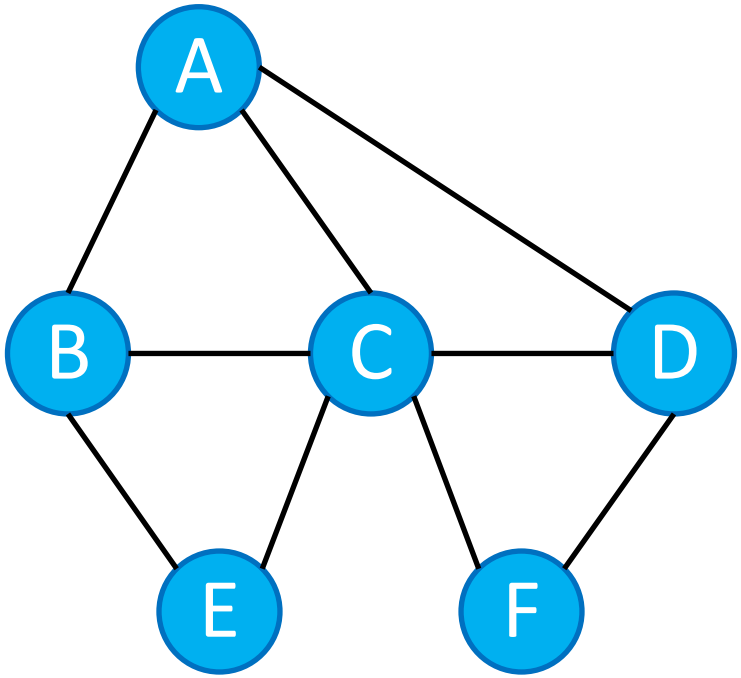
Способы представления графов в компьютере. Обходы графов.

Практикум 3 курс
Осень 2015

Поиск в ширину

```
BFS(G, s)
        foreach v∈V[G]\s
                color[u]:=WHITE
                d[u]:=INF
                p[u]:=NIL
        color[s]:=GRAY
        d[s]:=0
        p[s]:=NIL
        Q.push(s)//Q – очередь
        while !Q.empty()
                u:=Q.pop()
                foreach v ∈Adj[u]
                        if color[v]==WHITE
                                color[v]:=GRAY
                                d[v]:=d[u]+1
                                p[v]:=u
                                Q.push(v)
                color[u]:=BLACK
```

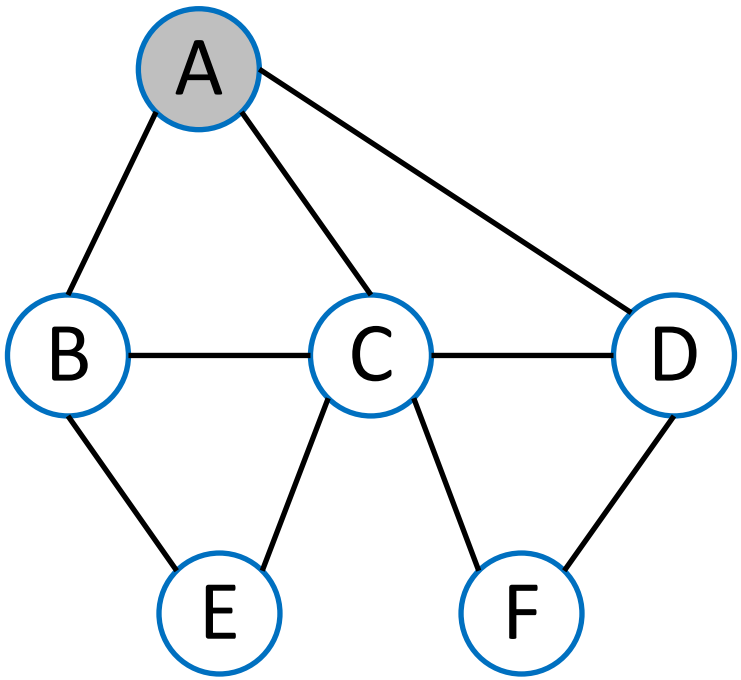
Поиск в ширину - пример



	d	p
A		
B		
C		
D		
E		
F		

Q

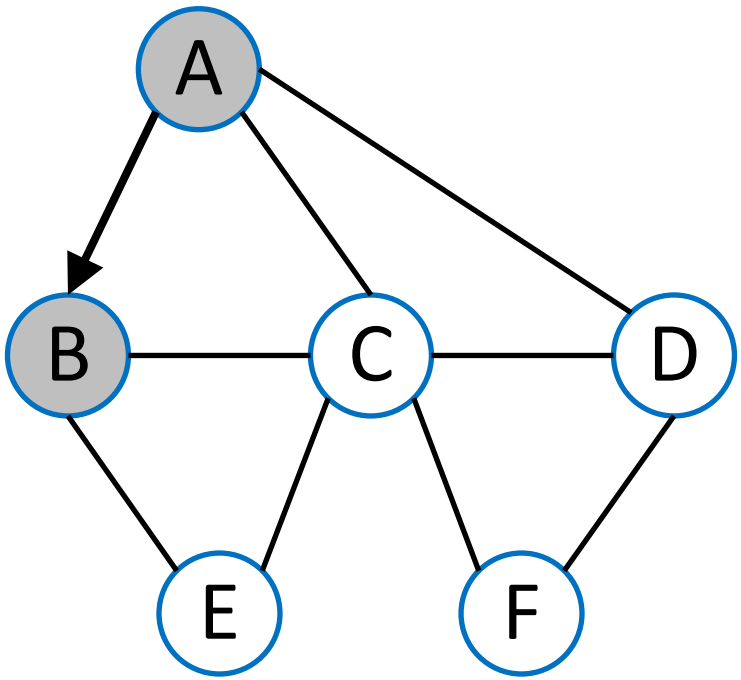
Поиск в ширину - пример



	d	p
A	0	NIL
B	INF	NIL
C	INF	NIL
D	INF	NIL
E	INF	NIL
F	INF	NIL

Q A

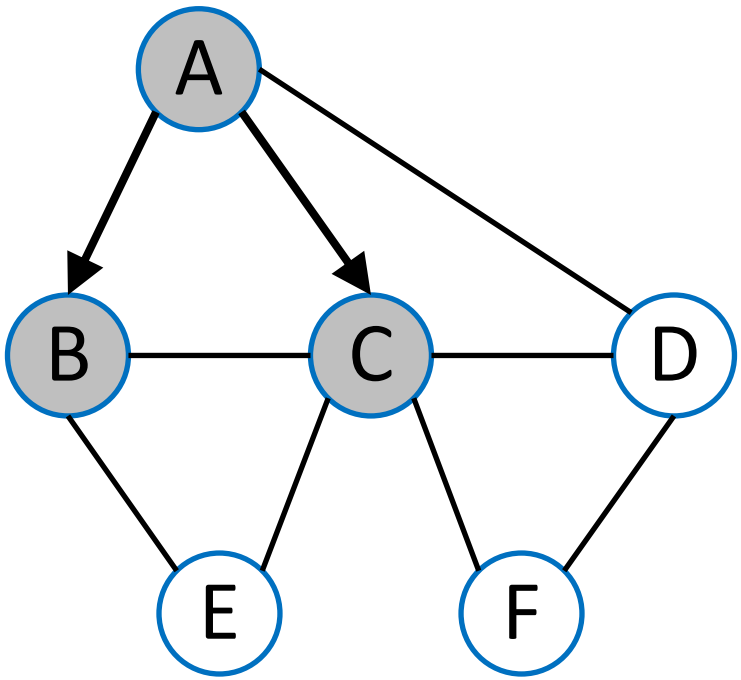
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	INF	NIL
D	INF	NIL
E	INF	NIL
F	INF	NIL

Q B

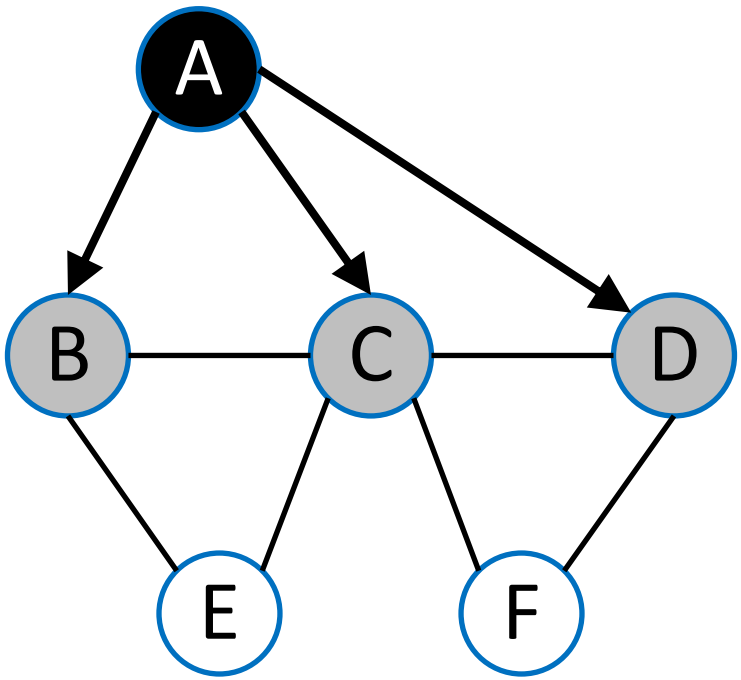
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	1	A
D	INF	NIL
E	INF	NIL
F	INF	NIL

Q

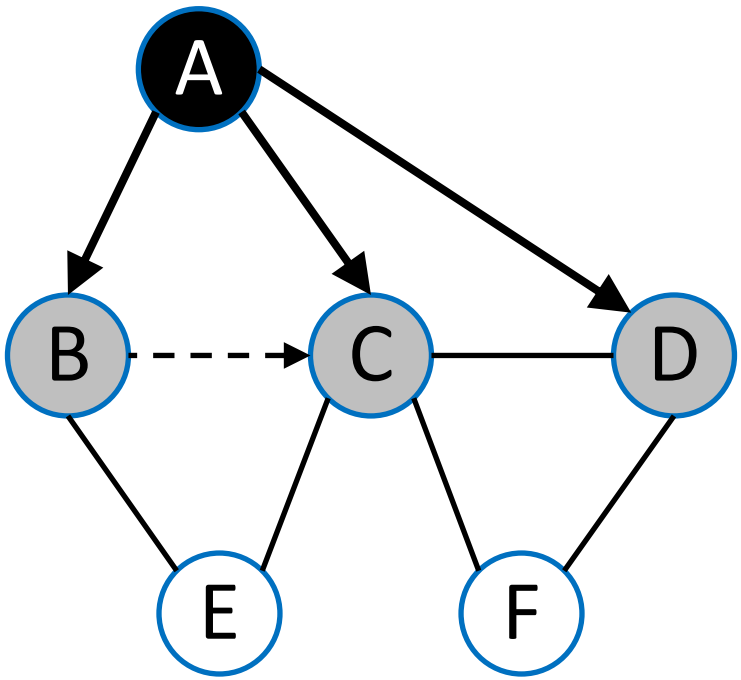
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	INF	NIL
F	INF	NIL

