Семинар 2

Графы. Деревья.

Бирюков Владимир

МФТИ

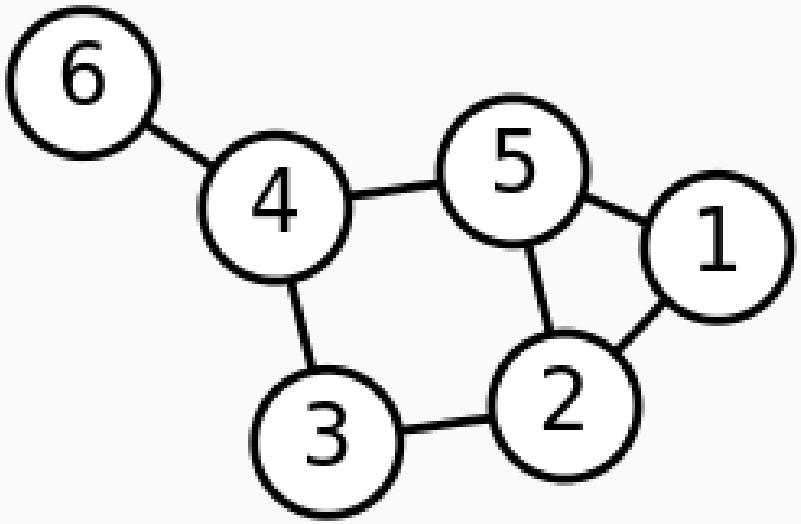
Введение в теорию графов

Граф

Граф – это математический объект, совокупность:

- $V =$ вершины
- $E =$ ребра

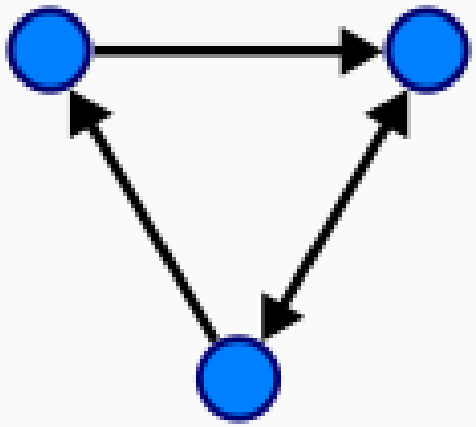
Обозначается как $G = (V, E)$
$n = |V|$ – число вершин $m = |E|$ – число рёбер



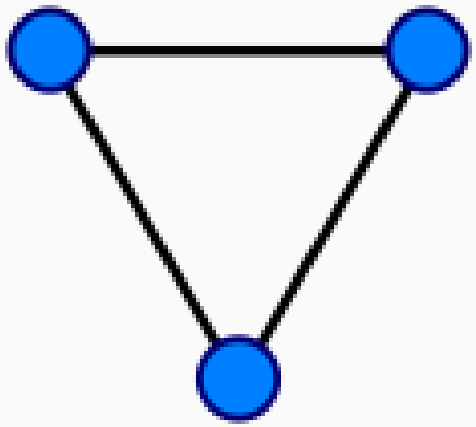
Граф

1. связный граф – если для любых вершин u,v есть путь из u в v.
2. взвешенный граф – если каждому ребру графа поставлено в соответствие некоторое число, называемое весом ребра.
3. простой граф – если он не имеет петель и кратных рёбер.
4. ациклический граф – если он не имеет циклов.
5. дерево – если он связный и ациклический.

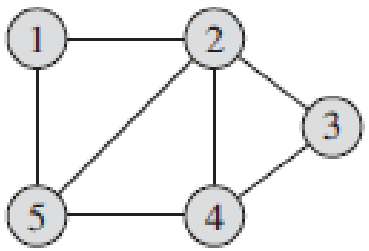
Ориентированный



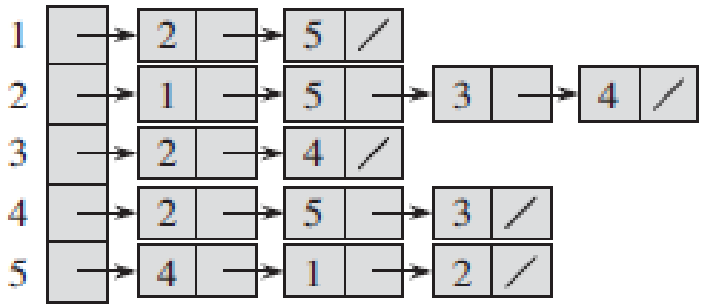
Неориентированный



Граф. Представления. Список смежных вершин. Матрица смежности.



(a)

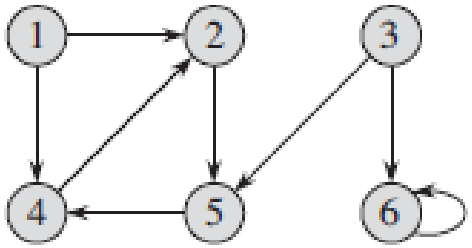


(b)

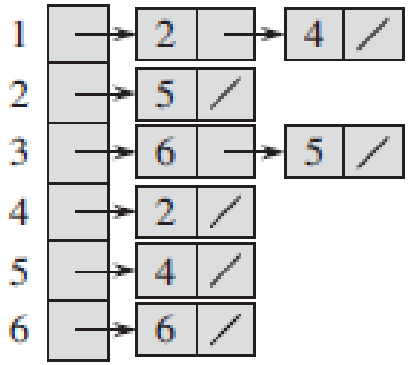
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

Граф. Представления. Список смежных вершин. Матрица смежности.



