

8. Графы

8.1. Типы графов

Ориентированный граф

Неориентированный граф

8.1. Ориентированный граф

Ориентированный граф G – пара (V, E) ,
где

V – конечное множество вершин графа G ,
 $V = \{v_i\}, i = 1, 2, \dots, n, v_i$ – вершина графа

E – множество ребер графа G ;

упорядоченные пары вершин из V ,

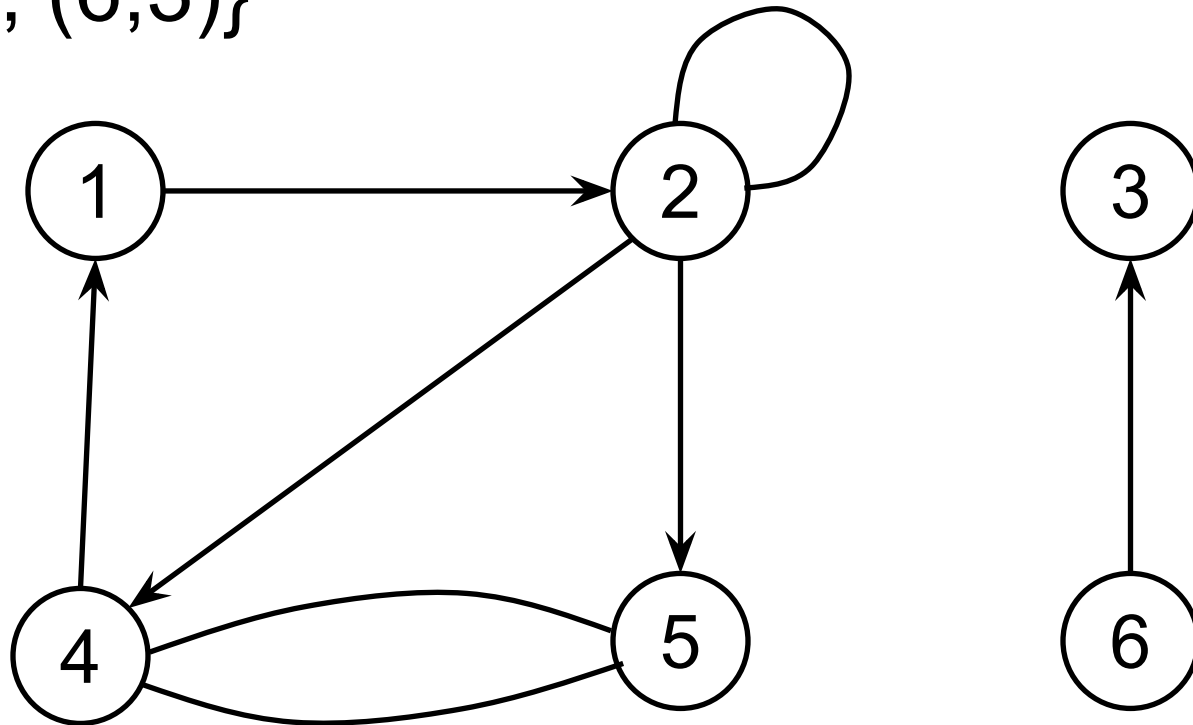
$E = \{e_i\}, e_i = (u, v), u \in V, v \in V,$

e_i – ребро графа; выходит из вершины u
и входит в вершину v

8.1. Ориентированный граф

Пример: $V = \{1, 2, 3, 4, 5, 6\}$,

$E = \{(1,2), (2,2), (2,4), (2,5), (4,1), (4,5),$
 $(5,4), (6,3)\}$



8.2. Неориентированный граф

Неориентированный граф G – пара (V, E) ,
где

V – конечное множество вершин графа G ,

$V = \{v_i\}, i = 1, 2, \dots, n, v_i$ – вершина графа

E – множество ребер графа G ;

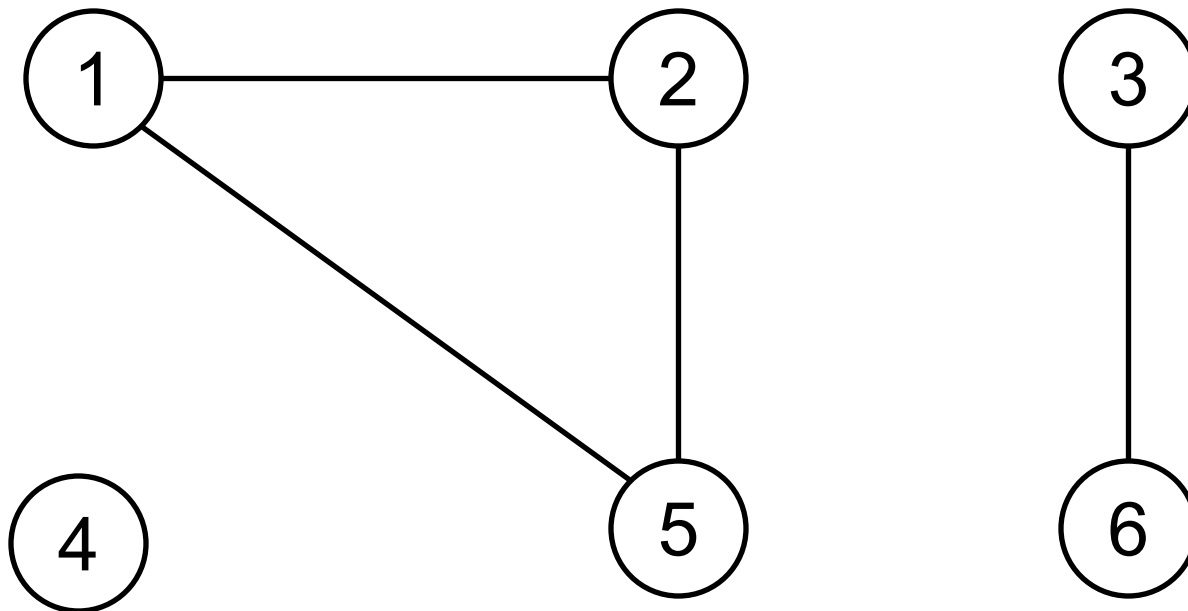
неупорядоченные пары вершин из V ,

$E = \{e_i\}, e_i = (u, v), u \in V, v \in V, u \neq v$

e_i – ребро графа; соединяет вершины u и v

8.2. Неориентированный граф

Пример: $V = \{1, 2, 3, 4, 5, 6\}$,
 $E = \{(1,2), (1,5), (2,5), (3,6)\}$



8.3. Определения

- Если (u, v) – ребро графа G , тогда вершина u графа – *смежная* с вершиной v : $u \square v$
- *Степень вершины* в неориентированном графе – число ребер, соединяющих ее с другими вершинами
- Вершина со степенью 0 – *изолированная*

