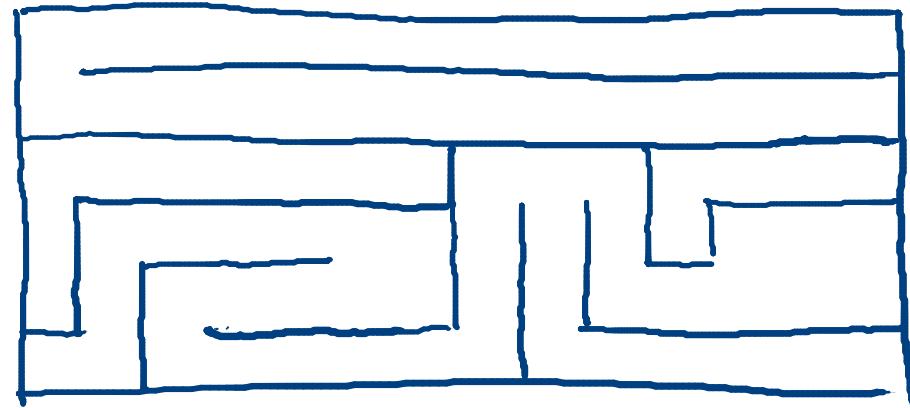


# Непересекающиеся множества

- MakeSet( $x$ ) : создать однотипное множество  $\{x\}$
- Find( $x$ ) : вернуть ID множества, содержащего  $x$
- Union( $x, y$ ) : объединить множества, содержащие  $x$  и  $y$

□ Auspunkt:



□ Ceb:

KOMP<sub>1</sub>

KOMP<sub>2</sub>

KOMP<sub>3</sub>

KOMP<sub>4</sub>

KOMP<sub>5</sub>

## Простейшие реализации

Дадем считать, что наши  
объекты - это просто  
числа  $1, 2, \dots, n$

ID = MUTHUMANBHIBIN nemetsz

Named: {9,3,2,4,7} {5} {6,1,8}

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

MakeSet(i)

smallest[i]  $\leftarrow i$

Find(i)

return smallest[i]

$O(1)$

Union(i,j)

i-id  $\leftarrow \text{Find}(i)$ , j-id  $\leftarrow \text{Find}(j)$

if i-id = j-id : return

m  $\leftarrow \min(i\text{-id}, j\text{-id})$

for k from 1 to n:

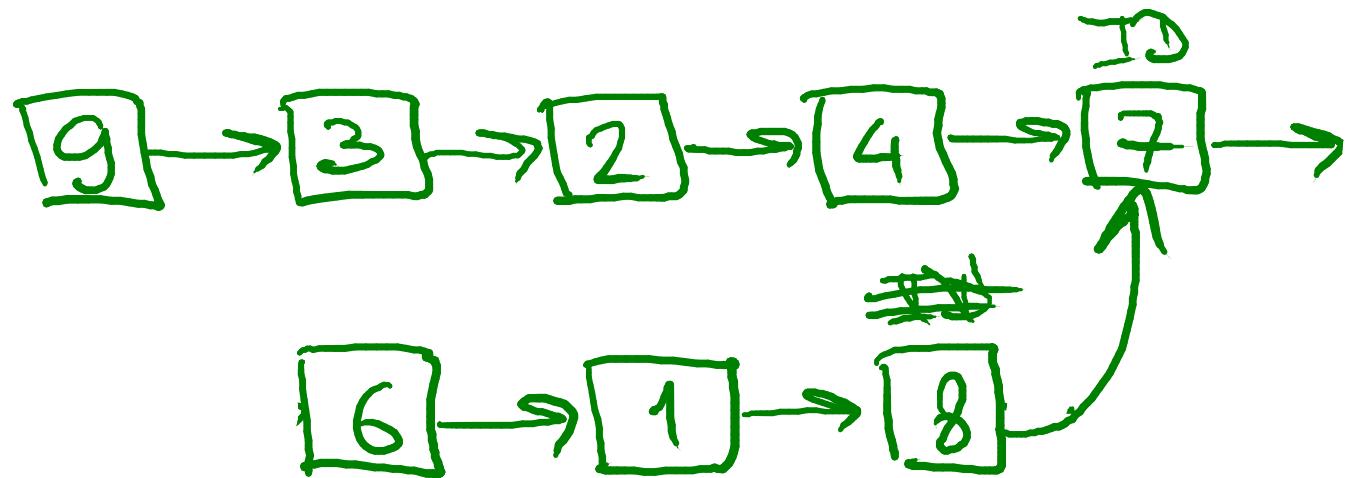
if smallest[k] in {i-id, j-id}:

