3/23/21

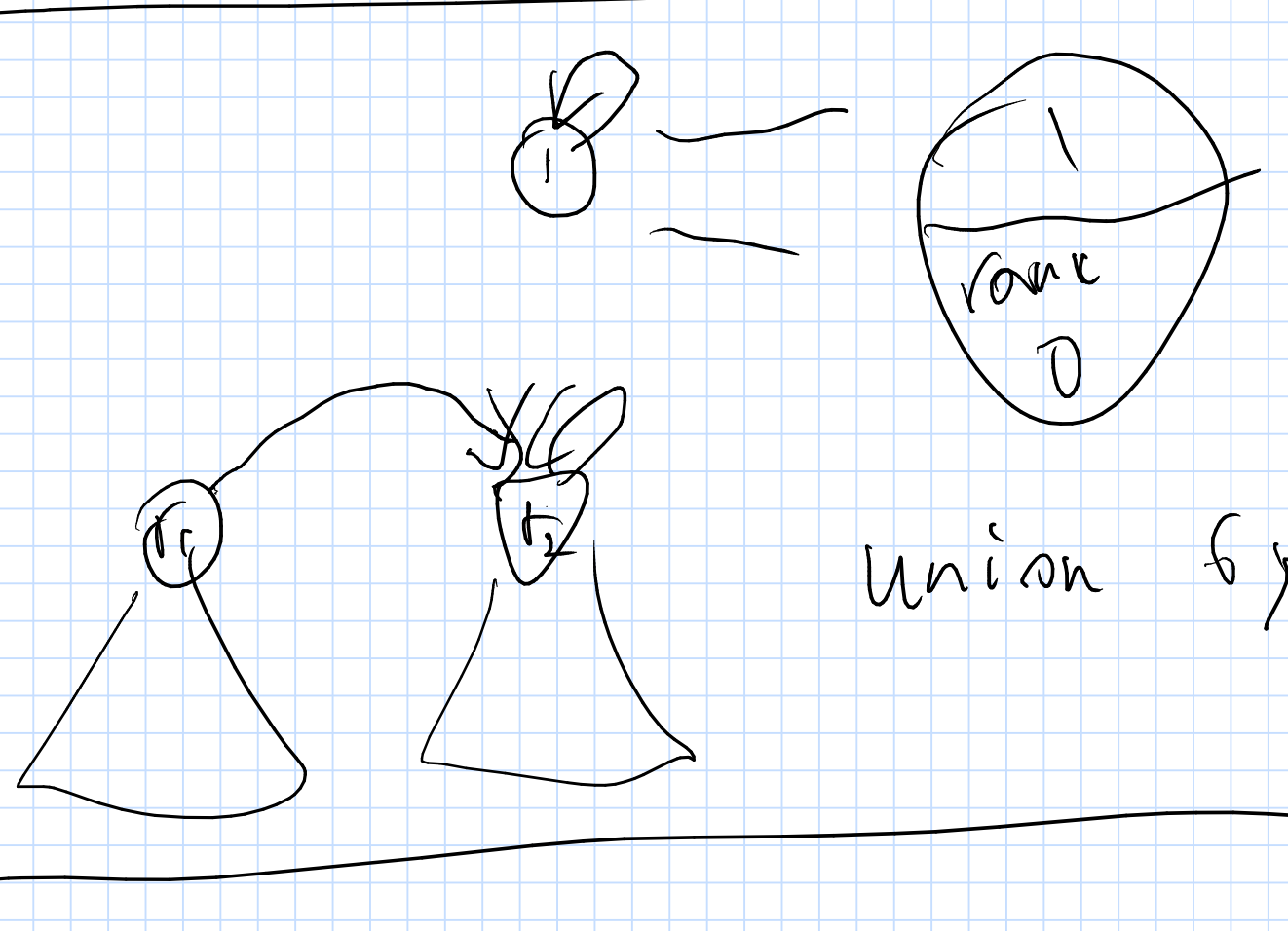
Union-find analysis

Disjoint sets

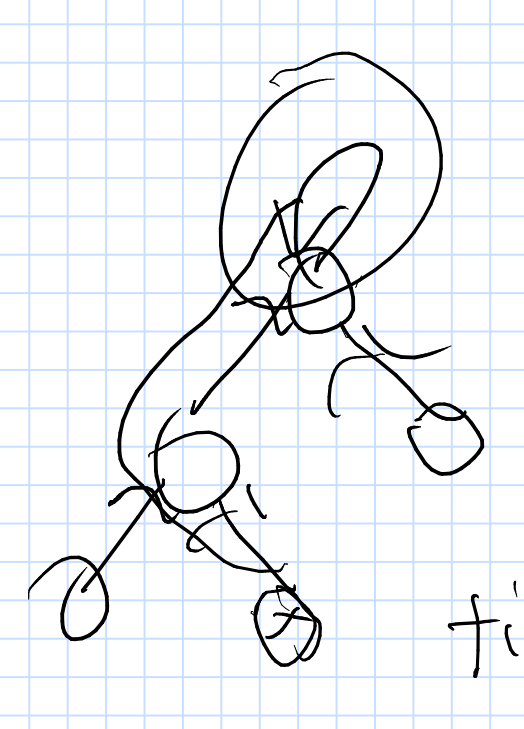
① ② ..... ⓝ

(1,4) —union— (3,8)   find(x)