8.3. Определения

- В ориентированном графе: *исходящая степень* – количество выходящих из вершины ребер, *входящая степень* – количество входящих в вершину ребер
- *Степень вершины* в ориентированном графе = исходящая степень + входящая степень

8.3. Определения

- *Путь (маршрут) длины k* от вершины u к вершине v в графе $G = (V, E)$ – последовательность вершин $\langle v_0, v_1, \dots, v_k \rangle$ такая, что $u = v_0$, $v = v_k$ и $(v_{i-1}, v_i) \in E$ для $i = 1, 2, \dots, k$
- *Длина пути* – количество составляющих его ребер

8.3. Определения

- Путь *содержит* вершины v_0, v_1, \dots, v_k и ребра $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$
- Если имеется путь p из вершины u в вершину v , говорят, что вершина u *достижима* из вершины v по пути p
- Путь является *простым*, если все вершины пути – различны

8.3. Определения

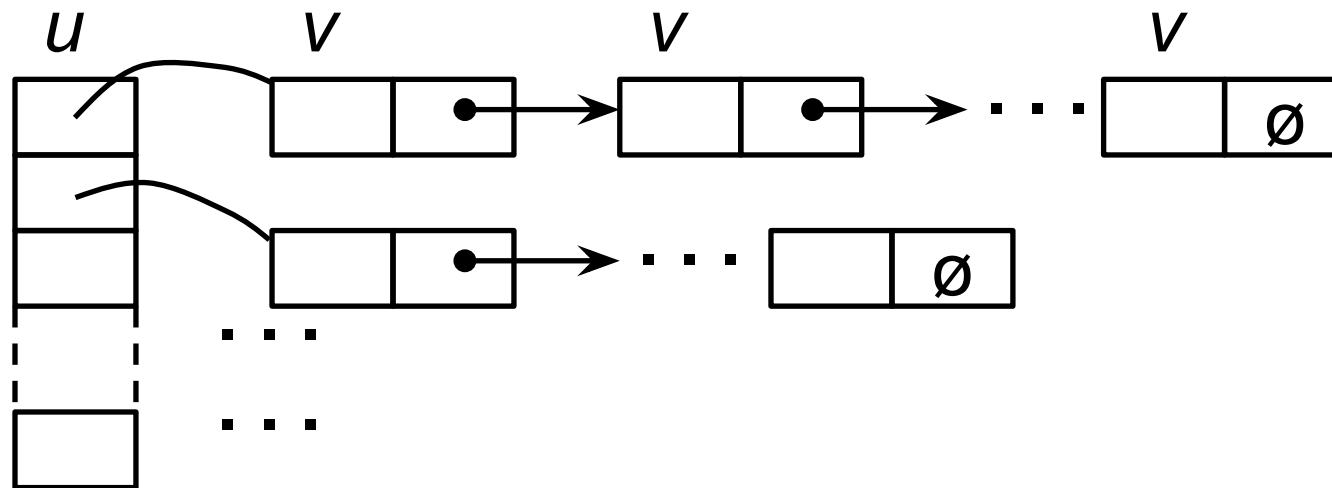
- В ориентированном графе путь $\langle v_0, v_1, \dots, v_k \rangle$ образует *цикл*, если $v_0 = v_k$; цикл *простой*, если все вершины v_0, v_1, \dots, v_{k-1} различны
- *Петля* – цикл с длиной 1
- В неориентированном графе путь $\langle v_0, v_1, \dots, v_k \rangle$ образует (*простой*) *цикл*, если $k \geq 3$, $v_0 = v_k$ и все вершины v_0, v_1, \dots, v_{k-1} различны
- Граф без циклов – *ациклический*

8.3. Определения

- *Взвешенный* граф – граф, с каждым ребром которого связан определенный вес, обычно определяемый *весовой функцией* $w: E \rightarrow R$