smallest[k]  $\leftarrow m$

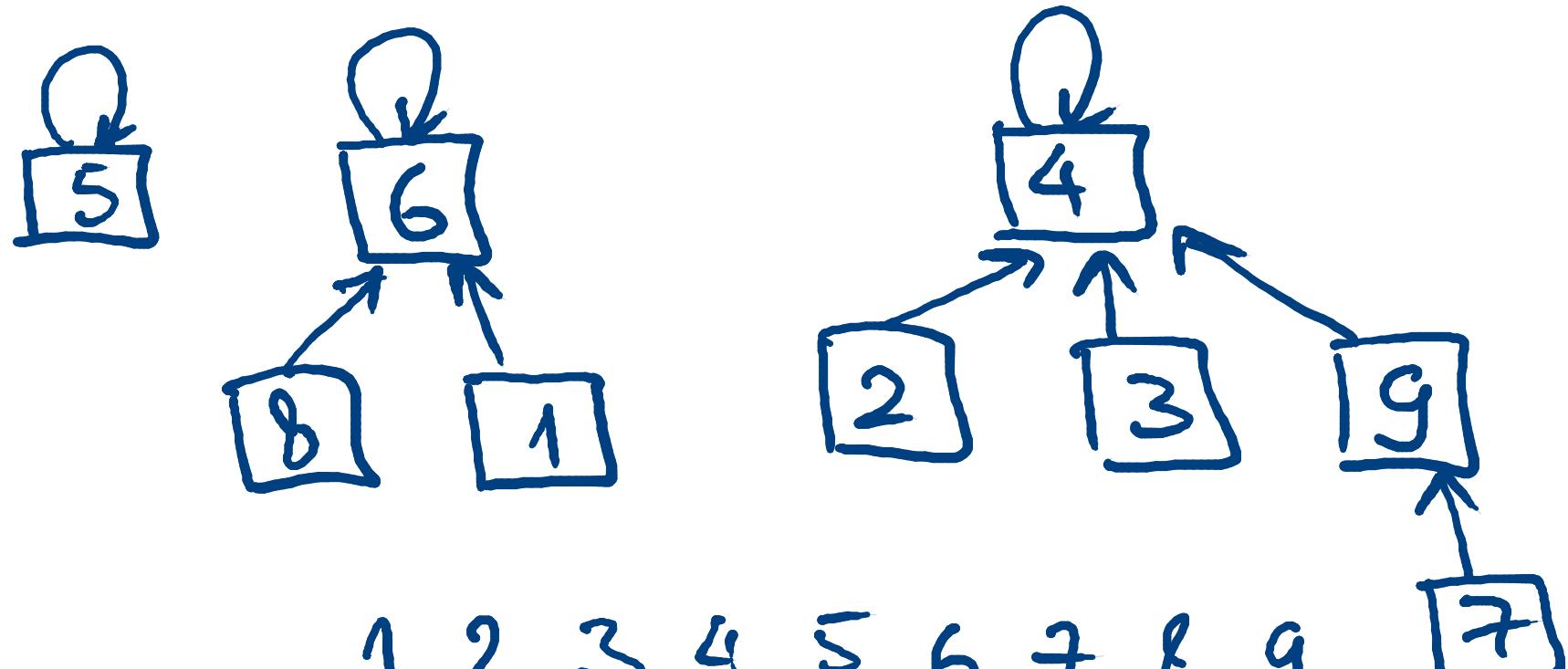
$O(n)$

Способы хранения списков!

