Построение и анализ алгоритмов

Лекция 6.1

Раздел: Алгоритмы на графах

Тема лекции:

Часть 1.

Минимальное остовное дерево (МОД)

1. Теорема о минимальном ребре
2. Алгоритм Краскала (*Жадный* алгоритм)
3. Алгоритм ЯПД (Ярника-Прима-Дейкстры)
4. Нижняя граница задачи нахождения МОД

Остовное дерево

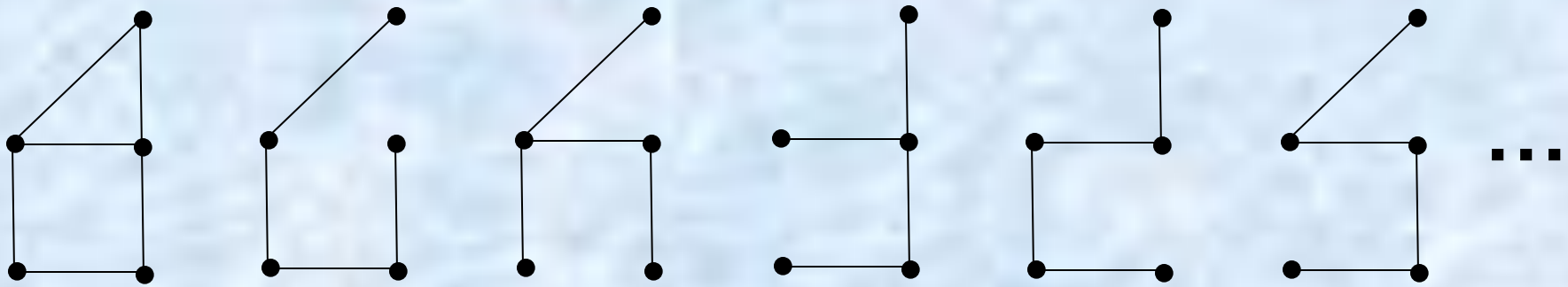
Дерево – связный граф, не содержащий циклов.

Граф *связный*, если каждая пара различных вершин графа связана маршрутом.

Для связного графа $G = (V, E)$ *остовным деревом (остовом, каркасом, стягивающим деревом, скелетом)* является граф (дерево) $T = (V, F)$, где $F \subseteq E$.

Рёбра дерева – *ветви*, остальные рёбра графа – *хорды*.

В графе много остовов, а именно, число остовов n^{n-2}.

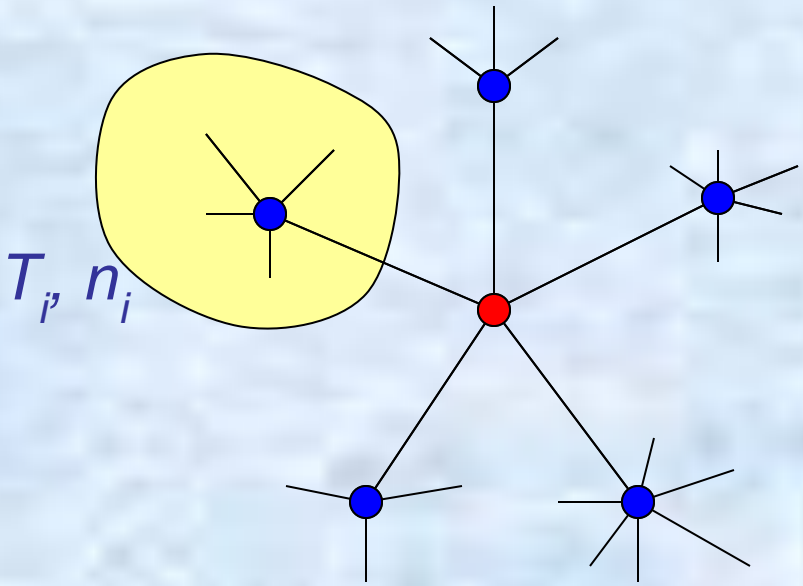


Повторение с предыдущей лекции

Остовное дерево

В дереве из n вершин имеется $m = n - 1$ рёбер

Доказательство (по индукции):



Удаляем некоторый узел и k инцидентных рёбер ($k \in 1..n-1$).

Образуется лес из деревьев T_i с числом узлов n_i ($i \in 1..k$).

$$\sum_{i=1}^{k}(n_i - 1) = \sum_{i=1}^{k} n_i - k = (n-1) - k$$

$$(n - 1 - k) + k = n - 1$$

Минимальное остовное дерево (МОД)

Граф $G = (V, E)$. Матрица весов $W[v, u]$.

Пусть $T = (V, F)$ – остов графа.

Вес остова определяется как

$$W(T) = \sum_{e \in F} W(e) = \sum_{(v,u \in V) \& (\{v,u\} \in F)} W[v,u]$$

МОД : $T_M = Arg\ min\ (\ W(T)\ )$

1. Теорема о минимальном ребре. Версия 0.

Формулировка 1: *Пусть веса всех рёбер различны*. Тогда оптимальное остовное дерево графа содержит ребро с наименьшим весом.

Доказательство (от противного): Пусть $\{v, u\}$ – кратчайшее из всех рёбер – не входит в МОД T.

Добавим $\{v, u\}$ к T.

Образуется цикл.

Удалим из этого цикла ребро, отличное от $\{v, u\}$.
Получим T^* дерево с весом, меньшим чем вес T, чего не может быть, поскольку T есть МОД (противоречие!).

1. Теорема о минимальном ребре. Версия 0*.

Формулировка 2: Существует (найдётся) оптимальное остовное дерево графа, содержащее ребро с наименьшим весом.

Доказательство (от противного): Пусть $\{v, u\}$ – кратчайшее из всех рёбер – не входит в МОД T.

Добавим $\{v, u\}$ к T. Образуется цикл.

Удалим из этого цикла ребро $\{v', u'\}$, отличное от $\{v, u\}$. Получим T^* дерево с весом, *меньшим или равным*, чем вес T.

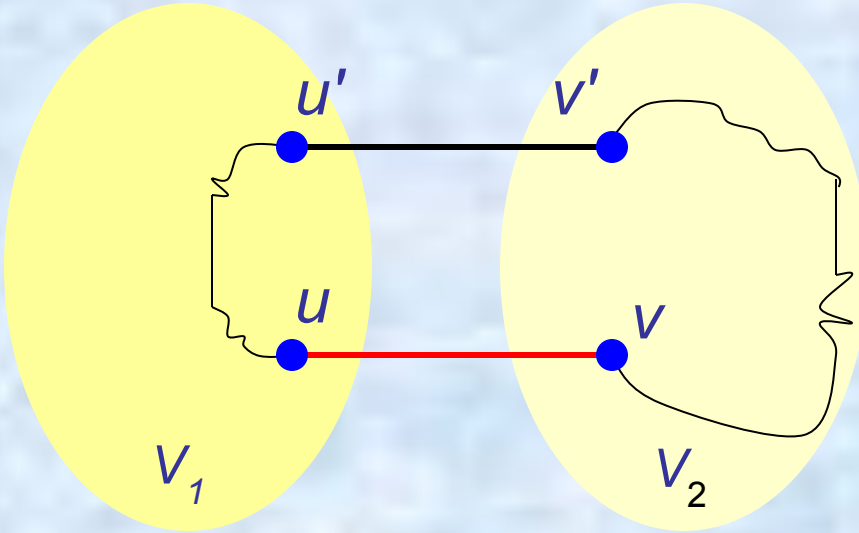
Если $W[v, u] = W[v', u']$, т.е. $W(T^*)=W(T)$, то теорема справедлива.

Если $W[v, u] < W[v', u']$, то $W(T^*) < W(T)$, чего не может быть, поскольку T есть МОД (противоречие!).

1. Теорема о минимальном ребре. Версия 1.

Пусть $G(W) = (V, E, W)$, а $\{V_1, V_2\}$ есть *разбиение* V. В G имеется МОД, содержащее кратчайшее из рёбер, одна вершина которого принадлежит V_1, а другая – V_2.

Доказательство (от противного): Как и ранее, пусть кратчайшее ребро $\{u, v\}$ ($u \in V_1$, $v \in V_2$) не входит в МОД …



На этом цикле есть ребро $\{u', v'\}$.

$W[u, v] \le W[u', v']$.

1. $W[u, v] < W[u', v']$. Тогда, удалив $\{u', v'\}$, получим другое ОД, содержащее T и ребро $\{u, v\}$ и имеющее меньший вес, чем F. Это *противоречит* «противному».

2. $W[u, v] = W[u', v']$. Случай равных весов. …

1. Теорема о минимальном ребре. Версия 2

Пусть

- $\{(V_1, T_1), (V_2, T_2), \ldots, (V_k, T_k)\}$ – *остовный лес* на множестве V (т. е. $\{V_1, V_2, \ldots, V_k\}$ есть *разбиение* V и $(\forall i \in 1..k: T_i \subset E)$),
- $\{v, u\}$ – кратчайшее из всех рёбер, у которых ровно один конец содержится в V_1.