(a)



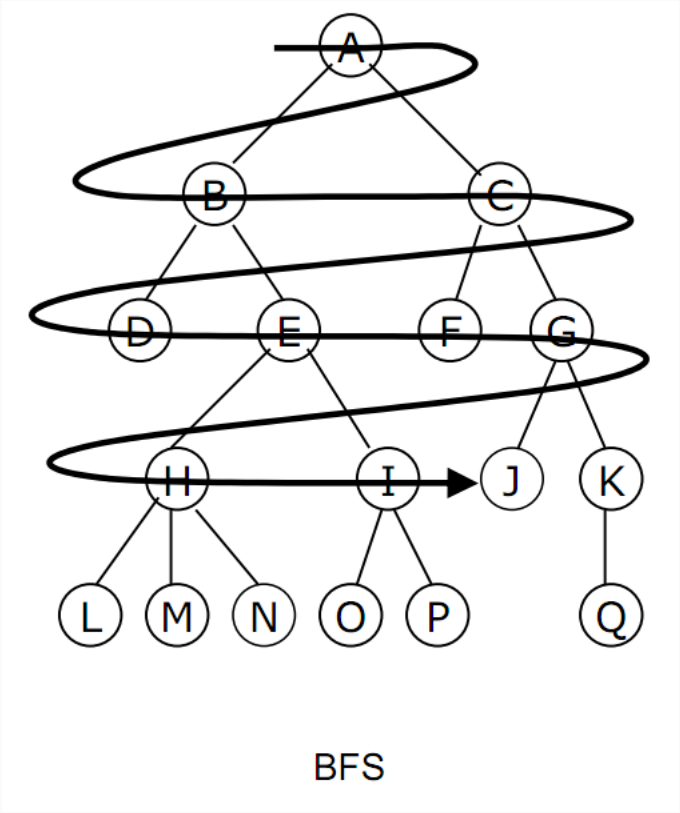
(b)

	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

(c)

