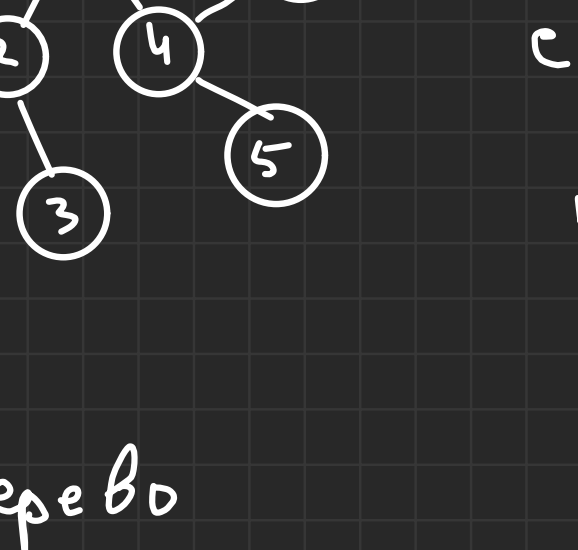
LCA. Наименьший общий предок. Sparse Table. Сведение к RMQ

1 Задача LCA (least common ancestor)

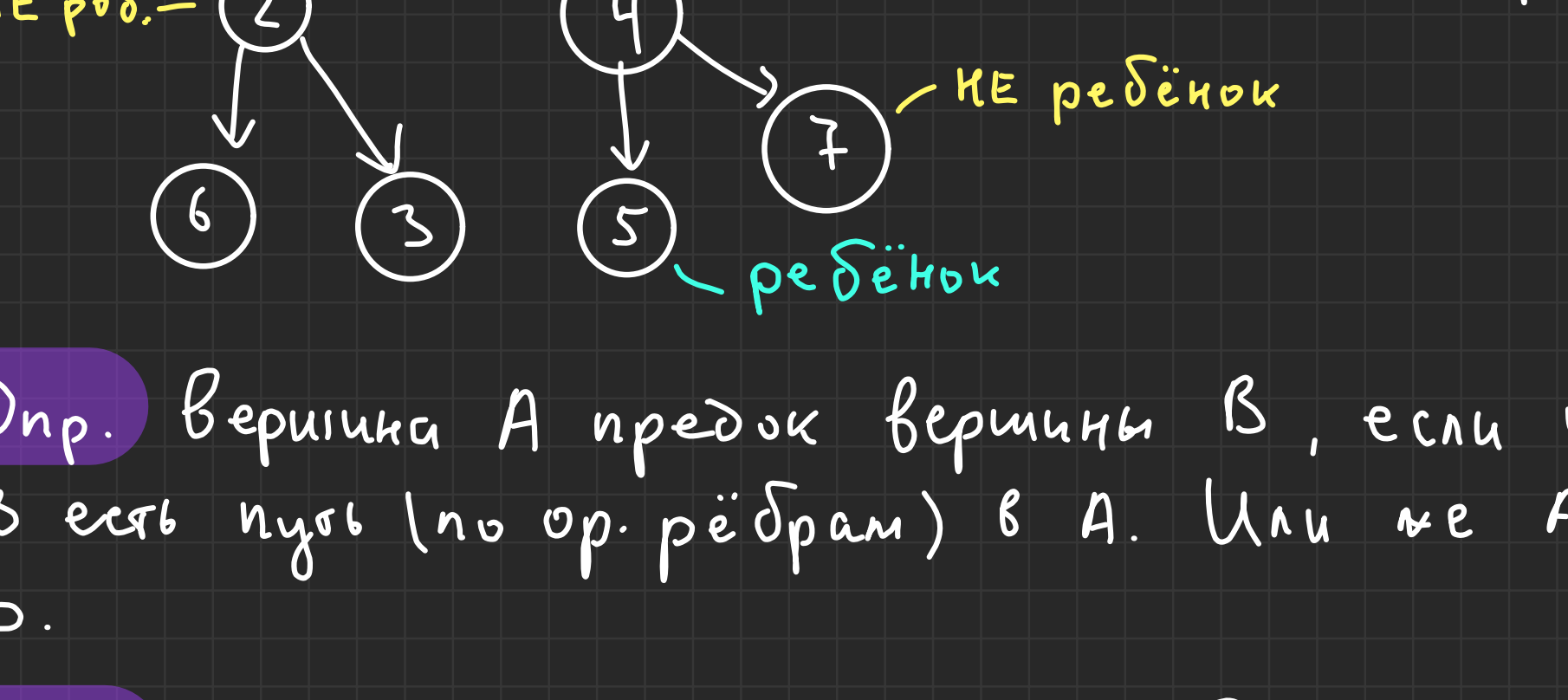
Дерево:



связный неор. графо с кол-вом рёбер = кол-во вершин - 1

между любыми 2 верш. есть ровно 1 путь.

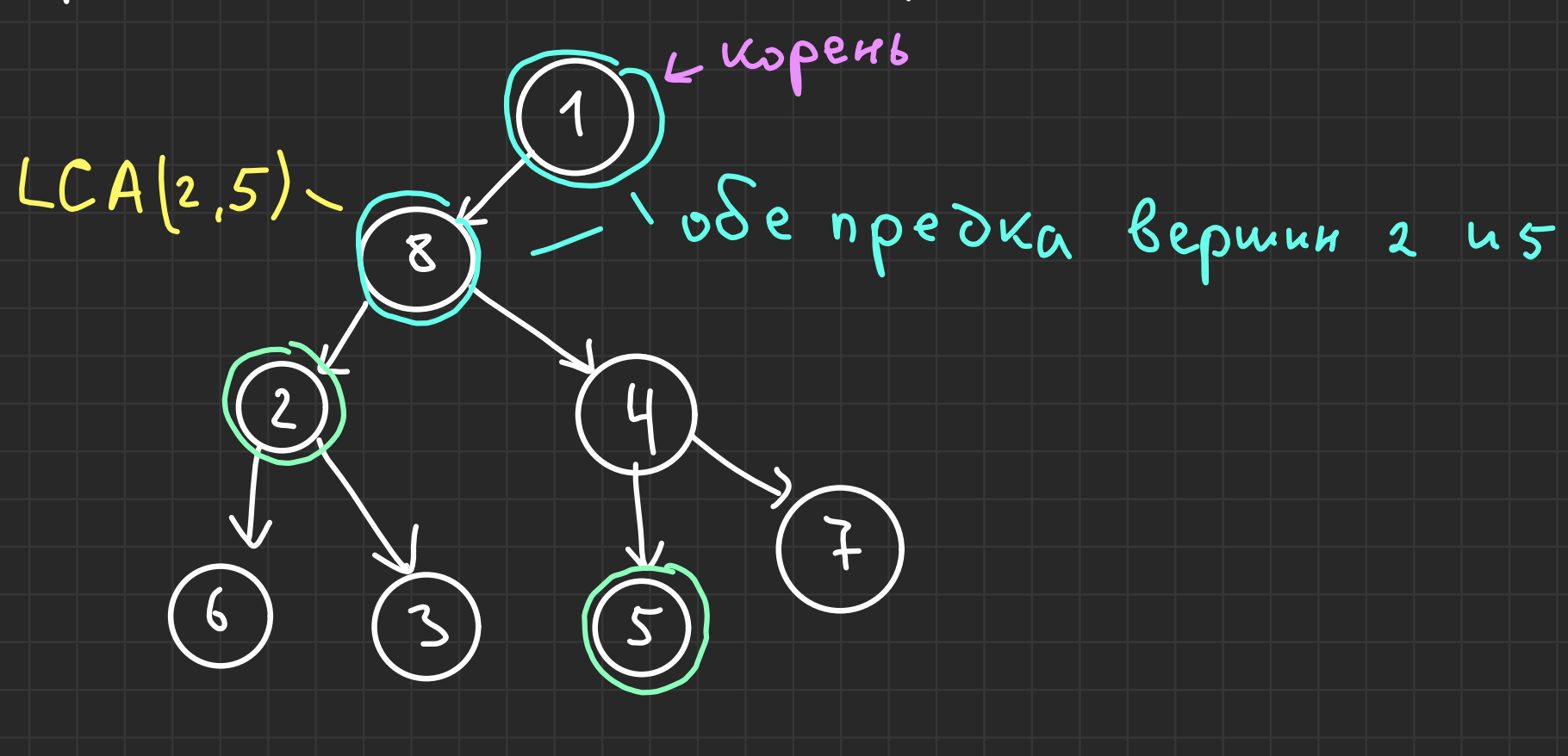
Подвешенное дерево



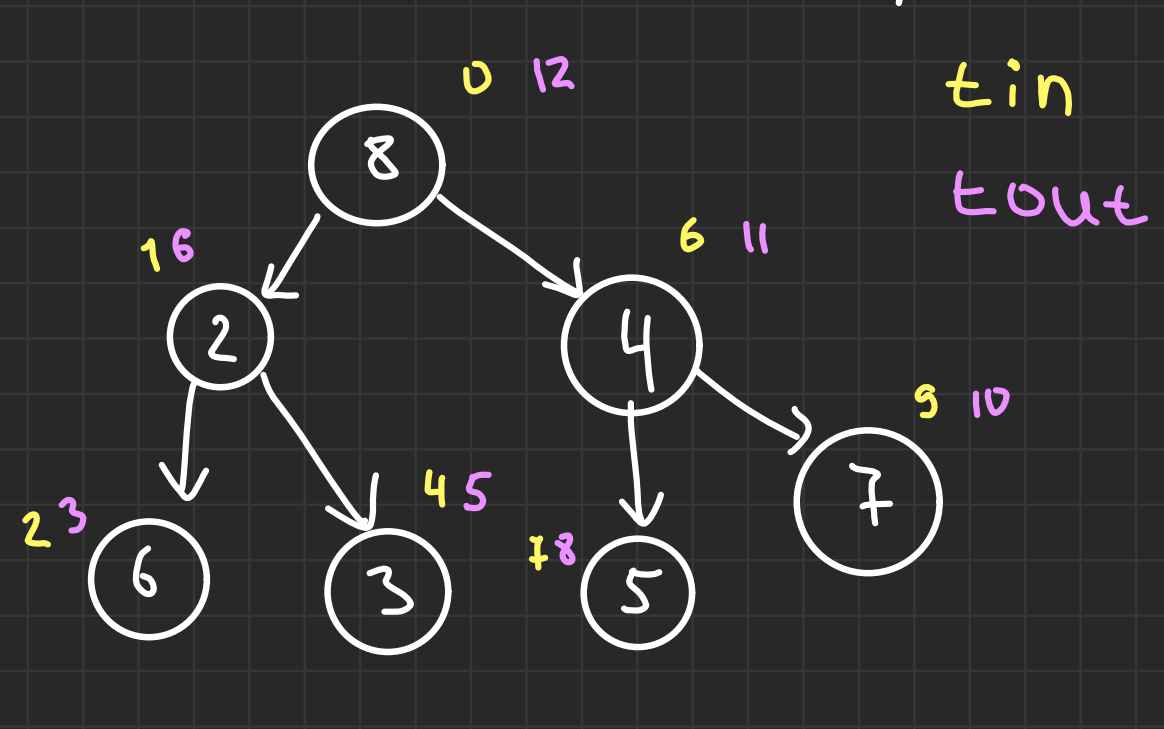
Босс - подчинённые
Родитель - ребёнок

Опр. Вершина A предок вершины B, если из вершины B есть путь (по ор. рёбрам) в A. Или же A в поддереве B.

Опр. LCA вершин A и B это ближайшая к ним вершина которая является родителем A и B.



Опр. $tin(v)$ - время входа в вершину v во время dfs
$tout(v)$ - время выхода (последнего) из вершины v.