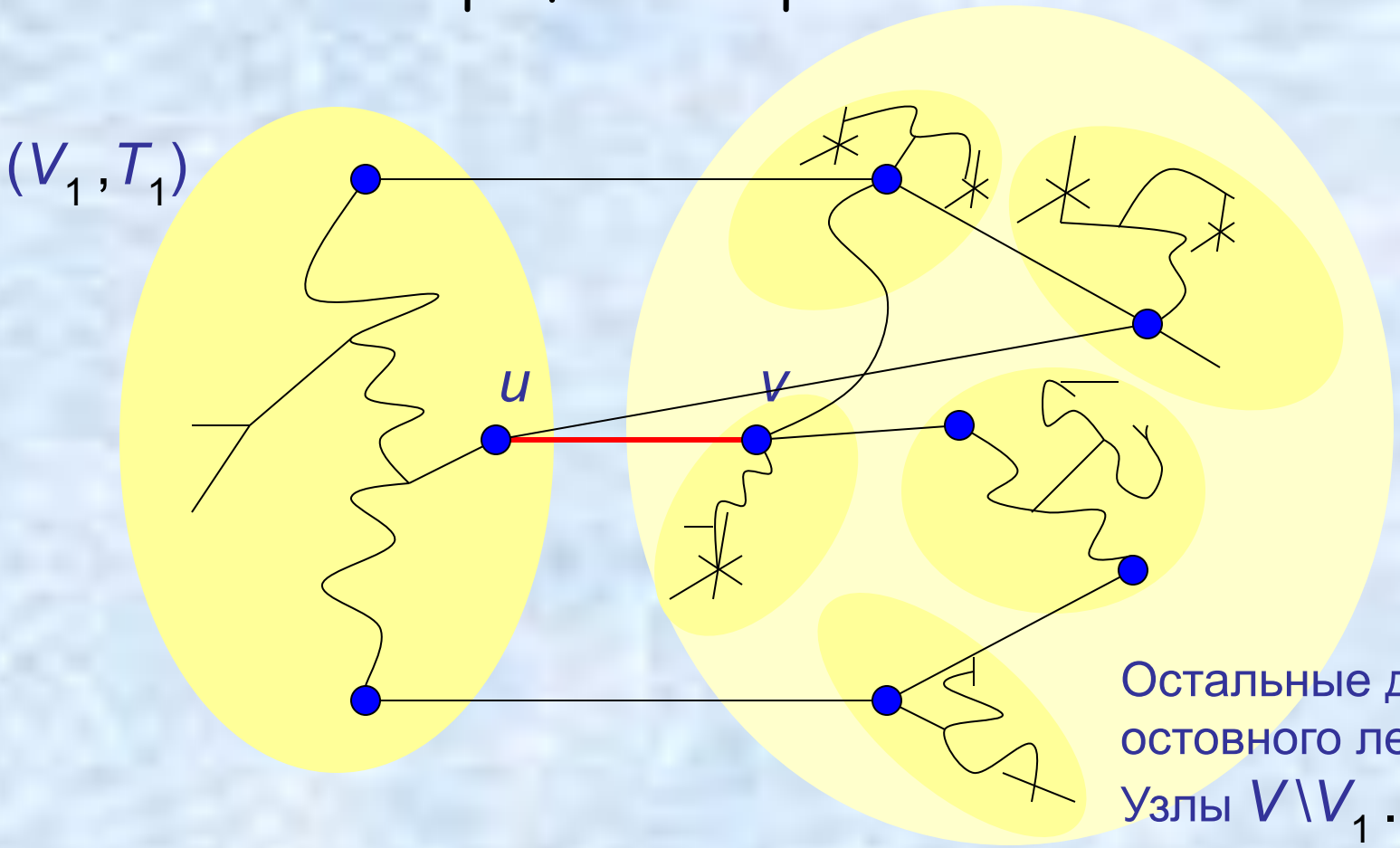
Тогда среди всех остовных деревьев, содержащих все рёбра из множества $T = \bigcup_{j=1}^{k} T_j$, существует оптимальное (для заданного леса) остовное дерево, содержащее $\{v, u\}$.

Т.е. найдётся остов, содержащий $T \cup \{\{v,u\}\}$ и имеющий минимальный вес среди всех остовов, содержащих T.

Замечание о рёбрах с равными весами и формулировке теоремы.

Теорема о минимальном ребре

Иллюстрация к теореме

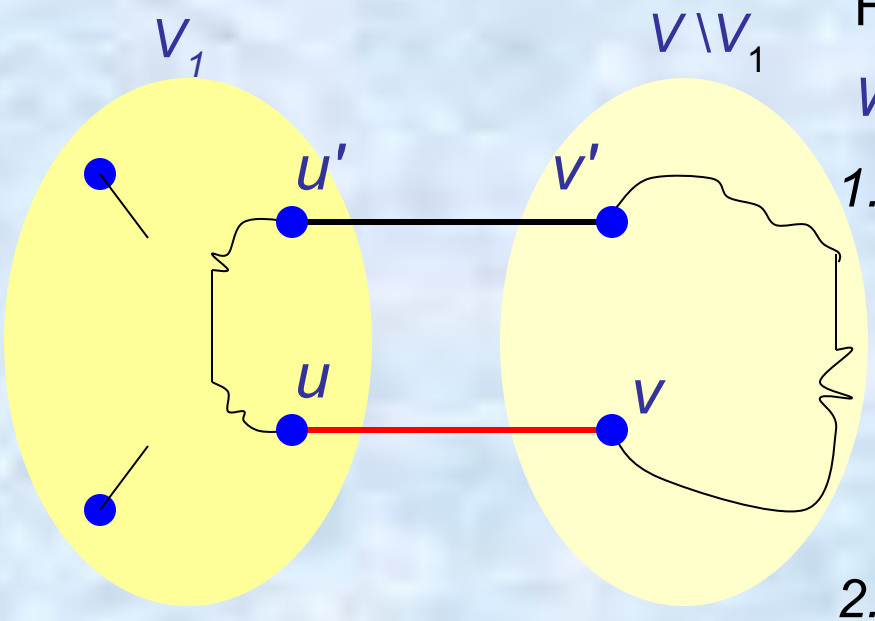


Теорема о минимальном ребре

Доказательство (от противного):

«*Противное*»: ∃ ОД (V, F), где $F \supset T$ и $\{u, v\} \notin F$, которое короче всех остальных ОД, содержащих T и ребро $\{u, v\}$.

Добавим к F ребро $\{u, v\}$. Образуется единственный цикл, не все вершины которого содержаться в V_1, например, $v \notin V_1$.



На этом цикле есть ребро $\{u', v'\}$.

$W[u, v] \le W[u', v']$.

1. $W[u, v] < W[u', v']$. Тогда, удалив $\{u', v'\}$, получим другое ОД, содержащее T и ребро $\{u, v\}$ и имеющее меньший вес, чем F. Это *противоречит* «противному».
2. $W[u, v] = W[u', v']$. Случай равных весов. …

Алгоритмы построения МОД

См. файл
«85_07_minimum_spanning_tree.pdf»

Борувка (O. Boruvka, 1926)
Ярник (V. Jarnik, 1930)
Крускал (J.B. Kruskal, Jr., 1956)
Прим (R.C. Prim, 1957)
Дейкстра (E.W. Dijkstra, 1959)

Мы рассмотрим:

1. Алгоритм Крускала (Жадный алгоритм)
2. Алгоритм ЯПД (Ярника-Прима-Дейкстры)
3. Алгоритм Борувки (позже, в лекции 7)

«Краскал – Крускал»

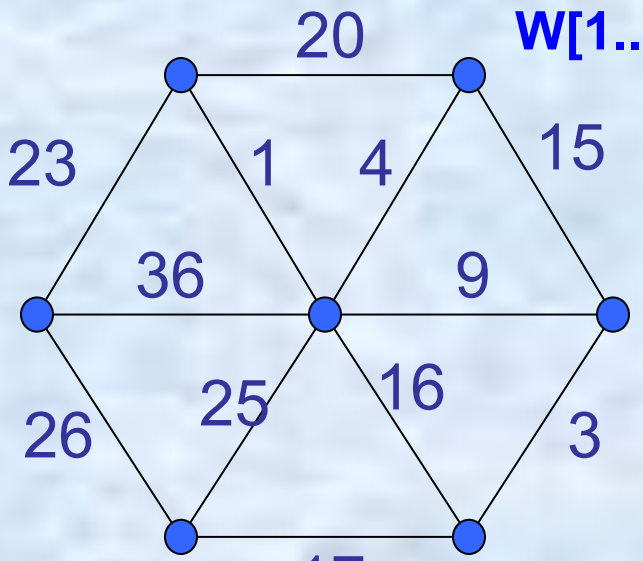
Материал из Википедии

«**Алгоритм Краскала** — алгоритм построения *минимального остовного дерева* взвешенного связного неориентированного графа. Открыт **Джозефом Крускалом** в 1956 году.»

Joseph B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. // Proc. AMS. 1956. Vol 7, No. 1. C. 48–50

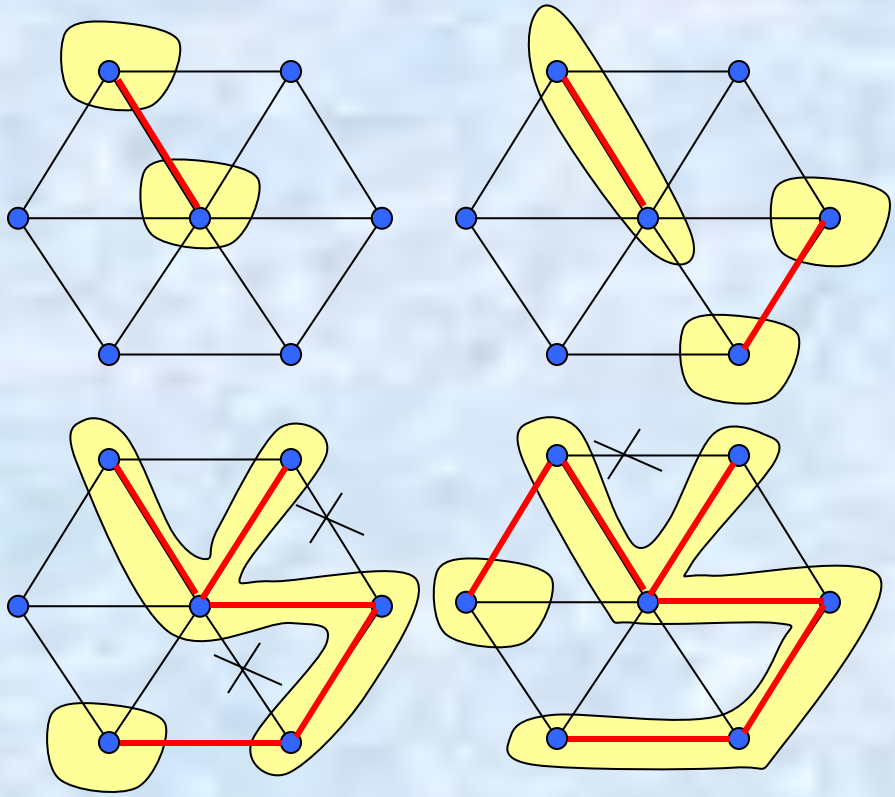
2. Жадный алгоритм построения МОД (Kruskal, 1956)

1. Упорядочить рёбра в порядке неубывания весов.
2. Поочерёдно выбирать рёбра минимального веса, не образующие циклов с ранее выбранными.