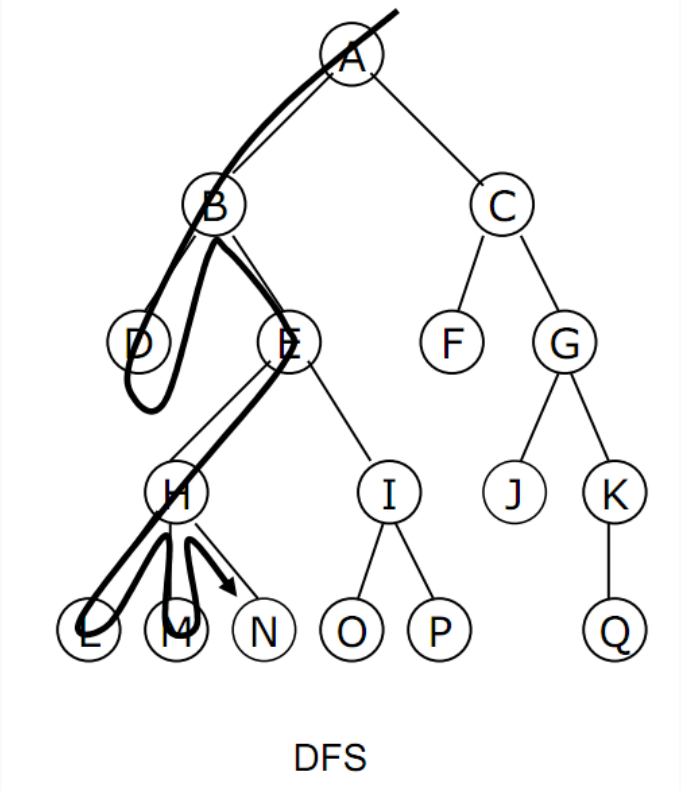
Граф

Поиск в ширину



BFS

Поиск в глубину

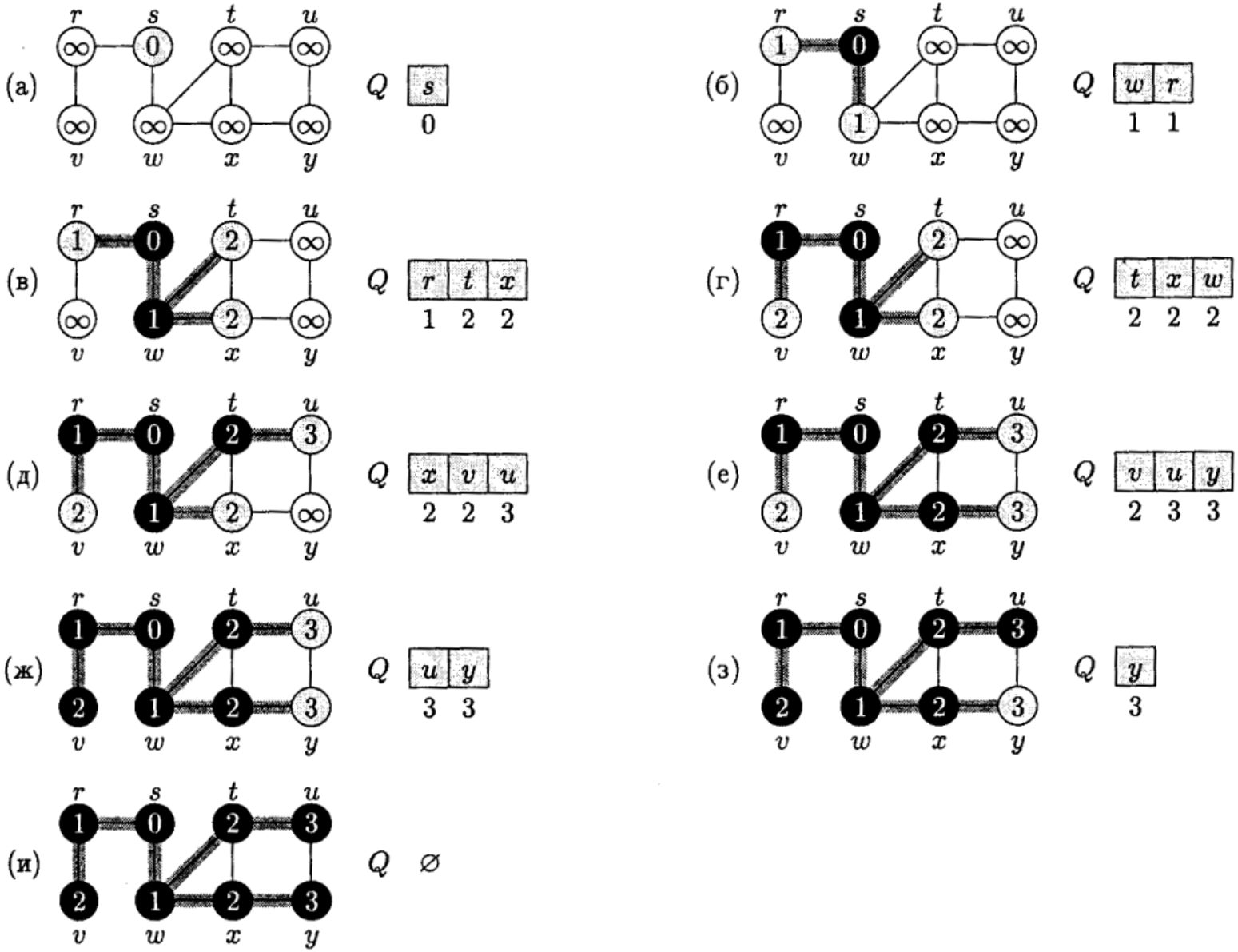


DFS

Граф. Псевдокод алгортма для BFS.

```
BFS(G, s)
 1  for (для) каждой вершины u ∈ V[G] − {s}
 2        do color[u] ← БЕЛЫЙ
 3           d[u] ← ∞
 4           π[u] ← NIL
 5  color[s] ← СЕРЫЙ
 6  d[s] ← 0
 7  π[s] ← NIL
 8  Q ← {s}
 9  while Q ≠ ∅
10        do u ← head[Q]
11           for (для) всех v ∈ Adj[u]
12                do if color[v] = БЕЛЫЙ
13                      then color[v] ← СЕРЫЙ
14                           d[v] ← d[u] + 1
15                           π[v] ← u
16                           ENQUEUE(Q, v)
17           DEQUEUE(Q)
18           color[u] ← ЧЁРНЫЙ
```

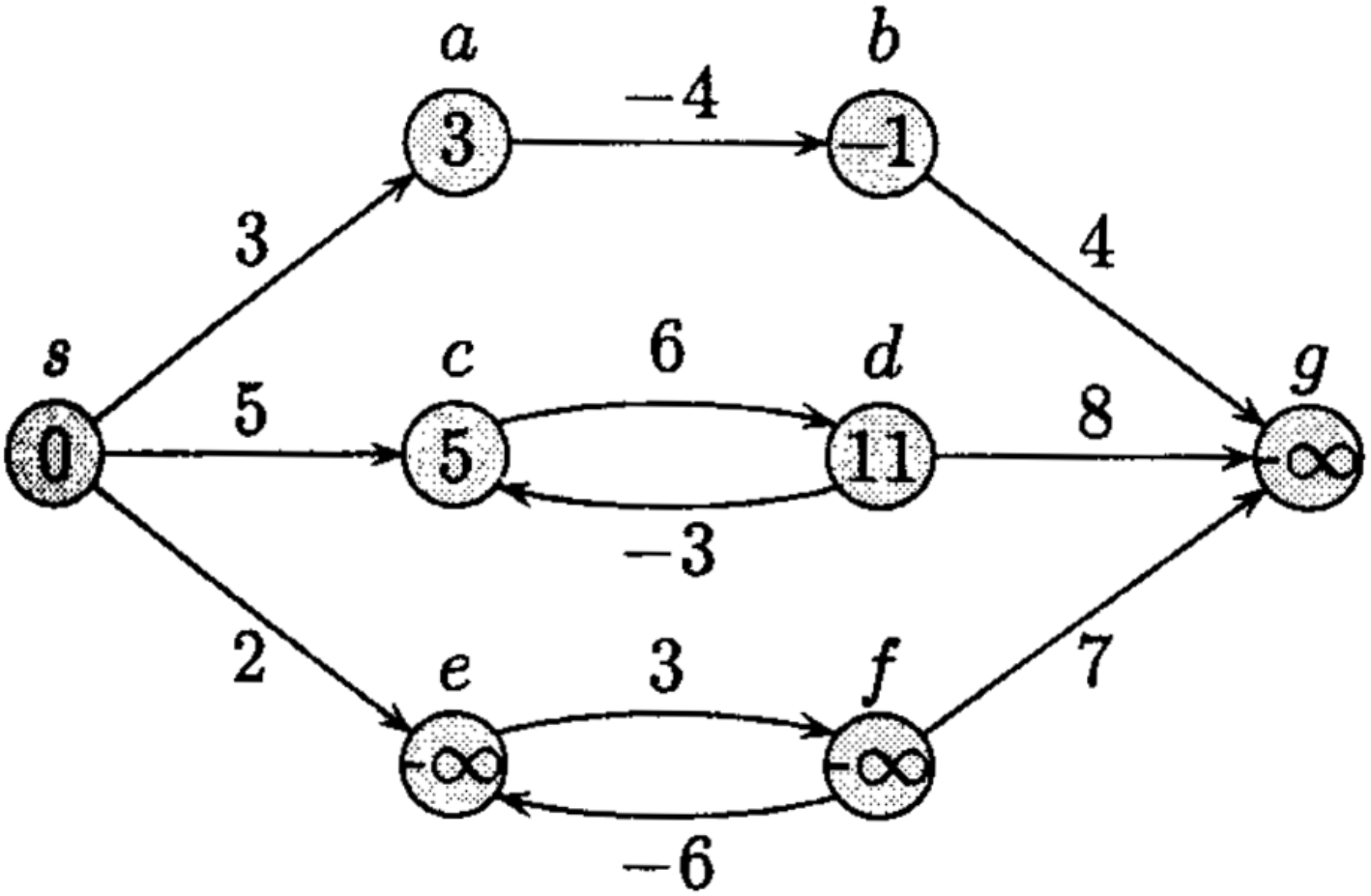
Граф. Алгортм BFS.