Можно использовать хвостовой  
заканчивая способы как ID



# Представление множеств в виде деревьев



1	2	3	4	5	6	7	8	9	10
6	4	4	4	5	6	9	6	4	1

MakeSet(i)

$\text{parent}[i] \leftarrow i$

$O(1)$

Find(i)

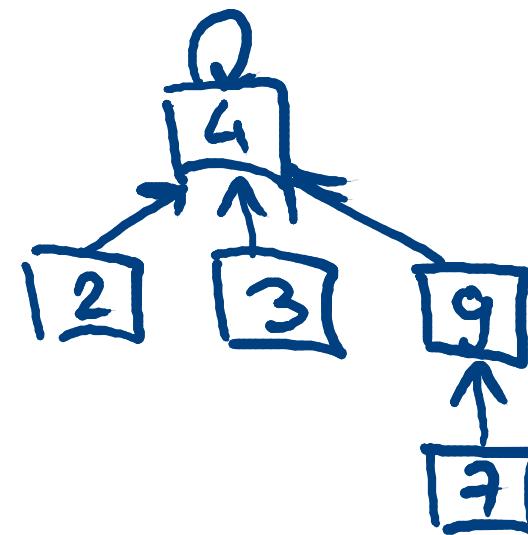
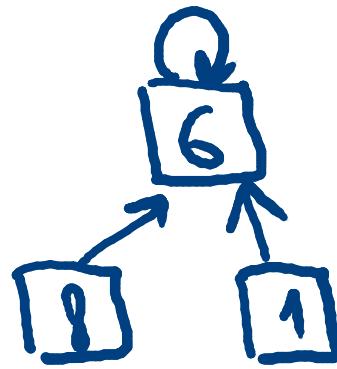
while  $i \neq \text{parent}[i]$ :

$i \leftarrow \text{parent}[i]$

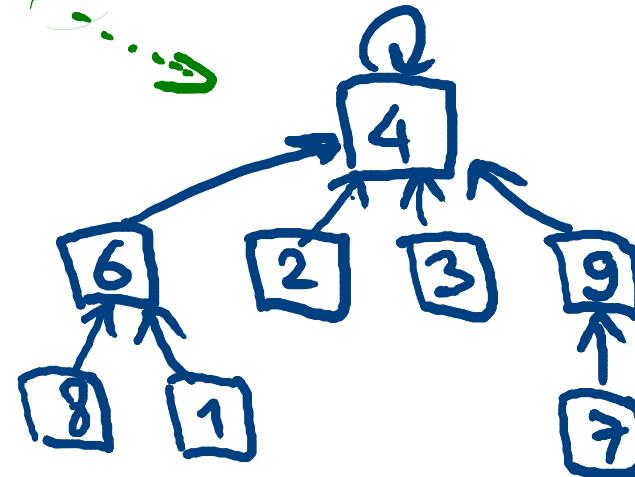
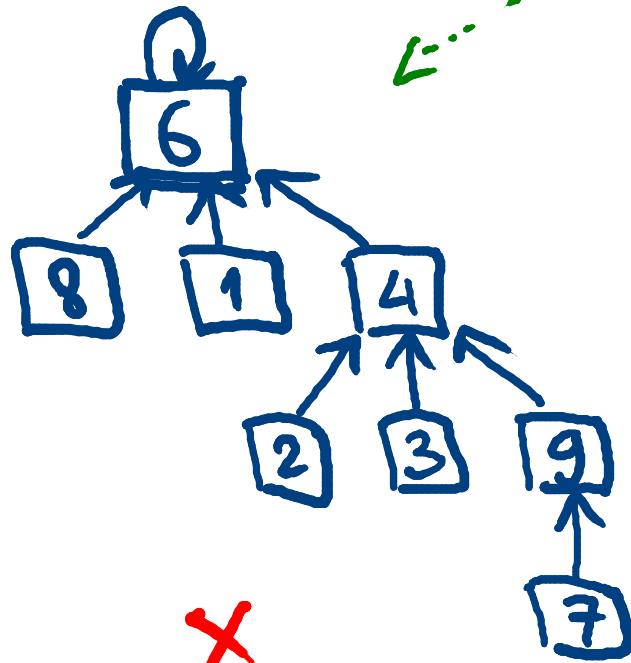
return  $i$

$O(\log n)$

Как обезпечить быструю?