8.4. Представление графа

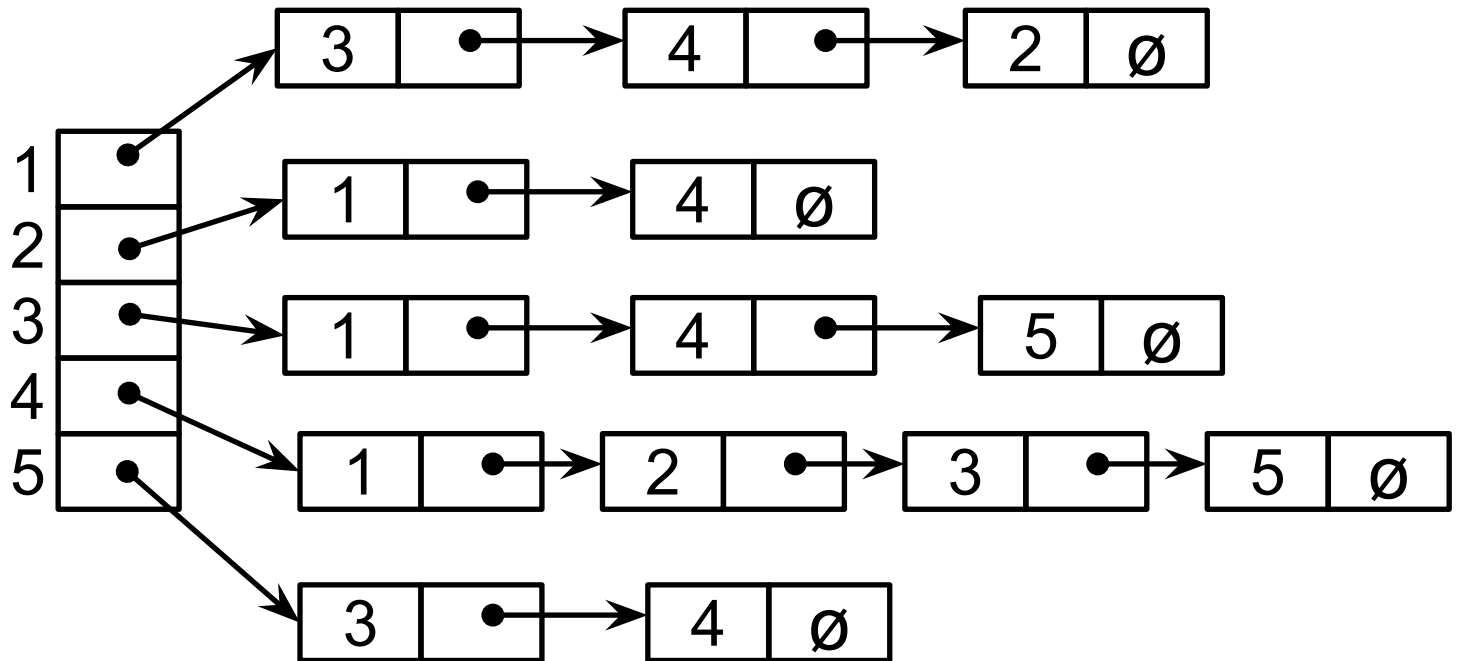
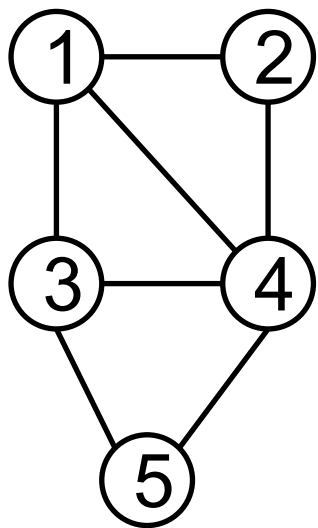
1. Набор списков смежных вершин



Для взвешенных графов вес ребра (u, v) хранится вместе с вершиной v в списке смежности u

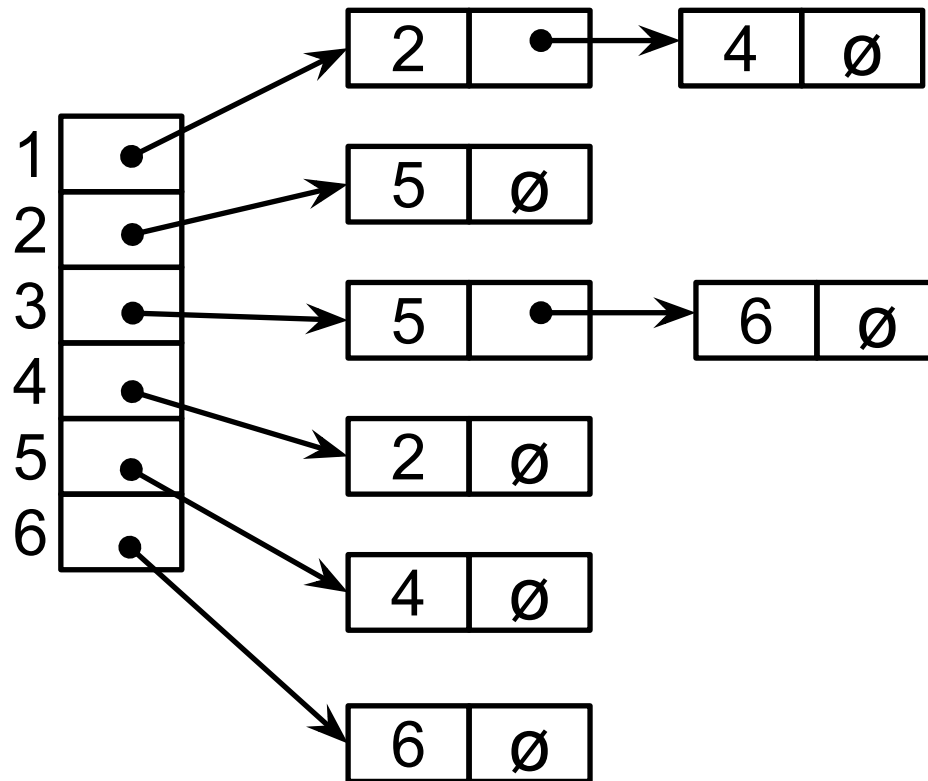
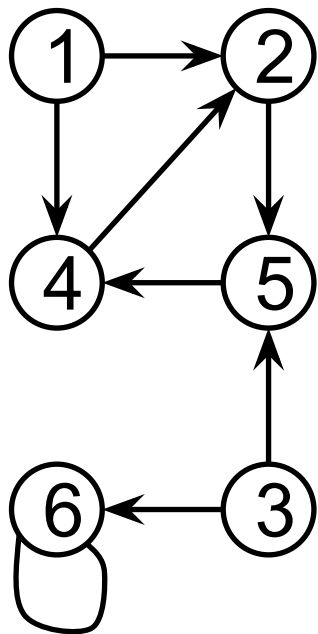
8.4. Представление графа