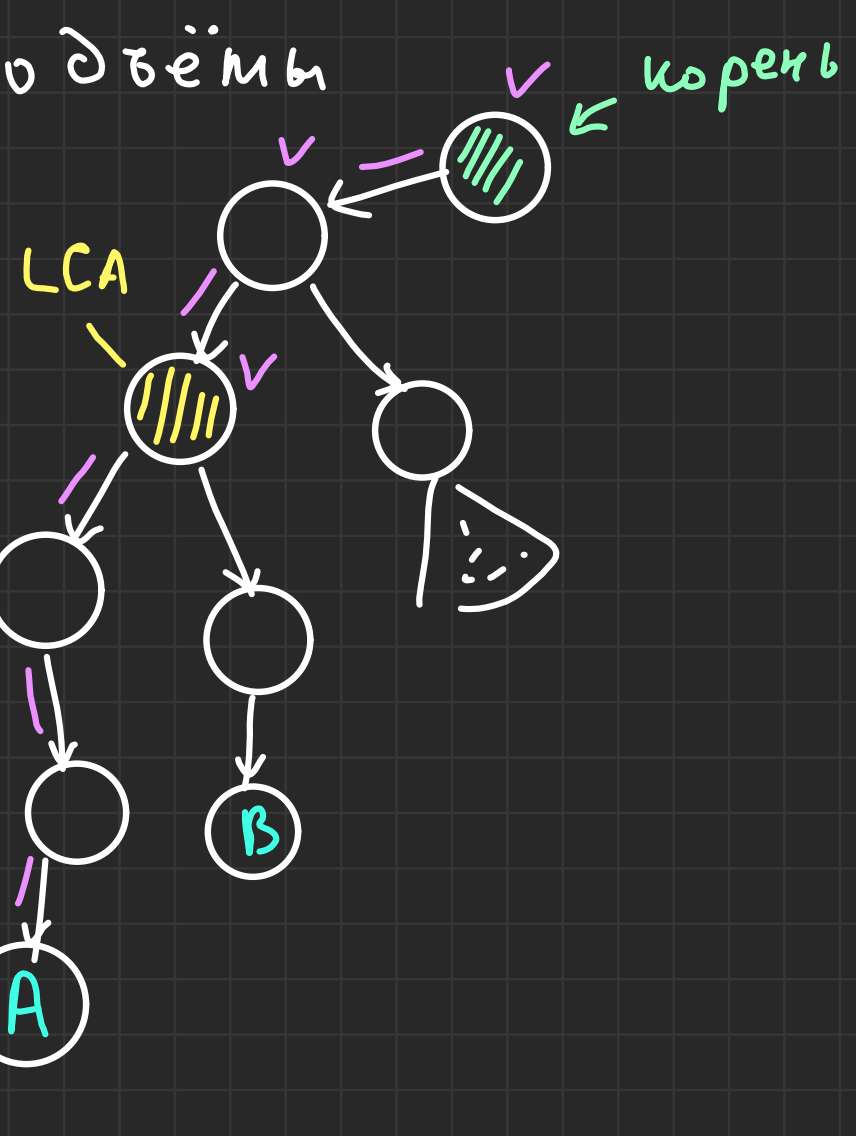
```
timer = 0
void dfs(v, p)
     tin[v] = timer++;
     for u in g[v]:
         if (u ≠ p):
             dfs(u, v)
     tout[v] = timer++;
```

Как определить, что одна вершина предок другой?

Между tin[v] и tout[v] находятся только вершины из поддерева v.

