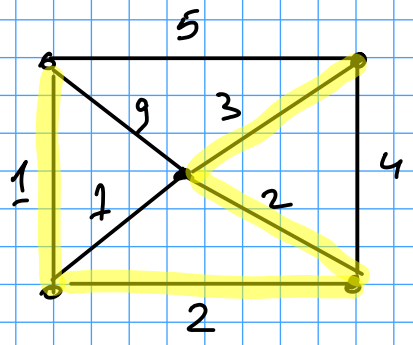
Минимальное остовное дерево (MST, Minimal Spanning Tree)

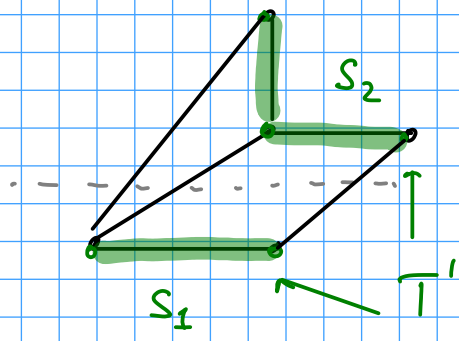


Свойство разреза

$\exists$ T – MST

$\nexists$ $T' \subset T$

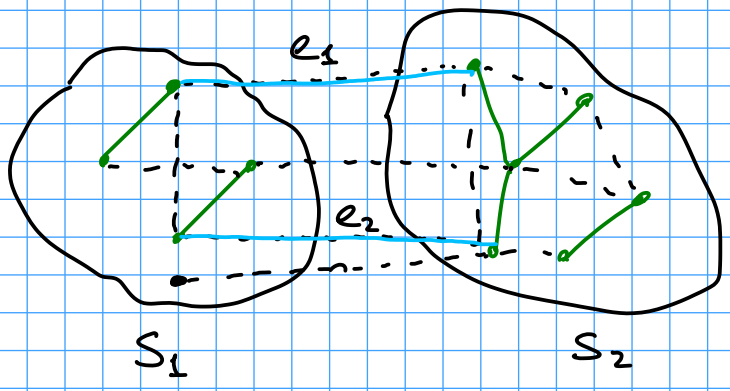
Δ $S_1 \sqcup S_2$ – разрез : в T' нет ребра из S_1 в S_2



Утв: $\exists$ e – min ребро из разреза S_1, S_2

$T' \cup \{e\}$ – подмножество некоторого MST T''.

Док-во:



$\exists$ e_1 – min ребро в разреза $S_1 \sqcup S_2$

$\exists$ $\exists$ MST T^* : $e_1 \notin T^*$

Добавим в T^* ребро e_1 $\Rightarrow$ появился цикл

$\exists$ e_2 – это второе ребро цикла, проход. через разрез $S_1 \sqcup S_2$.

Δ $T = T^* \cup \{e_1\} \setminus \{e_2\}$

1. T — дерево
2. стоимость T ≤ стоимость T^*

т.к. $w(e_1) \le w(e_2)$ ◁

Алгоритм Прима

Идея: поддерживаем одну связную компоненту и на # к ней мин ребро.

```
Prim ((V, E))
    for v ∈ V:
        dist[v] = ∞, prev[v] = -1
    dist[1] = 0
    T = ∅
    Q = make_priority_queue((1, 0))
    while Q.size() > 0:
        v = Q.extract_min()
        for (v, u) ∈ E:
            if dist[u] = ∞:
                Q.insert((u, w(v,u)))
                dist[u] = w(v,u); prev[u] = v
            else if dist[u] > w(v,u):
                dist[u] = w(v,u); prev[u] = v
                Q.decrease_key(u, w(v,u))
        if v ≠ 1: T = T ∪ {(prev[v], v)}
    return T
```

Время работы: $O((V+E)\log V)$

$O(V^2)$

$O(E)$, $E = V^{1+\varepsilon}$

Алгоритм Краскала

Система непересек. множеств СНМ

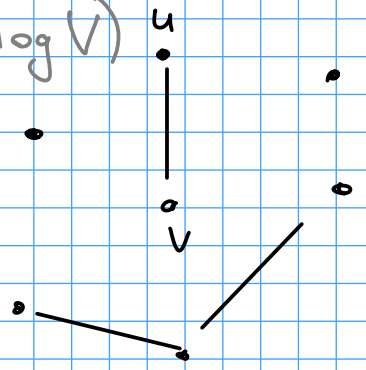
```
Kruskal ((V, E))
    for v ∈ V:                O(V)
        MakeSet (v)
    T = ∅
    Sort (E)                  O(E)
    for (u,v) ∈ E :           O(E)
        if Find (u) ≠ Find (v)   O(log V)
            Union (u, v)
            T = T ∪ {(u,v)}
    return T
```