W[1..12]: 1, 3, 4, 9, 15, 16, 17, 20, 23, 25, 26, 36

W(T)= 1+3+4+9+17+23 = 57



Жадный алгоритм построения МОД (Kruskal, 1956)

```
Vs - множество множеств узлов (множество деревьев остовного леса)
begin {алгоритм МОД }
    Vs := { }; T:= { };
    for ∀ v ∈ V do добавить {v} к Vs;
    Построить очередь Q из всех рёбер e ∈ E, упорядочив их по
    возрастанию весов;
    while Card(Vs) > 1 do
    begin
        Выбрать из Q ребро e = {u,v} с минимальным весом;
        Удалить e из Q;
        If u и v принадлежат разным подмножествам W1 и W2 из Vs
        then
            begin Заменить W1 и W2 на W1 ∪ W2 в Vs;
                T := T ∪ {e}
            end
    end {while}
end {алгоритма МОД };
```

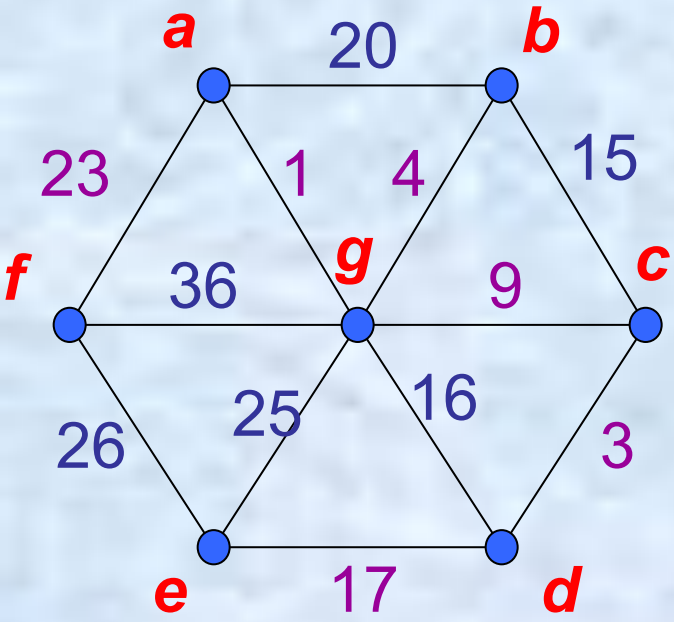
Замечание. Ср. с примером о связности. Найти W1∈Vs, такое, что u∈W1. То же для v и W2. Это операция НАЙТИ. Заменить в Vs подмножества W1 и W2 на их объединение W1∪W2 - это операция ОБЪЕДИНИТЬ.

Жадный алгоритм построения МОД (Kruskal, 1956)

```
// алгоритм МОД
// Vs - множество деревьев остовного леса = мн-во подмножеств
узлов, T –множество рёбер МОД
    Vs = { }; T= { };
    for (∀ v ∈ V) добавить {v} к Vs;
    Построить очередь Q из всех рёбер e ∈ E, упорядочив их по
        возрастанию весов;
    while  (Card(Vs) > 1) {
        Выбрать из Q ребро e = {u,v} с минимальным весом;
        Удалить e из Q;
        if (u и v принадлежат разным подмножествам W1 и W2 из Vs)
{
            Заменить W1 и W2 на W1 ∪ W2 в Vs;
          T = T ∪ {e}
        }
    } // end-while
// конец алгоритма МОД
```

Замечание. Ср. с примером о связности. Найти $W1 \in Vs$, такое, что $u \in W1$. То же для v и W2. Это операция НАЙТИ. Заменить в Vs подмножества W1 и W2 на их объединение $W1 \cup W2$ - это операция ОБЪЕДИНИТЬ.

Жадный алгоритм построения МОД (Kruskal, 1956)



1, 3, 4, 9, 15, 16, 17, 20, 23

Сложность (быстрый поиск – медленное объединение)

~ $m \log m + n^2$ (!)

	a	b	c	d	e	f	g
a-g 1	g	b	c	d	e	f	g
c-d 3	g	b	d	d	e	f	g
b-g 4	g	g	d	d	e	f	g
c-g 9	g	g	g	g	e	f	g
b-c 15							
d-g 16							
e-d 17	g	g	g	g	g	f	g
a-b 20							
a-f 23	f	f	f	f	f	f	f

Жадный алгоритм построения МОД (Kruskal, 1956)

1.Корректность алгоритма (Теорема+индукция).

2.Сложность алгоритма $O(m \log m)$ при соответствующей реализации набора непересекающихся подмножеств Vs.

Сортировка $O(m \log m)$. Очередь с приоритетами – пирамида → $O(k \log m)$. В худшем случае $O(m \log m)$.

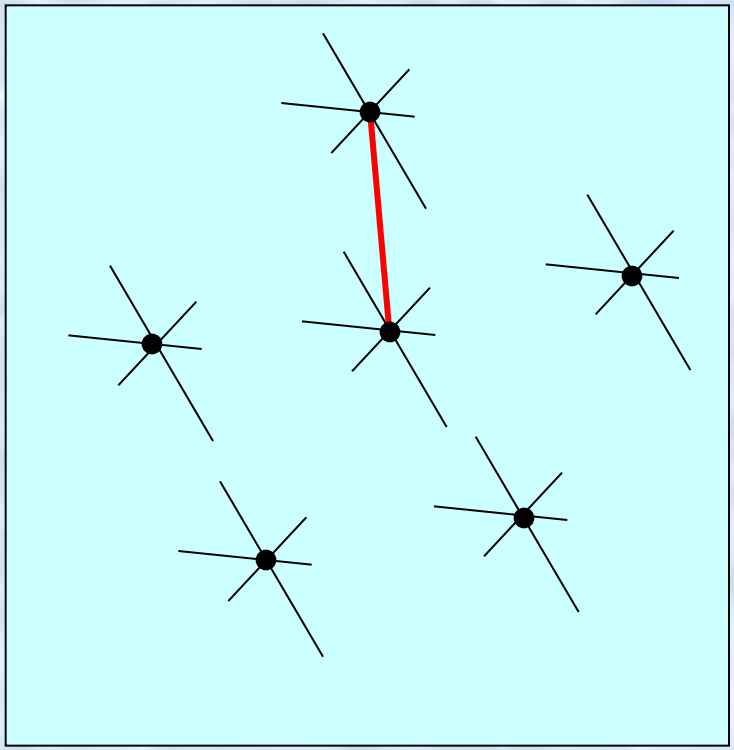
Базовые операции с набором непересекающихся подмножеств:

- принадлежность заданной вершины к одному из подмножеств (НАЙТИ);
- ОБЪЕДИНИТЬ подмножества.

(См. следующую лекцию)

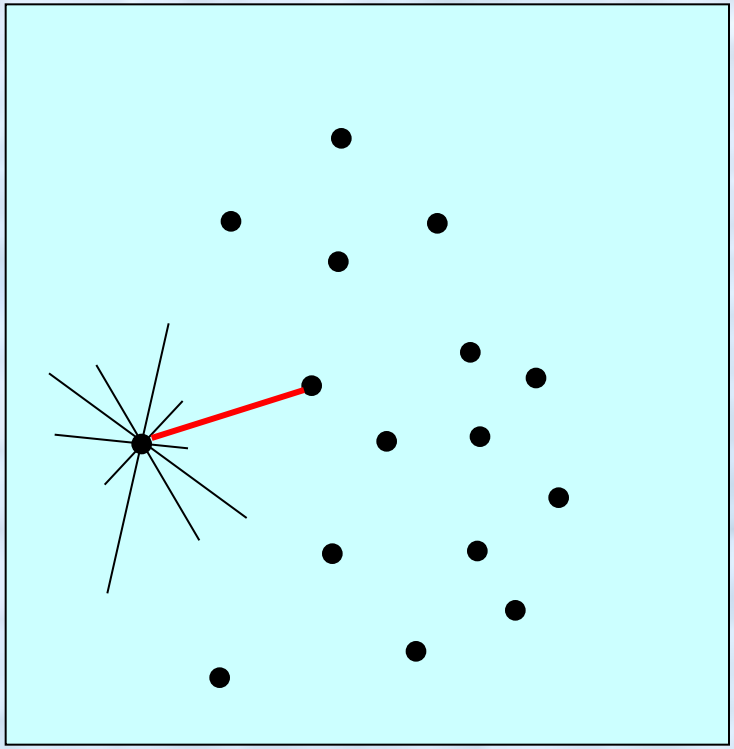
3. Алгоритм Ярника-Прима-Дейкстры построения МОД

(Jarnik, Prim, Dijkstra; алгоритм "ближайшего соседа")



Жадный алгоритм

(на каждом шаге дерево растёт за счёт слияния поддеревьев)



Алгоритм ЯПД

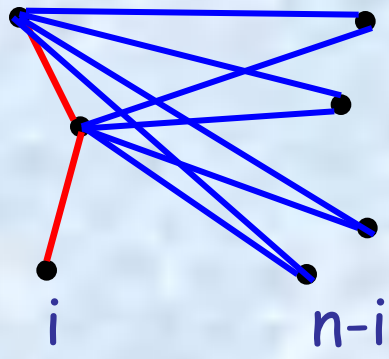
(выбор *ближайшей вершины к дереву*, на каждом шаге дерево вырастает на одну ветку)

Алгоритм Ярника-Прима-Дейкстры

Корректность алгоритма:
из теоремы, при этом (V_1, T_1) – дерево, а остальные (V_i, T_i) – изолированные вершины, т.е. $V_i = \{v_{j(i)}\}$ и $T_i = \varnothing$.

