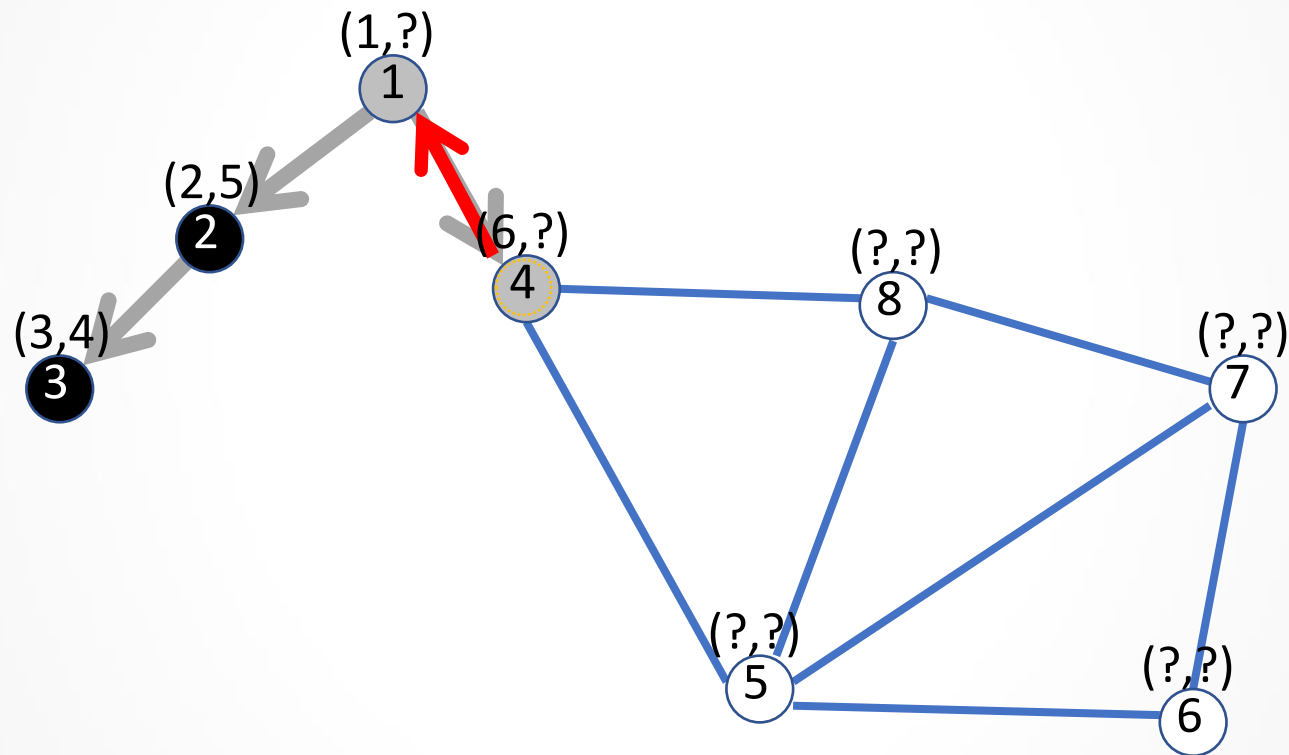
Обход в глубину: пример



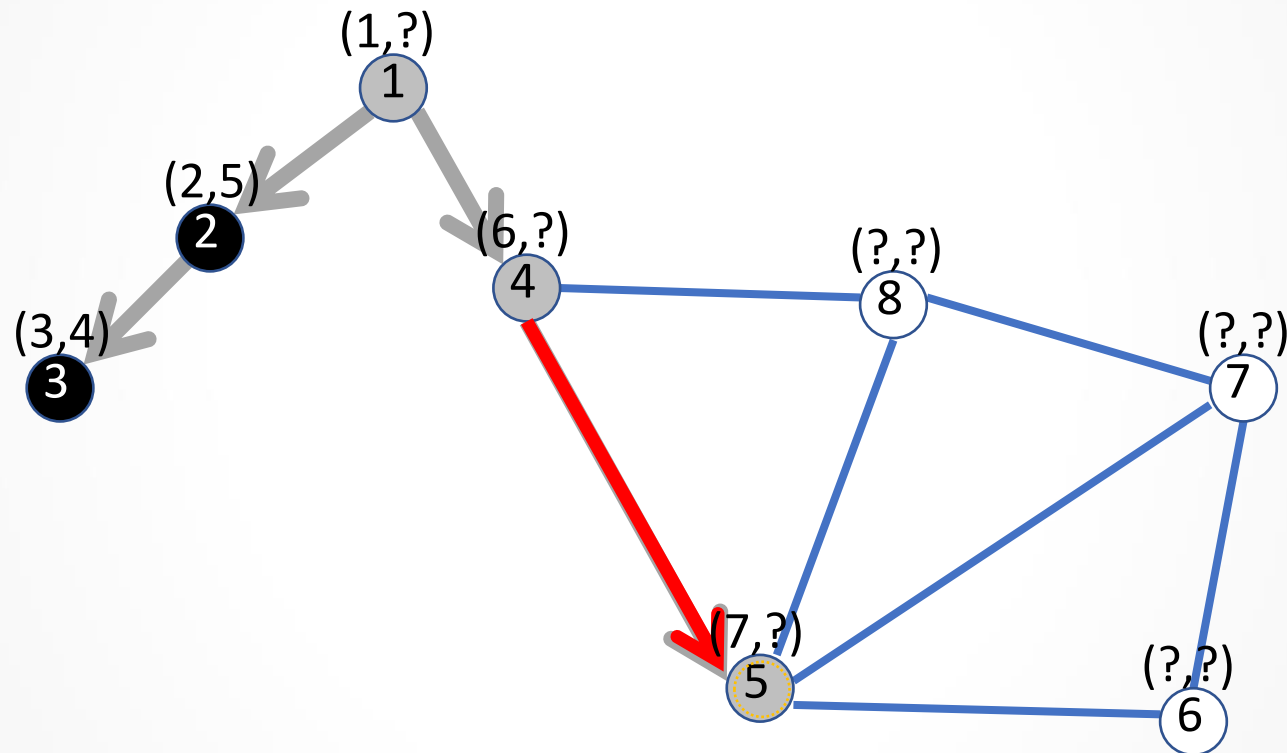
Обход в глубину: пример



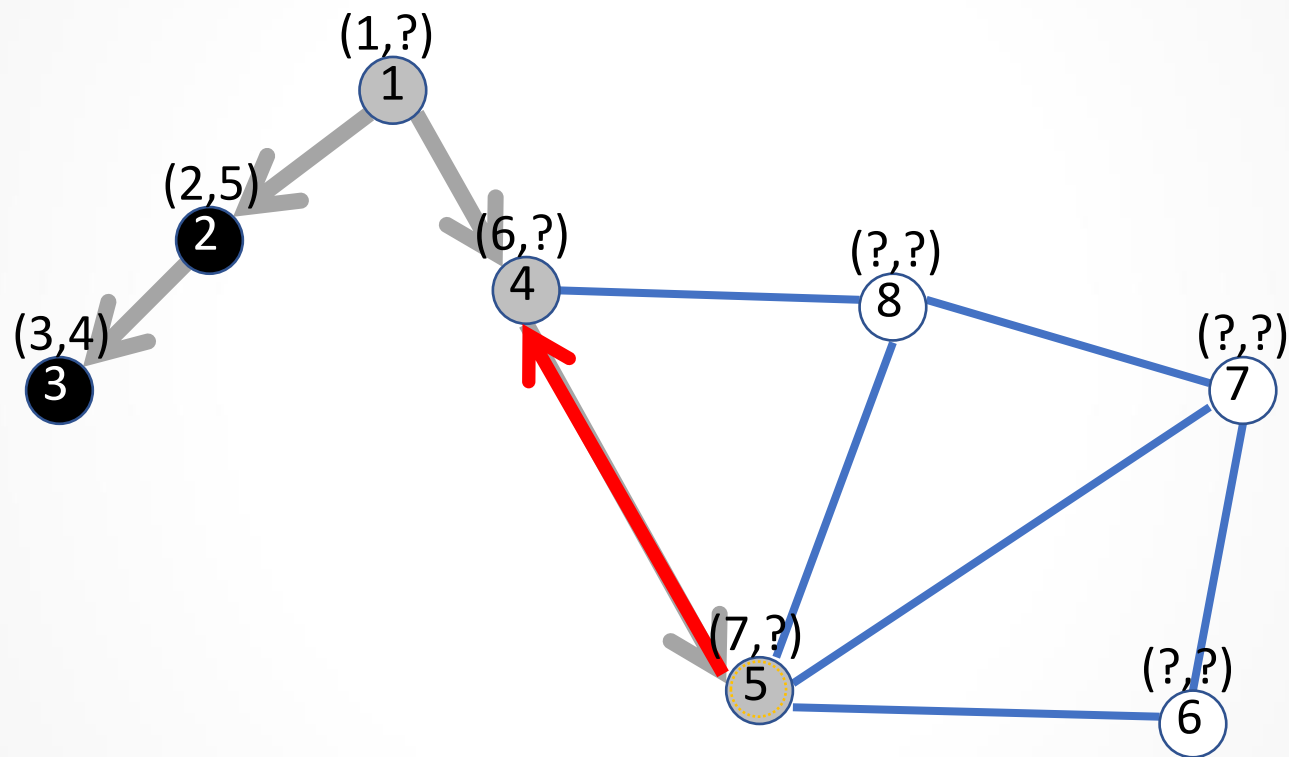
Обход в глубину: пример



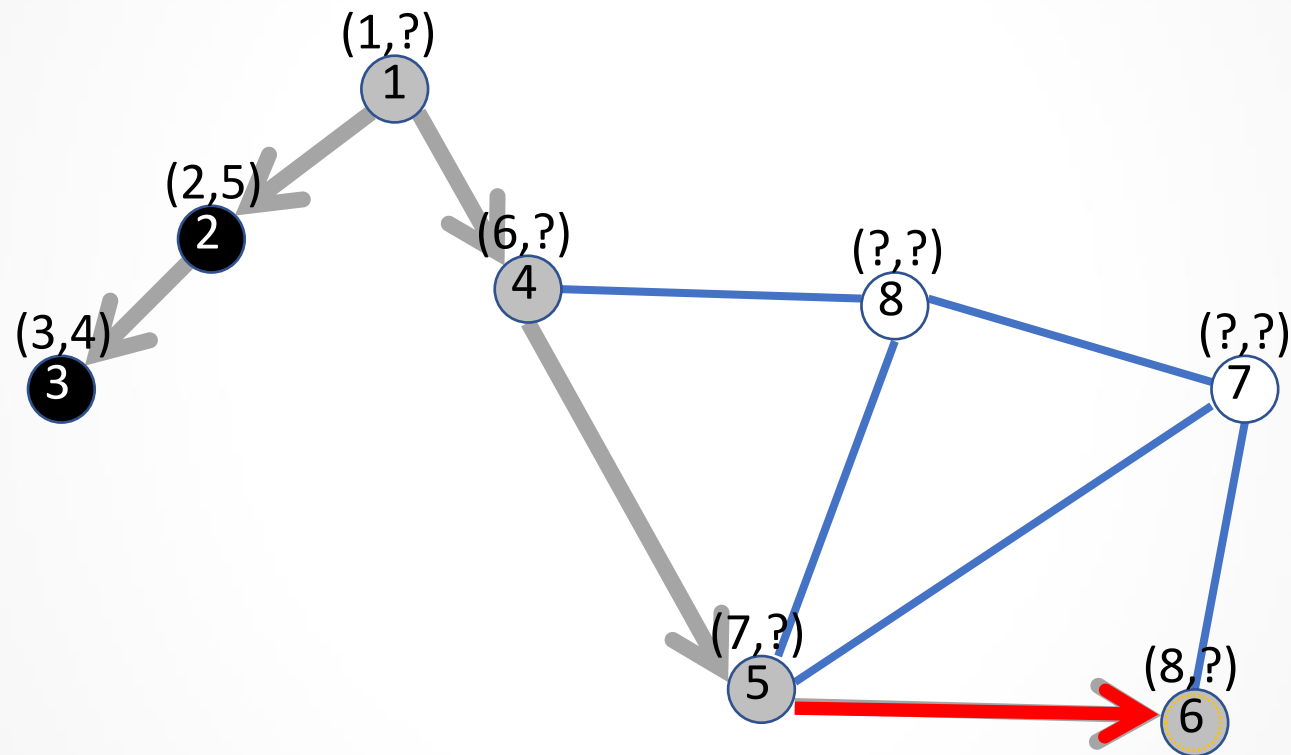
Обход в глубину: пример



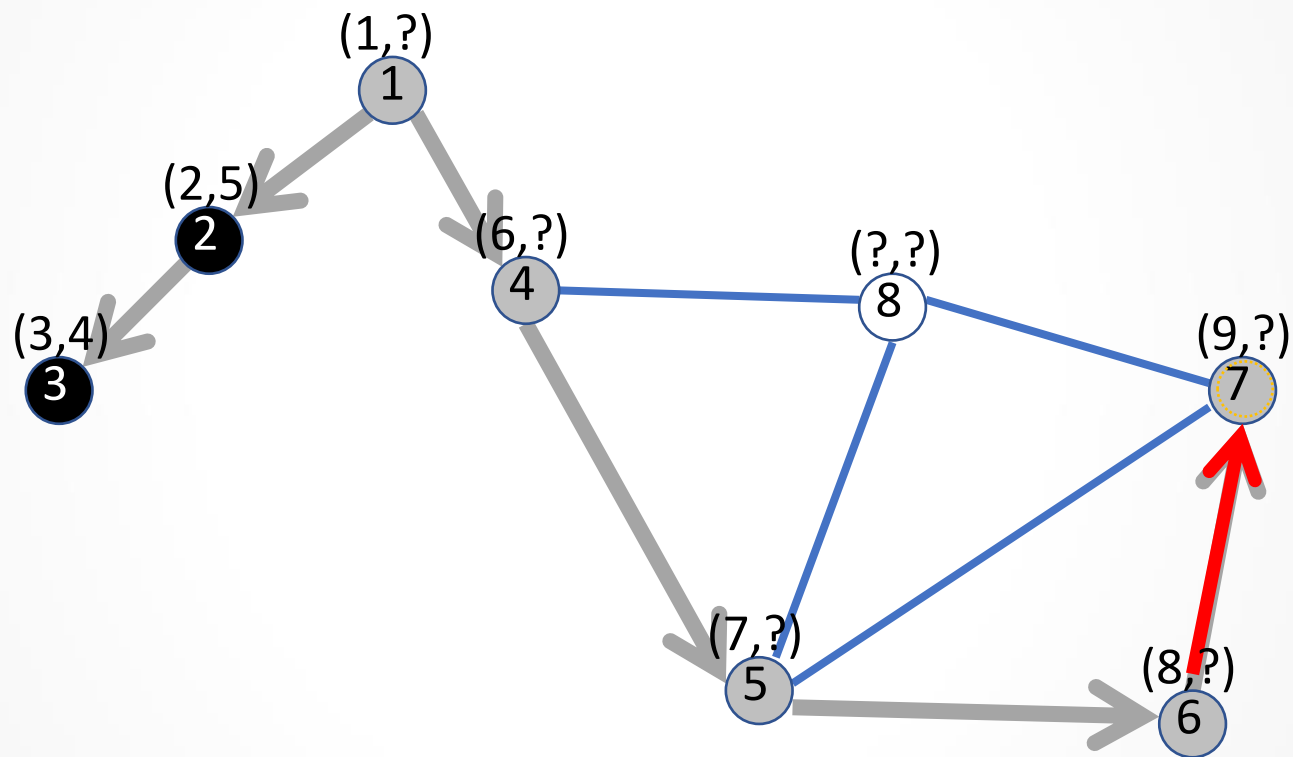
Обход в глубину: пример



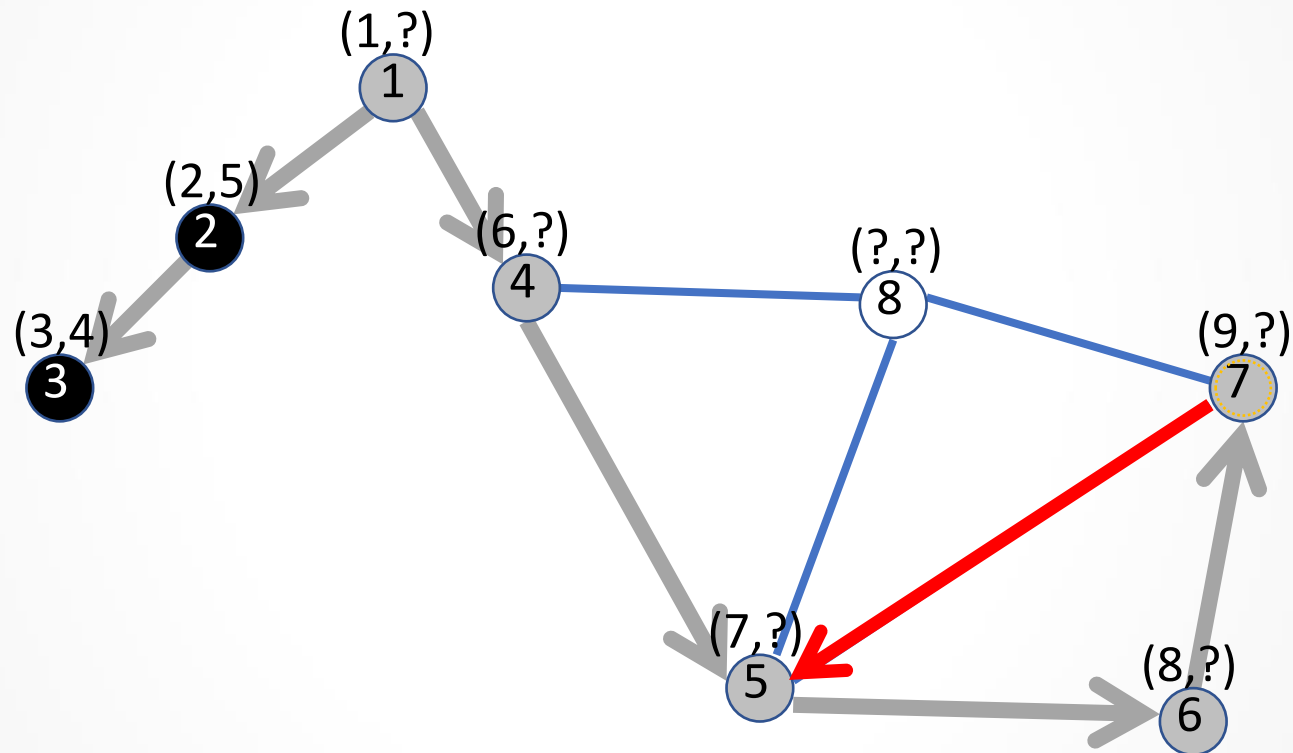