Union(3, 8)



# OSZEDUTIETUE NO PARITY

Makeset(i)

$\text{parent}[i] \leftarrow i$   
 $\text{rank}[i] \leftarrow 0$

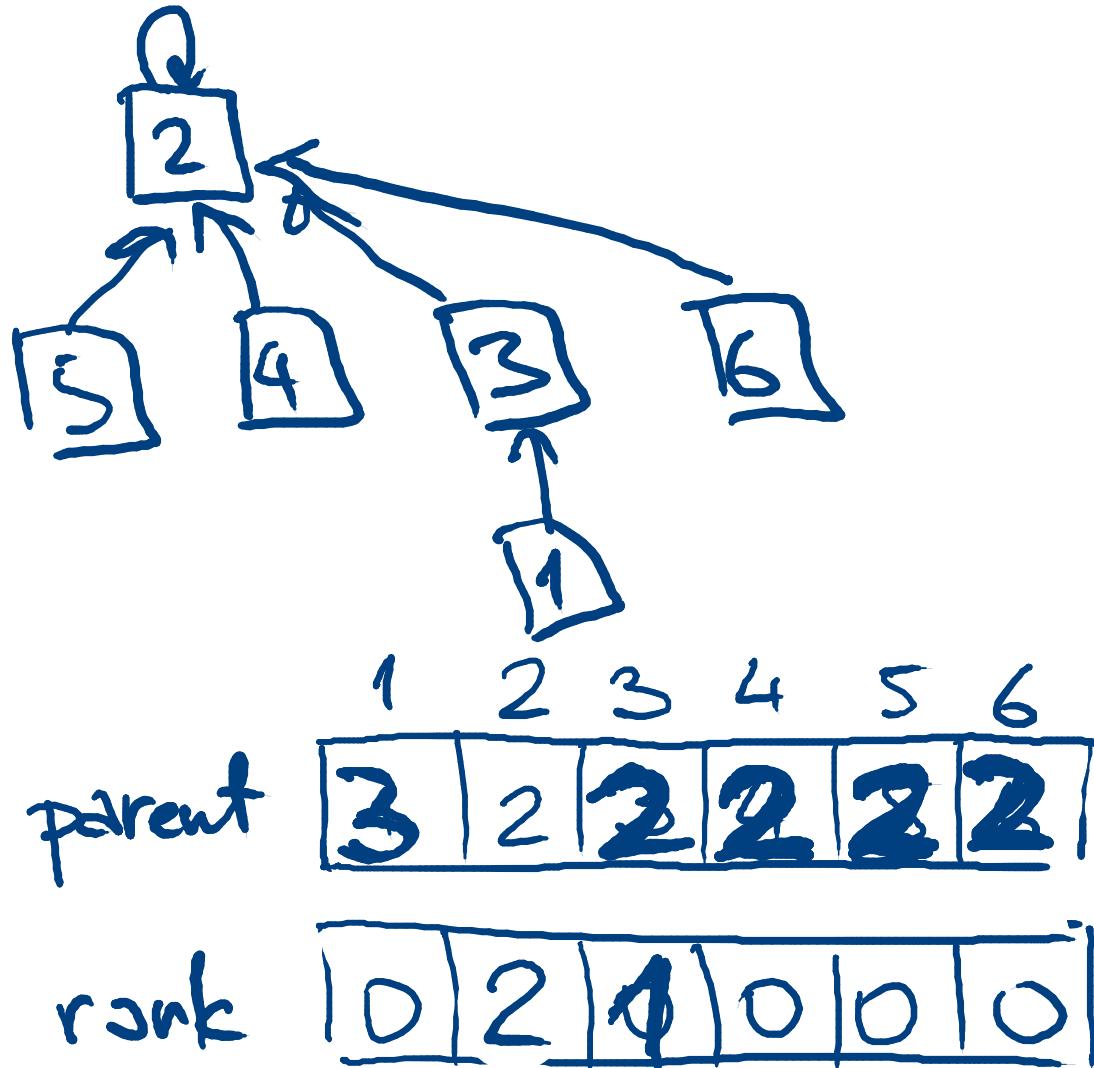
Find(i)

while  $i \neq \text{parent}[i]$ :  
     $i \leftarrow \text{parent}[i]$   
return  $i$

Union(i,j)

