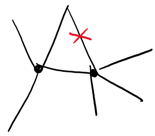
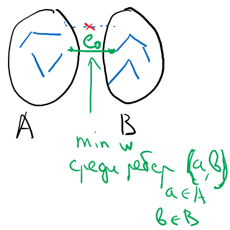
Минимальное остовное дерево (Minimal spanning tree, MST)

G – неор., взвеш., $w_e \geq 0$, связный
Выбрать подмн-во ребер min веса, чтобы граф остался связным.
Такое мн-во будет обр. дерево, т.е. связный граф без циклов.
(точнее, сущ. мн.-во, обр. дерево)
Остовный подграф — содержит все вершины исх. графа.

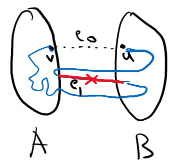


Lm. (о разрезе, о безопасном ребре)
$V = A \sqcup B$, $T_1 \subseteq E$, T_1 можно продолжить до MST
$\nexists\, e \in T_1 : e = \{a, b\}, a \in A, b \in B$
Тогда пусть e_0 — ребро min w среди ребер, пересек. разрез, т.е. $\{\{a,b\} \mid a \in A, b \in B\}$
Тогда e_0 - безопасное, т.е. $T_1 \cup \{e_0\}$ можно продолжить до MST.



Д-во T – MST: $T \supseteq T_1$.
1). $T \ni e_0$ — (+)
2). $T \not\ni e_0$. $e_0 = \{v, u\}$, $v \in A$, $u \in B$. T - связн. $\Rightarrow \exists\, p$ - путь $v \to u$.
p пересекает разрез ребром e_1. (могут быть и другие ребра)

$$T' = T \setminus \{e_1\} \cup \{e_0\}$$



• $T' \supseteq T_1$: $T \supseteq T_1$, $e_1 \notin T_1$

• $T' \ni e_0$

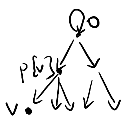
• T' – MST?
$\begin{cases} |E(T')| = |E(T)| - 1 + 1 = |E(T)| = |V(G)| - 1 \\ T' \text{ - связно? } e_1 = \{v_1, u_1\} \end{cases}$
$v_1 \to u_1$ в T': используем путь p и ребро e_0
$\Downarrow$
T' – дерево.
$w(T') = w(T) - w(e_1) + w(e_0) \leq w(T)$ – min вес дерева
$w(e_0) \leq w(e_1)$
$\Rightarrow w(T') = w(T) = w(MST) \Rightarrow T'$ - MST.

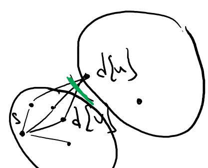
■

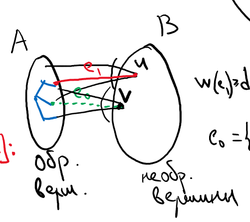
Алгоритмы построения MST.

Алгоритм Прима

```
q = PriorityQueue()
d = [+∞]·n;  p = [-1]·n;  used = [False]·n
d[0] = 0;  q.push((d[0], 0));  ~~p[0] = 0~~
while q is not empty:
    d_cur, v = q.popMin();  used[v] = True
    for u, w in edges[v]:
        if d[u] > w and ~~p[u] == -1~~ not used[u]:
            d[u] = w
            q.changeKey(u, w)
            p[u] = v
```

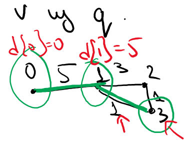






$w(e_1) \geq d[u] \geq d[v] = w(e_0)$
$\Downarrow$ по Lm
e_0 - безопасное
$e_0 = \{v, p[v]\}$

Берём ребро $\{v, p[v]\}$ в T_1 в тот момент, когда достаём v из q.



Time = $O(V^2)$ – на массиве
Time = $O(E \log V)$ – на куче
Mem = $O(V)$

Алгоритм Краскала

all_edges = [(v, u, w)]

```
sort(all_edges, key=w_e)   w↗
for e(v,u,w) in all_edges:
    if v и u в разных комп. связности:
        добавить ребро e в ответ
```