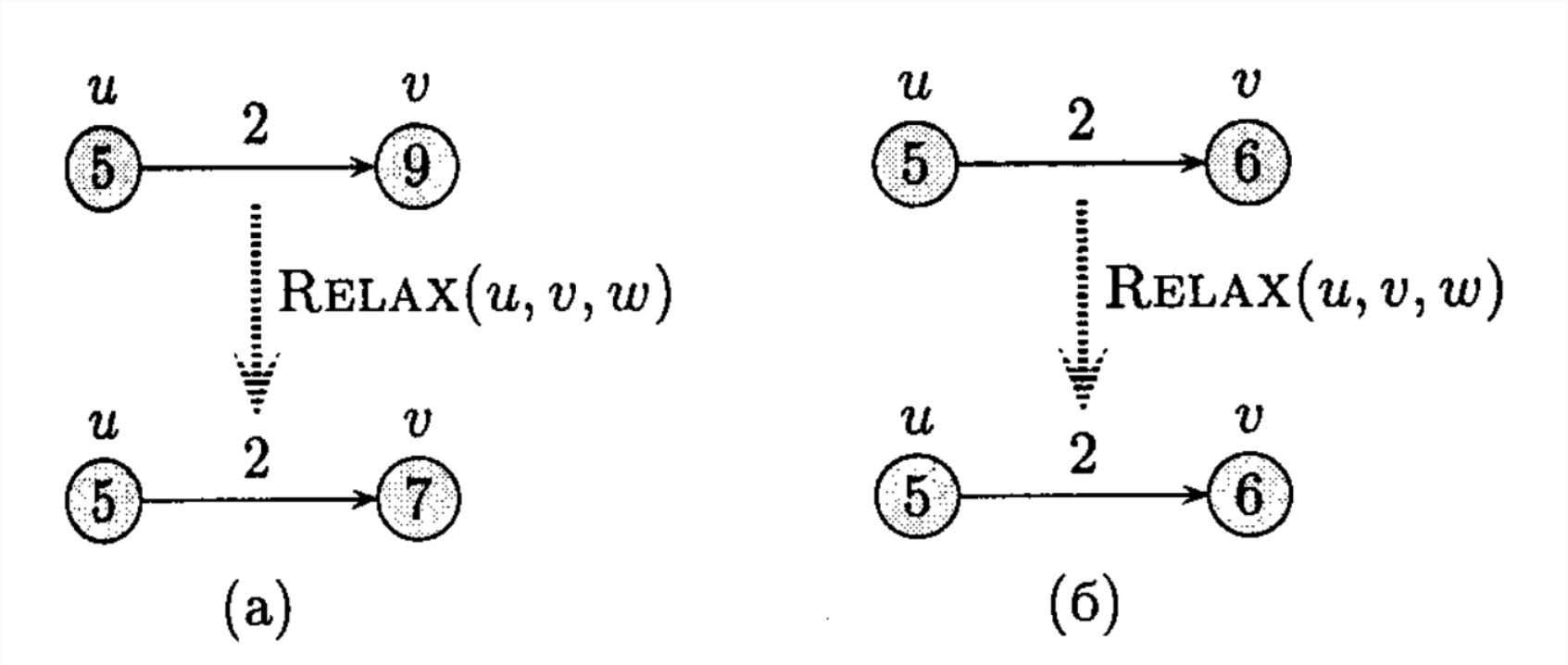
Очередь с приоритетом.

- Абстрактный тип данных в программировании, поддерживающий две обязательные операции — добавить элемент и извлечь минимум.
- Предполагается, что для каждого элемента можно вычислить его приоритет.
- Простейшая реализация – с помощью структуры данных куча(heap).
- Асимптотическая сложность операций добавления и извлечения минимального в простейшей реализации – $O(log(n))$.