$i\text{-id} \leftarrow \text{Find}(i)$ ,  $j\text{-id} \leftarrow \text{Find}(j)$   
if  $i\text{-id} = j\text{-id}$  : return  
if  $\text{rank}[i\text{-id}] > \text{rank}[j\text{-id}]$  :  
     $\text{parent}[j\text{-id}] \leftarrow i\text{-id}$   
else :  
     $\text{parent}[i\text{-id}] \leftarrow j\text{-id}$   
    if  $\text{rank}[i\text{-id}] = \text{rank}[j\text{-id}]$  :  
         $\text{rank}[j\text{-id}] += 1$

# Prüfer



Union(2,4)  
Union(5,2)  
Union(3,1)  
Union(2,3)  
Union(2,6)

Важное свойство:  $\text{rank}[i]$  — высота поддерева с корнем  $i$  в дереве  $\text{rank}$

Лемма: высота любого дерева в лесе не больше  $\log_2 n$ .

Dok. Bo'.

Индукцией по  $k$  покажем, что  
в дереве высоты к корням есть  
 $2^k$  вершин.

База:  $k=0$ , одна вершина:  $2^0=1$

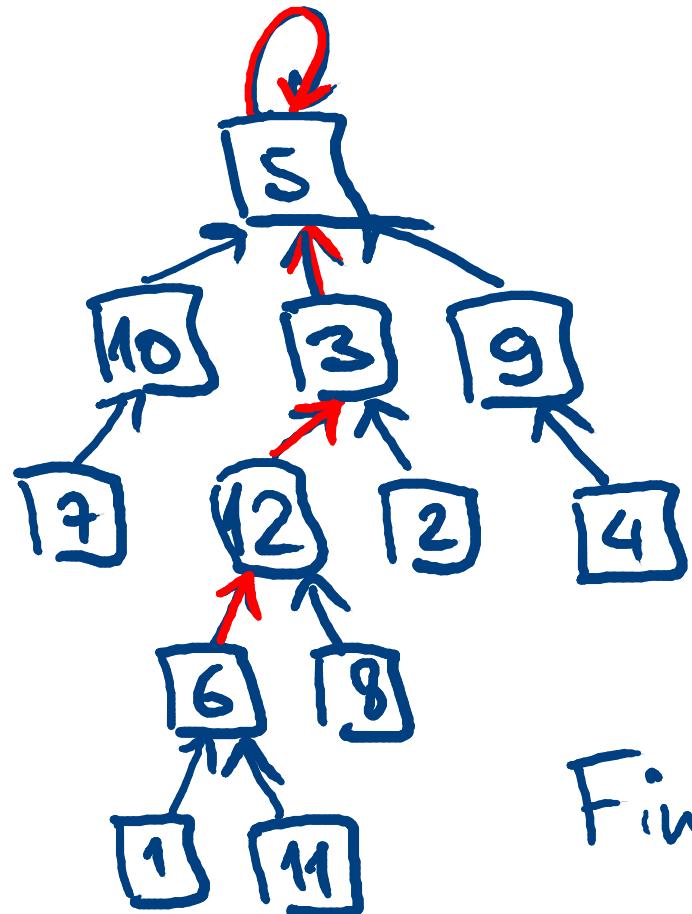
Переход: дерево высоты  $k$  получается  
объединением двух деревьев  
высоты  $k-1$ :

$$\begin{array}{c} k-1 \\ \uparrow \\ \triangle \\ \downarrow \\ \geq 2^{k-1} \end{array} + \begin{array}{c} k-1 \\ \uparrow \\ \triangle \\ \downarrow \\ \geq 2^{k-1} \end{array} = \begin{array}{c} \triangle \\ \quad \quad \quad \triangle \\ \uparrow \quad \downarrow \\ \geq 2^k \end{array}$$