Обход в глубину: пример



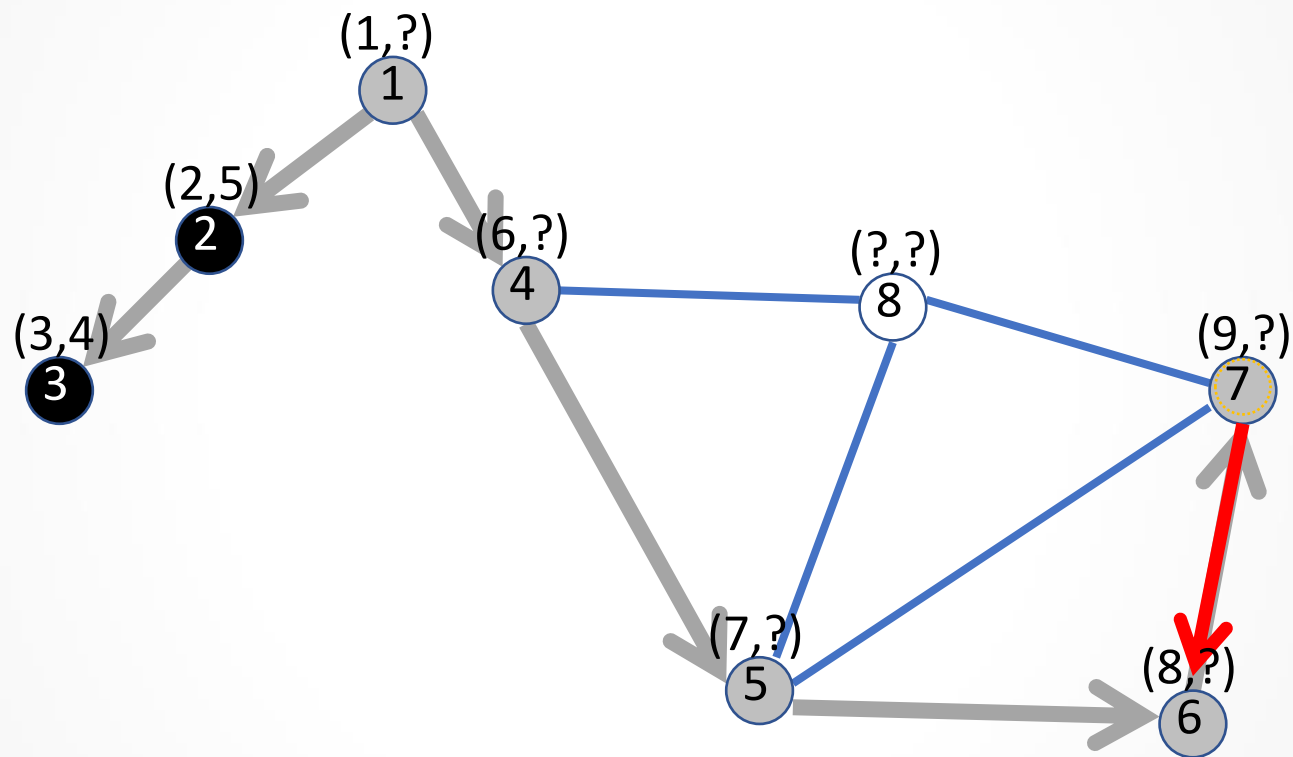
Обход в глубину: пример



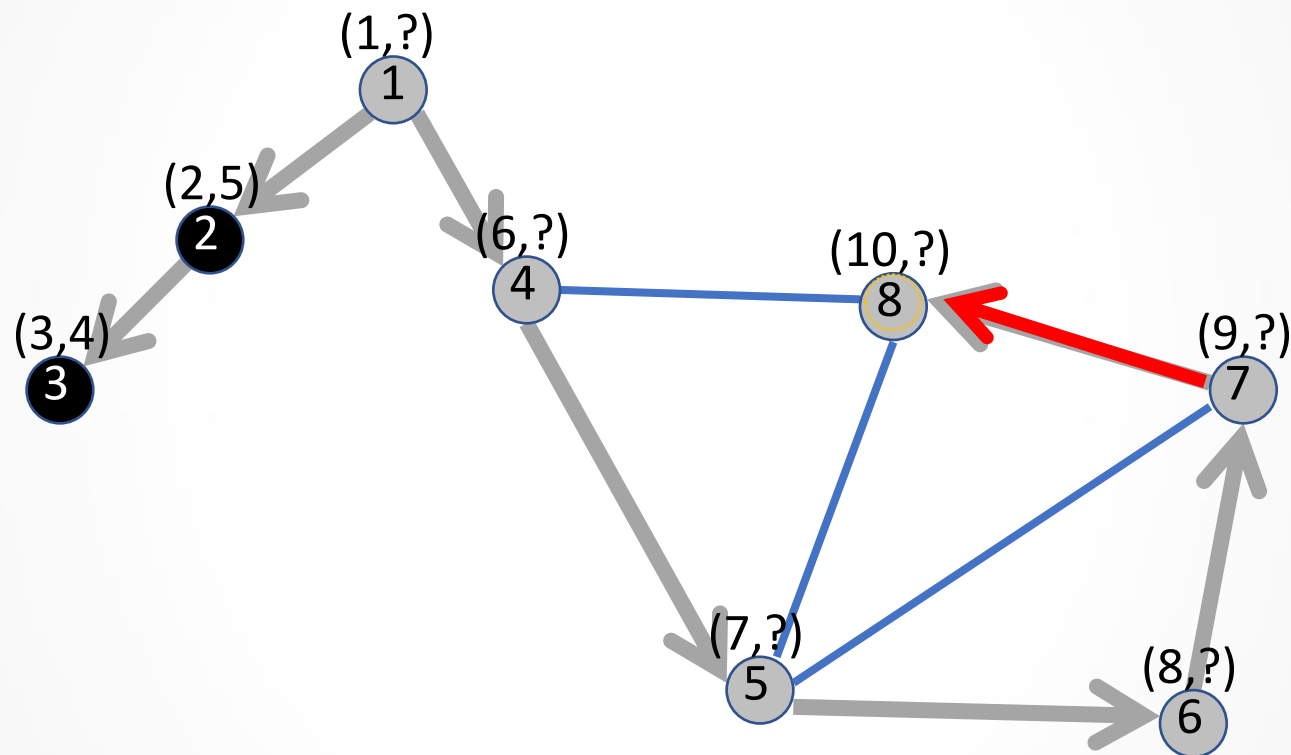
Обход в глубину: пример



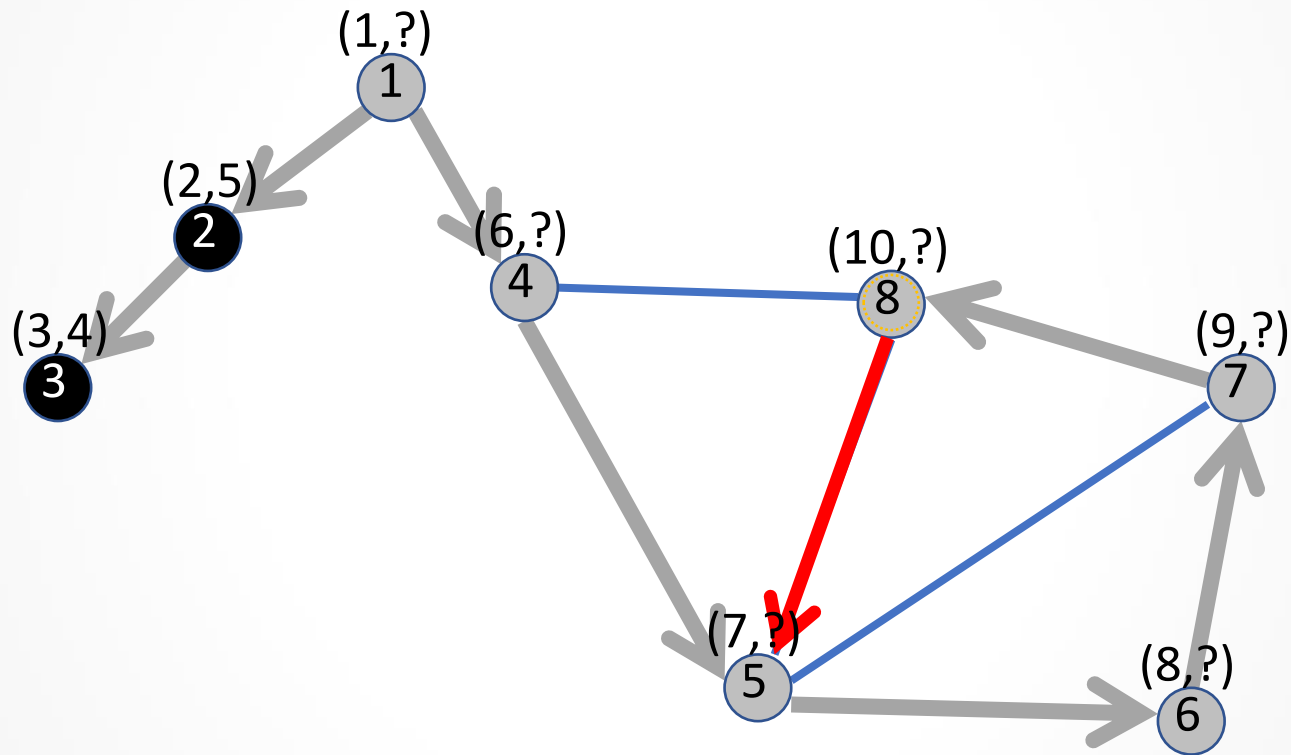
Обход в глубину: пример



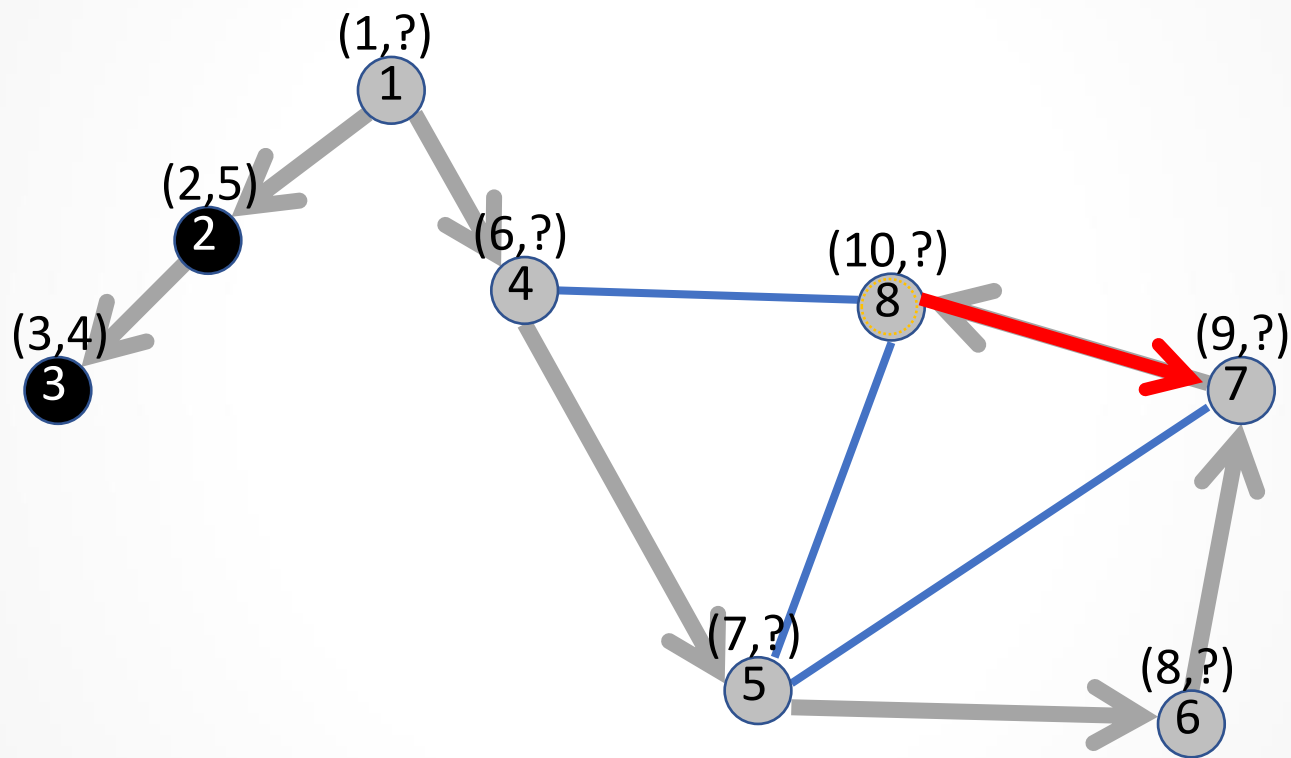
Обход в глубину: пример



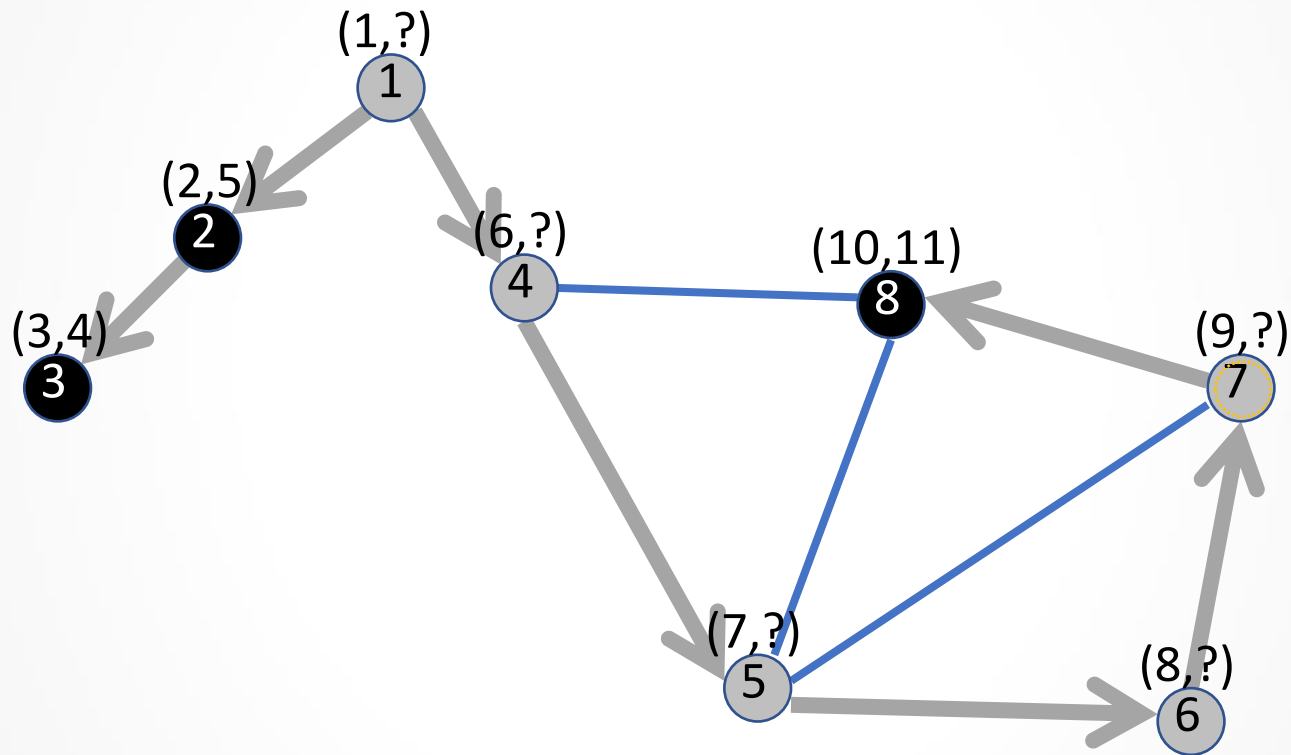
Обход в глубину: пример