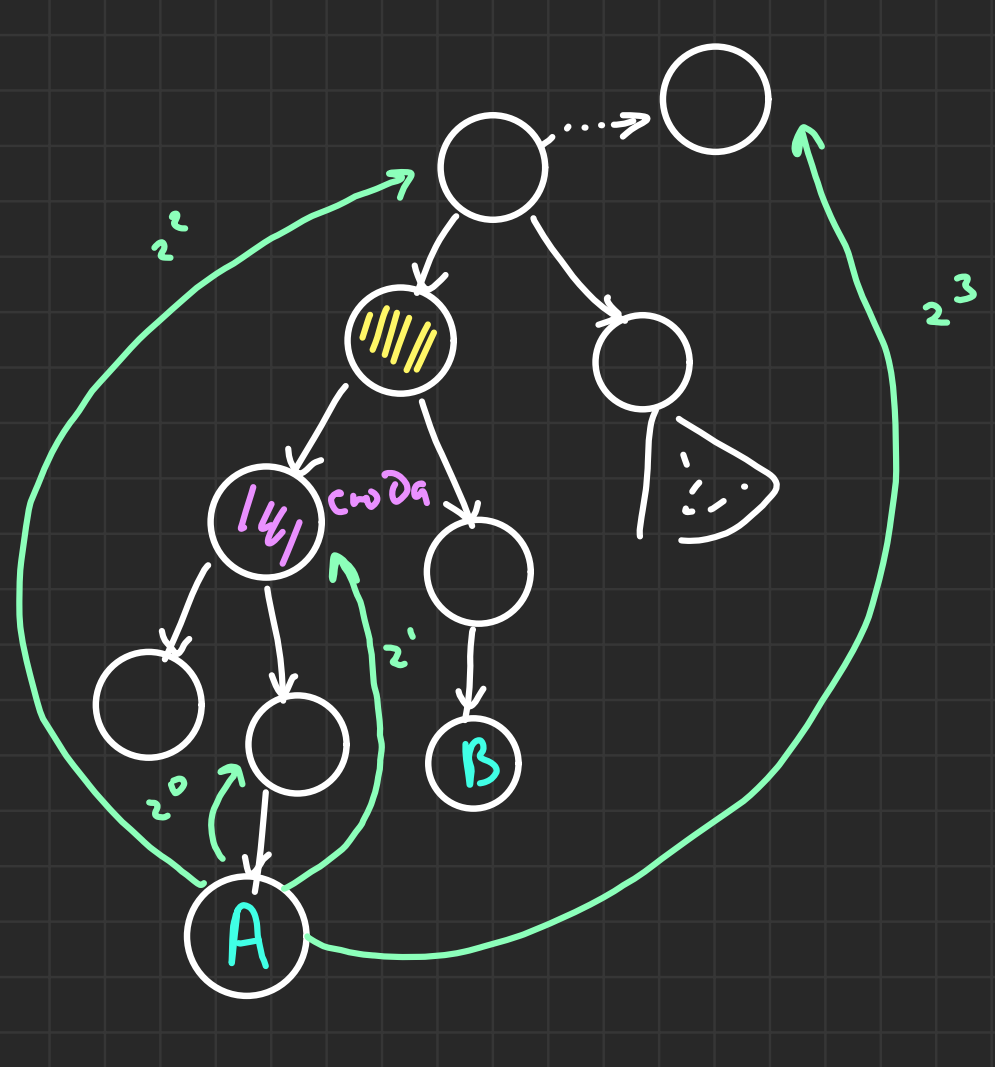
```
bool pred(a, b):  ( a предок b? )
    return tin[a] ≤ tin[b] && tout[a] ≥ tout[b]
```

2 Бинподъёмы



Если C предок A и B то pred(C) тоже предок A и B

Будем прыгать на степени двойки:



Для каждой вершины храним предка на расстоянии 2^k

Каждый раз прыгаем на макс. расстояние так что бы быть всегда НЕ предком вершины B

В конце окажемся в вершине, которая НЕ предок B, но уже следующая вершина предок B.

Давайте заметим, что если мы

$v \xrightarrow{2^k} v' \Rightarrow v \xrightarrow{2^{k+1}}$ ведёт не туда $\Rightarrow$
$v' \xrightarrow{2^k}$ ведёт не туда
$(2^k + 2^k = 2^{k+1})$

$$\underbrace{v \xrightarrow{2^k} v' \xrightarrow{2^k} a}_{2^{k+1}}$$

```
int lca(a, b):
    if pred(a, b):
            return a
    for i = logN-1; i ≥ 0; --i:
         if(!pred(up[a][i], b)):
                 a = up[a][i]
    return up[a][0]
```

предок a на расстоянии 2^i

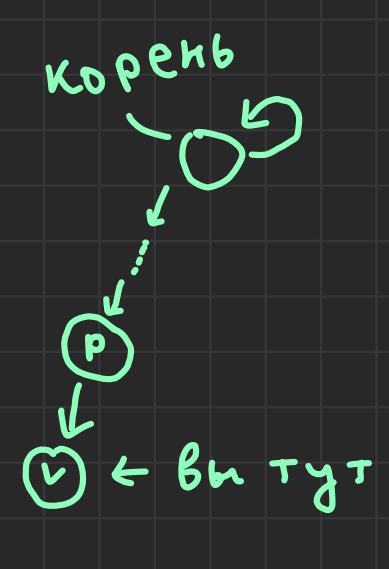
Как насчитать up?