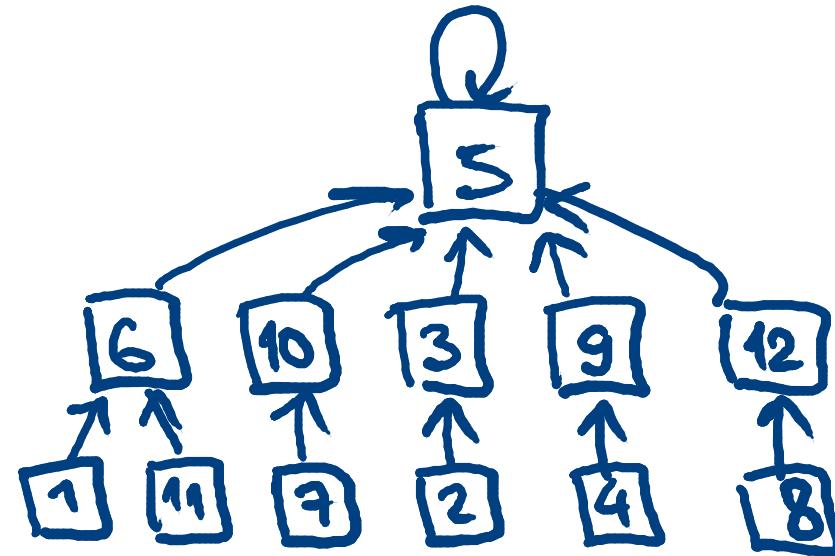
## Фьюжн

Объединение по Path  
гарантирует, что операции  
Union и Find имеют время  
работы  $O(\log n)$ .

# C<sub>h</sub>atue Tree



Find(6)



Find(i)

if  $i \neq \text{parent}[i]$ :

$\text{parent}[i] \leftarrow \text{Find}(\text{parent}[i])$

return  $\text{parent}[i]$

ОТВ:  $\log^* n$  – шерифованный логарифм  $n$  – число раз, которое нужно применить функцию  $\log_2$  к  $n$ , чтобы результат стал  $\leq 1$ :

$$\log^* n = \begin{cases} 0, & \text{если } n \leq 1 \\ 1 + \log^*(\log_2 n), & \text{если } n > 1 \end{cases}$$

<u><math>n</math></u>	<u><math>\log^* n</math></u>
$n = 1$	0
$n = 2$	1
$n = 3, 4$	2
$n = 5, 6, \dots, 16$	3
$n = 17, 18, \dots, 65536$	4
$n = 65537, \dots, 2^{65536}$	5

Лемма: Такъ изначально структура  
датческих пуста. Сделаем и  
вызовов операций, включая  
и  $\leq m$  вызовов MaxSel.  
Суммарное время работы  
будет  $O(m \log^* n)$ .

Другими словами, среднее (амортизированное)  
время работы одной операции -  
 $O(\log^* n)$ . Это константа при  
значениях  $n$ , возникающих на  
практике!

# Заключение

- Представляем множества как деревья
- Представитель — корень
- Обединение по рангу
- Сжатие путей
- Практически константное амортизированное время работы операций:  
 $O(\log^* n)$ .