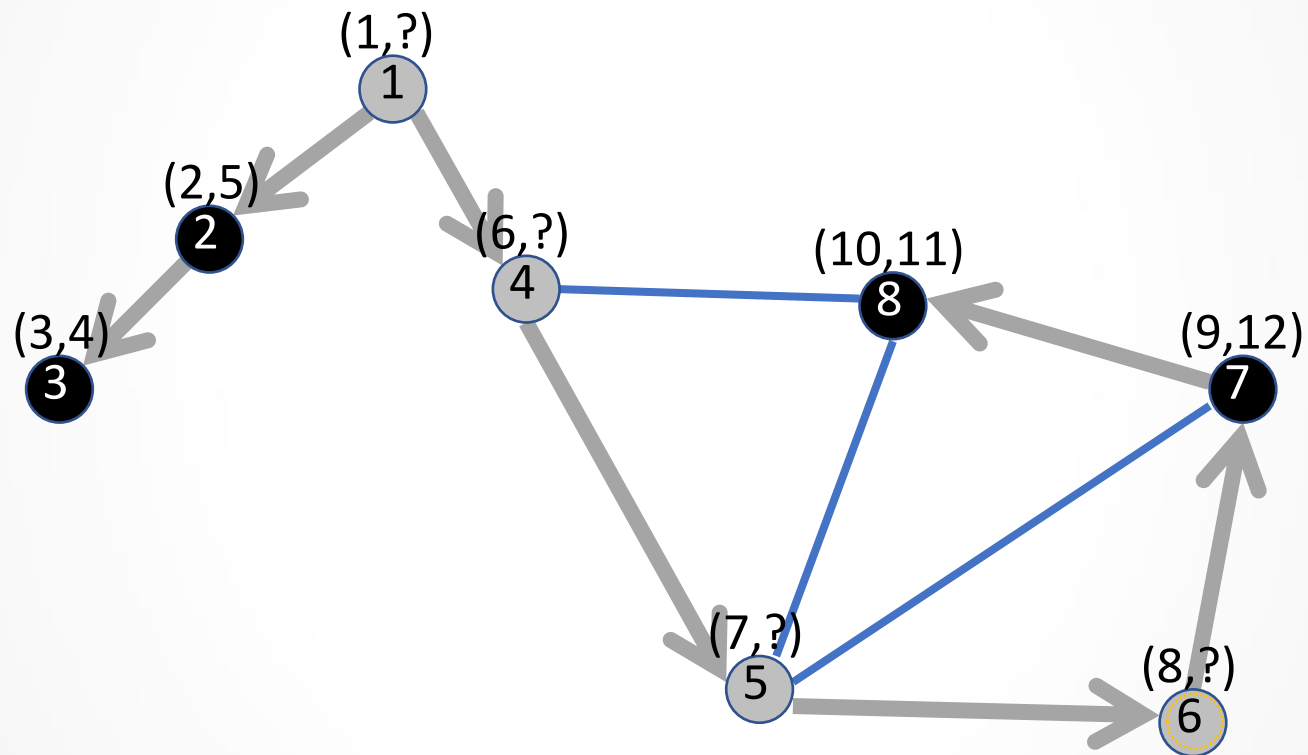
Обход в глубину: пример



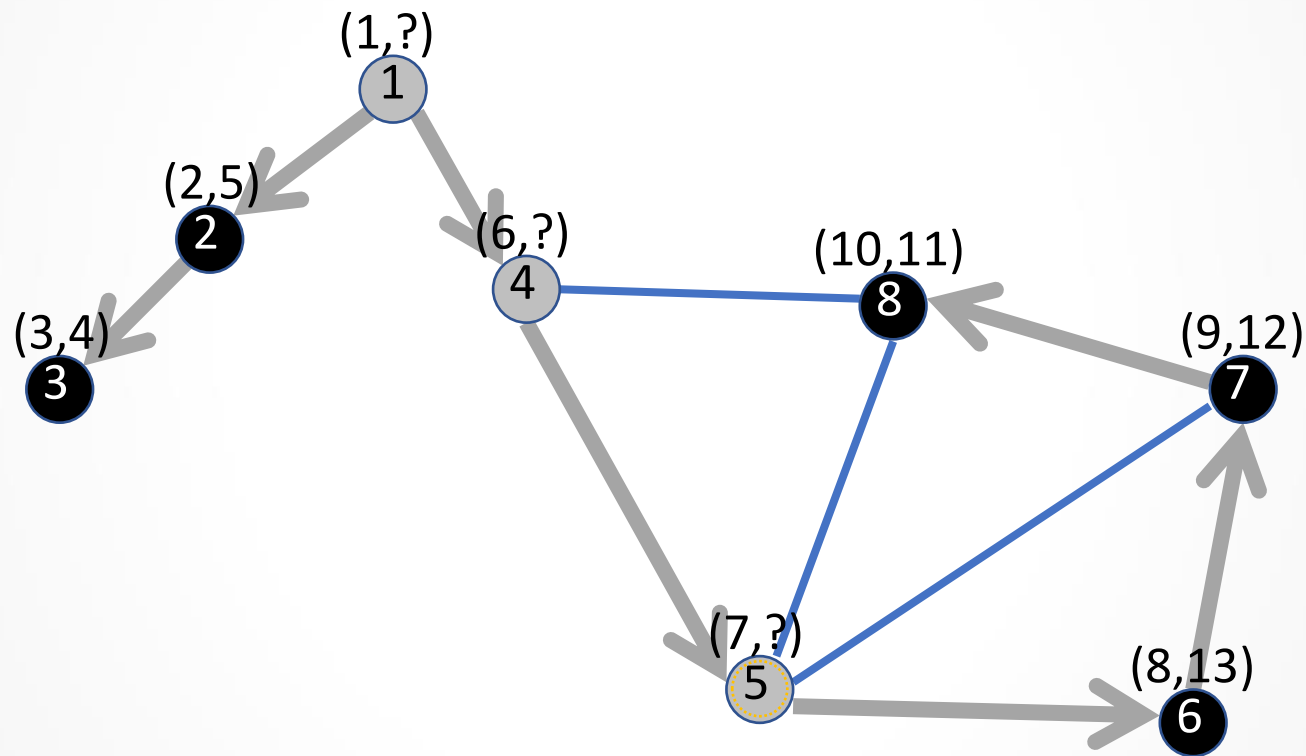
Обход в глубину: пример



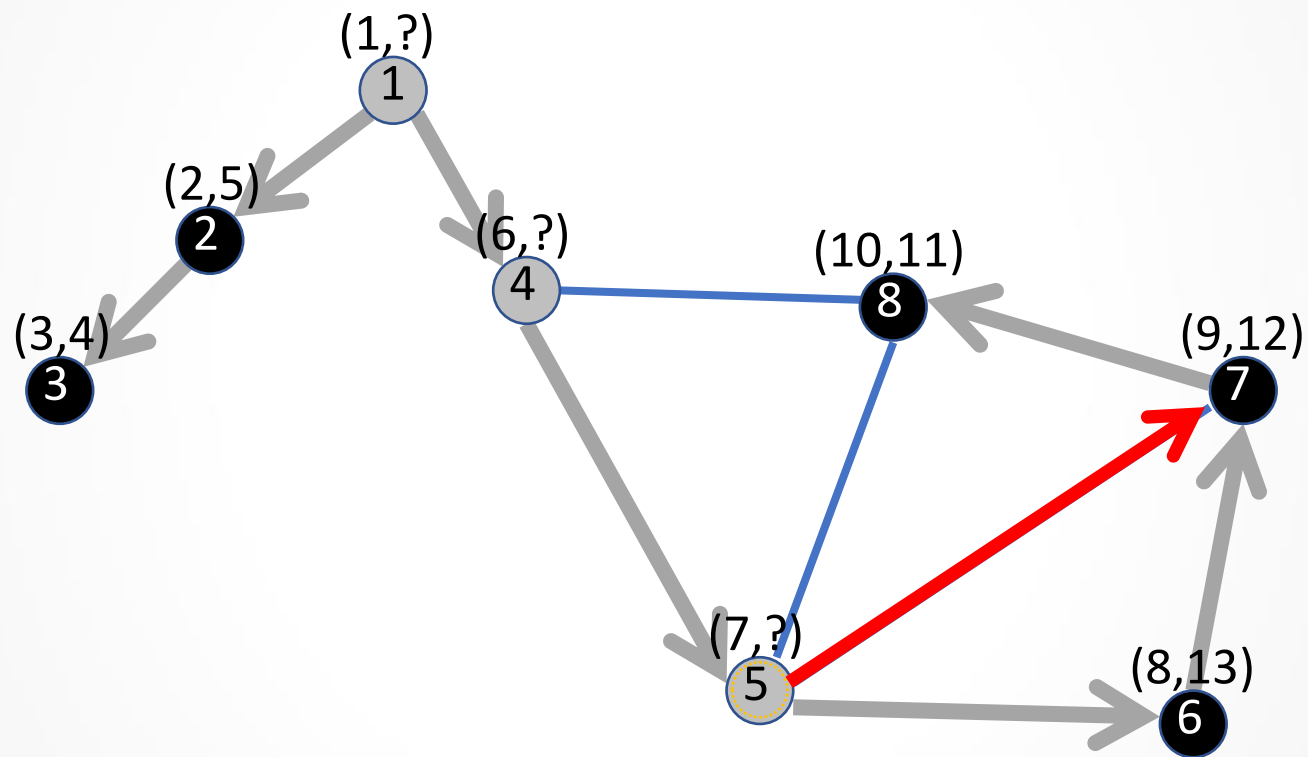
Обход в глубину: пример



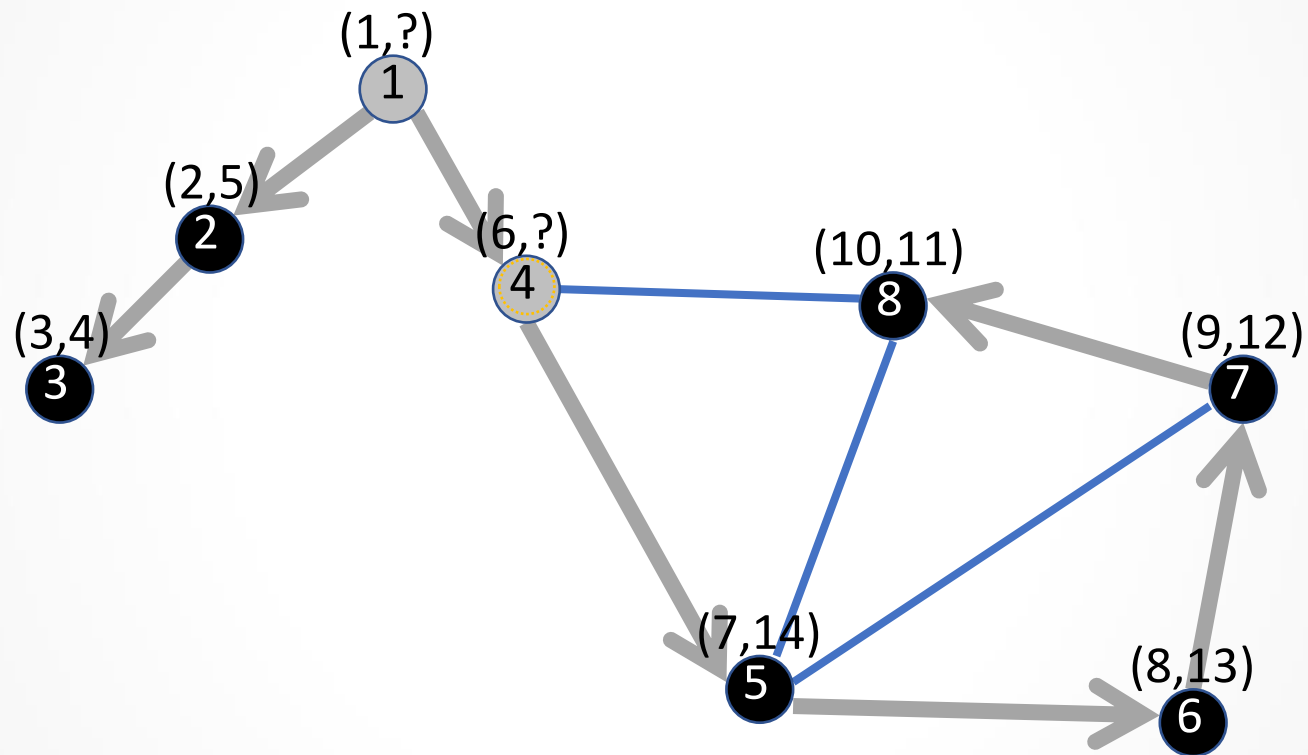
Обход в глубину: пример



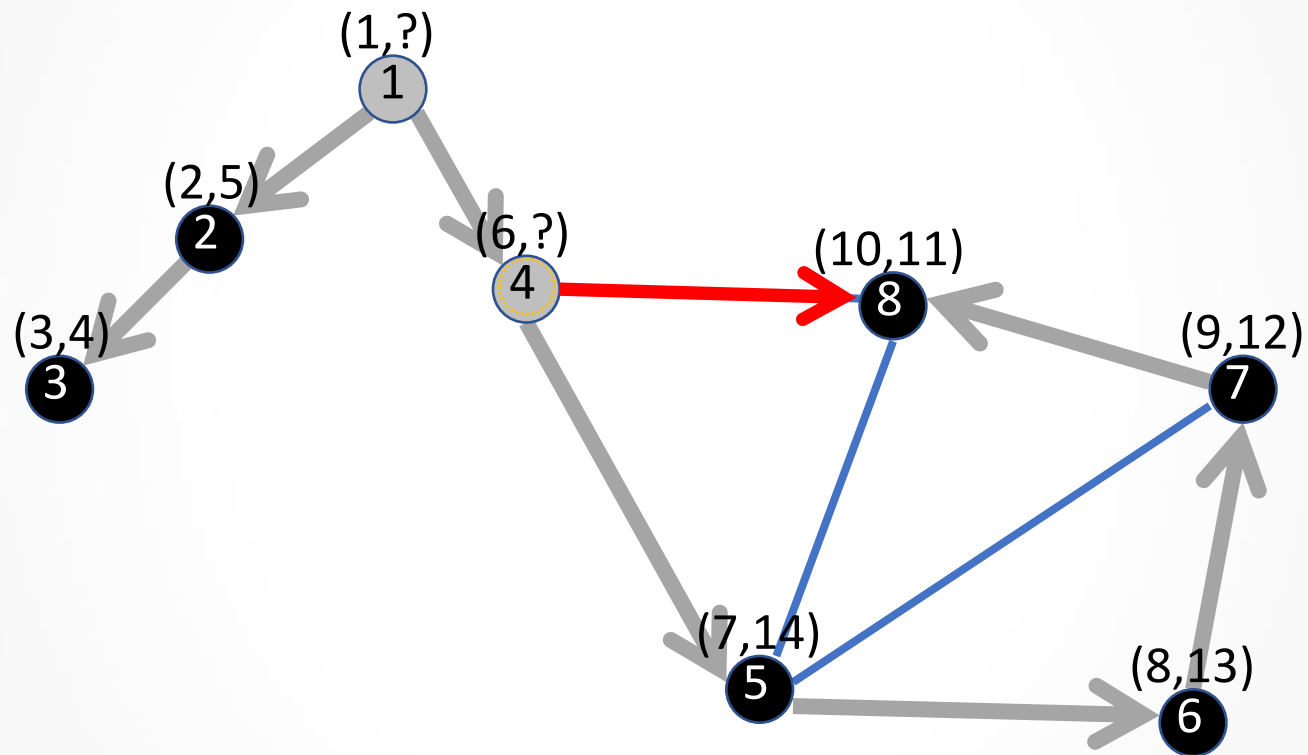
Обход в глубину: пример



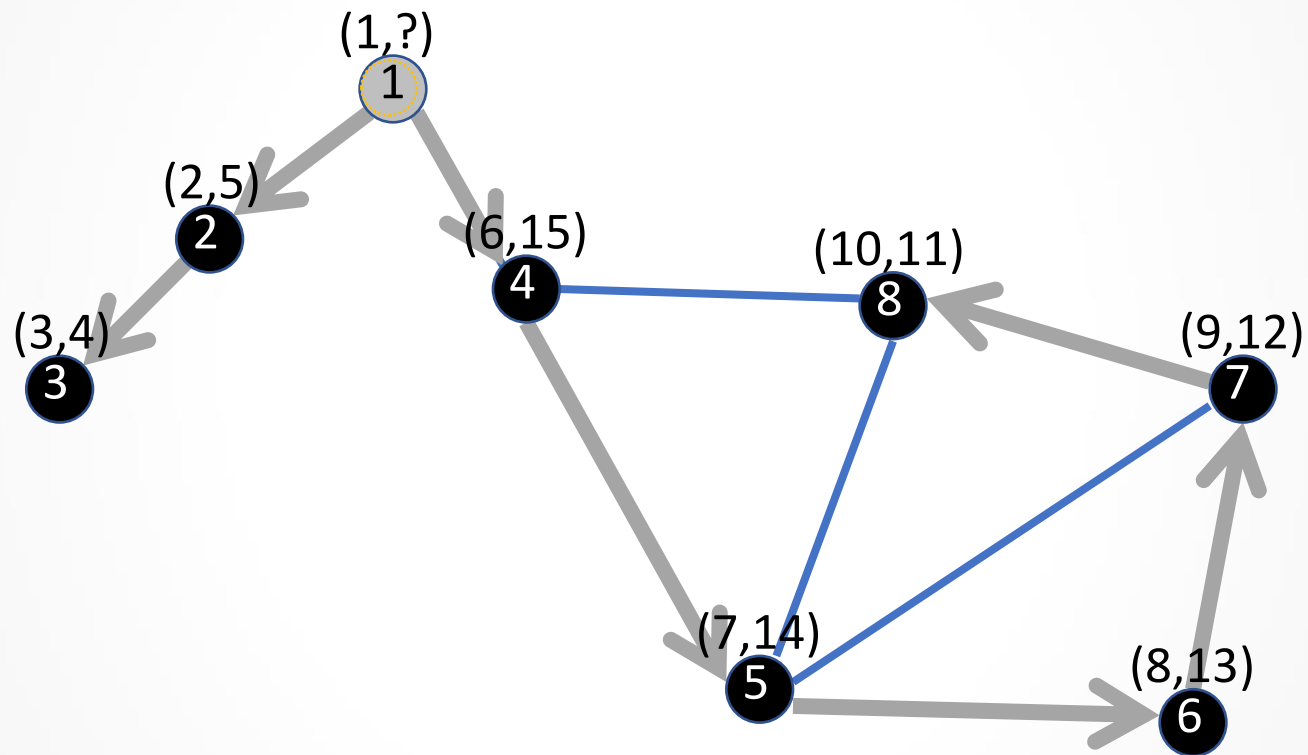
Обход в глубину: пример