Пример – неориентированный граф



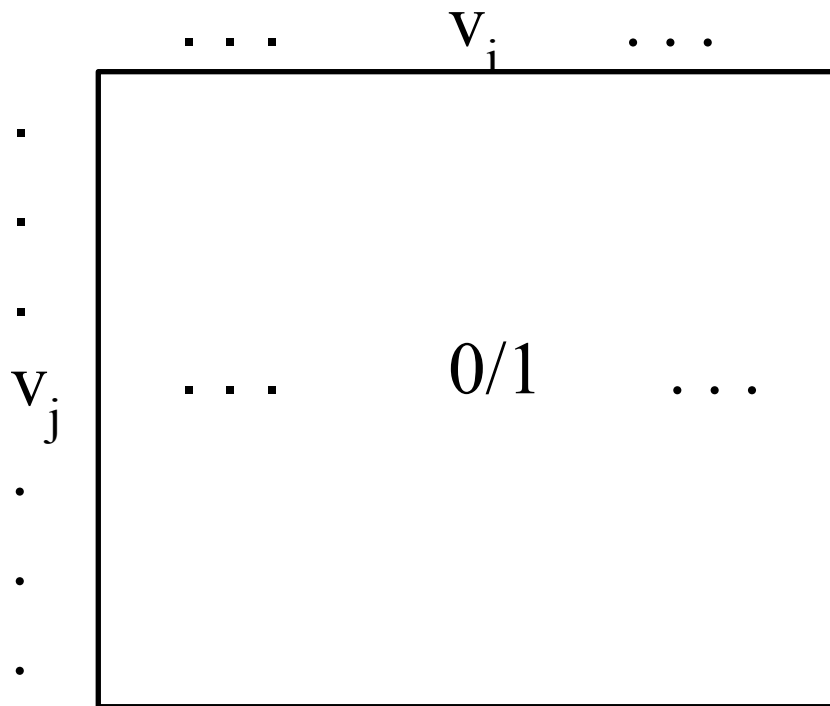
8.4. Представление графа

Пример – ориентированный граф



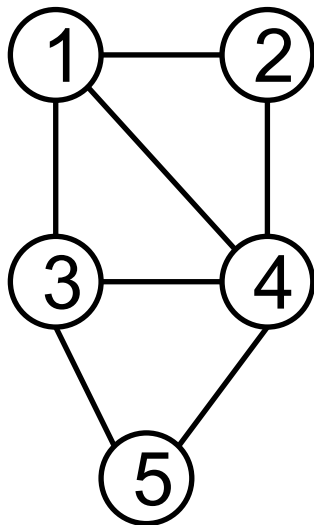
8.4. Представление графа

2. Матрица смежности



8.4. Представление графа

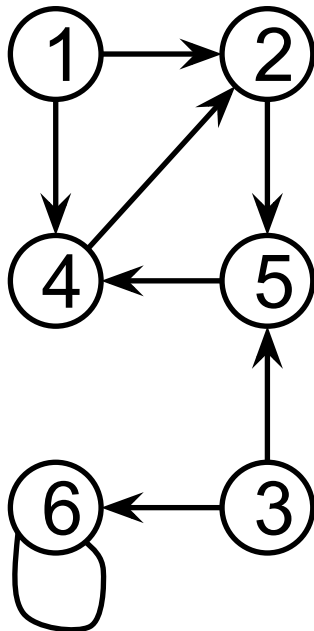
Пример – неориентированный граф



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 |
| 2 | 1 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 0 | 1 |
| 5 | 0 | 0 | 1 | 1 | 0 |

8.4. Представление графа

Пример – ориентированный граф



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 |

8.5. Поиск в ширину

Один из простейших алгоритмов обхода графа

Задан граф $G = (V, E)$

Выделена исходная вершина s

Находятся все вершины, достижимые из s

Строится *дерево поиска в ширину* с корнем s

8.5. Поиск в ширину

Окрашивание вершин графа:

белые – еще не открытые вершины,

серые и черные – открытые вершины,
которые обрабатываются по-разному:

вершины, смежные с черной, серые
или черные;

вершины, смежные с серой, могут быть
белыми

8.5. Поиск в ширину

Корень дерева – s

Сканируется список смежности открытой вершины u : если открывается белая вершина v , то вершина v и ребро (u, v) добавляются в дерево

u – предшественник (или родитель) v в дереве поиска вширь, v – потомок u

8.5. Алгоритм поиска BFS(G, s)

Обозначения:

$Adj[u]$ – список смежности для вершины u

$color[u]$ – цвет вершины u

$pred[u]$ – предшественник вершины u ; если предшественника нет, $pred[u] = NULL$

$d[u]$ – расстояние от s до вершины u

Q – очередь для работы с множеством серых вершин

8.5. Алгоритм поиска BFS(G, s)

Инициализация:

Для каждой вершины $u \in V[G]$, кроме s {

$color[u] = \text{белый}$

$d[u] = \infty$

$pred[u] = \text{NULL}$

}

$color[s] = \text{серый}$

$d[s] = 0$

$pred[s] = \text{NULL}$

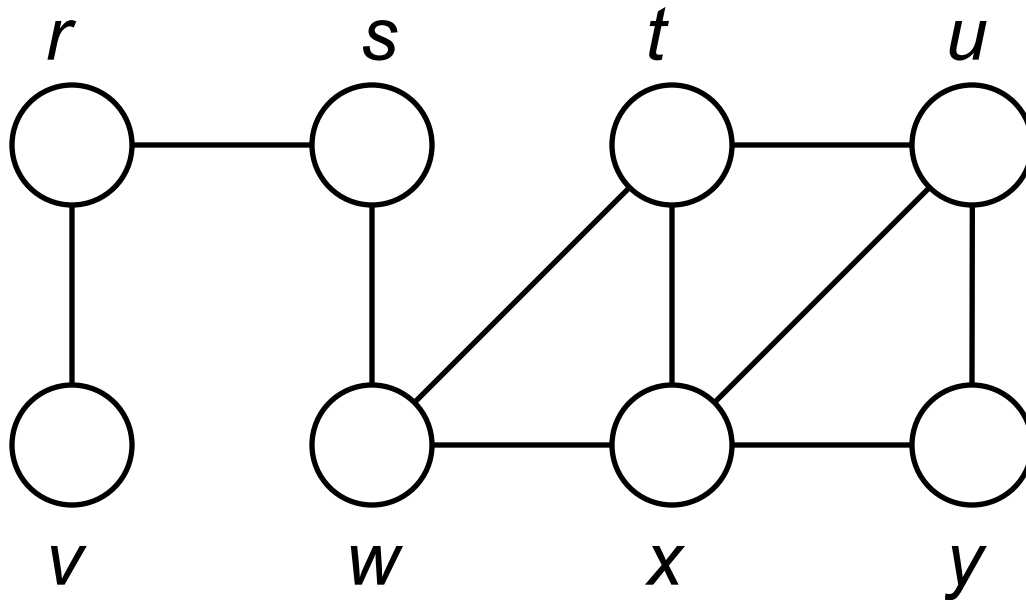
Записать s в очередь Q

8.5. Алгоритм поиска BFS(G, s)

```
while Q не пуста {  
     $u$  = очередная вершина из Q  
    для каждой  $v \in Adj[u]$   
        if  $color[v]$  = белый {  
             $color[v]$  = серый  
             $d[v] = d[u] + 1$   
             $pred[v] = u$   
            Записать  $v$  в Q  
        }  
     $color[u]$  = черный  
}
```