Q

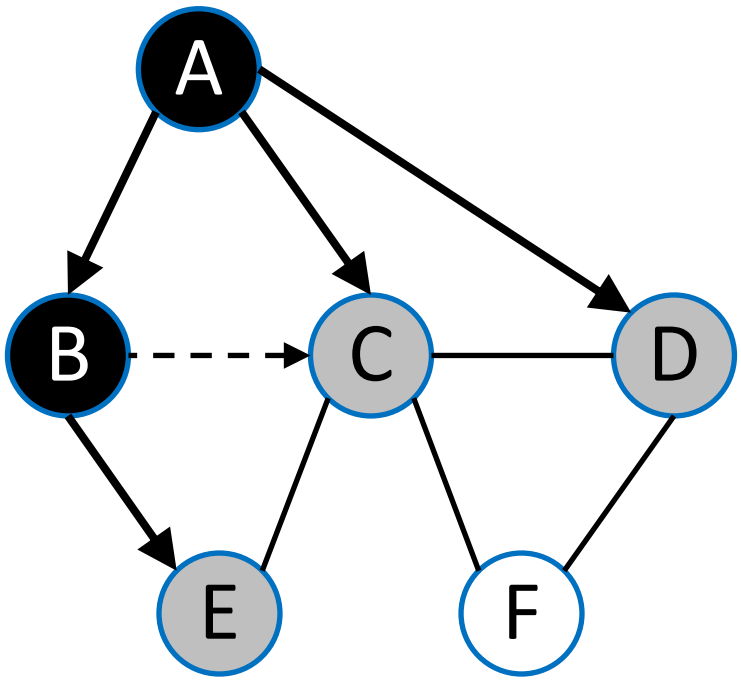
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	INF	NIL
F	INF	NIL

Q

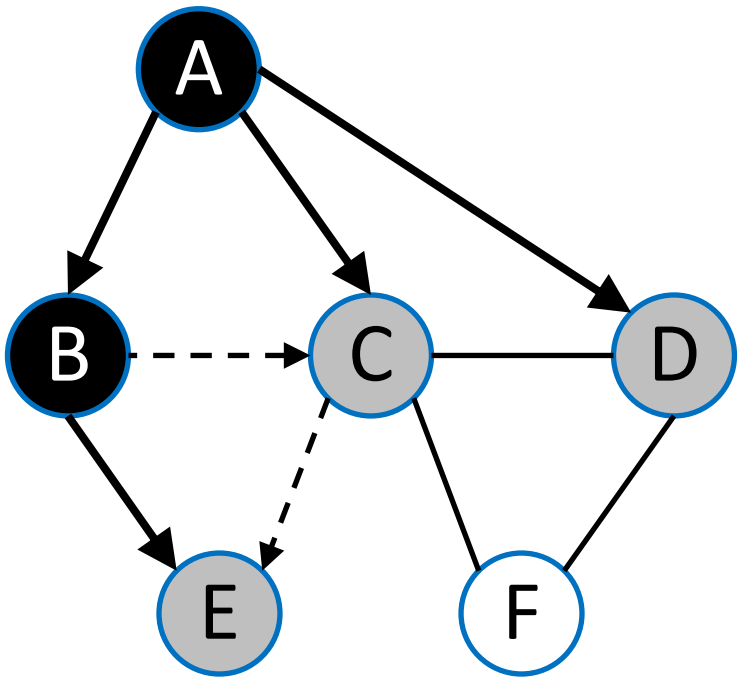
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	INF	NIL

Q

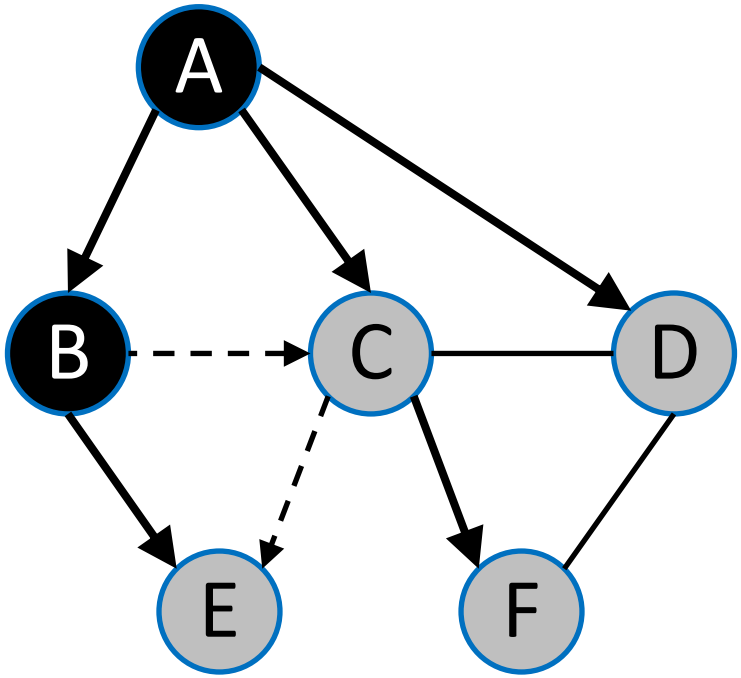
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	INF	NIL

Q

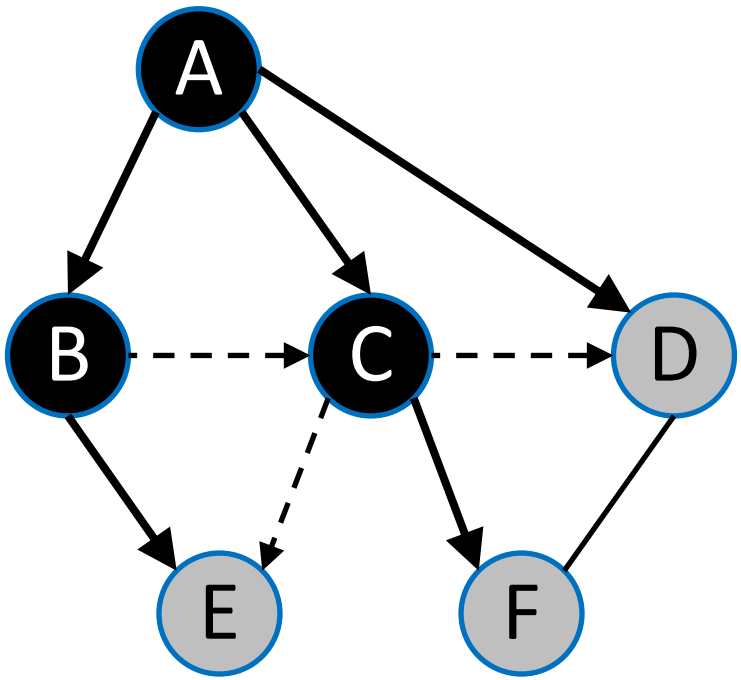
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	2	C

Q

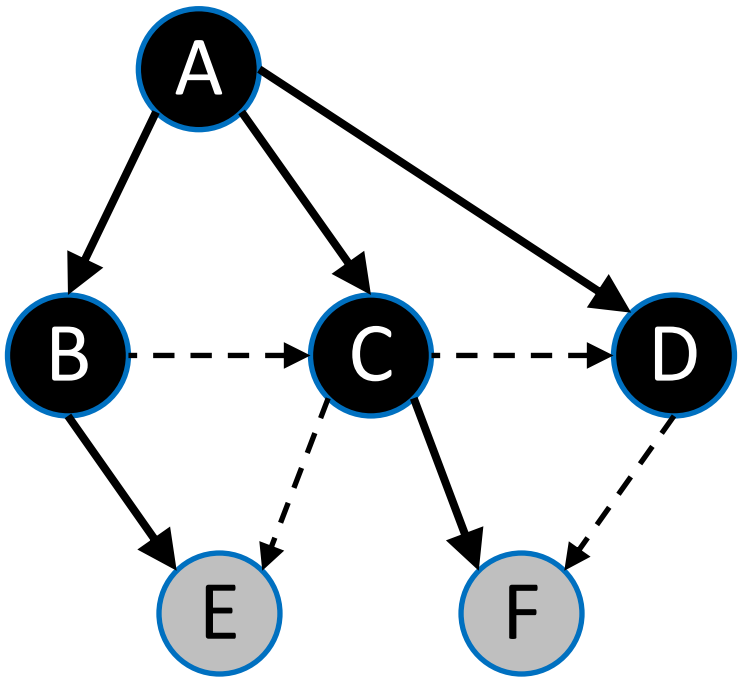
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	2	C