Предполагаемая сложность: пусть G – полный граф, тогда на i-ом шаге при выборе «ближайшей» для всех i вершин текущего дерева необходимо просмотреть все остальные $(n - i)$ изолированные вершины, т.е. всего $i\,(n - i)$ вариантов. Суммарно по всем шагам получим

$$\sum_{i=1}^{n-1} i(n-i) = \frac{1}{6}n^3 + O(n^2).$$



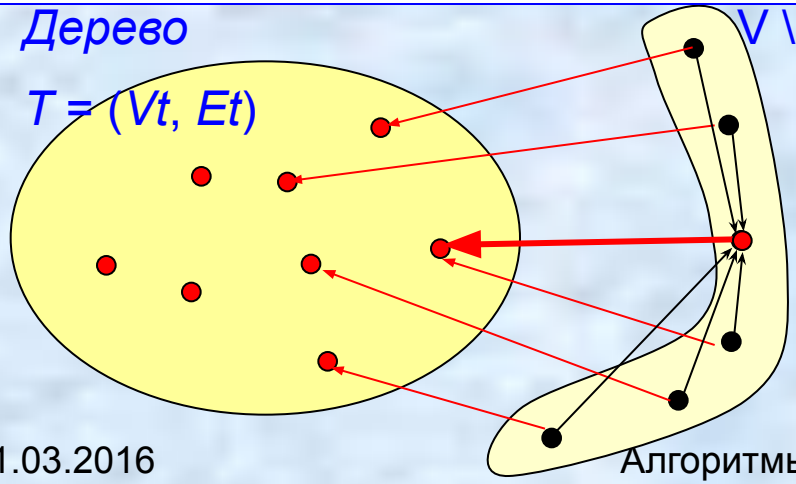
Можно улучшить алгоритм до $O(n^2)$.

Алгоритм Ярника-Прима-Дейкстры

Используем специальную СД, которая облегчает выбор ближайшей вершины, но требует коррекции на каждом шаге.

Пусть *Near*[1..*n*] - массив вершин, *Near* [*v*] – вершина дерева *T* = (*Vt*, *Et*), ближайшая к вершине *v*, ещё не включённой в *T*.

Тогда на *i*-ом шаге находим ближайшую к дереву вершину за (*n* – *i*) сравнений. Коррекция оставшихся *Near* [*] требует (*n* – *i* – 1) действий, т.е. всего на *i*-ом шаге не более 2(*n* – *i*), а суммарно по всем шагам будет $O(n^2)$.



$$\sum_{i=1}^{n-1} 2(n-i) = 2n(n-1) - 2\sum_{i=1}^{n-1} i =$$

$$= 2n(n-1) - n(n-1) = n(n-1)$$

Алгоритм Ярника-Прима-Дейкстры

Вход : V - множество вершин графа G=(V,E), a d [1..n][1..n] – матрица весов

Выход: множества Vt - вершин, Et - ветвей МОД, Wt - общий вес МОД

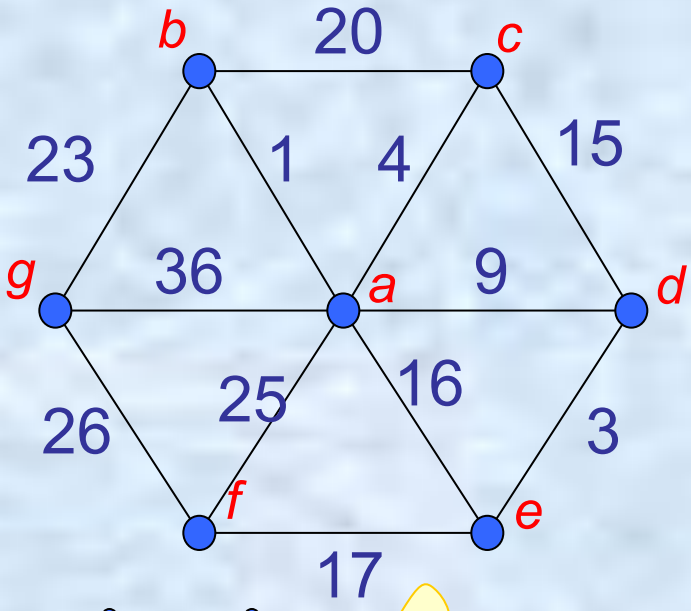
Рабочие: near[1..n] - массив вершин

```
//алгоритм МОД
//выбор произвольной вершины v0
 Vt = {v0};   Et = { } ;   Wt = 0;
for (∀ v ∈(V \ Vt) ) near[v] = v0; //инициализация near[*]
while  (Card(Vt) < n) {
 } //end-while
} // конец алгоритма МОД
```

Алгоритм Ярника-Прима-Дейкстры

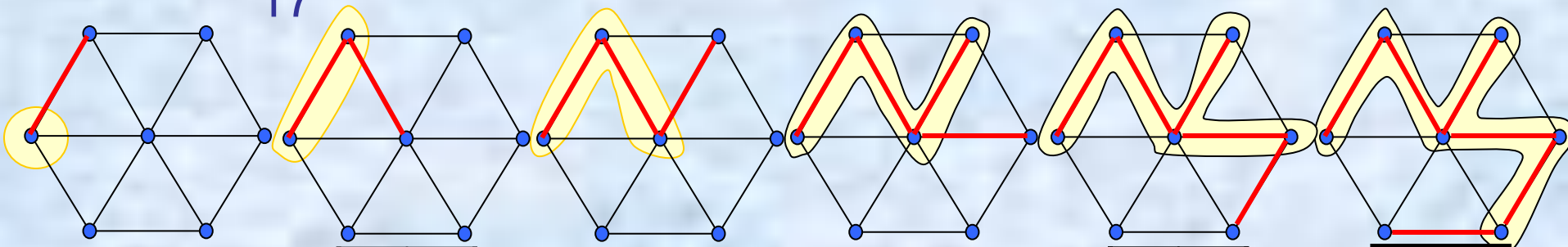
```
//Детализация фрагмента ф1:
 min = +∞ ;
 for (∀ x ∈(V \ Vt))
    if (d[x][near[x]] < min) {
         min = d[x][near[x]] ;
    v = x;
    }

/* Можно наряду с near[*] использовать рабочий массив
расстояний  dist[x] = d[x][near[x]]  */
```

Алгоритм Ярника-Прима-Дейкстры



Начальная вершина – *g*

W(T) = 23+1+4+9+3+17 = 57



a	g
b	g
c	♦
d	♦
e	♦
f	g

a	b
b	
c	b
d	♦
e	♦
f	g

a	
b	
c	a
d	a
e	a
f	a

a	
b	
c	
d	a
e	a
f	a

a	
b	
c	
d	
e	d
f	a

a	
b	
c	
d	
e	
f	e

4. Нижняя граница задачи нахождения МОД

Худший случай: полный граф $m = n(n-1)/2$.

Алгоритм ЯПД – (асимптотически) *оптимальный*.

Действительно, входная симметричная матрица весов содержит $m=n(n-1)/2$ элементов. В том числе и вес минимального ребра. Минимум из m чисел нельзя найти быстрее, чем за $(m-1)$ сравнений.

Пусть для заданной матрицы решена задача МОД за число шагов (операций) $X(n)$. Решением является дерево, содержащее $(n-1)$ ребро, в том числе кратчайшее. За $(n-2)$ шага можно извлечь это ребро, т.е. *минимальный элемент матрицы*.

Тогда если $X(n) + (n-2) < (m-1)$, т.е. $X(n) < (m-1) - (n-2)$, то и задачу нахождения минимума из $n(n-1)/2$ элементов можно решить за менее, чем $n(n-1)/2 - 1$ шагов, что неверно.

Отсюда $X(n) \geq (n-1)(n-2)/2$.

Итак, нижняя граница задачи МОД есть $\Omega(n^2)$,

и алгоритм ЯПД – (асимптотически) *оптимальный*.

4. Нижняя граница задачи нахождения МОД (2)

Можно рассуждать иначе (чуть менее формально).

Во входной весовой матрице $m=n(n-1)/2$ элементов. Ни один из них не может быть исключён из анализа. Пусть всё-таки один из элементов алгоритмом был проигнорирован. Он точно не войдёт в построенное ОД. Именно он мог оказаться минимальным ребром, и, следовательно, построенный остов не будет являться минимальным (!).

Реализация алгоритма Ярника-Прима-Дейкстры для разрежённых графов

Пусть наряду с near[*] используется рабочий массив расстояний dist[x] = d[x][near[x]].

Реализуем из данных (x, near[x], dist[x]) очередь с приоритетами по ключу dist[x].

Минимальный элемент извлекается (с восстановлением пирамиды) за $O(\log n)$ действий. При помещении вершины v в остовное дерево производим коррекцию данных о вершинах, смежных с v и находящихся в очереди. Коррекция каждой такой вершины x (ребра {v, x}) требует $O(\log n)$ действий, но при этом такое ребро «возникает» и «исчезает» лишь один раз за всё время.

Т.о. суммарно требуется $O((n+m)\log n)$ операций, или, что то же, $\boldsymbol{O(m \log n)}$.