Граф. Кратчайшие пути из одной вершины.



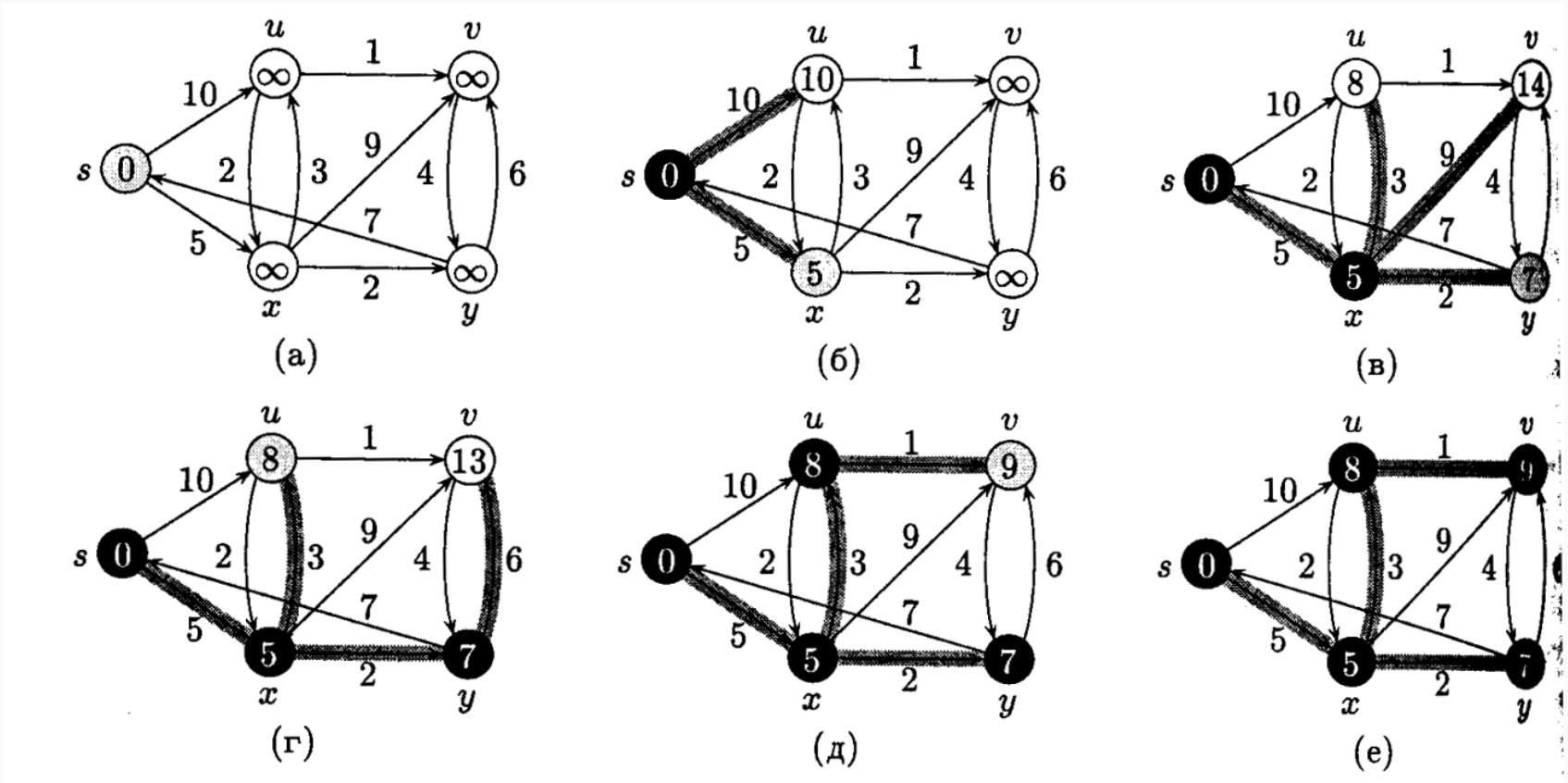
Граф. Релаксация.



Граф. Алгоритм Дейкстры.