Q

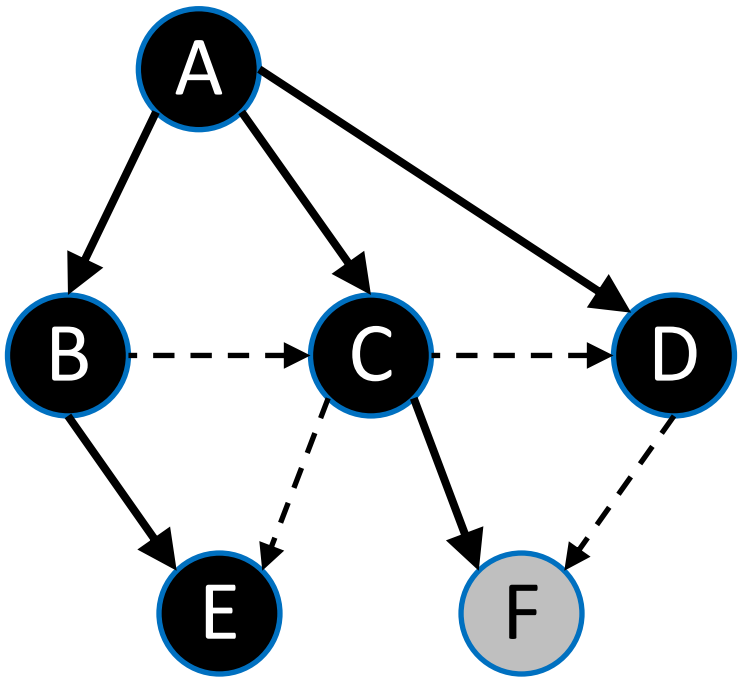
Поиск в ширину - пример



Q | E | F |

	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	2	C

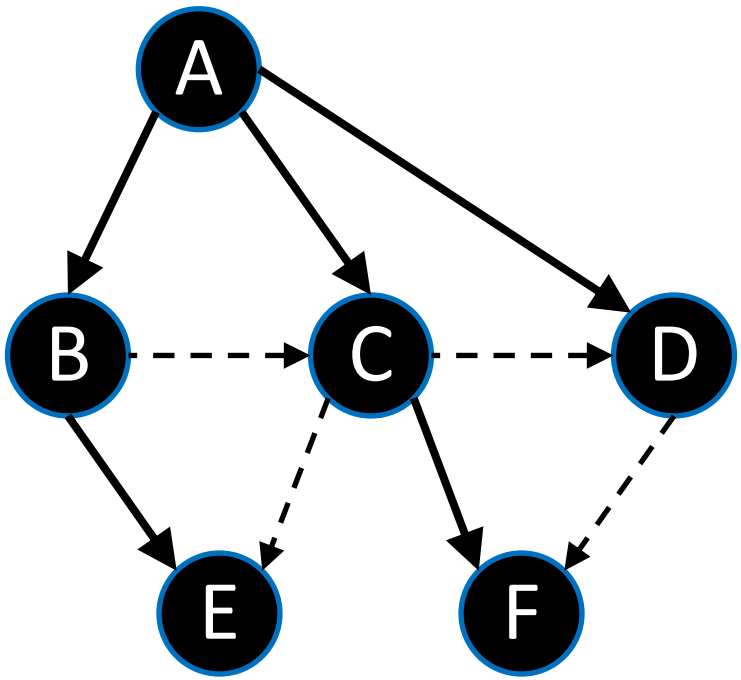
Поиск в ширину - пример



Q F

	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	2	C

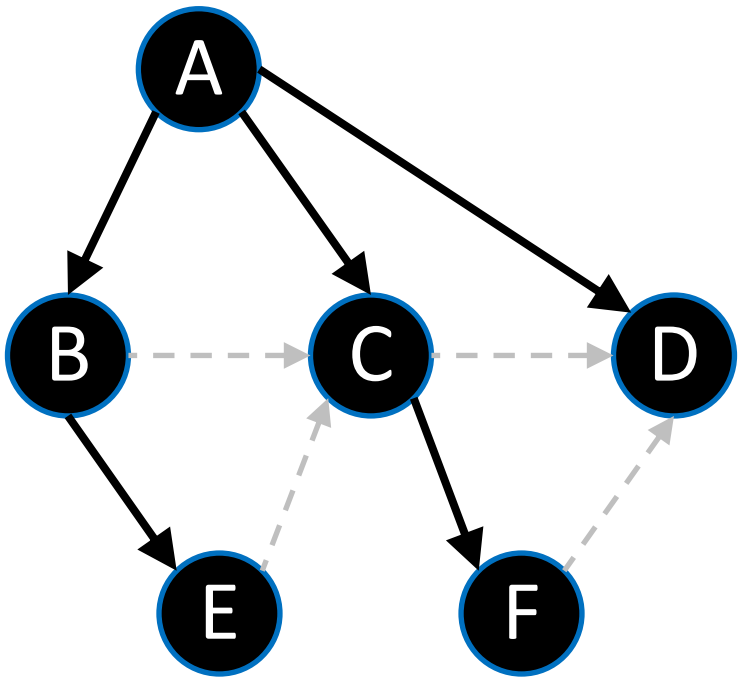
Поиск в ширину - пример



	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	2	C

Q

Дерево поиска в ширину



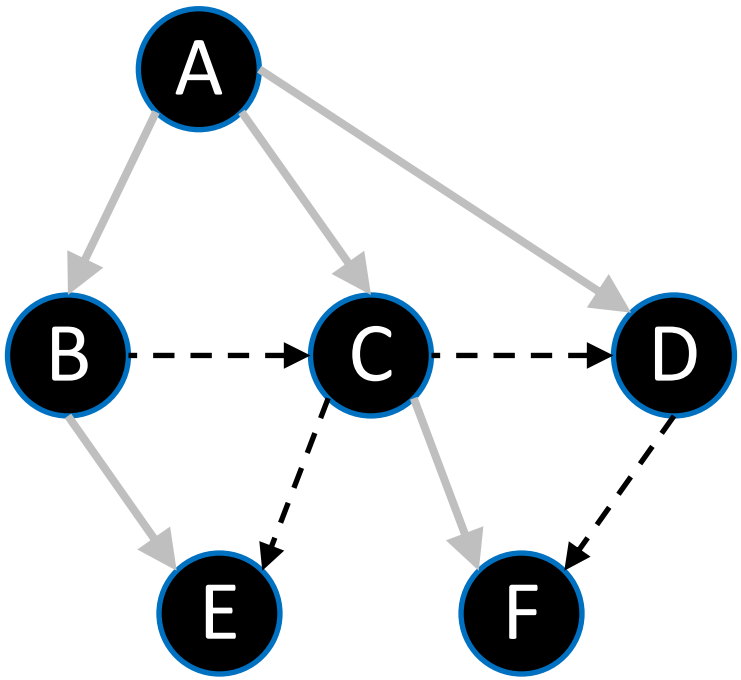
	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	2	C

Подграф предшествования $G_p(V_p, E_p)$, где

$$V_p = \{v \in V : p[v] \neq \mathrm{NIL}\} \cup \{\mathrm{A}\}$$

$$E_p = \{(p[v], v) : v \in V_p \setminus \{\mathrm{A}\}\}$$

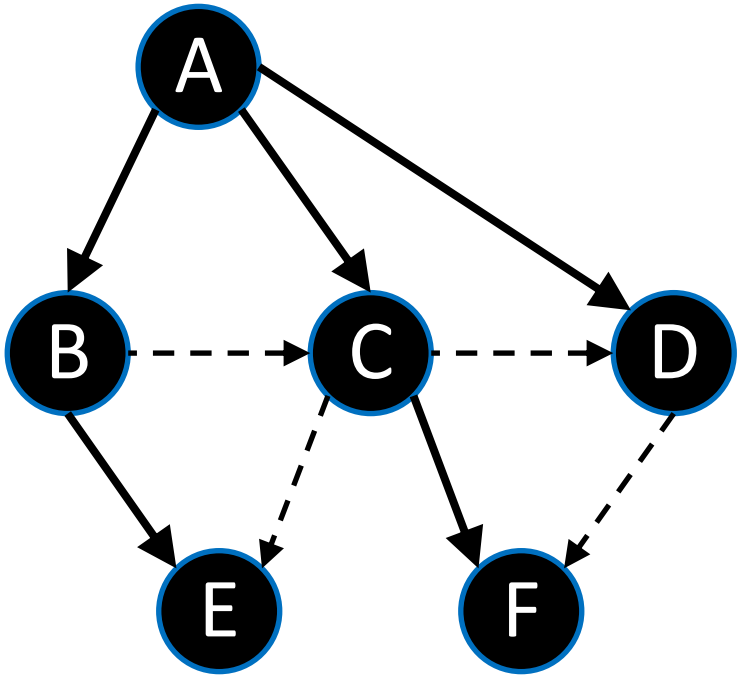
Перекрестные ребра – cross edges



	d	p
A	0	NIL
B	1	A
C	1	A
D	1	A
E	2	B
F	2	C

Для каждого перекрестного ребра (u, v) имеем $d[v] = d[u]$ или $d[v] = d[u] + 1$

Нахождение кратчайших путей при помощи поиска в ширину



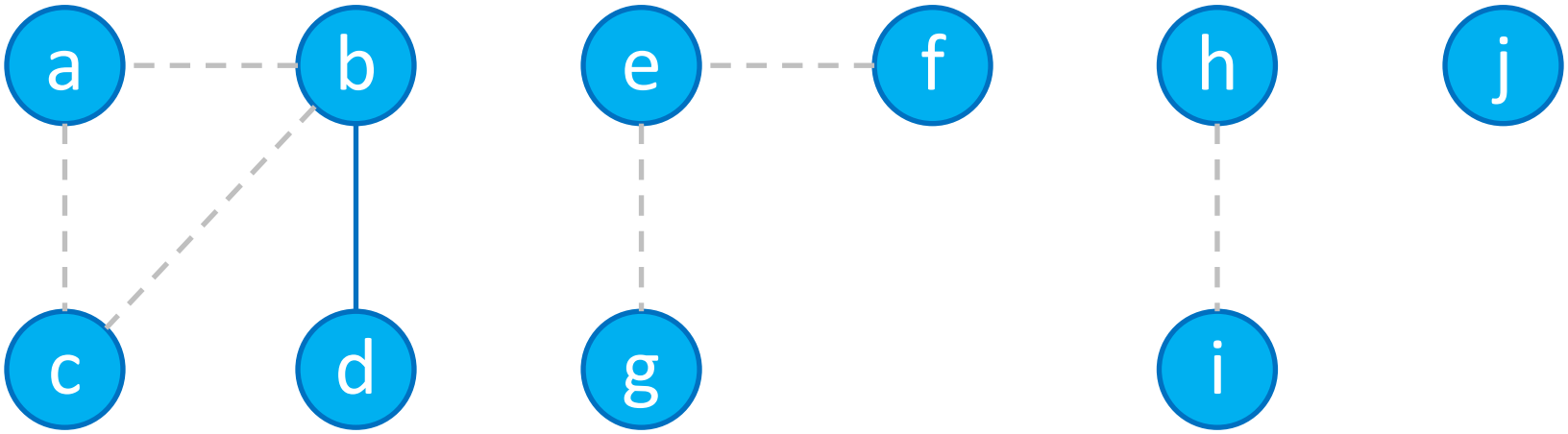

```
Print_Path(G, s, v)
        if v == s
                print s
        else
                if p[v] == NIL
                        print “No path exist from s to v”
                else
                        Print_Path(G,s,p[v])
                        print v
```

Поиск в глубину

```
DFS(G)
        foreach v∈V[G]
                color[u] := WHITE
                p[u] := NIL
        time := 0
        foreach v∈V[G]
                if color[u] == WHITE
                        DFS_Visit(G,u)

DFS_Visit(G,u)
        color[u] := GRAY
        time++
        d[u] := time//discovery time
        foreach v ∈Adj[u]
                if color[v] == WHITE
                        p[v] := u
                        DFS_Visit(G, v)
        color[u] := BLACK
        f[u] := ++time//finishing time
```