АТД Disjoint Sets

MakeSet (v)

Find (v)

Union (v_1, v_2)



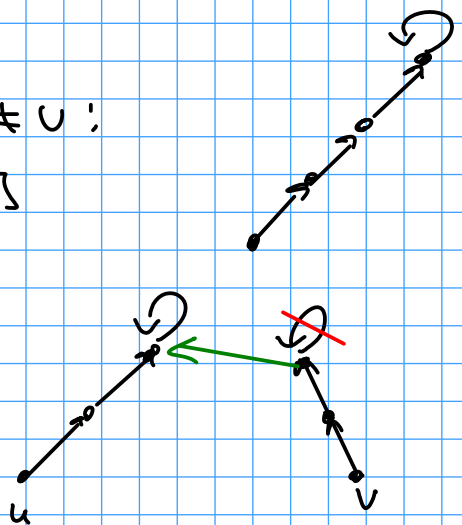
Множество соответствует поддереву дерева.

MakeSet

Find: идём по стрелочкам пока не дойдём.

Union: Объединили два дерева в одно

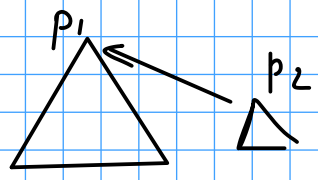
```
MakeSet (v):
    parent[v] = v
Find (v):
    while parent[v] ≠ v:
        v = parent[v]
    return v
Union (v, u):
    p1 = Find (v)
    parent[p1] = Find[u]
```



Вспомогательные функции:

```
MakeSet(v):
    parent[v] = v
    rank[v] = 0

Union(u, v):
    p1 = Find(u)
    p2 = Find(v)
    if rank[p1] > rank[p2]:
        parent[p2] = p1
    if rank[p2] > rank[p1]:
        parent[p1] = p2
    if rank[p1] = rank[p2]:
        parent[p2] = p1
        rank[p1] = rank[p1] + 1
```



Утв: rank[x] = высота поддерева

Утв: у поддерева с корнем ранга k $\geq 2^k$ вершин

(по индукции)

$\geq 2^k + 2^k = 2^{k+1}$

Всего n вершин $\Rightarrow$ размер поддерева

$$2^k \leq n \Rightarrow k \leq \log(n)$$

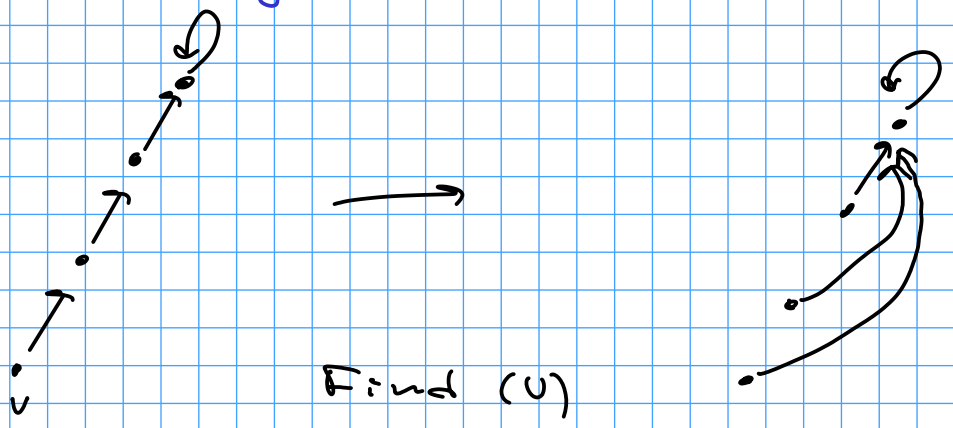
$\Rightarrow$ Время max пути в Find $= O(\log(n))$

MakeSet $O(1)$, Union $= O(\text{Find})$

Find $= O(\log n)$

$\Rightarrow$ Алгоритм Краскала за $O(E \cdot \log V)$

Сжатие путей




```
Find (v):
    if parent[v] ≠ v:
        parent[v] = Find(parent[v])
    return parent[v]
```

Утв: rank[z] ≥ высота

Th: Сжатие путей даёт оценку $O(\log^* n)$ для всякого Find(v) в среднем

$\log^* x = 1$	$1 \le x < 2$
$\log^* x = 2$	$2 \le x < 4$
$\log^* x = 3$	$4 \le x < 2^4$
$\log^* x = 4$	$2^4 \le x < 2^{2^4}$
$\log^* x = 5$	$2^{16} \le x < 2^{2^{16}}$

Следствие: Алгоритм Крускала работает за $O(E \cdot \log^* V)$

При условии, что E отсортировано по ↑ w