$up[v][k] = up[\underbrace{up[v][k-1]}_{v'}][k-1]$

$$\underbrace{v \xrightarrow{2^{k-1}} v' \xrightarrow{2^{k-1}} a}_{2^k}$$

```
void dfs(v, p)
     up[v][0] = p
     for i = 1... logN:
         up[v][i] = up[up[v][i-1]][i-1]
      for u in g[v]:
          if (u ≠ p):
              dfs(u, v)

  dfs(root, root)
```



Асимптотика:

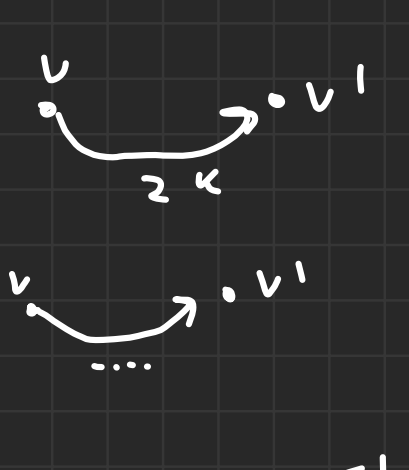
Память: N log N
Время на построение: N log N
Время на запрос: log N

3 Запросы на пути (статичные)

- Минимум / максимум на пути между A и B
- Сумму на пути ...
- GCD, хеш, xor, ...

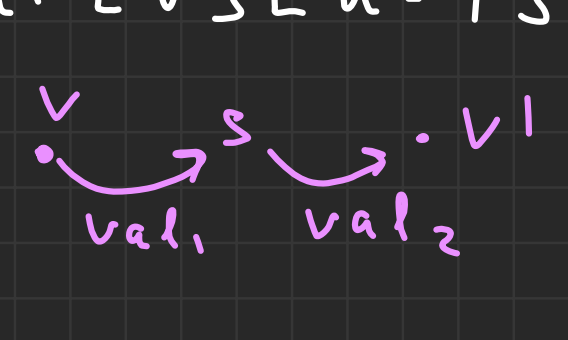
up[v][k] = v'

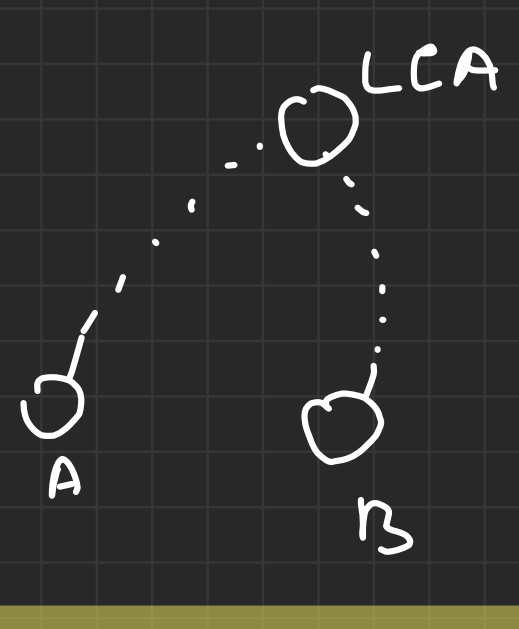
val[v][k]



up[v][k] = up[up[v][k-1]][k-1]

val[v][k] = val[v][k-1] + val[up[v][k-1]][k-1]