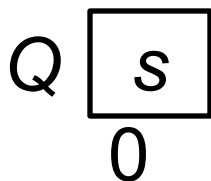
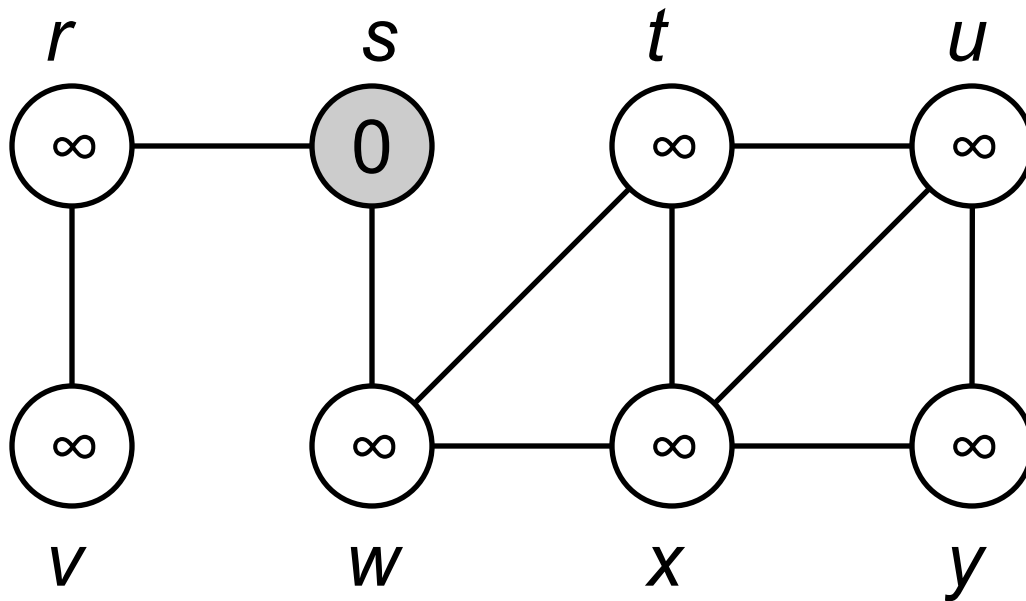

8.6. Пример

Исходный граф



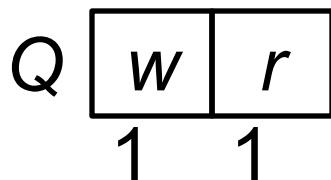
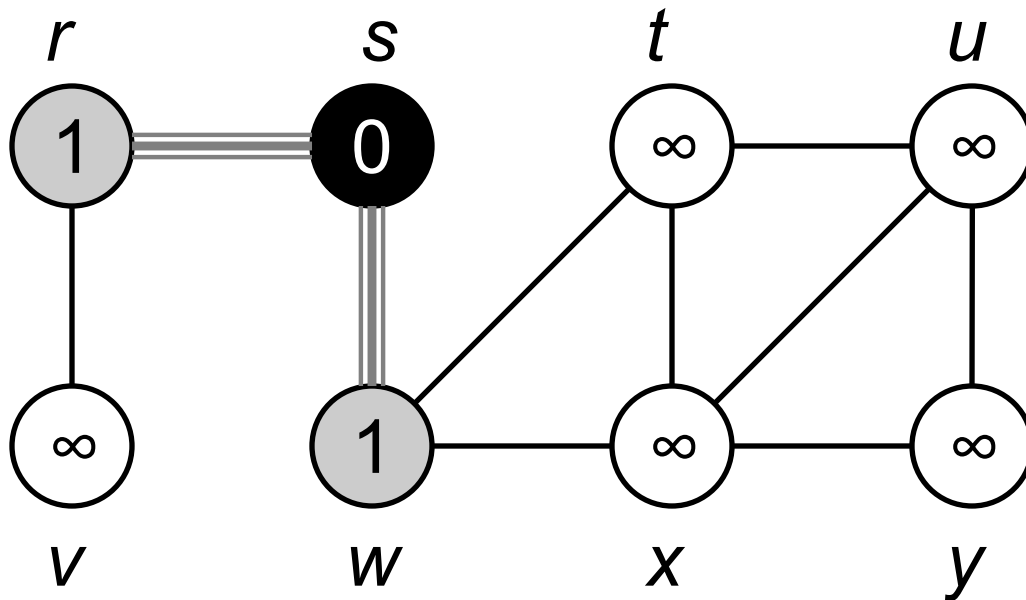
8.6. Пример

Инициализация



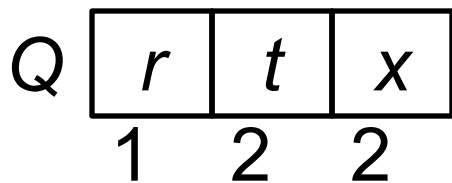
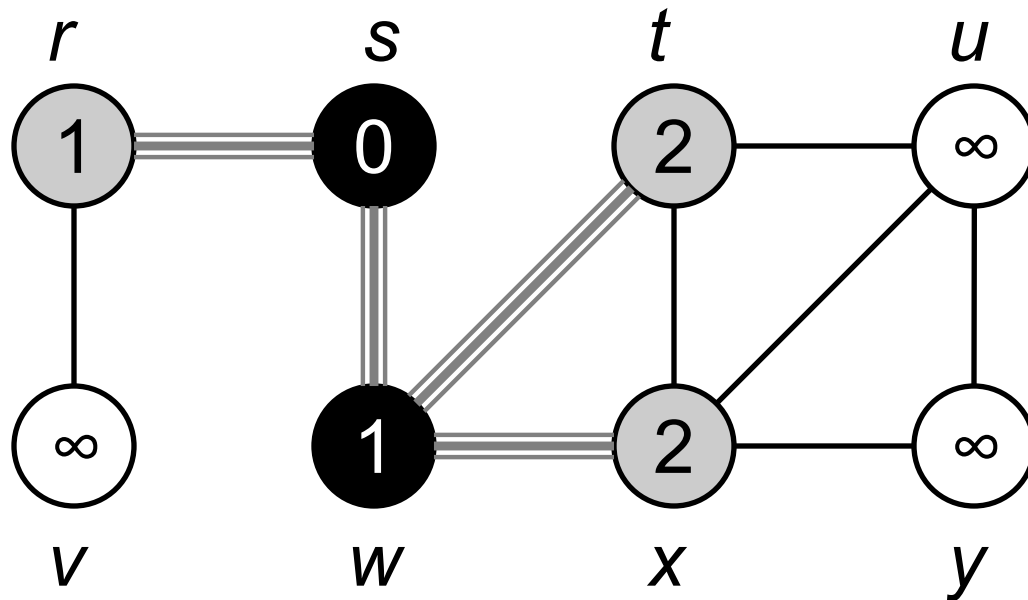
8.6. Пример