Ср. с $O(n^2)$ (например, при $m = O(n)$)

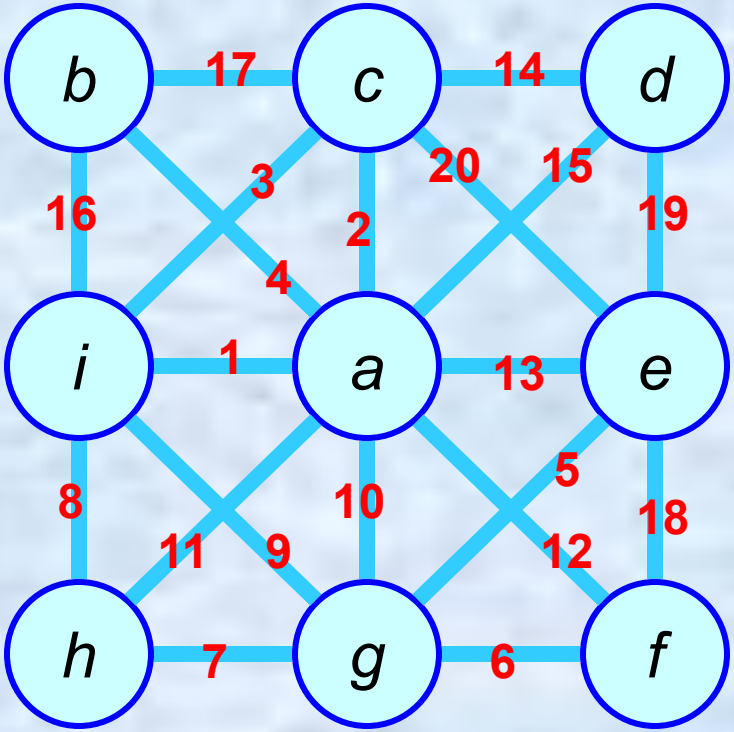
Алгоритм Ярника-Прима-Дейкстры построения МОД

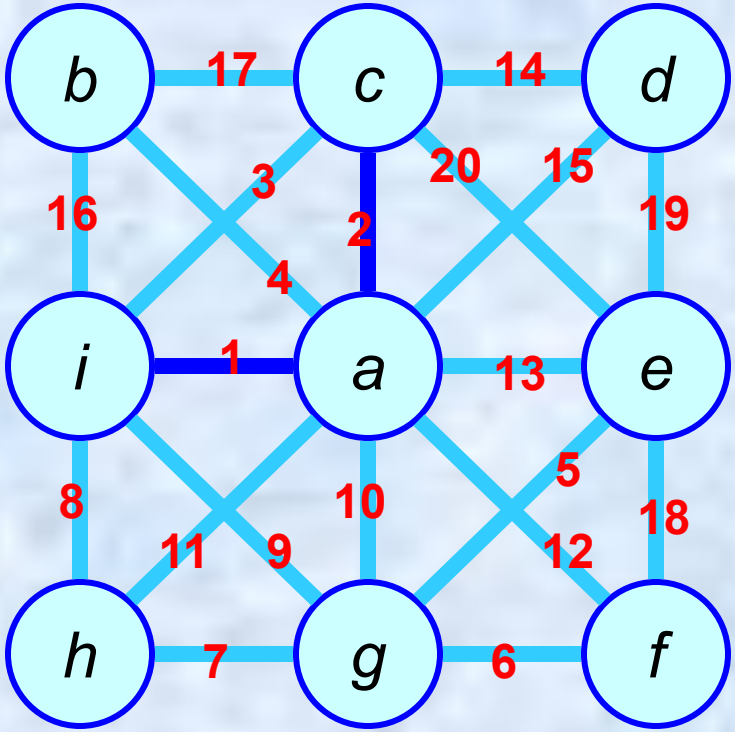
Примеры выполнения.

Варианты:

- Стандартный - для «плотных» графов ($m = O(n^2)$)
- Для разрежённых графов ($m = O(n)$)

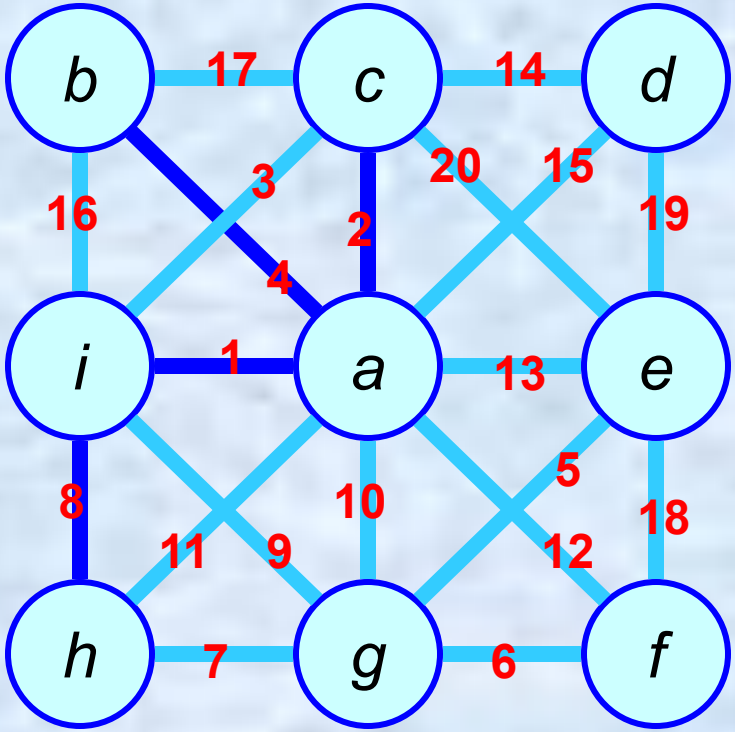
Пример 1. МОД. Алгоритм ЯПД (n = 9; m = 20)





*	a	b	c	d	e	f	g	h	i
0 шаг	-	4	2	15	13	12	10	11	1
1 шаг	-	4	2	15	13	12	9	8	!-
2 шаг	-	4	!-	14	13	12	9	8	-

Пример 1. МОД. Алгоритм ЯПД (n = 9; m = 20) 2



После двух шагов

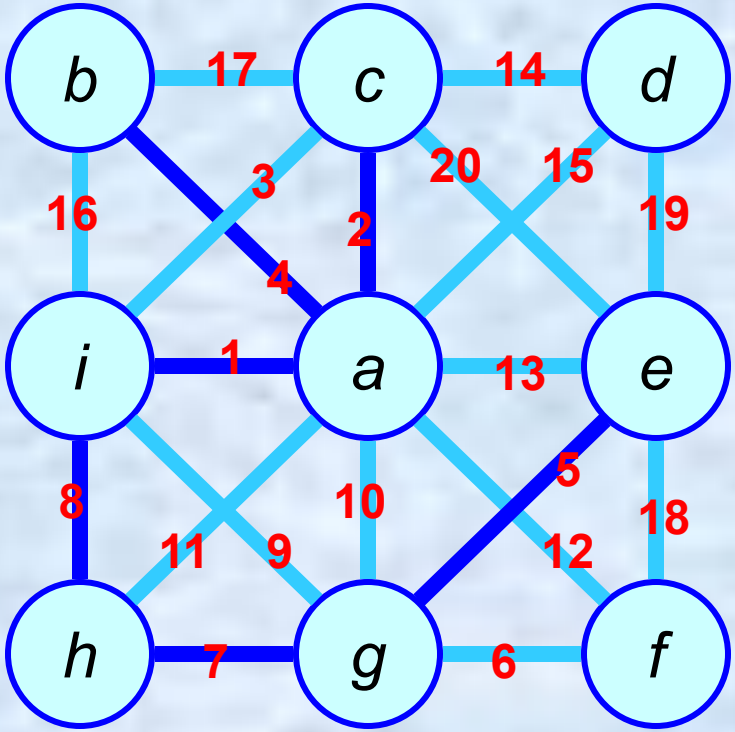
Текущее МОД: {a, i, c}

После четырёх шагов

Текущее МОД: {a, i, c, b, h}

	a	b	c	d	e	f	g	h	i
2 шаг	-	4	!-	14	13	12	9	8	-
3 шаг	-	!-	-	14	13	12	9	8	-
4 шаг	-	-	-	14	13	12	7 (7)	!-	-

Пример 1. МОД. Алгоритм ЯПД (n = 9; m = 20)



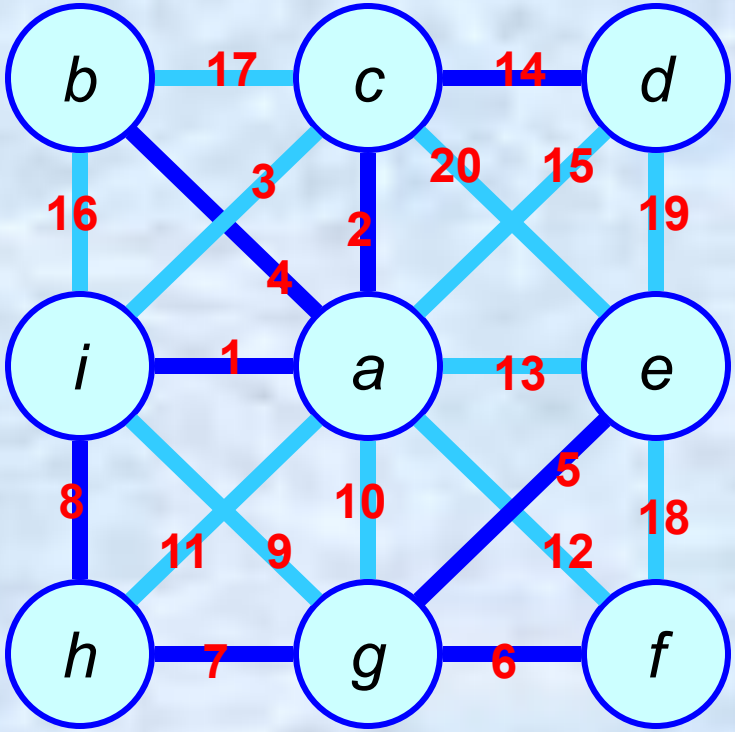
После четырёх шагов
Текущее МОД: {a, i, c, b, h}

После шести шагов

Текущее МОД: {a, i, c, b, h, g, e}

	a	b	c	d	e	f	g	h	i
4 шаг	-	-	-	14	13	12	7	!-	-
5 шаг	-	-	-	14	5 (5)	6	!-	-	-
6 шаг	-	-	-	14	!-	6	-	-	-