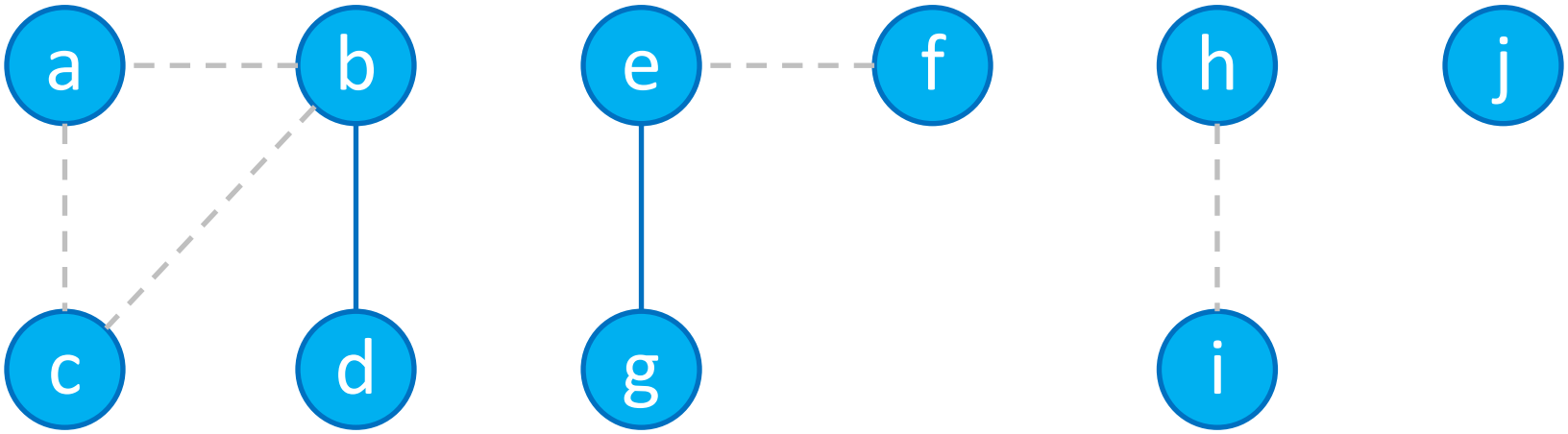
Поиск связанных компонент



Старт $\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}$

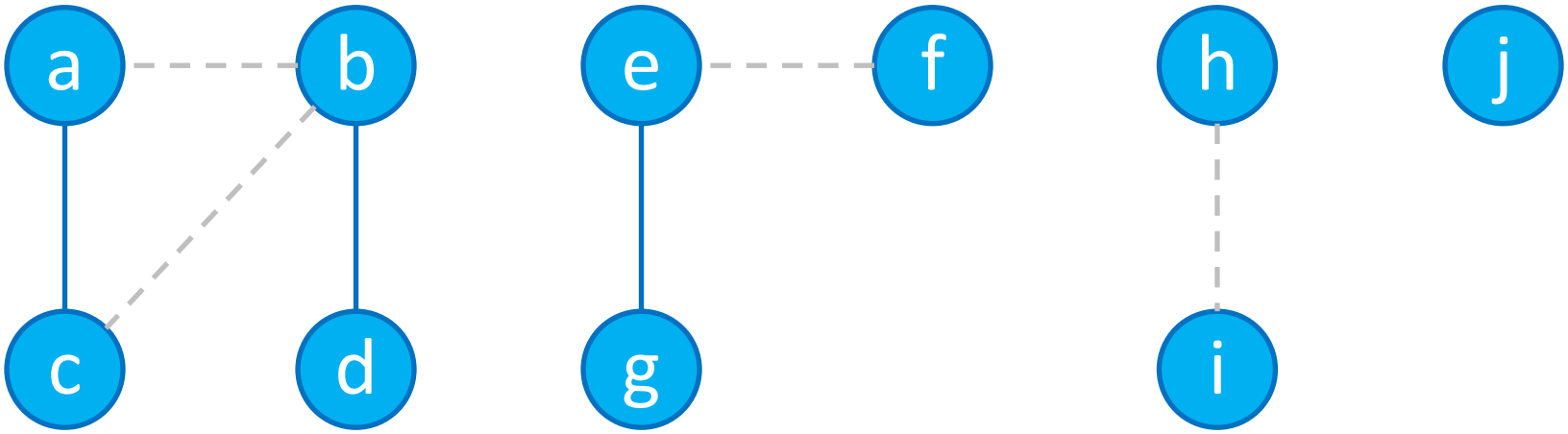
(b,d) $\{\mathrm{a}\}, \{\mathrm{b}, \mathrm{d}\}, \{\mathrm{c}\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}$

Поиск связанных компонент



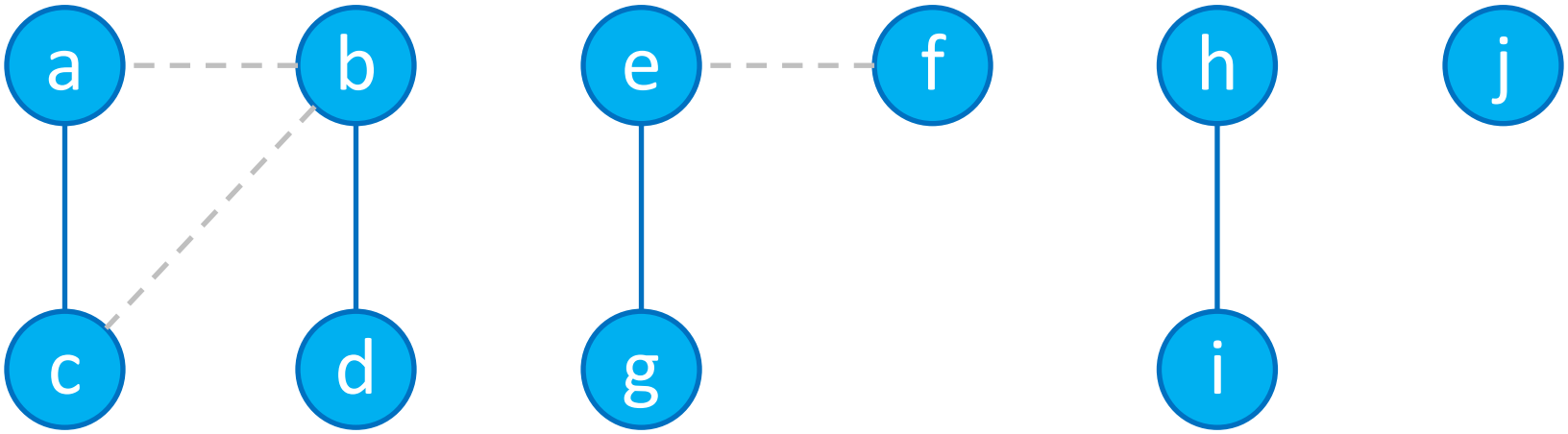
Старт	$\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}$
(b,d)	$\{\mathrm{a}\}, \{\mathrm{b}, \mathrm{d}\}, \{\mathrm{c}\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}$
(e,g)	$\{\mathrm{a}\}, \{\mathrm{b}, \mathrm{d}\}, \{\mathrm{c}\}, \{e, g\}, \{f\}, \{h\}, \{i\}, \{j\}$

Поиск связанных компонент



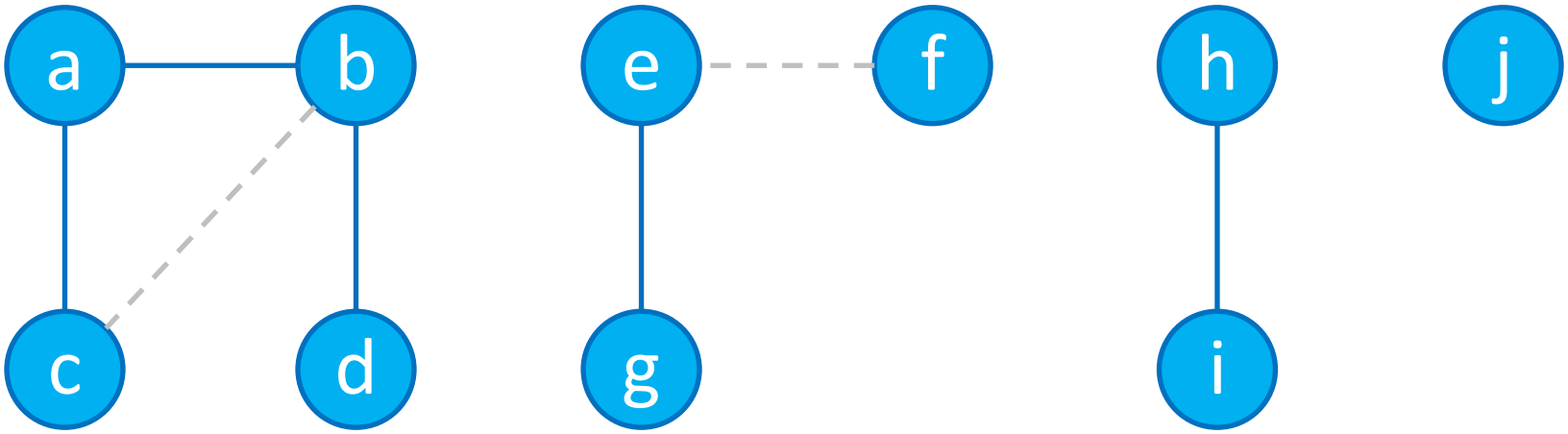
Старт	$\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}$
(b,d)	$\{\mathrm{a}\}, \{\mathrm{b}, \mathrm{d}\}, \{\mathrm{c}\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}$
(e,g)	$\{\mathrm{a}\}, \{\mathrm{b}, \mathrm{d}\}, \{\mathrm{c}\}, \{e, g\}, \{f\}, \{h\}, \{i\}, \{j\}$
(a,c)	$\{\mathrm{a}, c\}, \{\mathrm{b}, \mathrm{d}\}, \{e, g\}, \{f\}, \{h\}, \{i\}, \{j\}$

Поиск связанных компонент



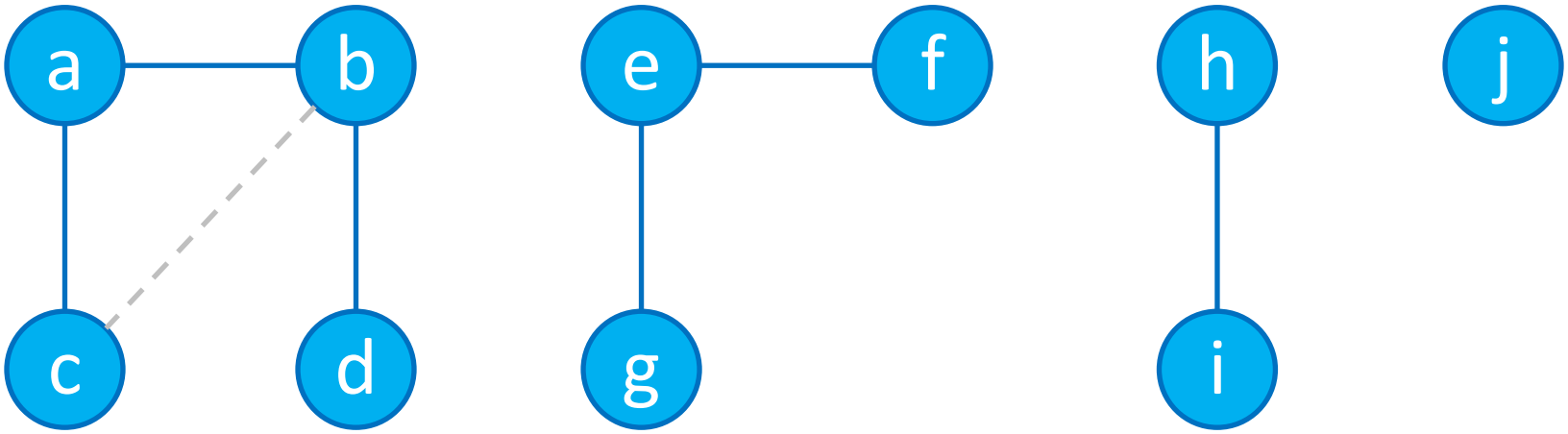
Старт	$\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}$
(b,d)	$\{\mathrm{a}\}, \{\mathrm{b}, \mathrm{d}\}, \{\mathrm{c}\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}$
(e,g)	$\{\mathrm{a}\}, \{\mathrm{b}, \mathrm{d}\}, \{\mathrm{c}\}, \{e, g\}, \{f\}, \{h\}, \{i\}, \{j\}$
(a,c)	$\{\mathrm{a}, c\}, \{\mathrm{b}, \mathrm{d}\}, \{e, g\}, \{f\}, \{h\}, \{i\}, \{j\}$
(h,i)	$\{\mathrm{a}, c\}, \{\mathrm{b}, \mathrm{d}\}, \{e, g\}, \{f\}, \{h, i\}, \{j\}$