1-я итерация цикла while



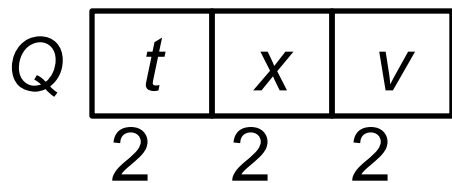
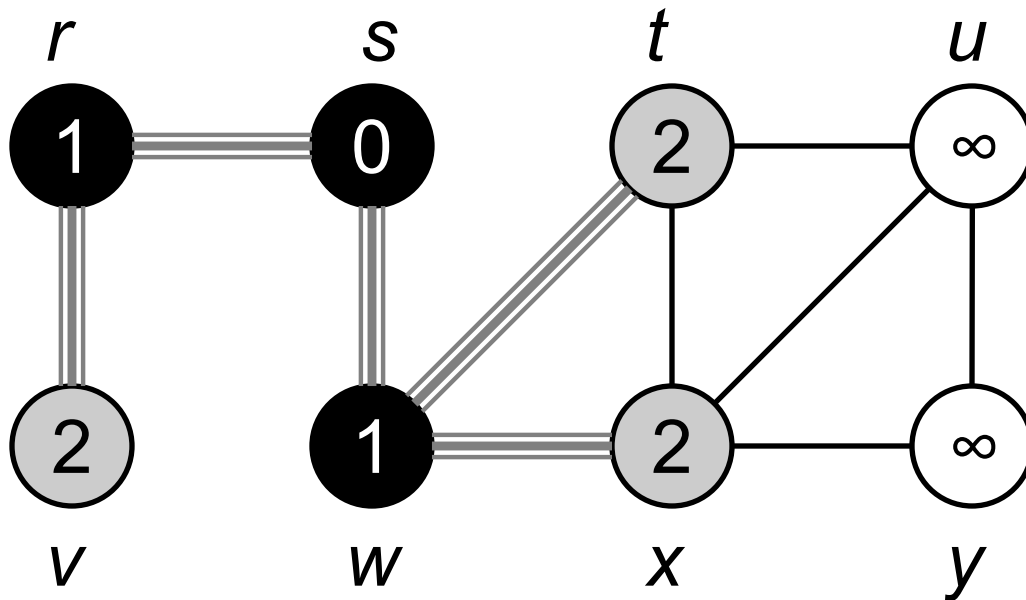
8.6. Пример

2-я итерация цикла while



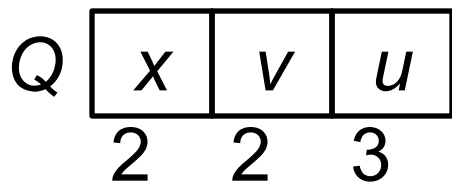
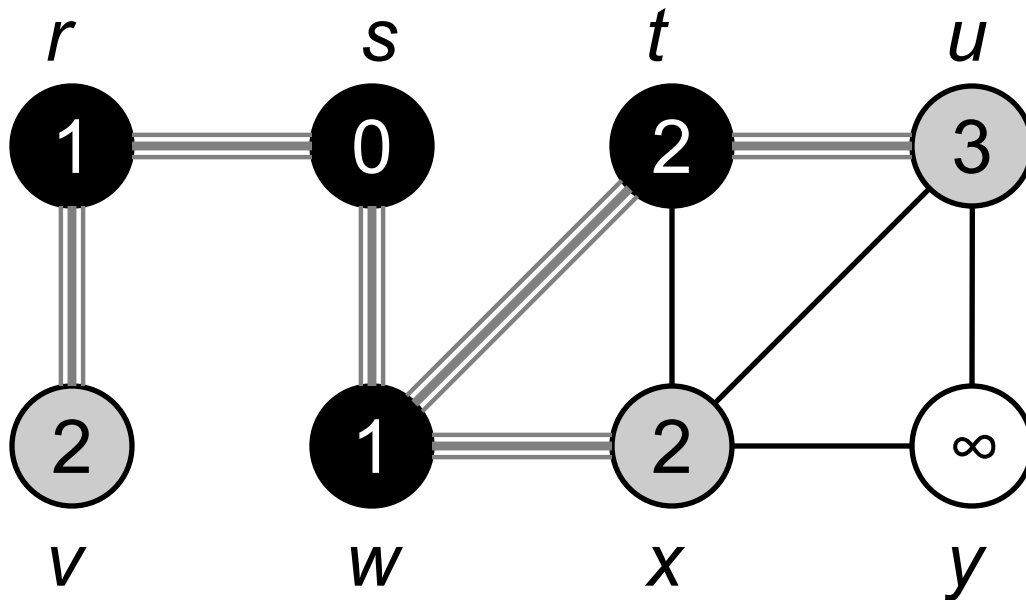
8.6. Пример