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S ← ∅
3  Q ← V[G]
4  while Q ≠ ∅
5        do u ← EXTRACT-MIN(Q)
6           S ← S ∪ {u}
7           for (для) всех вершин v ∈ Adj[u]
8                 do RELAX(u, v, w)
```

Граф. Алгоритм Дейкстры.



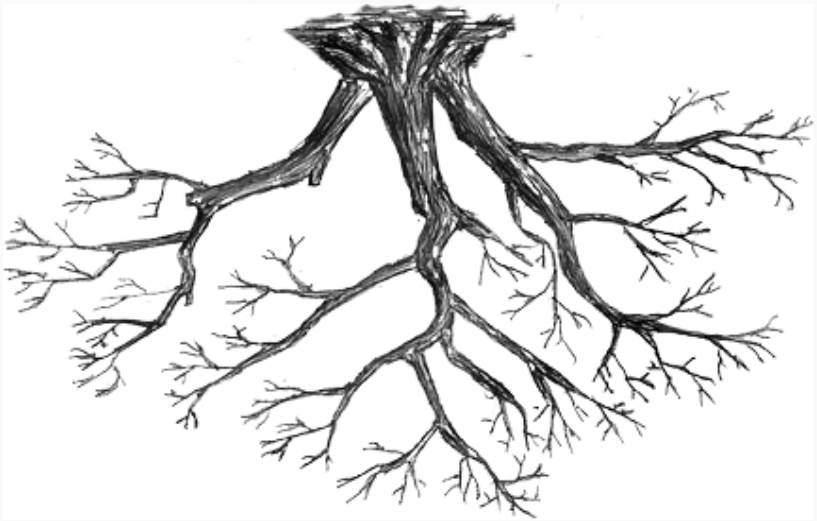
Граф. Сложности работы алгоритмов.

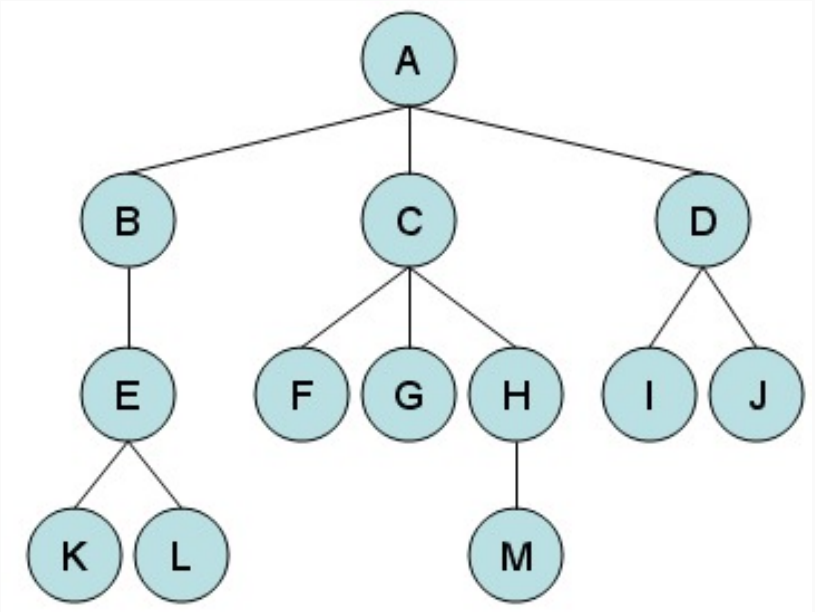
- BFS – $O(|V| + |E|)$
- DFS – $O(|V| + |E|)$
- Алгоритм Дейкстры – $O(|V|^2 + |E|)$
- Алгоритм Беллмана-Форда – $O(|V| * |E|)$
  (алгоритм нахождения кратчайших путей из одной вершины если есть отрицательные веса)
- Алгоритм Флойда-Уоршолла – $O(|V|^3)$
  (алгоритм нахождения кратчайших путей для всех пар вершин)

Деревья

Деревья

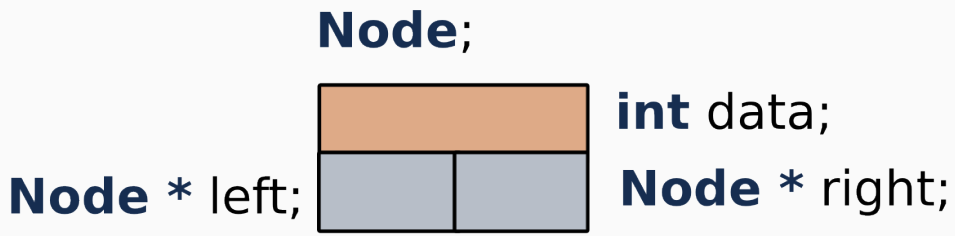
Дерево – это связный граф без циклов.





Двоичное дерево – если у каждого узла не более двух потомков.

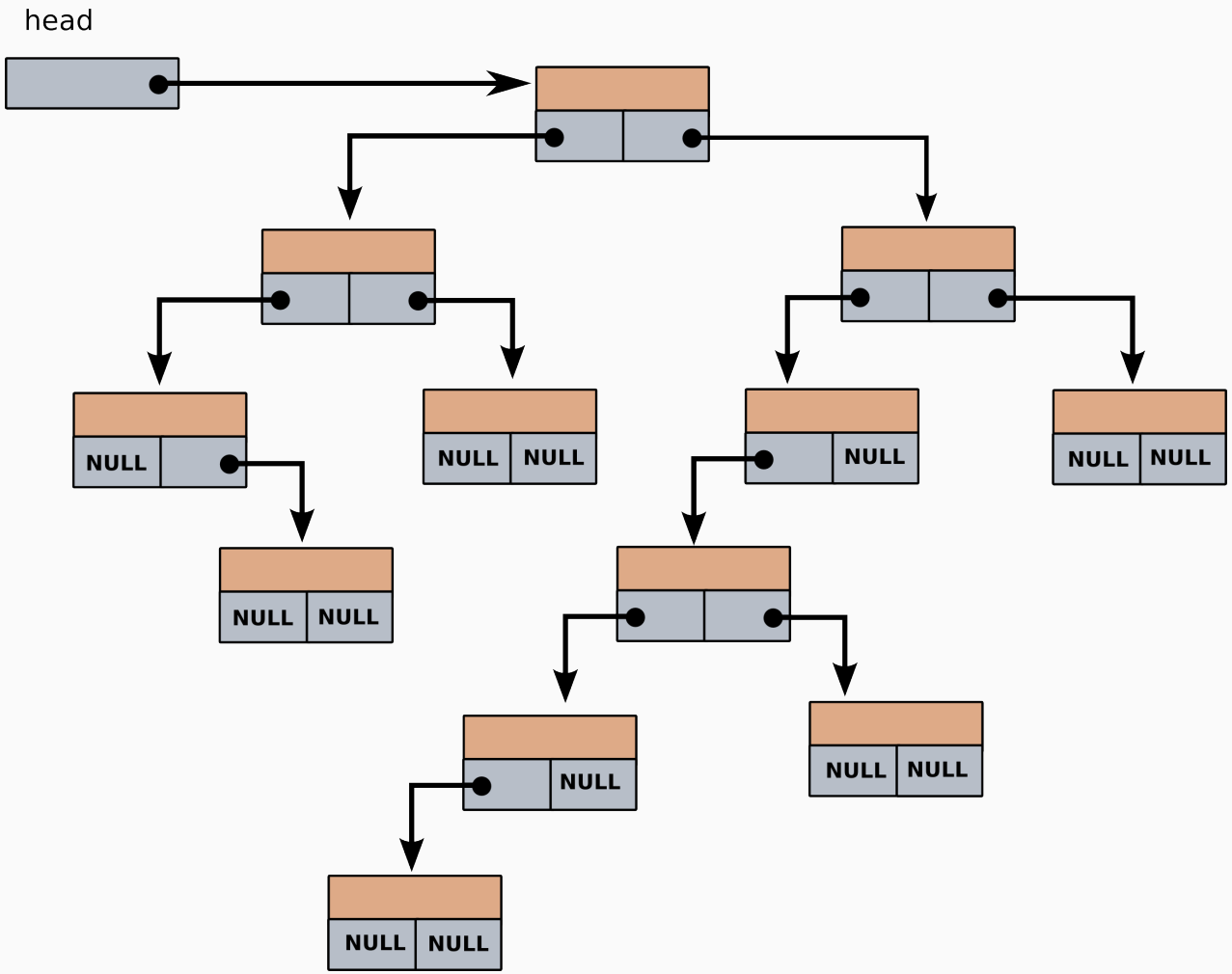
Двоичные деревья



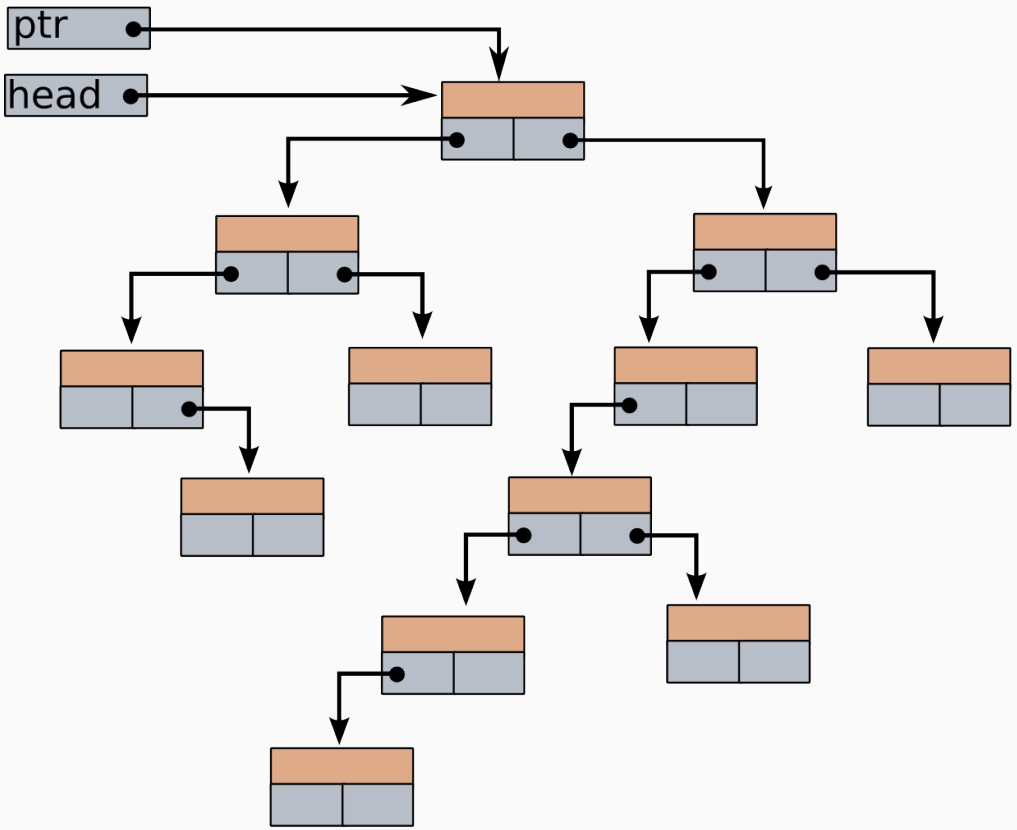
Код: Узел бинарного дерева