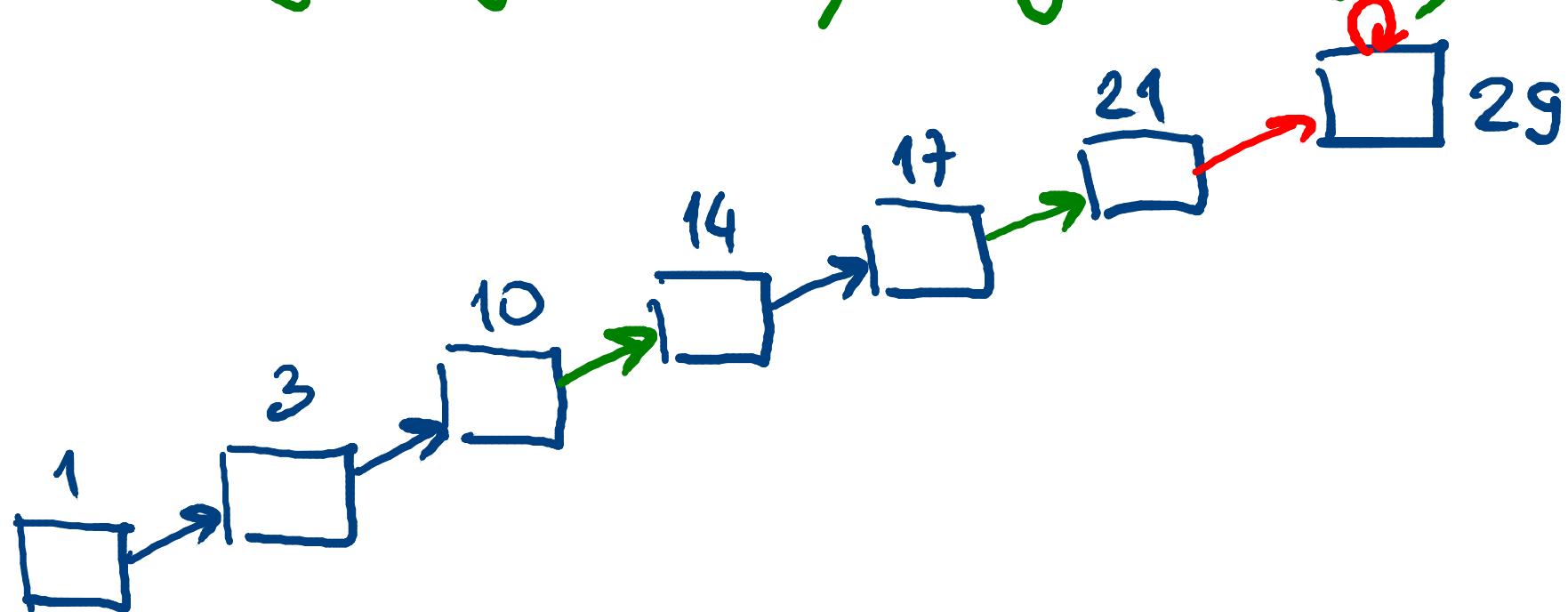
## Аналіз Зразка Роботи

- С звичайшої структурі путь rank[i] уже не обов'язково буде рівен висоті
- Но rank[i] буде по-прежньому менше або рівен висоті
- Более того, для корневих вершин  $\text{rank}_k$  по-прежньому буде верним то, що в них піддерево хотія бути  $2^k$  вершин

## Важные свойства

1. Есть не более  $\frac{n}{2^k}$  вершин  
Родителей
2. Для любой вершины  $i$   
 $\text{rank}[i] < \text{rank}[\text{parent}[i]]$
3. Одни из внутренних вершин -  
единственные внутренние вершины

$$\begin{aligned}
 T(\text{БСЕ зъброби Find}) &= \#(i \rightarrow j) = \\
 &= \#(i \rightarrow j : j - \text{корен}) + \\
 &+ \#(i \rightarrow j : \log^*(\text{rank}[i]) < \log^*(\text{rank}[j])) + \\
 &+ \#(i \rightarrow j : \log^*(\text{rank}[i]) = \log^*(\text{rank}[j]))
 \end{aligned}$$



Утв:  $\#(i \rightarrow j : j - \text{корень}) \leq O(n)$ .

Док-бо: не более  $n$  вызовов Find.  $\square$

Утв:  $\#(i \rightarrow j : \log^*(\text{rank}[i]) < \log^*(\text{rank}[j])) \leq$   
 $\leq O(n \log^* n)$ .

Док-бо: не более  $\log^* n$  разных  
значений  $\log^*(\text{rank}[\cdot])$ .  $\square$

Утв:  $\#(i \rightarrow j : \log^*(\text{rank}[i]) = \log^*(\text{rank}[j])) \leq$   
 $\leq O(n \log^* n).$

Док-вд: Пусть  $\text{rank}[i] \in \{k+1, \dots, 2^k\}$ .

Число таких вершин  $\leq$

$$\frac{n}{2^{k+1}} + \frac{n}{2^{k+2}} + \dots \leq \frac{n}{2^k}.$$

После вызова  $\text{Find}(i)$  вершина  $i$  связывается  
вершиной со старого большего рангом.

После не более  $2^k$  вызовов  $\text{Find}(i)$  родитель  
 $\text{find}$  может быть равен из другого интервала.

Каждая вершина с рангом из  $\{k+1, \dots, 2^k\}$   
даёт вклад в сумму  $\leq 2^k$ , всего таких  
вершин  $\leq \frac{n}{2^k}$ . Всего интервалов  $\leq \log^* n$ .  $\square$