Пример 1. МОД. Алгоритм ЯПД (n = 9; m = 20)



После шести шагов

Текущее МОД: {a, i, c, b, h, g, e}

После восьми шагов

Результат - МОД:

{a, i, c, b, h, g, e, f, d}

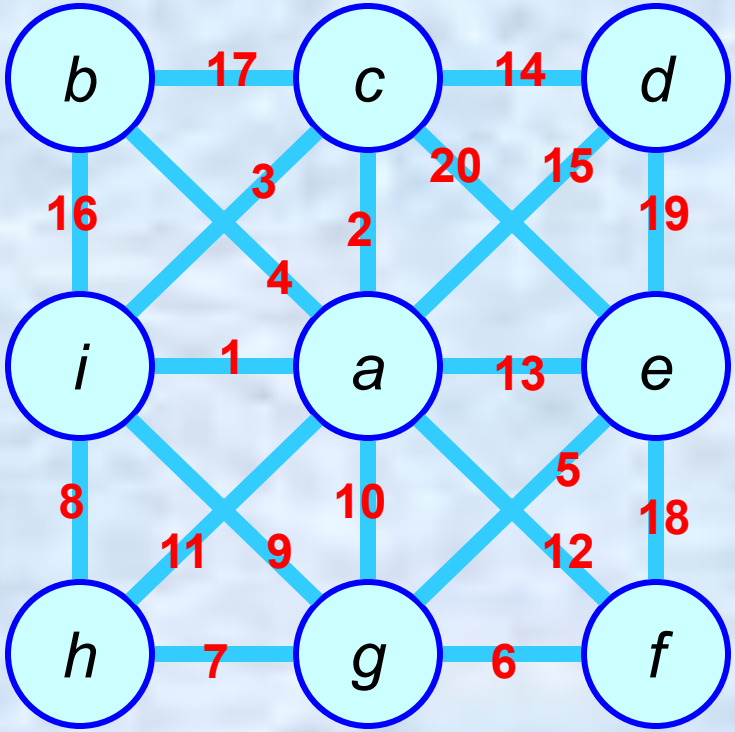
W = 1+2+4+8+7+5+6+14 = 47

	a	b	c	d	e	f	g	h	i
6 шаг	-	-	-	14	!-	6	-	-	-
7 шаг	-	-	-	14	-	!-	-	-	-
8 шаг	-	-	-	!-	-	-	-	-	-

Пример выполнения алгоритма Ярника-Прима-Дейкстры (вариант для разрежённых графов)

См. слайд 26

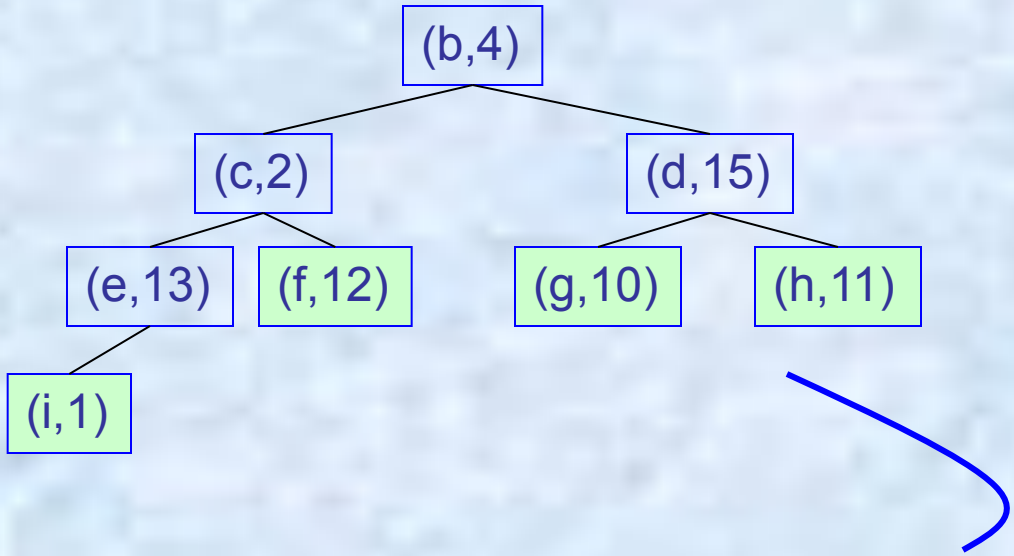
Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)



(x, near[x], dist[x])

dist[x] = d[x][near[x]].

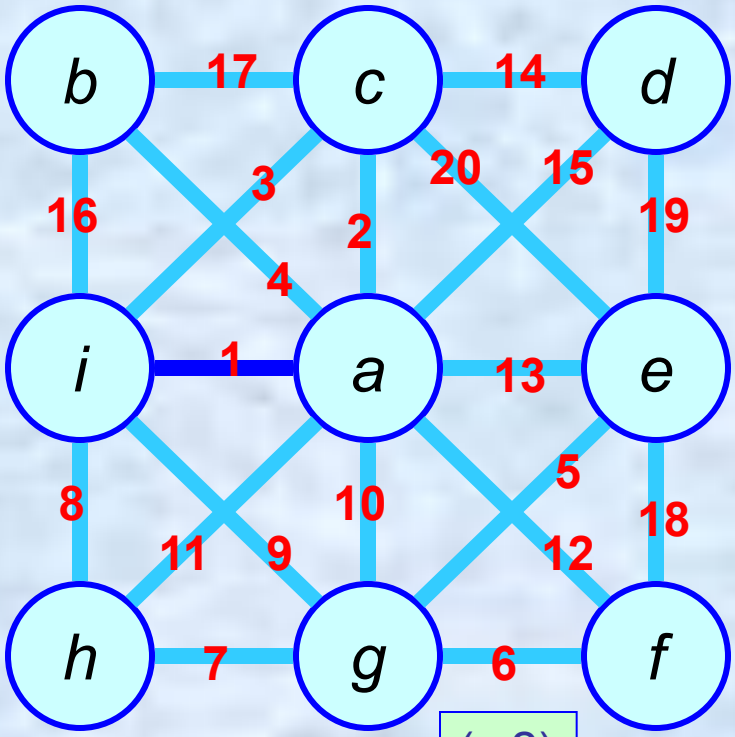
(x, dist[x])



(b,4) (c,2) (d,15) (e,13) (f,12) (g,10) (h,11) (i,1)

	a	b	c	d	e	f	g	h	i
0 шаг	-	4	2	15	13	12	10	11	1

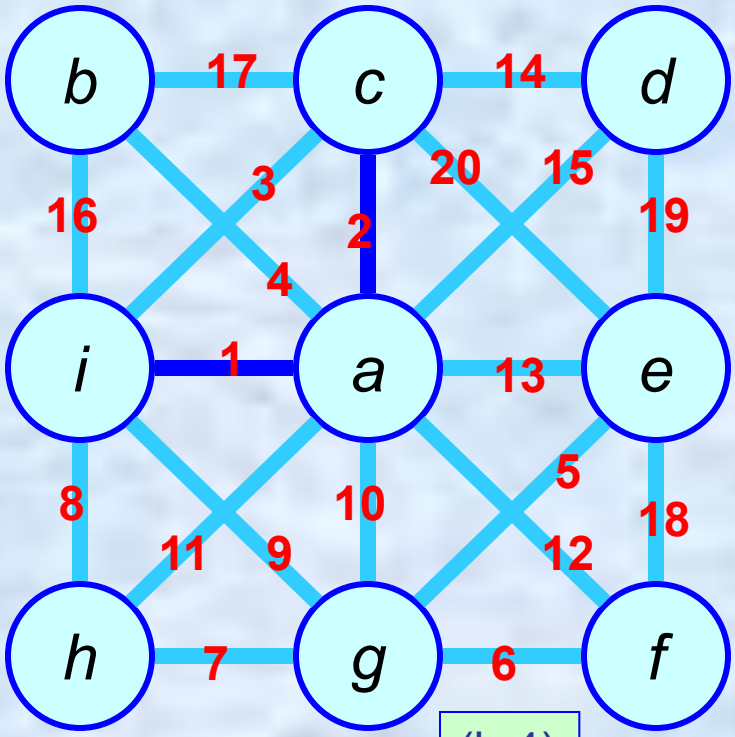
Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)



Шаг 1

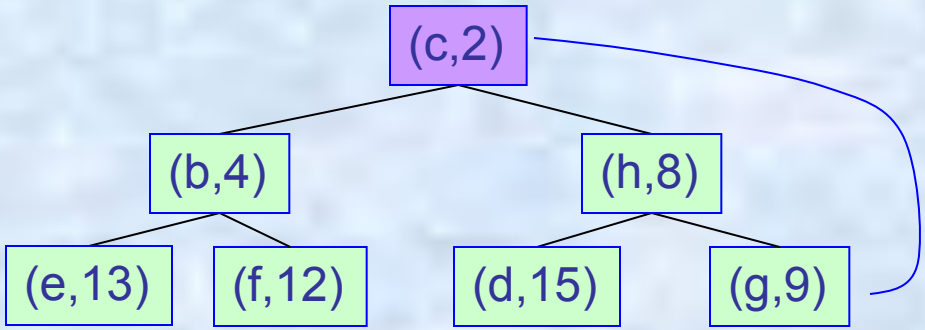
ТМОД = {a, i}

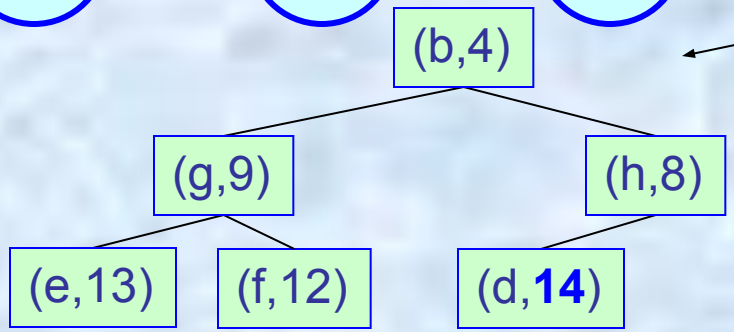
Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)