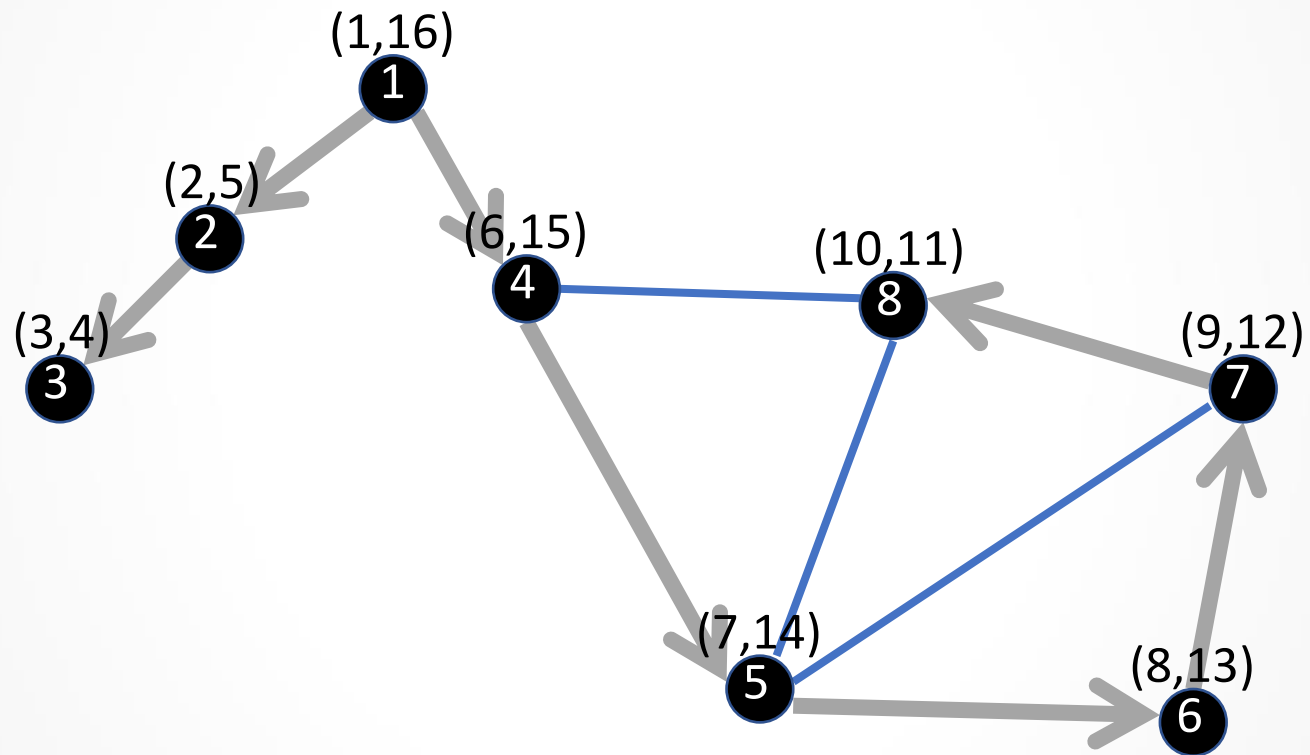
Обход в глубину: пример



Обход в глубину: пример



Обход в глубину: пример



Алгоритм обхода в глубину

- Для учёта правильного порядка обхода вершин будем применять стек или рекурсию.
- Для обхода графа в глубину нужны такие структуры данных:
 - ✓ Для учёта статуса вершины – массив `State[]`.
 - ✓ Сохранение предшественников вершин в массиве `Pred[]`.