Поиск связанных компонент



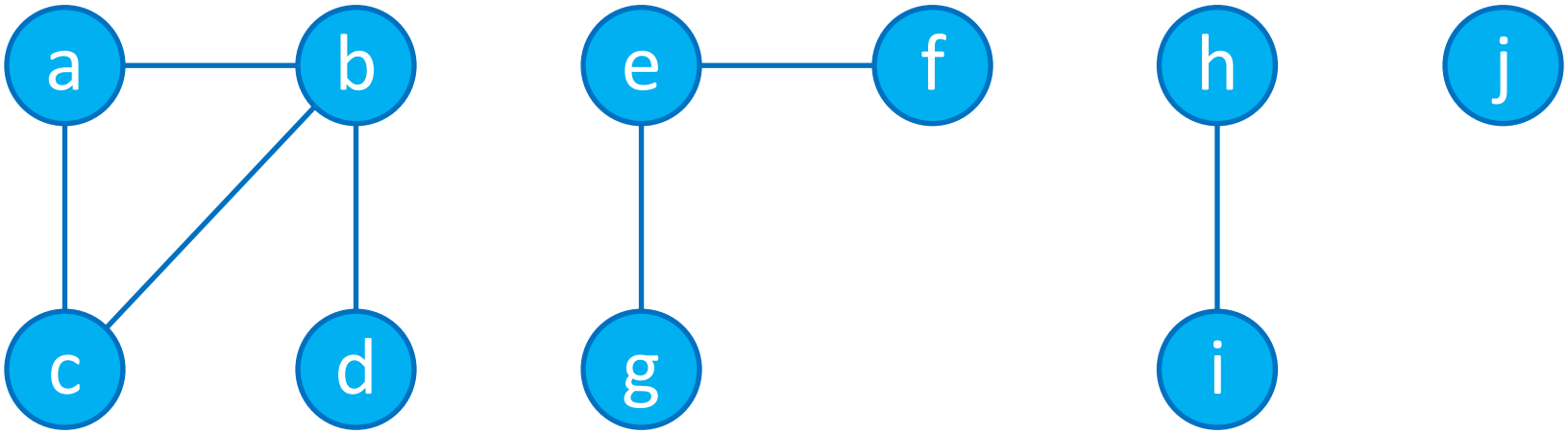
Старт	$\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\},\{h\},\{i\},\{j\}$
(b,d)	$\{\mathrm{a}\},\{\mathrm{b},\mathrm{d}\},\{\mathrm{c}\},\{e\},\{f\},\{g\},\{h\},\{i\},\{j\}$
(e,g)	$\{\mathrm{a}\},\{\mathrm{b},\mathrm{d}\},\{\mathrm{c}\},\{e,g\},\{f\},\{h\},\{i\},\{j\}$
(a,c)	$\{\mathrm{a},c\},\{\mathrm{b},\mathrm{d}\},\{e,g\},\{f\},\{h\},\{i\},\{j\}$
(h,i)	$\{\mathrm{a},c\},\{b,d\},\{e,g\},\{f\},\{h,i\},\{j\}$
(a,b)	$\{a,b,c,d\},\{e,g\},\{f\},\{h,i\},\{j\}$

Поиск связанных компонент



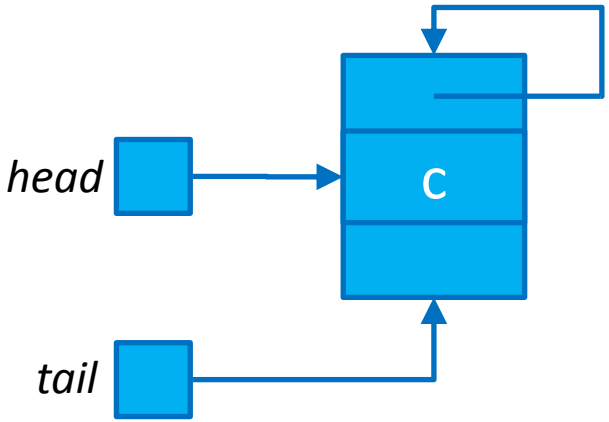
Старт	$\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\},\{h\},\{i\},\{j\}$
(b,d)	$\{\mathrm{a}\},\{\mathrm{b},\mathrm{d}\},\{\mathrm{c}\},\{e\},\{f\},\{g\},\{h\},\{i\},\{j\}$
(e,g)	$\{\mathrm{a}\},\{\mathrm{b},\mathrm{d}\},\{\mathrm{c}\},\{e,g\},\{f\},\{h\},\{i\},\{j\}$
(a,c)	$\{\mathrm{a},c\},\{\mathrm{b},\mathrm{d}\},\{e,g\},\{f\},\{h\},\{i\},\{j\}$
(h,i)	$\{\mathrm{a},c\},\{b,d\},\{e,g\},\{f\},\{h,i\},\{j\}$
(a,b)	$\{a,b,c,d\},\{e,g\},\{f\},\{h,i\},\{j\}$
(e,f)	$\{a,b,c,d\},\{e,f,g\},\{h,i\},\{j\}$

Поиск связанных компонент



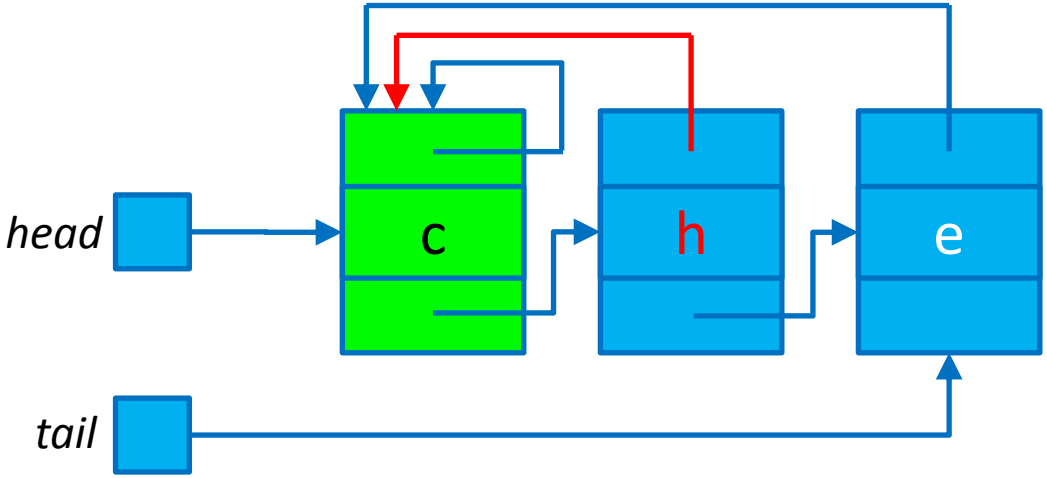
Старт	$\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\},\{h\},\{i\},\{j\}$
(b,d)	$\{\mathrm{a}\},\{\mathrm{b},\mathrm{d}\},\{\mathrm{c}\},\{e\},\{f\},\{g\},\{h\},\{i\},\{j\}$
(e,g)	$\{\mathrm{a}\},\{\mathrm{b},\mathrm{d}\},\{\mathrm{c}\},\{e,g\},\{f\},\{h\},\{i\},\{j\}$
(a,c)	$\{\mathrm{a},c\},\{\mathrm{b},\mathrm{d}\},\{e,g\},\{f\},\{h\},\{i\},\{j\}$
(h,i)	$\{\mathrm{a},c\},\{b,d\},\{e,g\},\{f\},\{h,i\},\{j\}$
(a,b)	$\{a,b,c,d\},\{e,g\},\{f\},\{h,i\},\{j\}$
(e,f)	$\{a,b,c,d\},\{e,f,g\},\{h,i\},\{j\}$
(b,c)	$\{a,b,c,d\},\{e,f,g\},\{h,i\},\{j\}$