3-я итерация цикла while



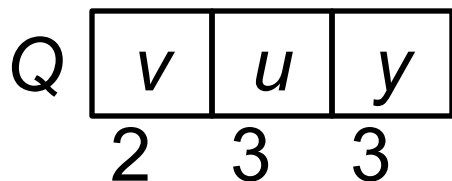
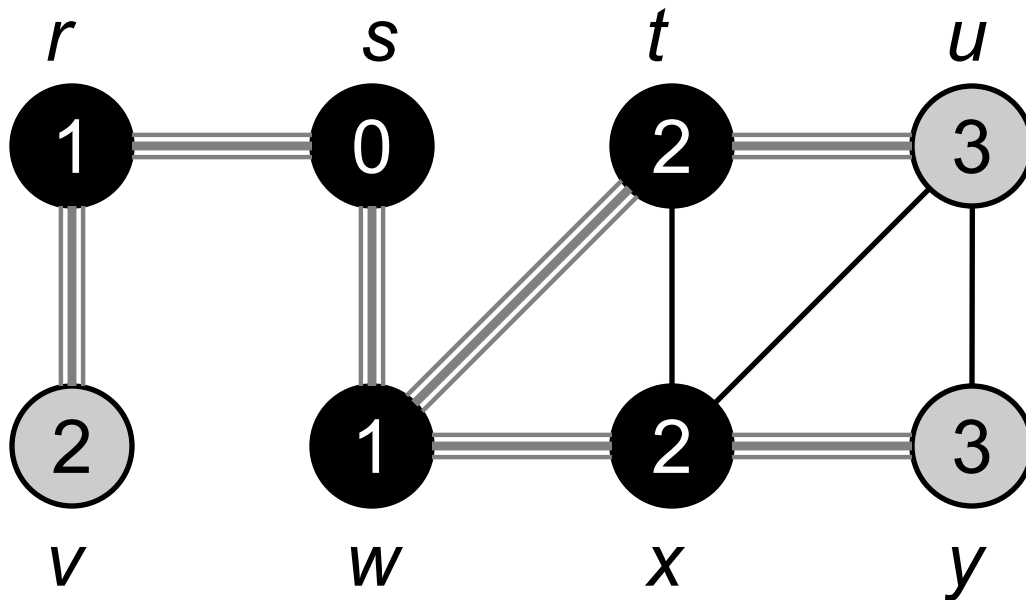
8.6. Пример

4-я итерация цикла while



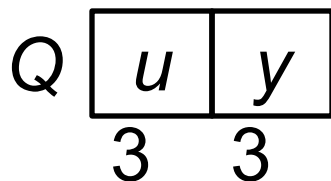
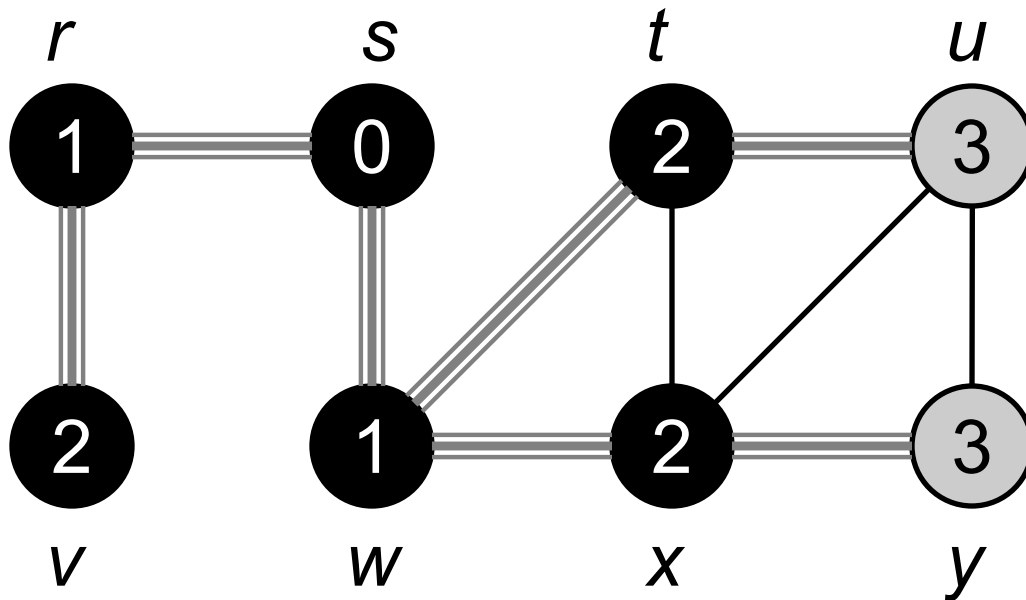
8.6. Пример