 - ✓ Для учёта времён входа и выхода в вершины – массивы `Time_In[]` и `Time_Out[]`.

Алгоритм обхода в глубину

DFS (G)

For each $v \in V$:

 State[v] := 'unvisited';

 Pred[v] := NULL;

 Time_In[v] := NULL;

 Time_Out[v] := NULL;

CurTime := 0;

For each $v \in V$:

 If State[v] = 'unvisited'

 DFS_Visit(v);

Алгоритм обхода в глубину

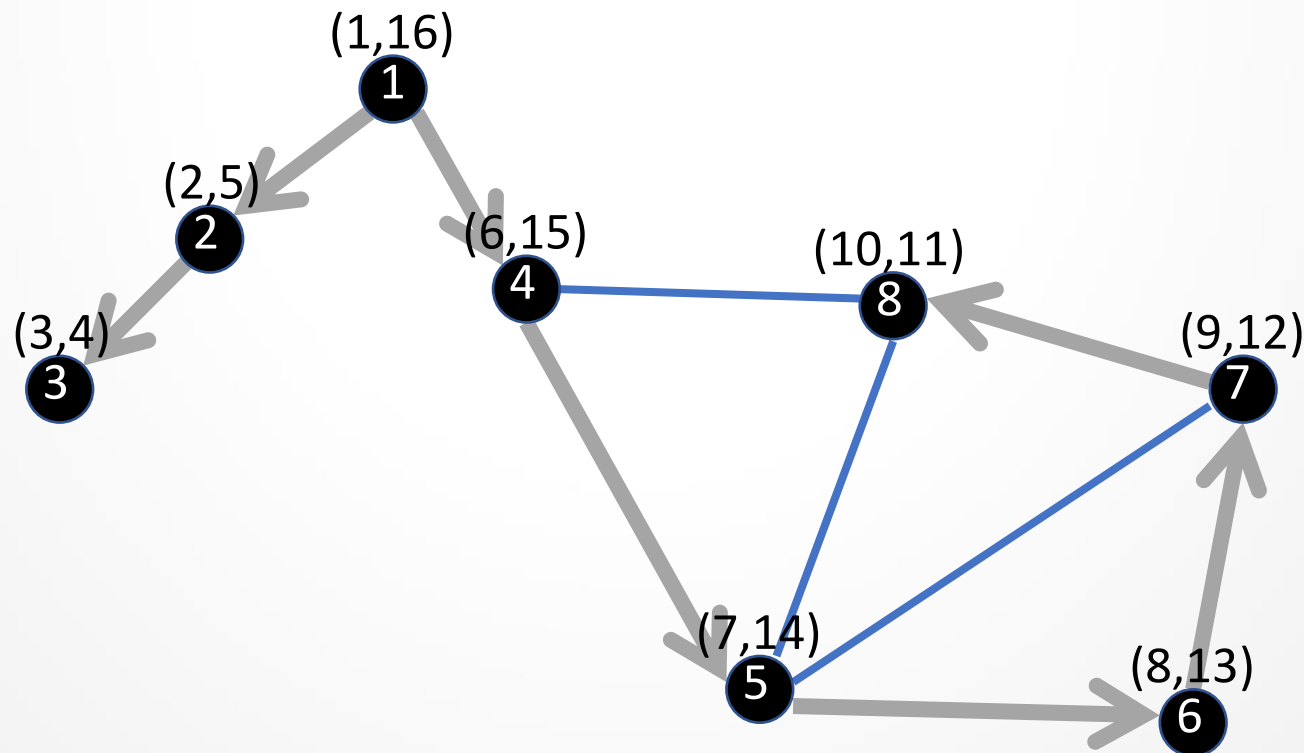
```
DFS Visit (v)  
State[v] := 'visited';  
CurTime := CurTime + 1;  
Time_In[v] := CurTime;  
For each u in Adj(v)  
    If State[u] = 'unvisited'  
        Pred[u] := v;  
        DFS_Visit(u);  
State[v] := 'processed';  
CurTime := CurTime + 1;  
Time_Out[v] := CurTime;
```

Алгоритм обхода в глубину

Полезные свойства пометок `Time_In[]` и `Time_Out[]`.