Реализация на основе связанных списков



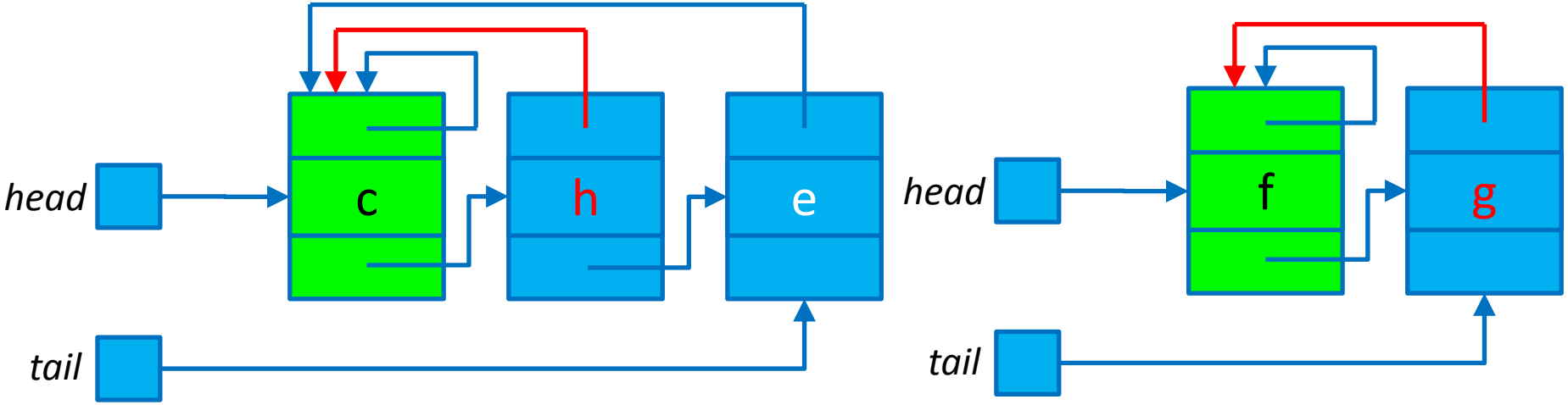

```
Make_Set(a)
Сложность = O(1)
```

Реализация на основе связанных списков



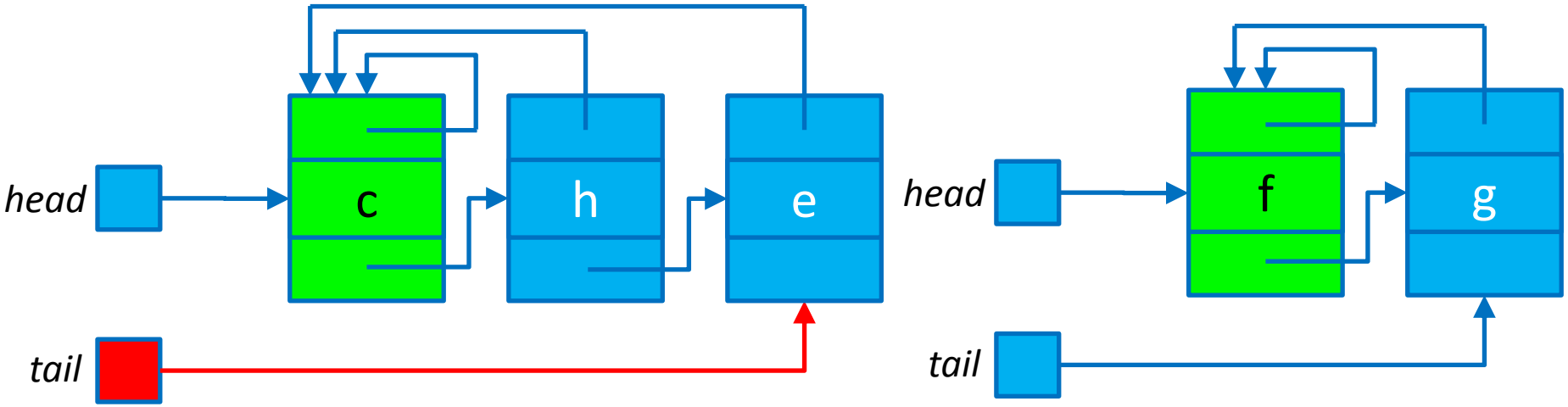

```
Find_Set(h)
Сложность = O(1)
```

Реализация на основе связанных списков



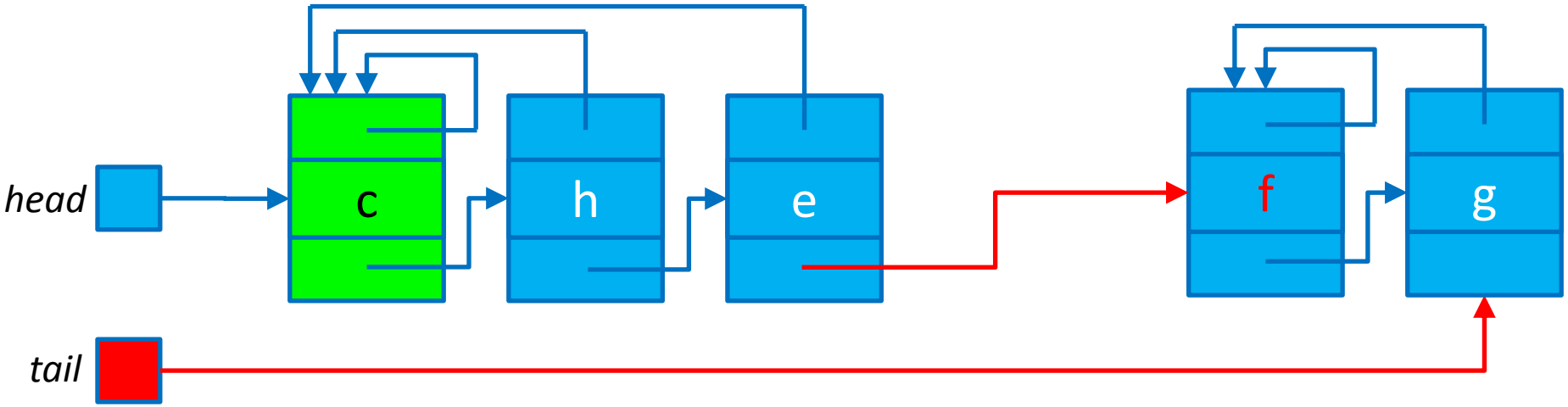
`Union(h,g)`

Реализация на основе связанных списков



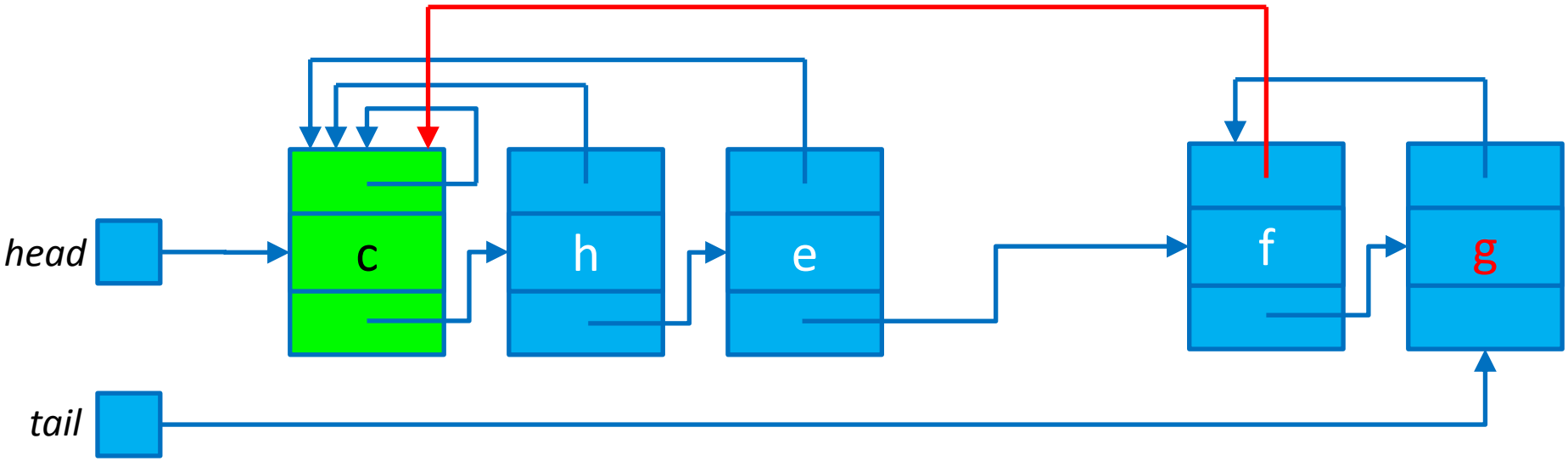
`Union(h,g)`

Реализация на основе связанных списков



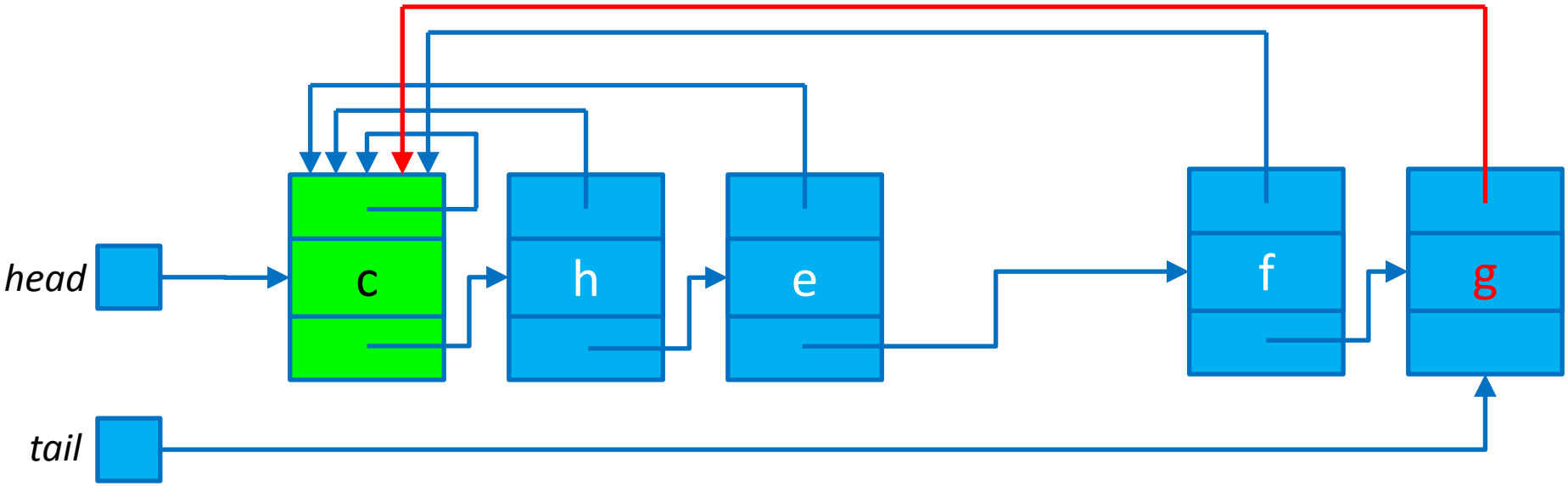
`Union(h,g)`

Реализация на основе связанных списков



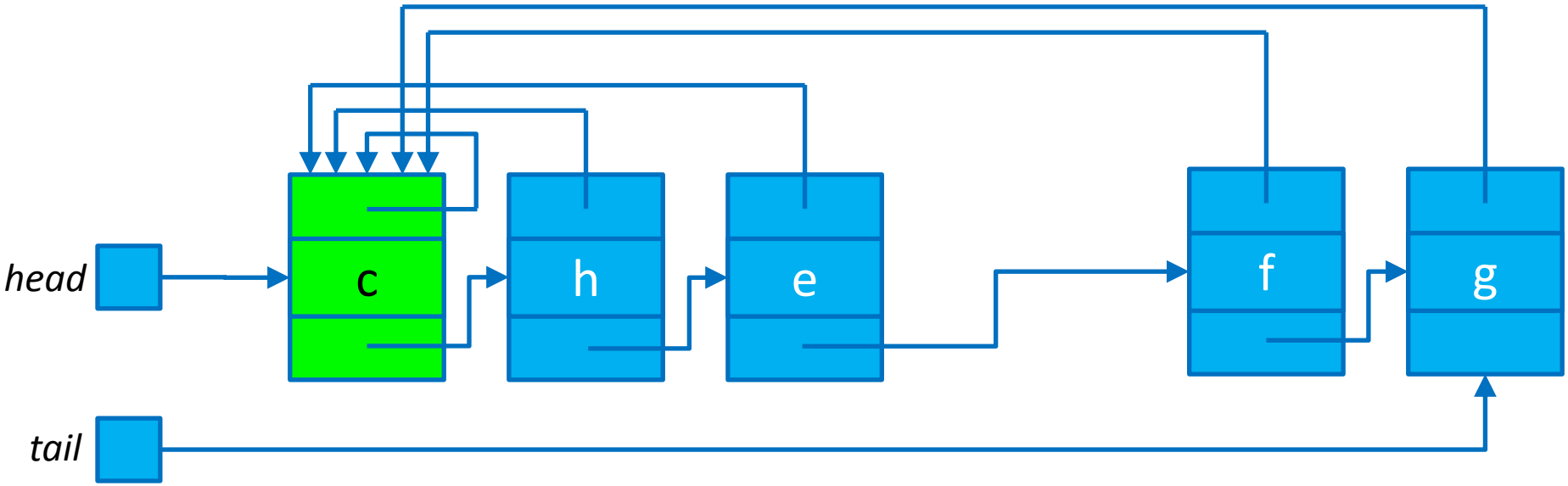
Union(h,g)