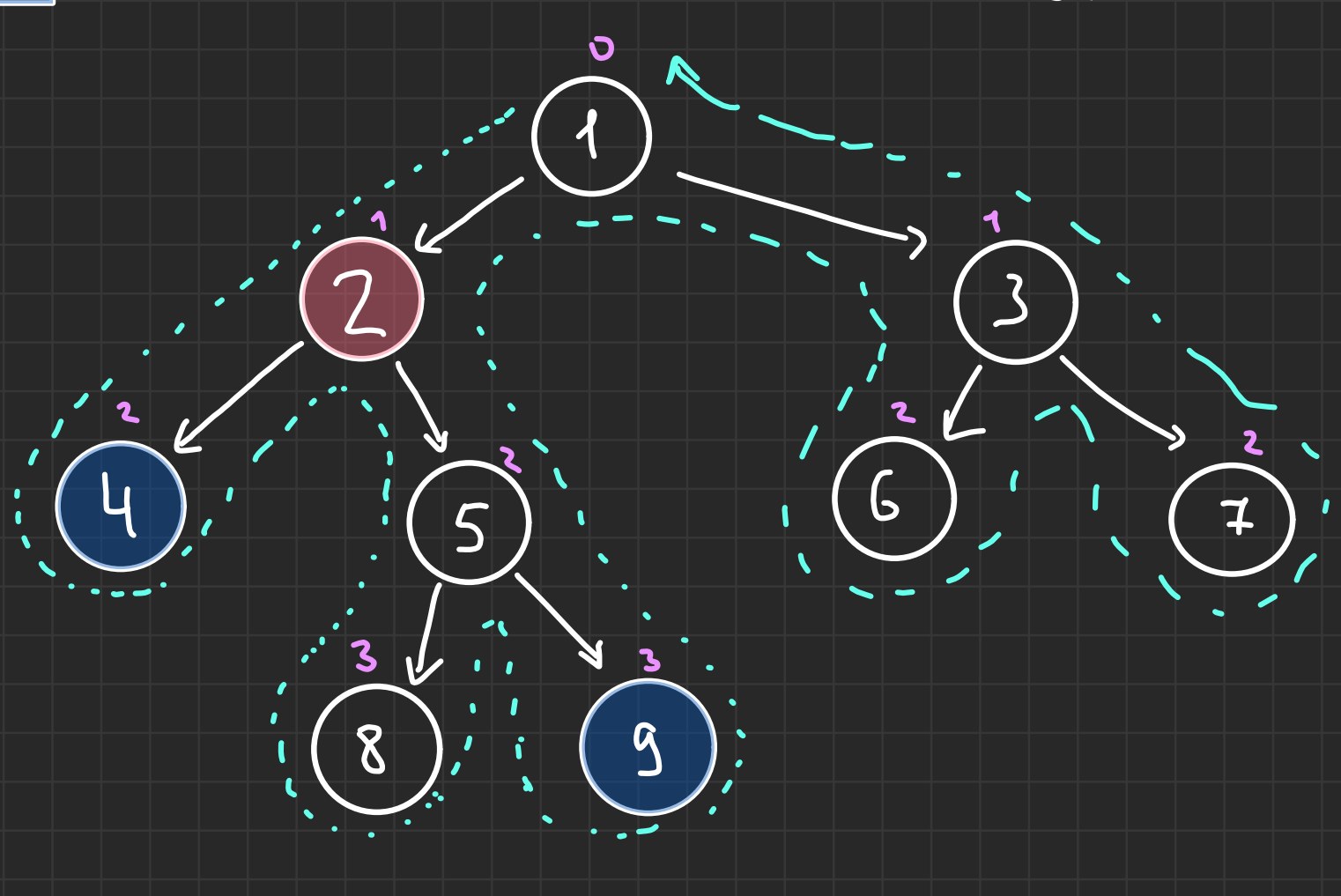
Нужно сделать два запуска

Код для минимума:

```
int lca_min(a, b)
      if pred(a, b):
             return INF
      res = INF

      for i = logN-1; i ≥ 0; --i:
           if !pred(up[a][i], b):
                  res = min(res, val[a][i])
                   a = up[a][i]
           return min(res, val[a][0])
```

4 Сведение LCA к RMQ



1 2 4 2 5 8 5 9 5 2 1 3 6 3 7 3 1
0 1 2 1 2 3 2 3 2 1 0 1 2 1 2 1 0

Между двумя вершинами ровно 1 путь
Это путь по всему дереву

LCA между A и B это min на [tin(A); tin(B)]
если tin(A) ≤ tin(B)

```
order = vec{};

void dfs(v, p, d)
      tin[v] = order.size()
      order.push(v)
       for u in g[v]:
            if (u ≠ p):
              dfs(u, v, d+1)

   int lca(a, b)
       l = tin[a]
       r = tin[b]
       if l > r:
            swap(l, r)

       индекс минимума на [l; r]
```

Корнячка:

Память: O(N)

Время на построение: O(N)