1. Если вершина x – предшественник (не обязательно непосредственный) вершины y , то

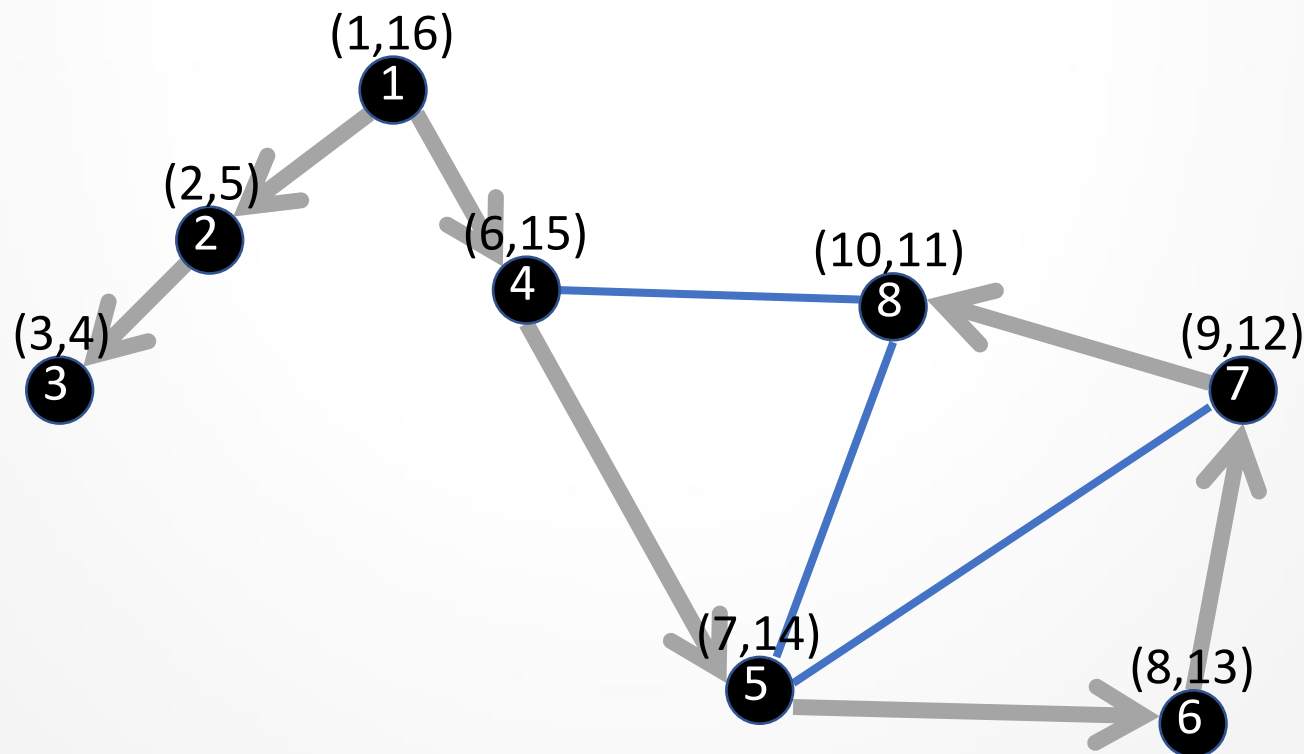
$$\text{Time_In}[x] < \text{Time_In}[y] < \text{Time_Out}[y] < \text{Time_Out}[x]$$



Алгоритм обхода в глубину

Полезные свойства пометок `Time_In[]` и `Time_Out[]`.

2. Величина $(Time_Out[x] - Time_In[x] - 1)/2$ равна количеству древесных потомков вершины x .



Сложность обходов

Оценим временную сложность процедур обхода в ширину и в глубину.

При каждой процедуре обхода мы ровно по 2 раза обрабатываем каждую вершину и каждое ребро.

Поэтому временная сложность составляет

$$O(n + m).$$

Применение обхода в глубину

- ✓ *Поиск компонент связности*
- ✓ *Обнаружение циклов*
- ✓ Проверка двудольности графа
- ✓ Поиск мостов на графе

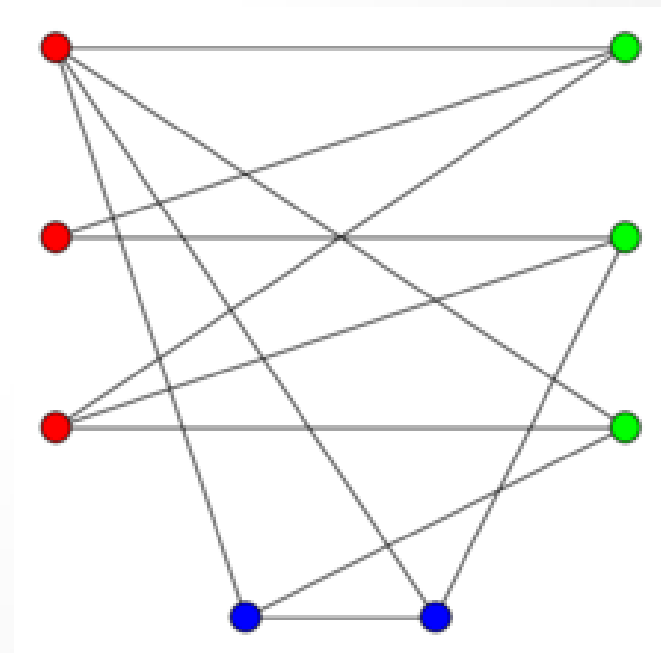
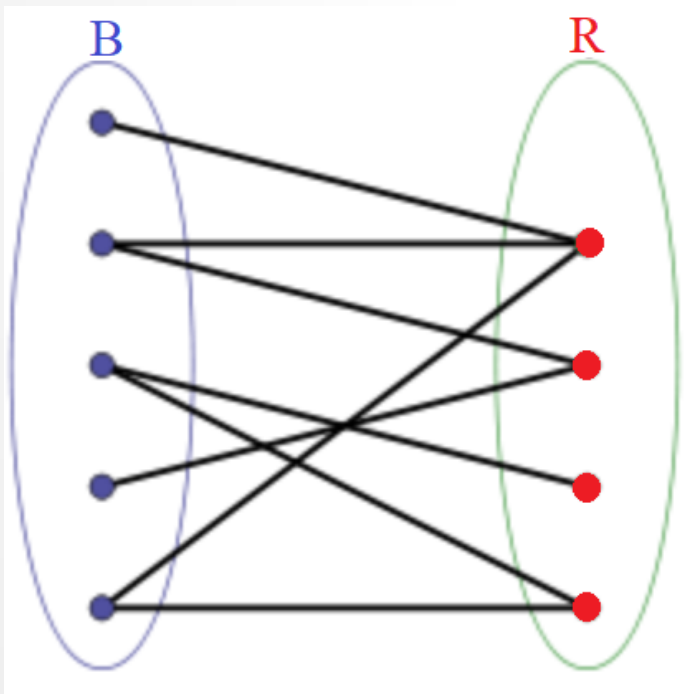
Применение обхода в глубину

- ✓ *Поиск компонент связности*
- ✓ *Обнаружение циклов*

Для каждой из этих двух задач алгоритм на основе обхода в глубину по своей сути совпадает с аналогичным алгоритмом, основанным на обходе в ширину.

Двудольность

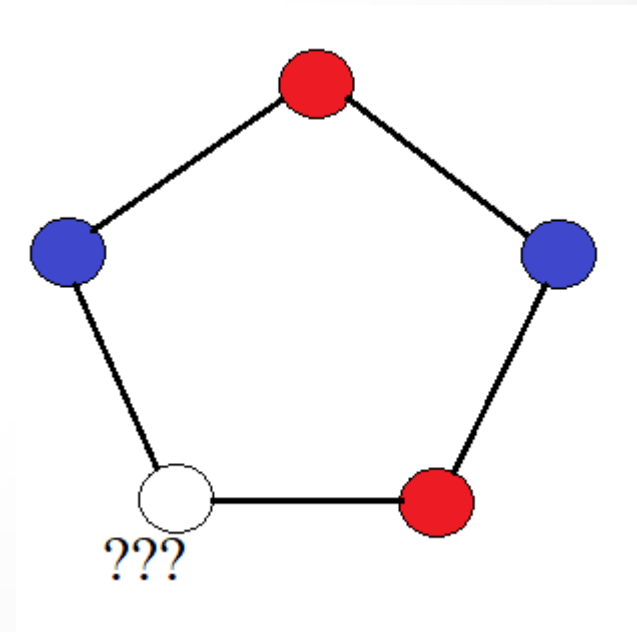
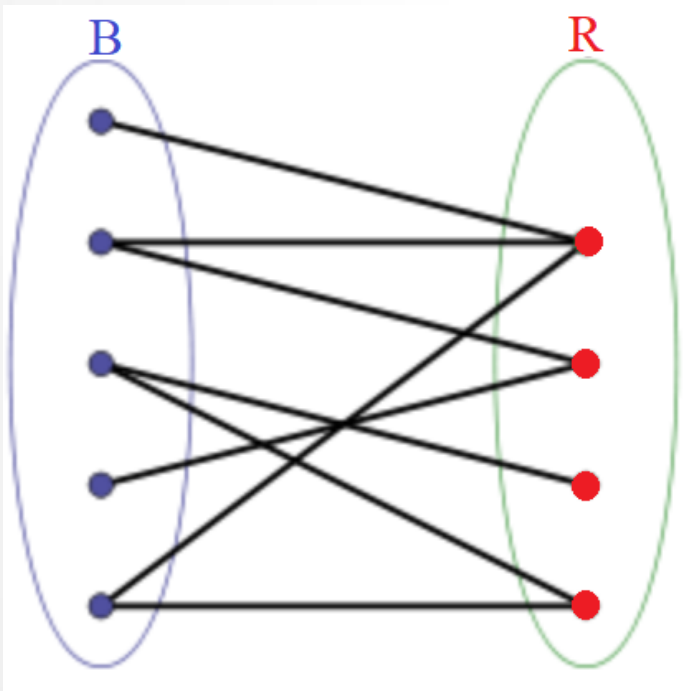
Определение. Граф $G(V, E)$ называется **двудольным**, если множество вершин можно разбить на два непересекающиеся подмножества $V = B \cup R$ таким образом, что концы каждого ребра лежат в разных подмножествах.