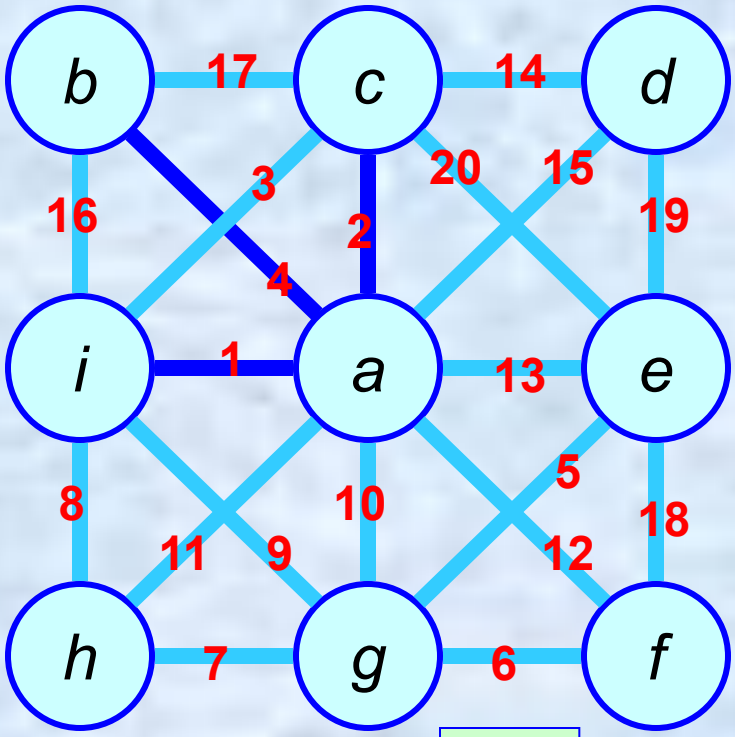
Шаг 2

ТМОД = {a, i, c}



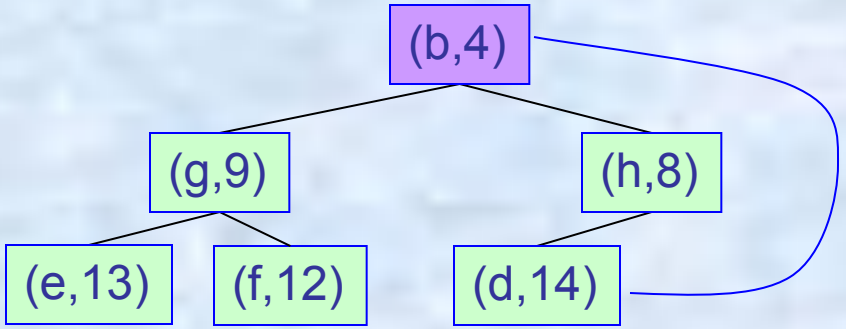


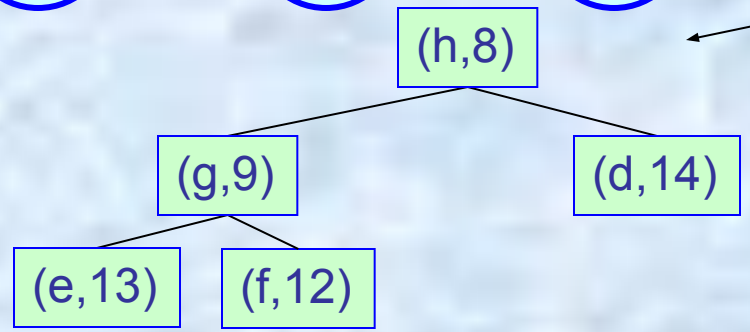
Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)



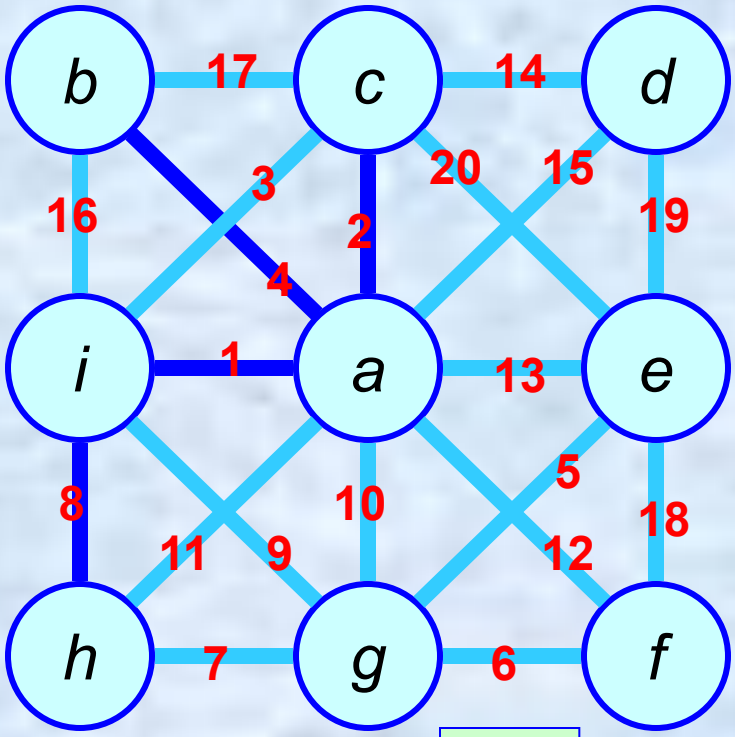
Шаг 3

ТМОД = {a, i, c, b}



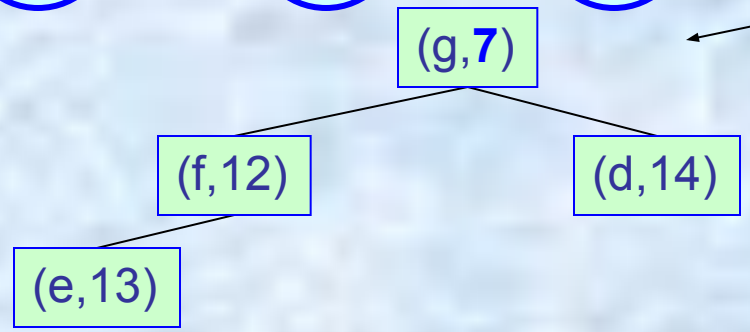


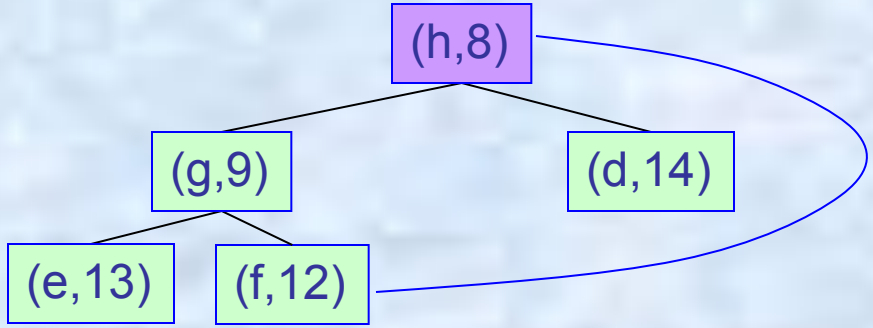
Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)



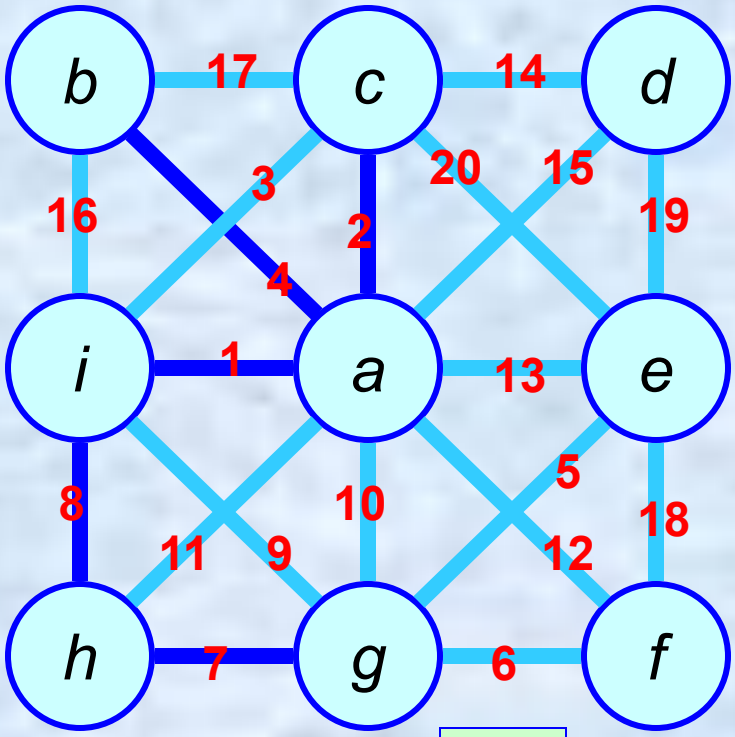
Шаг 4

ТМОД = {a, i, c, b, h}



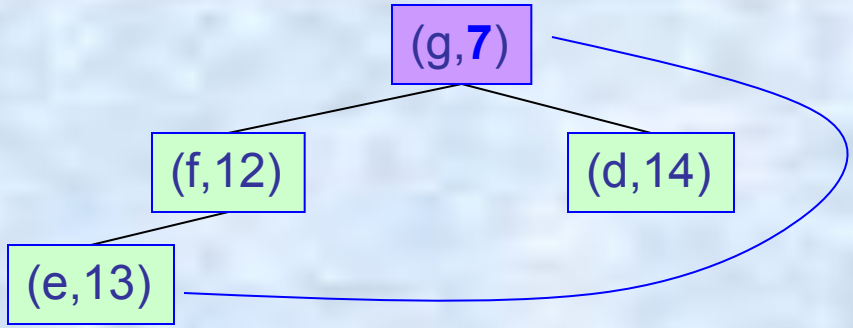


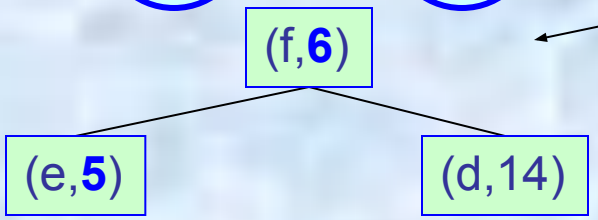
Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)