Время на запрос: $O(\sqrt{N})$

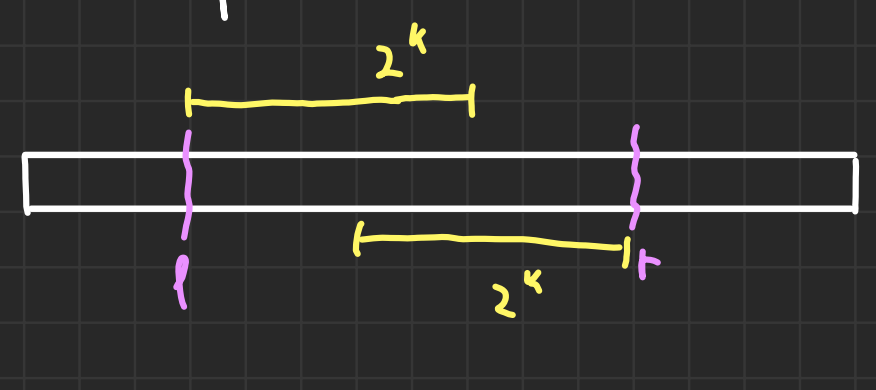
Дерево отрезков:

Память: O(N)

Время на построение: O(N)

Время на запрос: O(log N)

5 Sparse Table (Разреженные Таблица)



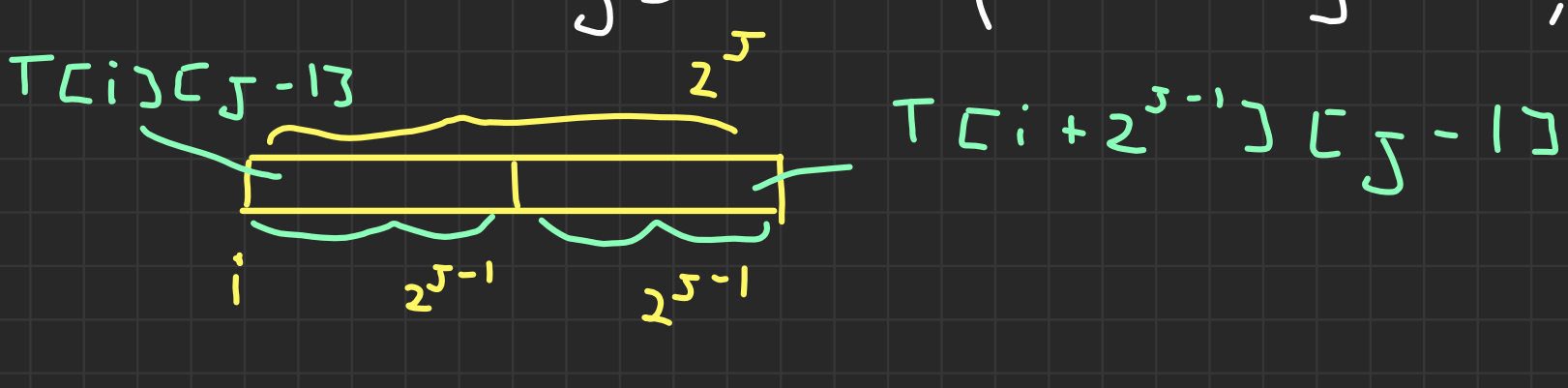
min/max
$a \circ a = a$

$2^k < r - l + 1$
$2^{k+1} > r - l + 1$
$\Rightarrow k = \log(r - l + 1)$

$min[l, r] = min(min[l; l+2^k], min[r-2^k; r])$

$T[pos][k] = min[pos][pos+2^k]$

```
void create(a)
  lg[1] = 0
  for (i=2; i<N; ++i) lg[i] = lg[i/2] + 1
  for (i=0; i<N; ++i) T[i][0] = a[i]
   for (j = 1; j < logN; ++j)
      for (i = 0; i + 2^j < N; ++ i)
           T[i][j] = min( T[i][j-1], T[i + 2^(j-1)][j-1])
```



```
int get_min(l, r)  [l; r]
    j = lg[r - l + 1]
    return min( T[l][j], T[r - 2^j][j])
                  [l; l + 2^j]      [r - 2^j; r]
```