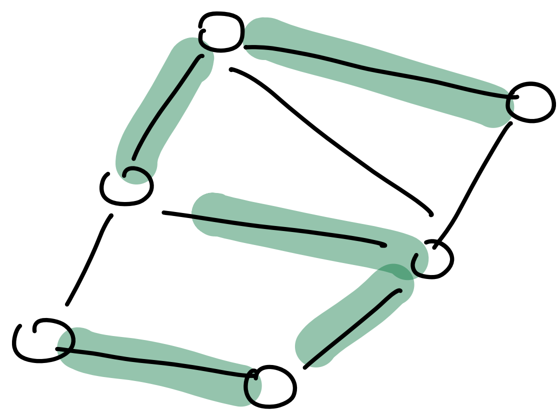
Минимальные остовные деревья

I

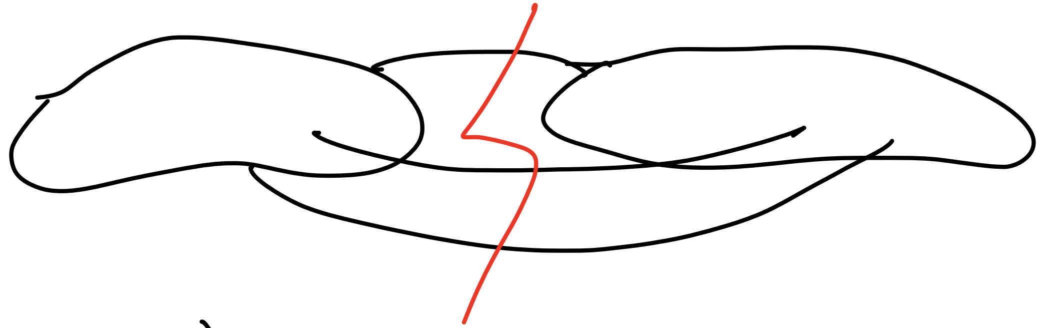
<u>Опр</u> Дерево — связный граф без циклов.

<u>Опр</u> Минимальное остовное дерево (миностов) - это дерево, вершинами кот-го являются <u>все</u> вершины изнач. графа и суммарный вес ребер которого мин.



<u>Опр</u> Подграф $T \subset G$ графа наз-ся <u>безопасным</u>, если T является подграфом миностова G. (а ребра и вершины T - безопасными)

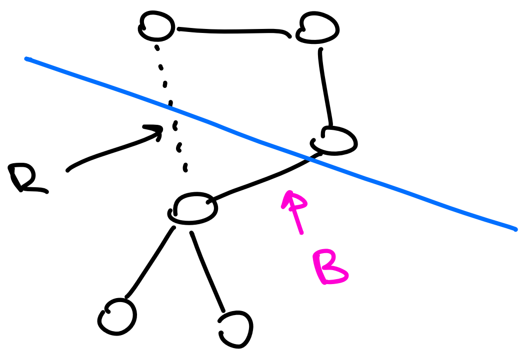
<u>Опр</u> <u>Разрезом</u> связного графа наз-ся подмн-во вершин и ребер графа, т.ч. образуется граф с двумя комп. связности



Лемма (о безопасном ребре)

Выберем некий разрез подграфа миностова. Тогда ребро минимального веса, пересекающее этот разрез является безопасным.

Д-во:



$\Rightarrow$ при добавлении ребра R мин. веса образуется цикл.

$\Downarrow$

удаляем ребро B, т.к $w(R) < w(B) \Rightarrow$

$\Rightarrow$ противоречие.

Сл-е: 1. Все ребра имеют разный вес

$\Downarrow$

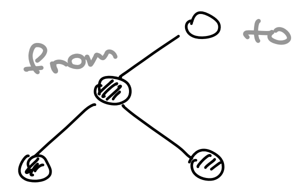
миностов уникален.

2. Произведение ребер можно также считать. веса $a \to \log(a)$

II Алгоритм Прима (поиска миностова)

1. Алгоритм за $O(NM)$

- Мин. ребро → соединяем used с не used
- Изначально миностов состоит из одной вершины




```
used = [False... False]   — не посещены
used[0] = True
for i = 0... N-1
    min_w = ∞
    to, from;
    for j = 0... M:          # кол-во ребер
    {
        edge = G.getEdges()[j]        || edge.to
                                         edge.from
                                         edge.weight
        if min_w > edge.weight and used[edge.from]
                and not used[edge.to]
            min_w = edge.weight
            from = edge.from
            to = edge.to
    }
    used[to] = True     # вершина to добавлена в миностов
    print(from, to)
```