Реализация на основе связанных списков



Union(h,g)

Реализация на основе связанных списков

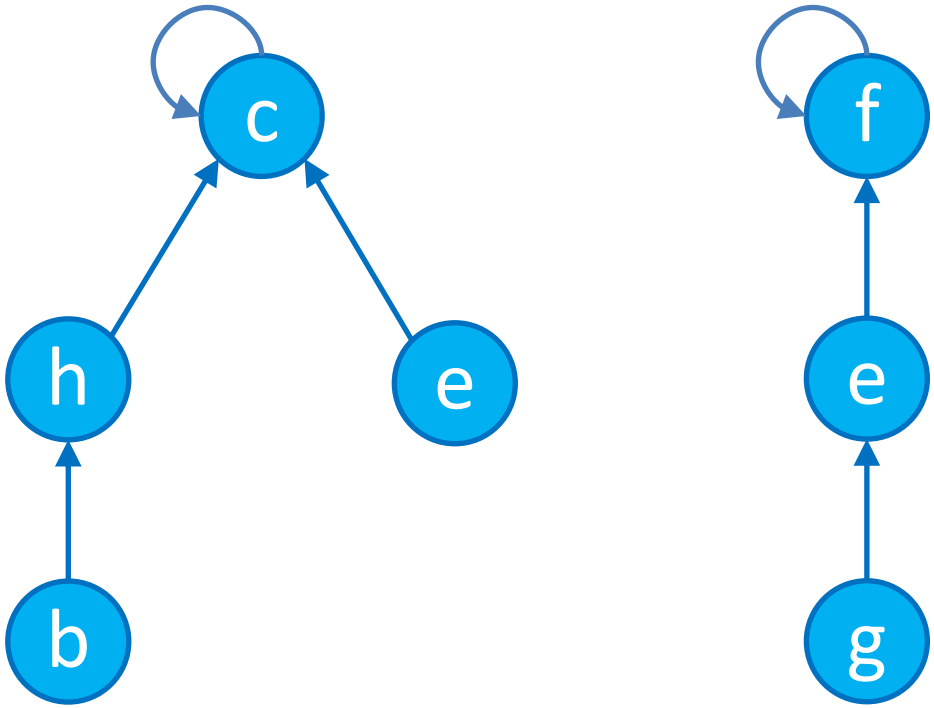



```
Union(h,g)
Сложность = O(n)
```

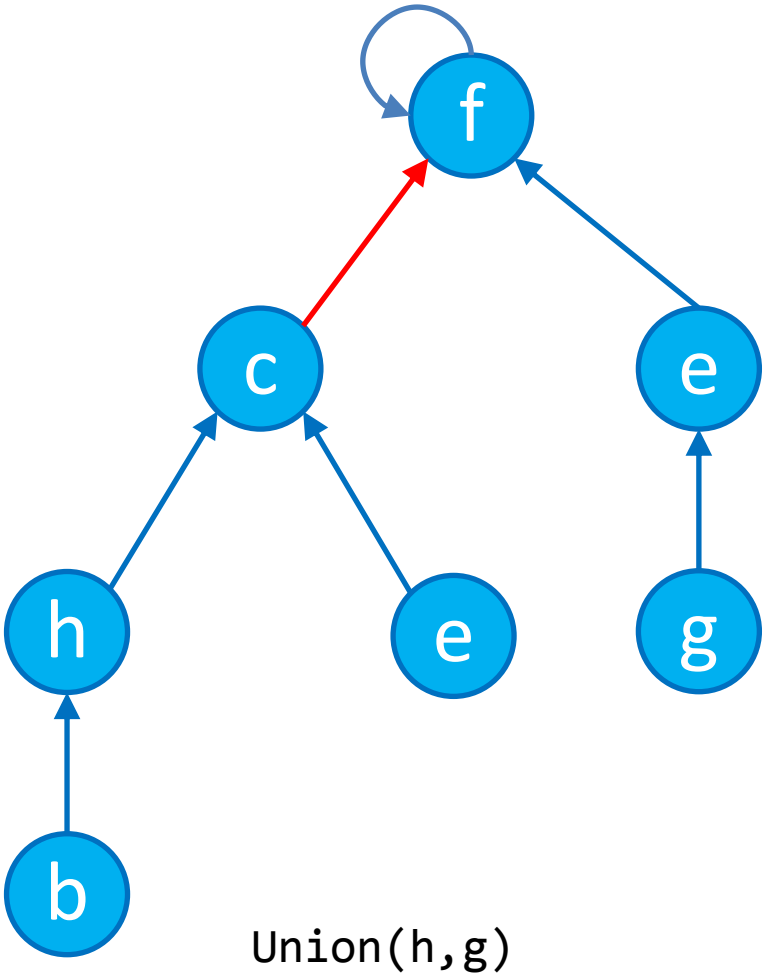
Весовая эвристика

- Для списков поддерживаем значение их длины
- Присоединяем более короткий список к более длинному
- Последовательность из m операций Make_Set, Union и Find_Set, n из которых составляют операции Make_Set требует для выполнения O(m+nlog n) времени

Лес непересекающихся множеств



Лес непересекающихся множеств



`Union(h,g)`

Объединение по рангу и сжатие пути

- Аналог весовой эвристики
- Идея: корень с меньшим количеством узлов должен указывать на корень с большим количеством узлов
- Ранг корня – верхняя граница высоты узла (длина максимального пути от листовых вершин до корня)
- Сжатие пути – перебрасывание ссылок на родителя при поиске канонического представителя

Объединение по рангу и сжатие пути

```
Make_Set(x)
        p[x] := x
        rank[x] := 0

Union(x, y)
        Link(Find_Set(x), Find_Set(y))

Link(x, y)
        if rank[x] > rank[y] then
                p[y] := x
        else
                p[x] := y
                if rank[x] = rank[y] then
                        rank[y]++

Find_Set(y)
        if x≠p[x] then
                p[x] := Find_Set(p[x])
        return p[x]
```

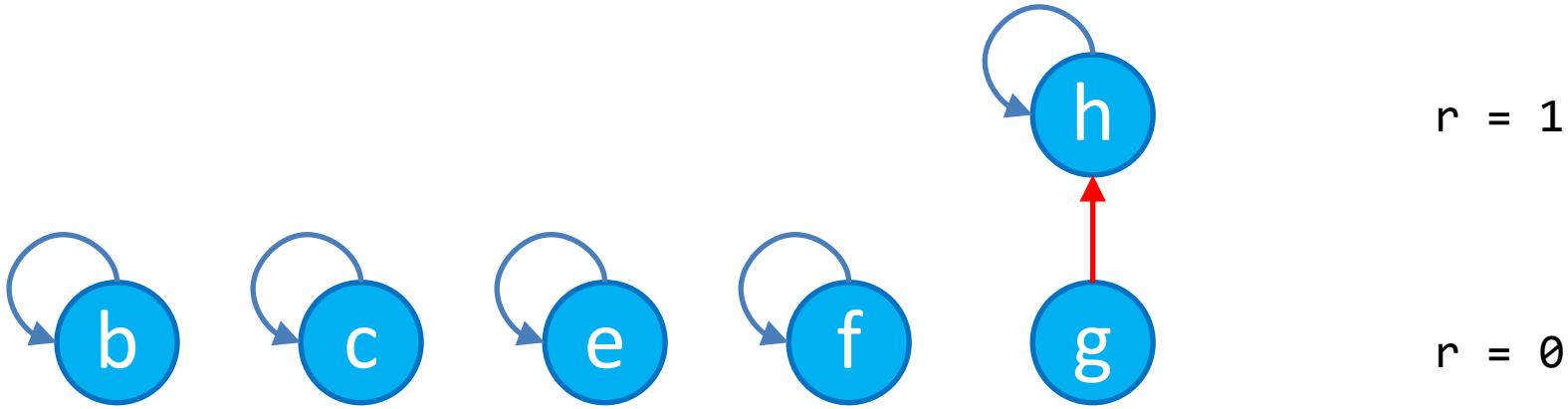
Объединение по рангу и сжатие пути

```
Make_Set(b)
Make_Set(c)
Make_Set(e)
Make_Set(f)
Make_Set(g)
Make_Set(h)
```

r = 0

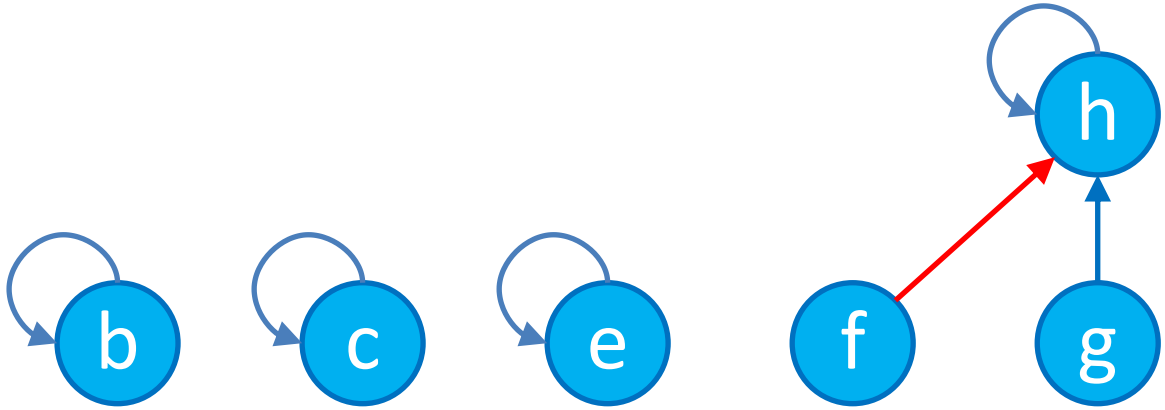
Объединение по рангу и сжатие пути

```
Make_Set(b)
Make_Set(c)
Make_Set(e)
Make_Set(f)
Make_Set(g)
Make_Set(h)
Union(g,h)
```