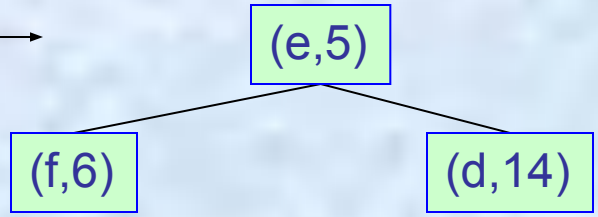


Шаг 5

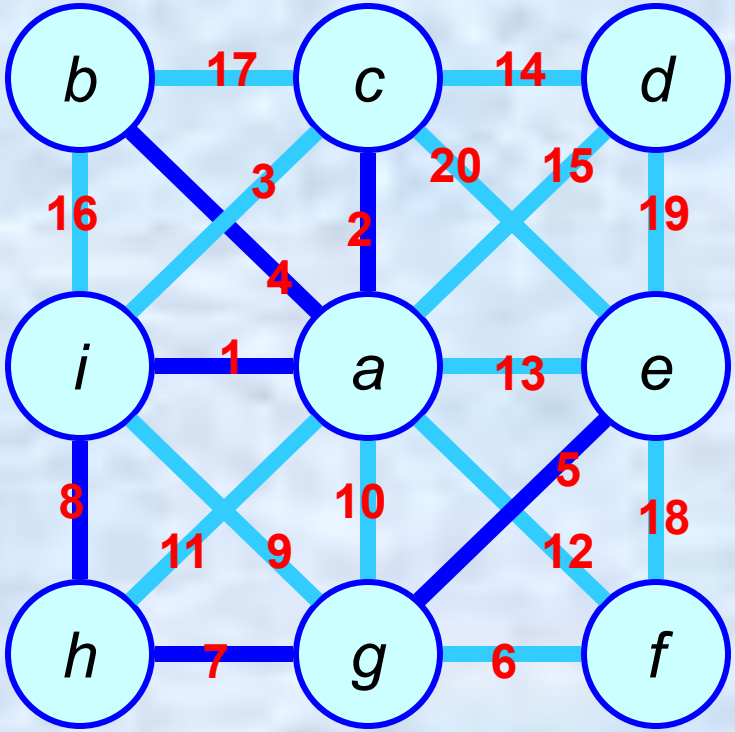
ТМОД = {a, i, c, b, h, g}





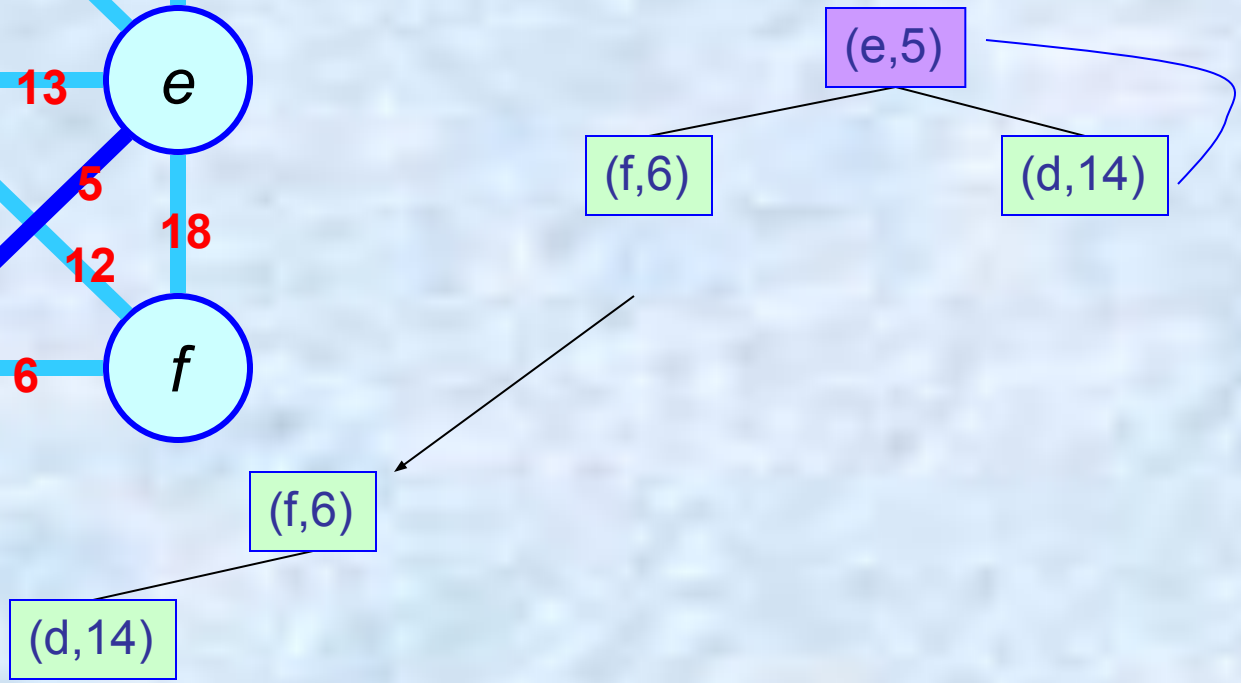


Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)

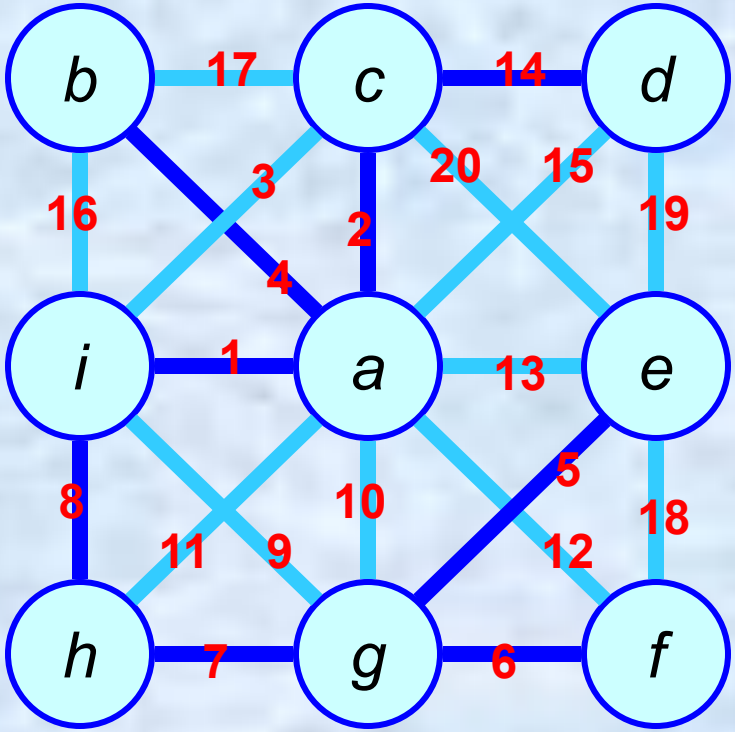


Шаг 6

ТМОД = {a, i, c, b, h, g, e}



Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)



Шаг 7

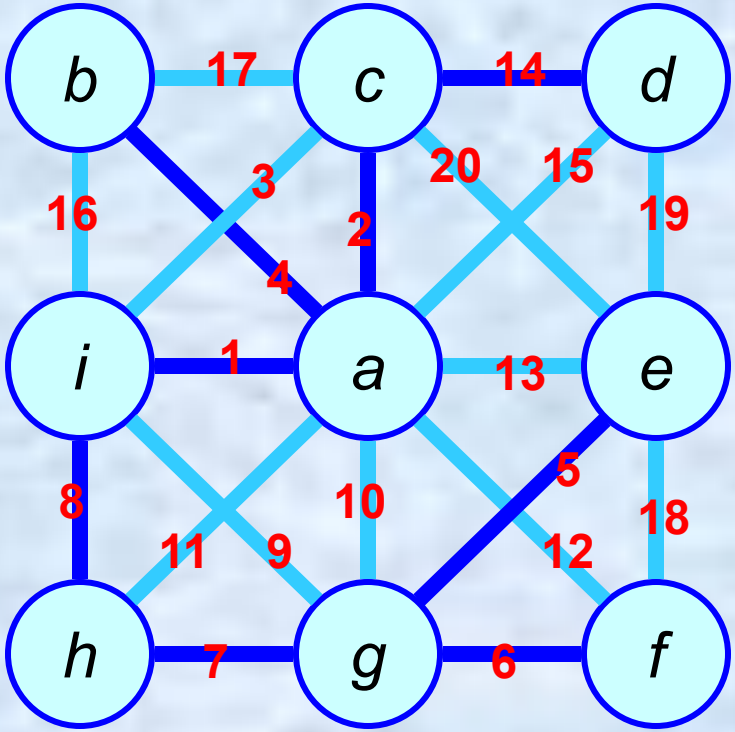
ТМОД = {a, i, c, b, h, g, e, f}

(f,6)

(d,14)

(d,14)

Пример 2. МОД. Алгоритм ЯПД (n = 9; m = 20)



Шаг 8

ТМОД = {a, i, c, b, h, g, e, f, d}

(d,14)

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