Двудольность

Теорема. Граф $G(V, E)$ является двудольным тогда и только тогда, когда на нём нет циклов нечётной длины.

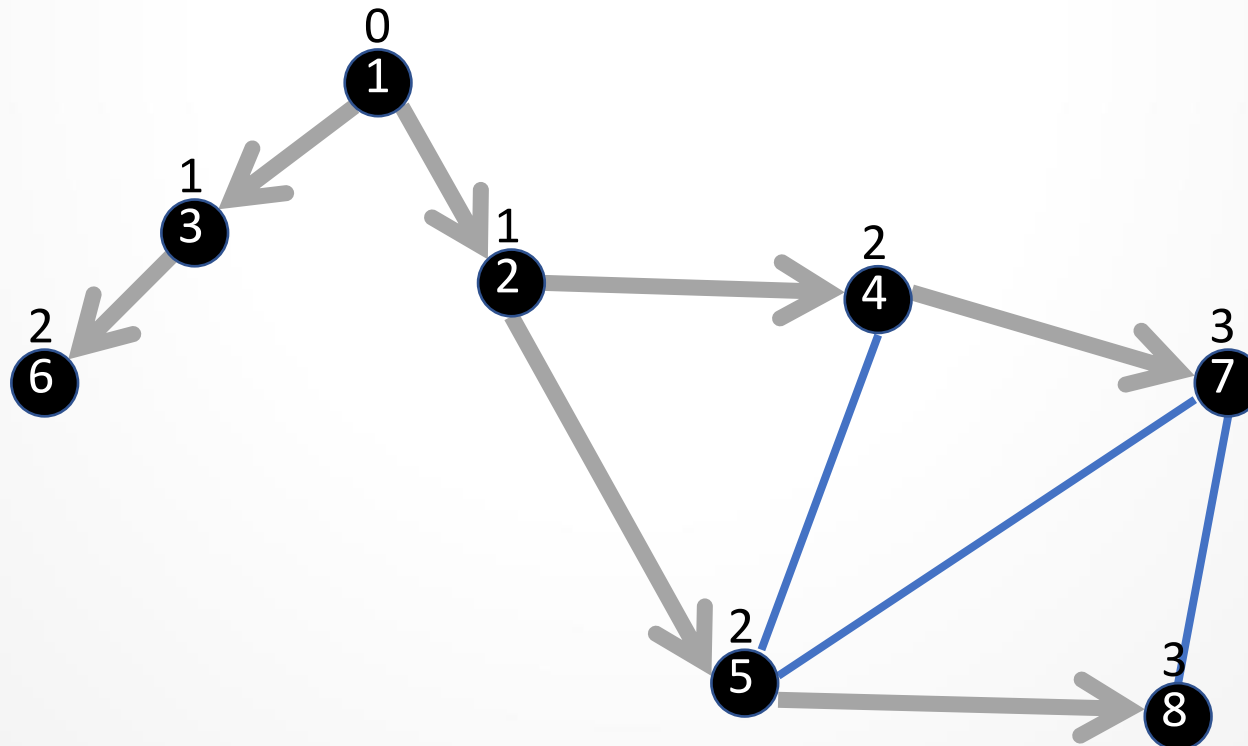
Следствие. Деревья и леса – двудольные графы.



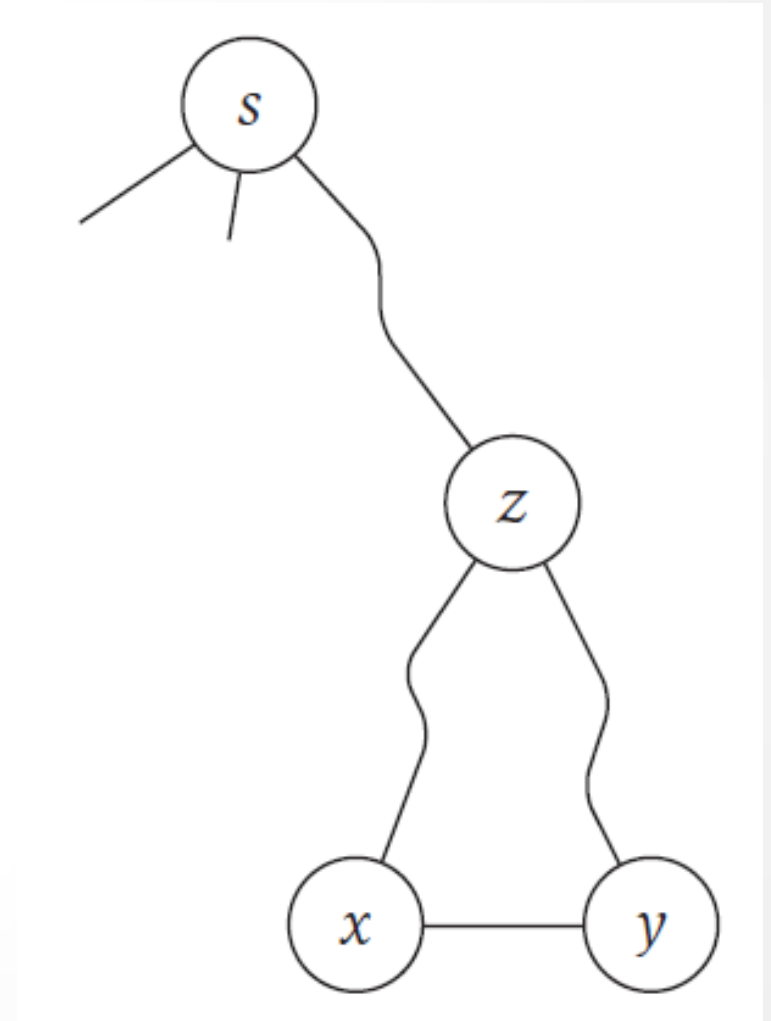
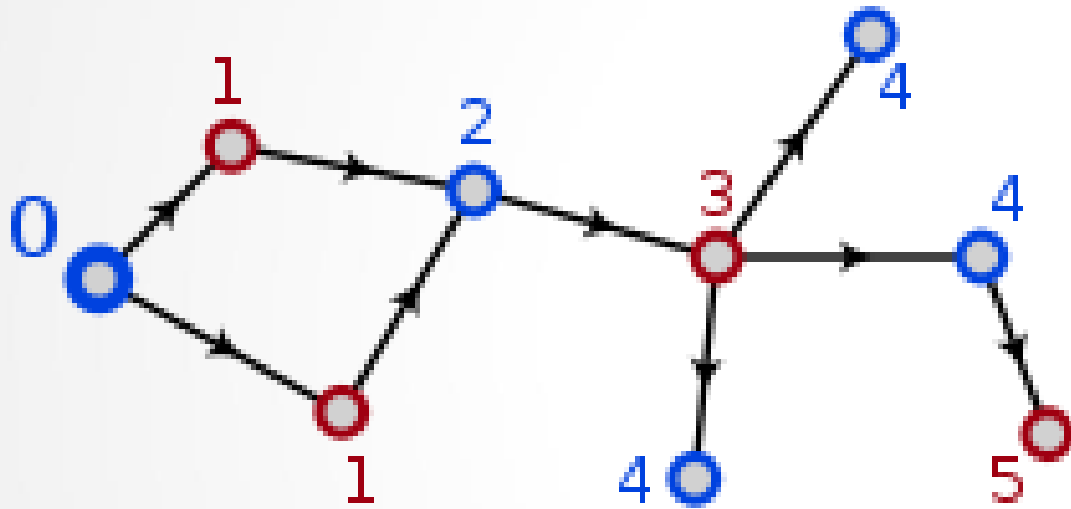
Двудольность

Проверка двудольности с помощью обхода в ширину.

1. Рассчитаем расстояния $\text{Dist}[]$ от начальной вершины до всех остальных.
2. Любой цикл нечётной длины соединяет вершины u и v , для которых $\text{Dist}[u]$ и $\text{Dist}[v]$ имеют одинаковую чётность.



Двудольность



Двудольность

Алгоритм проверки двудольности с помощью обхода в ширину.

Пусть $G(V, E)$ - связный граф.

1. $R = B = \emptyset$
2. Выбрать произвольную вершину $s \in V$. $\text{Dist}[s]=0$.
3. Рассчитать $\text{Dist}[]$ - расстояния от s до всех остальных вершин.
4. For each $v \in V$:
 if $\text{Dist}[v]$ - нечётное: $R = R \cup \{v\}$
 else: $B = B \cup \{v\}$
5. Для всех рёбер проверить выполнение условия.

Временная сложность: $O(n + m)$.

Двудольность

Проверка двудольности с помощью обхода в глубину.

В процессе обхода графа в глубину, разделяем вершины на две доли (R и B):

1. Выбираем произвольную начальную вершину и помещаем её в R .
2. Каждую новую вершину u помещаем в долю, отличную от доли, в которой находится её непосредственный предшественник $\text{Pred}[u]$.
3. Если рассматриваемое ребро (v, u) ведёт в посещённую вершину, то проверяем, что u и v лежат в разных долях.

Временная сложность: $O(n + m)$.

Обход в глубину: пример