② Алгоритм за $O(N^2)$

{ Что лучше? $O(NM)$ или $O(N^2)$

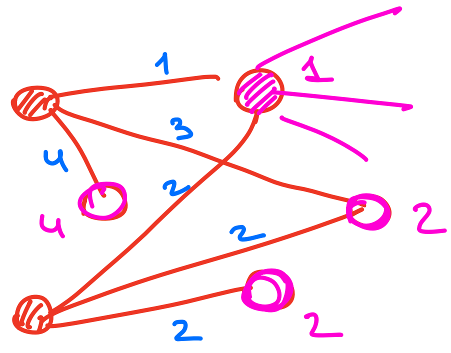
Вводим сущности:

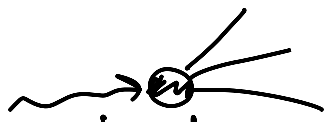
min_edge[u] ~ минимальный вес входящего ребра в u $(v \to u)$

best_edge[u] ~ вершину v, из кот-й пришли в u

```
min-edge = [+∞ ... +∞]
min_edge[0] = 0
best_edge = [... ]

for i = 0 ... N:
{
    v = -1
    for u = 0 .... N:
    {
        if not used[u] and (v == -1 or
                            min_edge[u] < min_edge[v])
            ↳ v = u
    }
    used[v] = true
    ch = G.get children(v)
    for j = 0... ch.size():
```



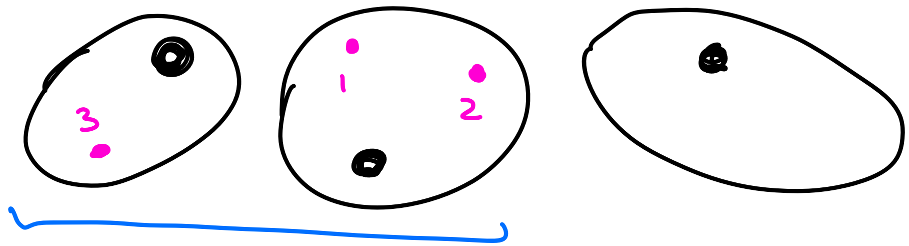


```
u = ch[j]
if    u.weight  <  min_edge[u.to]:
          min_edge[u.to] = u.weight
          best_edge[u.to] = v
}
```

$\underline{O}(N^2 + M) = \underline{O}(N^2)$

III Система непересекающихся множеств.

• - лидер



union
объединение

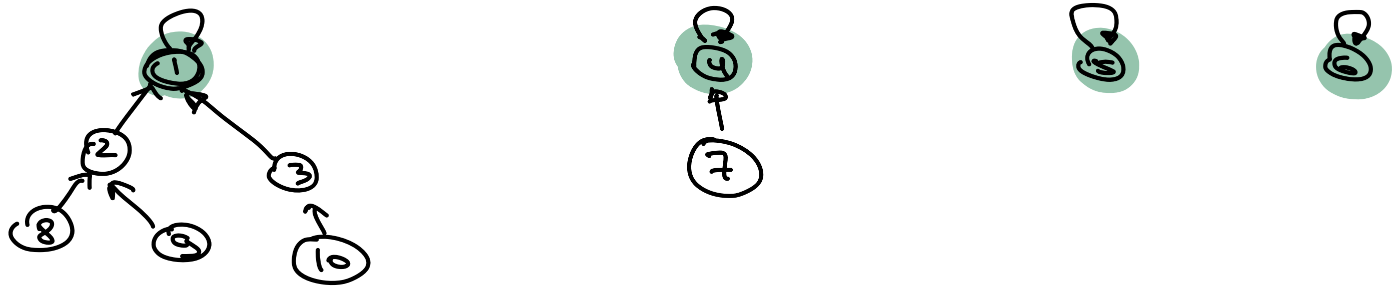
leader(1) = leader(2)
⇓
лежат в одном мн-ве

Храним:

1 ... N - элементы мн-в.

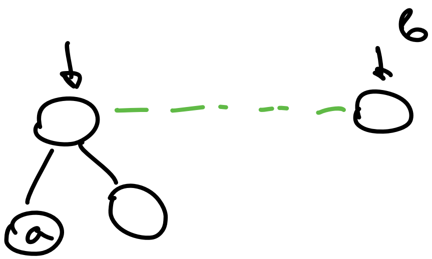
parent[i] - родитель вершины i

лидер имеет предком себя



Тривиальный алгоритм:

```
Функция leader(v):
    if parent[v] = v:
        ↳ return v
    return leader(paren[v])
```