```
struct node {
    int data;
    struct node * left;
    struct node * right;
}
typedef struct node Node;
```

Двоичные деревья



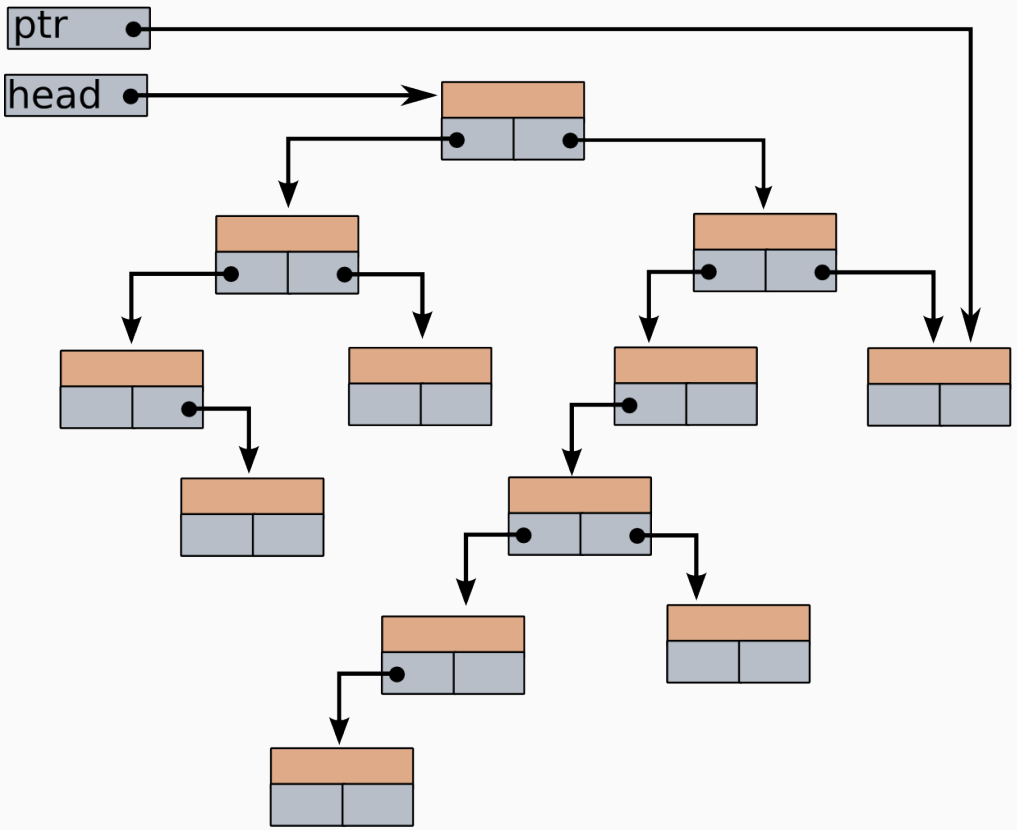
Двоичные деревья (обход)



Код: Обход дерева