Объединение по рангу и сжатие пути

```
Make_Set(b)
Make_Set(c)
Make_Set(e)
Make_Set(f)
Make_Set(g)
Make_Set(h)
Union(g,h)
Union(f,g)
```

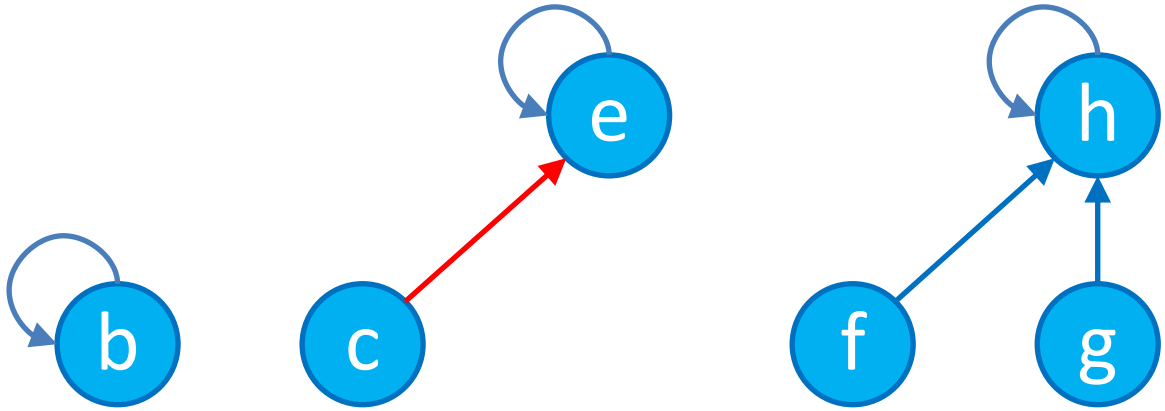


r = 1

r = 0

Объединение по рангу и сжатие пути

```
Make_Set(b)
Make_Set(c)
Make_Set(e)
Make_Set(f)
Make_Set(g)
Make_Set(h)
Union(g,h)
Union(f,g)
Union(c,e)
```

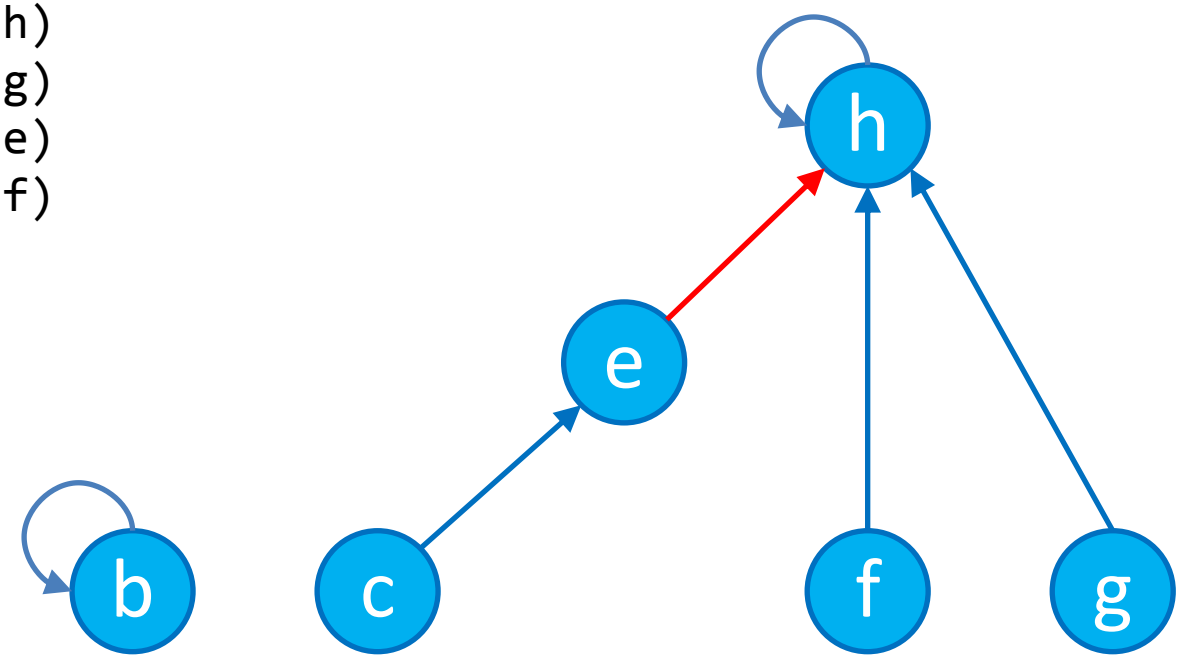


r = 1

r = 0

Объединение по рангу и сжатие пути

```
Make_Set(b)
Make_Set(c)
Make_Set(e)
Make_Set(f)
Make_Set(g)
Make_Set(h)
Union(g,h)
Union(f,g)
Union(c,e)
Union(c,f)
```



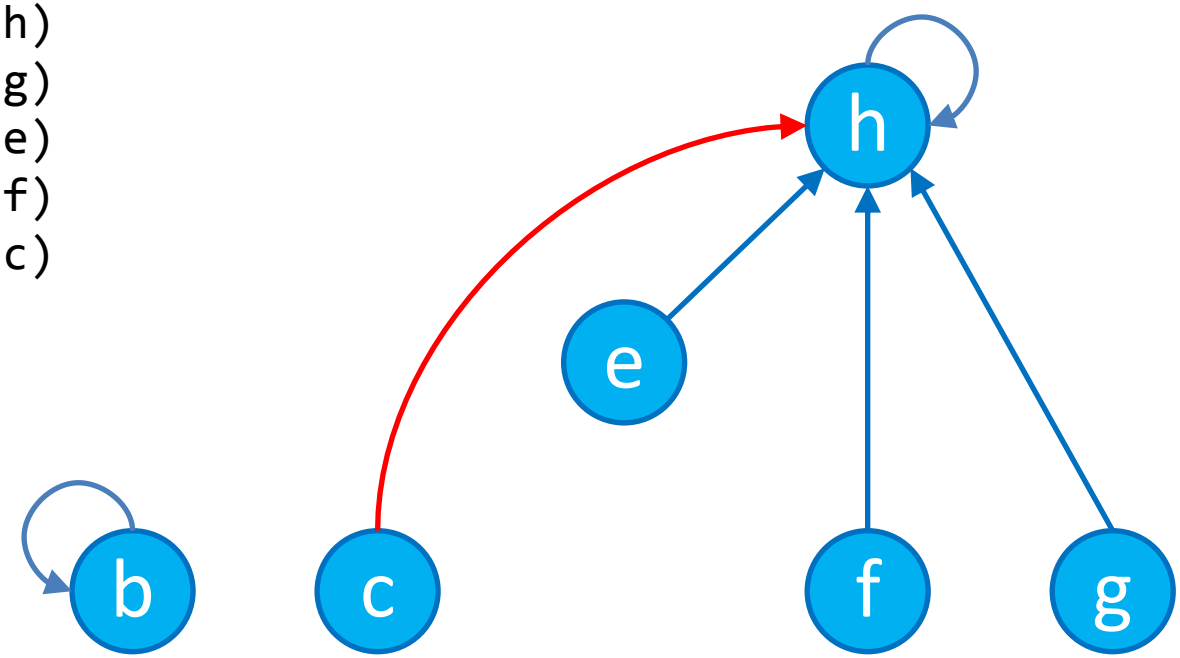
r = 2

r = 1

r = 0

Объединение по рангу и сжатие пути

```
Make_Set(b)
Make_Set(c)
Make_Set(e)
Make_Set(f)
Make_Set(g)
Make_Set(h)
Union(g,h)
Union(f,g)
Union(c,e)
Union(c,f)
Union(b,c)
```



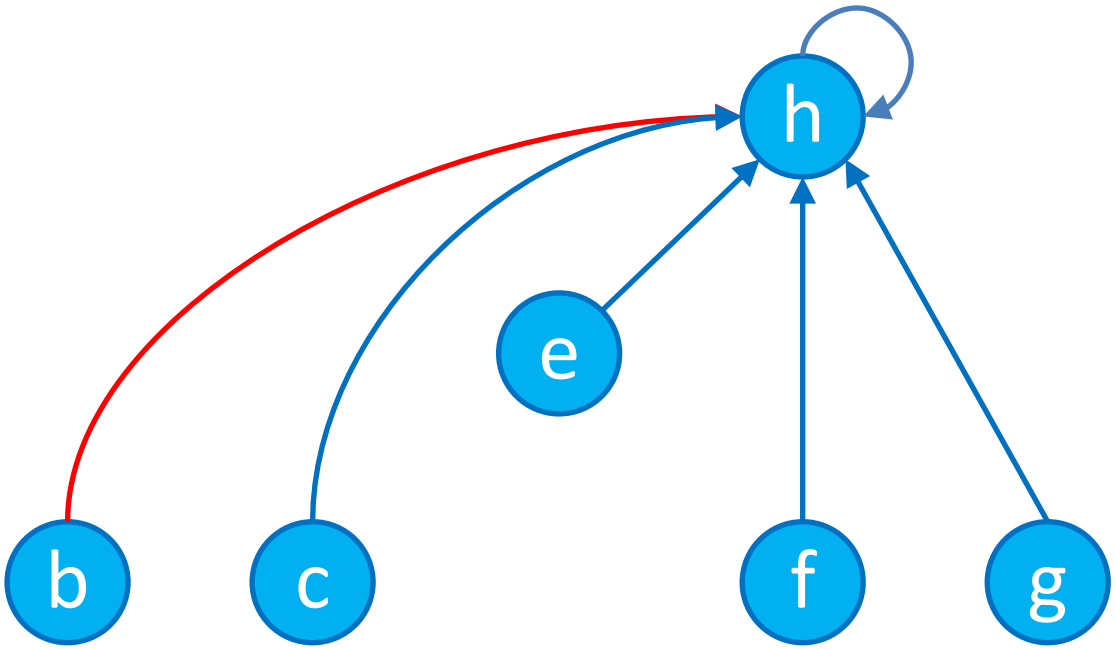
r = 2

r = 1

r = 0

Объединение по рангу и сжатие пути

```
Make_Set(b)
Make_Set(c)
Make_Set(e)
Make_Set(f)
Make_Set(g)
Make_Set(h)
Union(g,h)
Union(f,g)
Union(c,e)
Union(c,f)
Union(b,c)
```



r = 2

r = 1

r = 0