5-я итерация цикла while



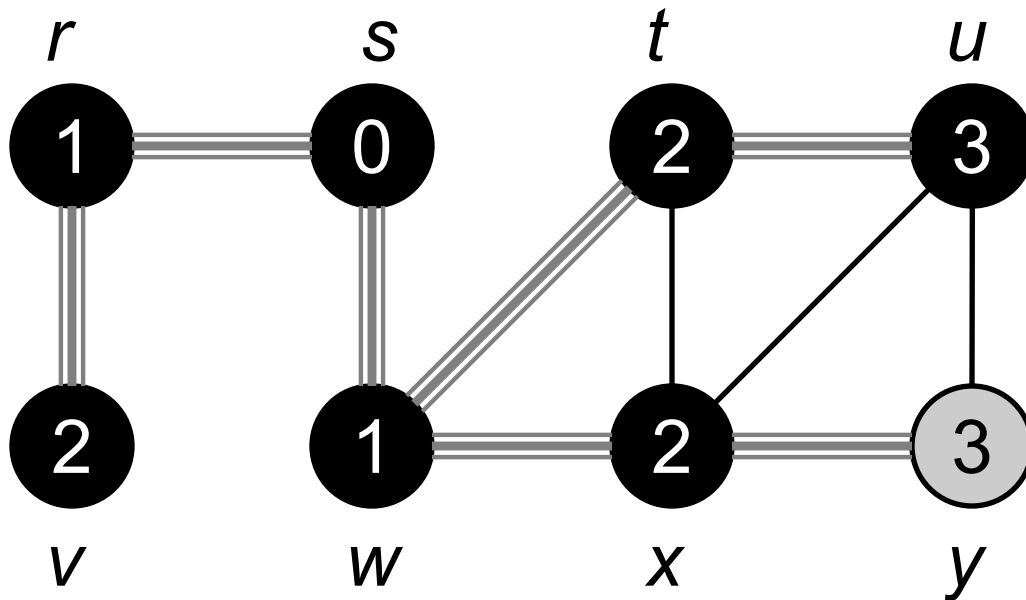
8.6. Пример

6-я итерация цикла while



8.6. Пример

7-я итерация цикла while



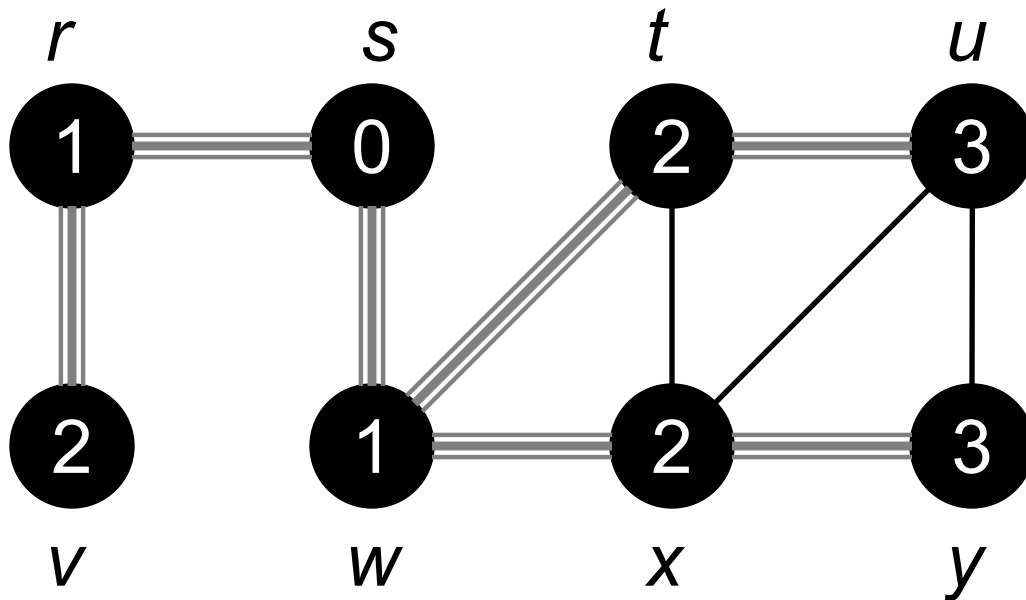
Q

| |
|-----|
| y |
|-----|

3

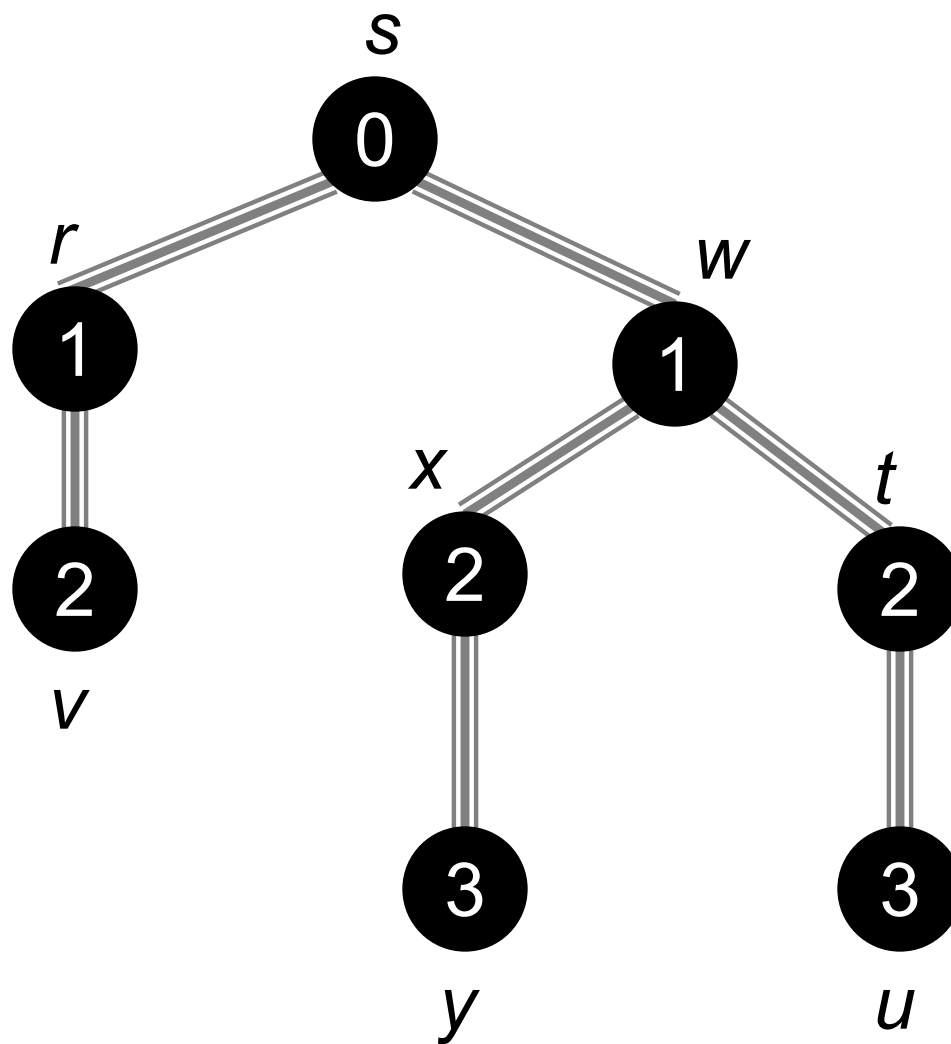
8.6. Пример

8-я итерация цикла while



$Q \ \emptyset$

8.6. Пример



8.7. Анализ алгоритма

Общее время операций с очередью – $O(V)$

Сумма длин всех списков смежности – $\Theta(E)$

Общее время сканирования списков – $O(E)$

Накладные расходы на инициализацию – $O(V)$

Общее время работы алгоритма BFS – $O(V + E)$

8.8.

8..

8..

8..

8..

8..

8..

8..

8..

8..

8..

8..

8..

8..

8..

8..

8..

8..