```
ptr = head;
```

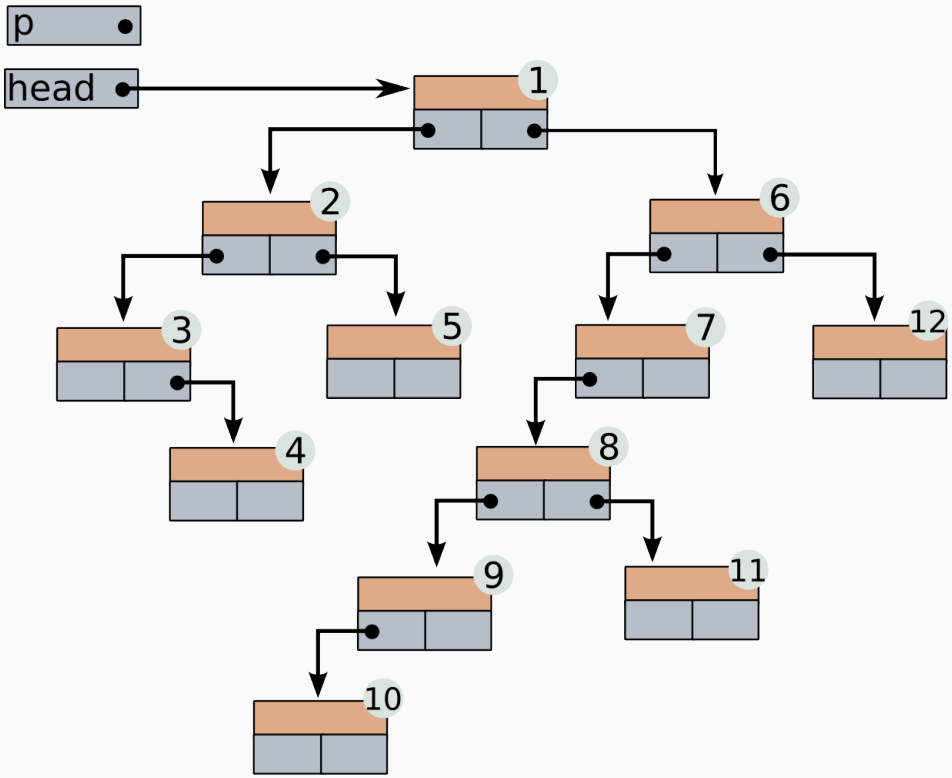
Двоичные деревья (обход)



Код: Обход дерева

```
ptr = head;
ptr = ptr->right;
ptr = ptr->right;
```

Двоичные деревья (обход)



Код: Обход дерева

```
int print_tree(Node* p)
{
  if ( p )
  {
    printf("%d ", p->data);
    print_tree(p->left);
    print_tree(p->right);
  }
}
```

Двоичные деревья поиска

- Двоичные деревья поиска – это двоичное дерево, обладающее следующим свойством:
- Пусть x – произвольная вершина двоичного дерева поиска. Если вершина y находится в левом поддереве вершины x, то $y.data \leq x.data$. Если y находится в правом поддереве вершины x, то $y.data \geq x.data$.

Двоичные деревья поиска