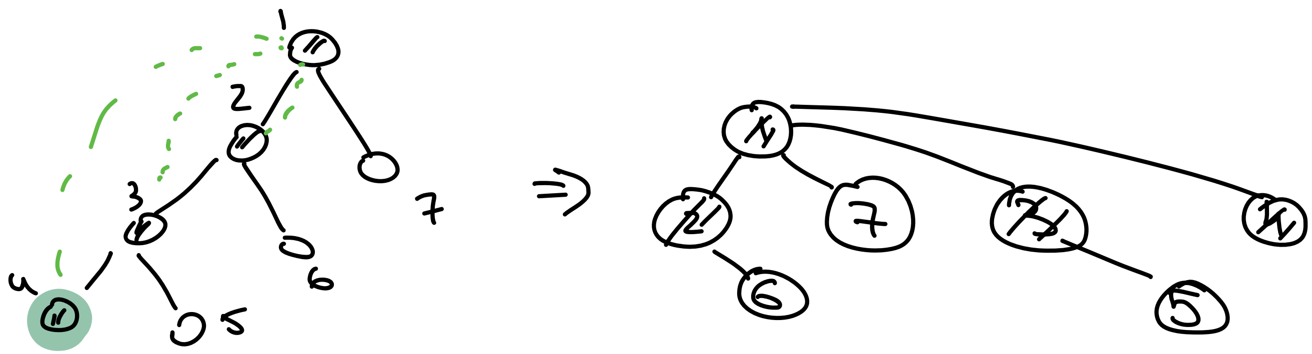



```
Функция unite(a, б):
    a = leader(a)
    б = leader(б)
    parent[a] = б
```

<u>Улучшение:</u> (а) Эвристика сжатия

```
Функция leader(v):
    if parent[v] = v
        ↳ return v
```

p[v] = leader(p[v])
return p[v]

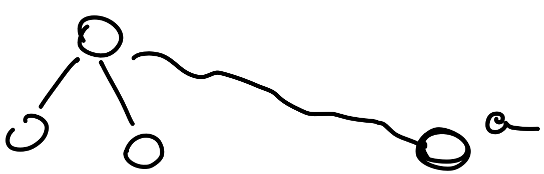


Асимптотика (б/д)
$\underline{O}(\log N)$

(б) Ранговая эвристика

храним высоту h дерева

функция



b h(b) = 2

a h(a) = 1

```
union(a, b):
a = leader(a)
b = leader(b)
if h(a) > h(b):
    swap(a, b)
h(b) = max(h(b), h(a)+1)
parent[a] = b
```

a - min
b - max

высота: $\underline{O}(\log N)$

(б) Весовая эвристика

храним кол-во вершин в поддереве

$s[a]$ = # вершин-детей a

Функция union (a, b):

a = leader(a)

b = leader(b)

if $s[a] > s[b]$:

    swap(a, b)

$s[b]$ += $s[a]$

parent$[a] = b$

a – min
b – max

Высота: $\underline{O}(\log N)$

При использовании эвр сжатия + (эвр. ранг или эвр весота)

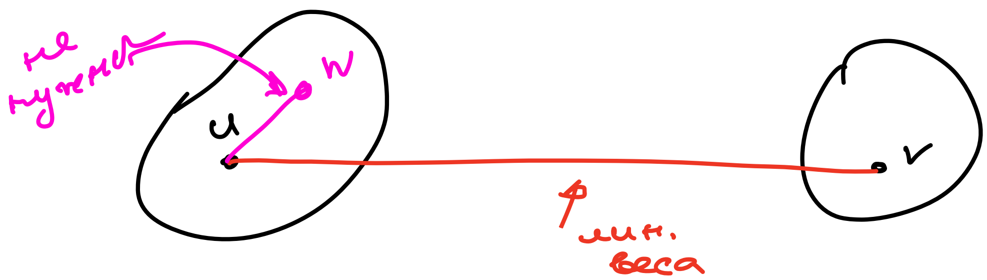
Асимптотика: $\underline{O}(\alpha(N))$

где $\alpha(N)$ – ф-я Аккермана

$(\forall N)^*$ $\alpha(N) < 4$

IV Алгоритм Краскала

1. Сортируем ребра → $\underline{O}(M \log M)$



2. Берем это ребро в случае, если оно соединяет два разных мн-ва

Алгоритм:

```
edges = G.GetEdges()
edges.sort(...)        // O(MlogM) по весу.
for i = 0...edges.size()
    edge = edges[i]
    if leader(edge.from) != leader(edge.to):
        unite(edge.from, edge.to)
        print(edge.from, edge.to)
```

$\underline{O}(M)$

Асимптотика : $\underline{O}(M\log M + M) = \underline{O}(M\log M)$