n make set
n-1 union operation
m find

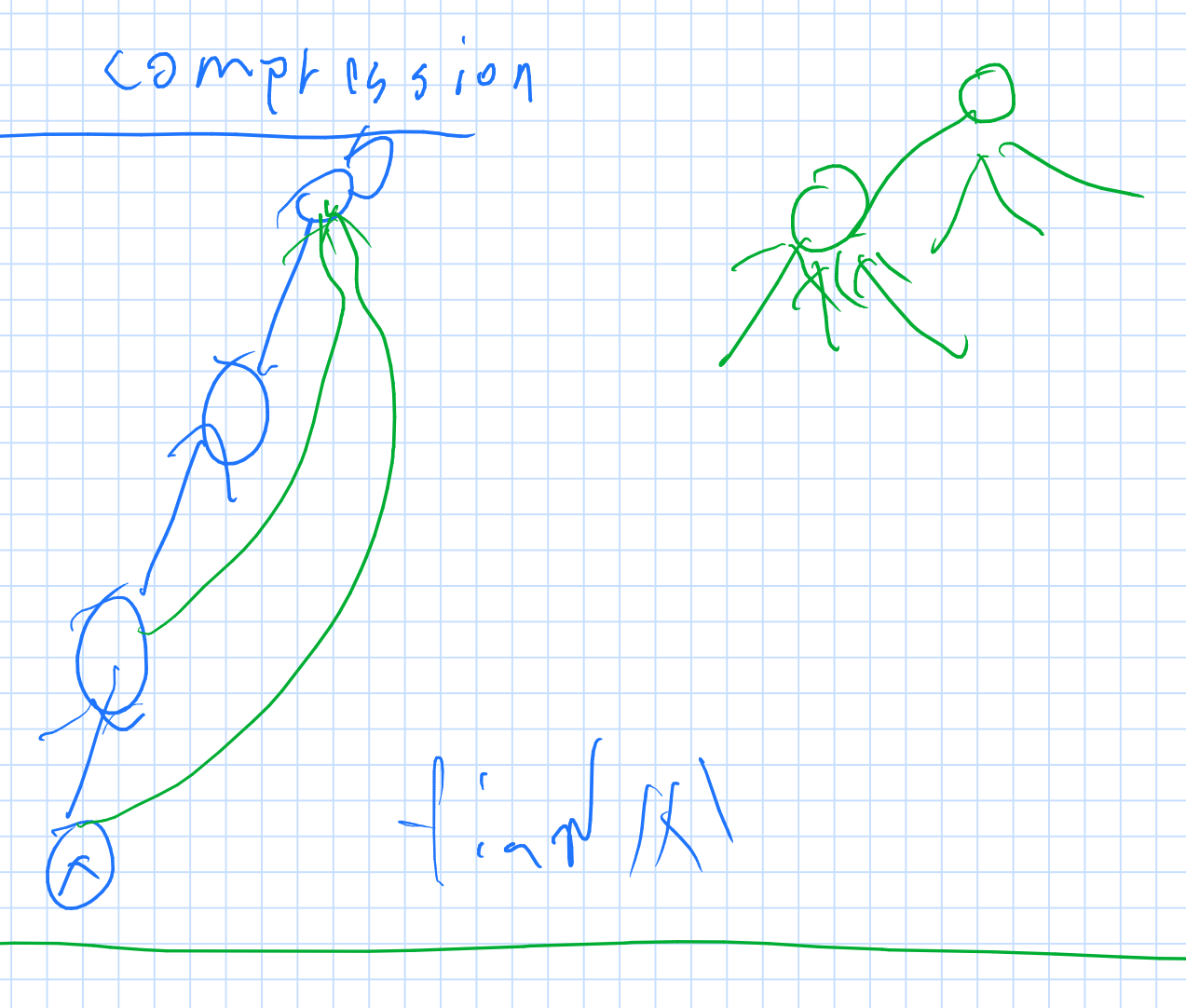


union by rank



find(1)   expensive

2. Path compression

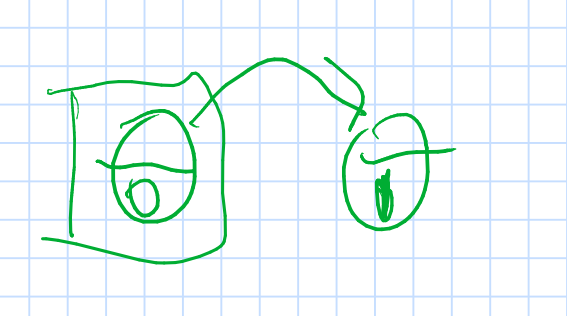


find(x)

Theorem
Union Find takes $O((n+m)\log^* n)$ time, for a sequence of m operations over n elements.

$O(m+n\log^* n)$

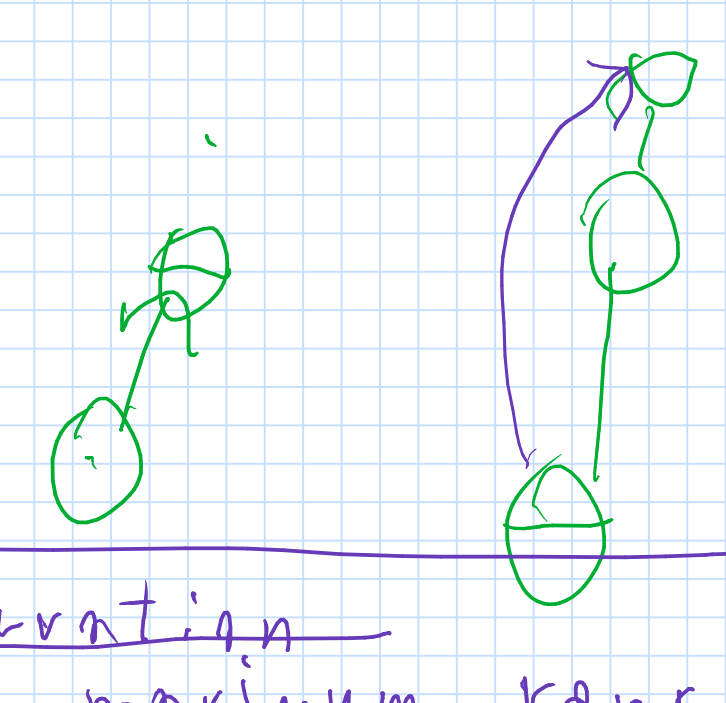
Lemma
- Once a node stop being a root, its rank is fixed.
- The number of elements of rank i through out the alg execution is at most $n/2^i$.



$\frac{n}{2}$ rank 1

$\frac{n}{i}$ rank i

Observation
Along a path in the reverse tree the ranks are strictly increasing.



Observation
The maximum rank in the UF data structure is

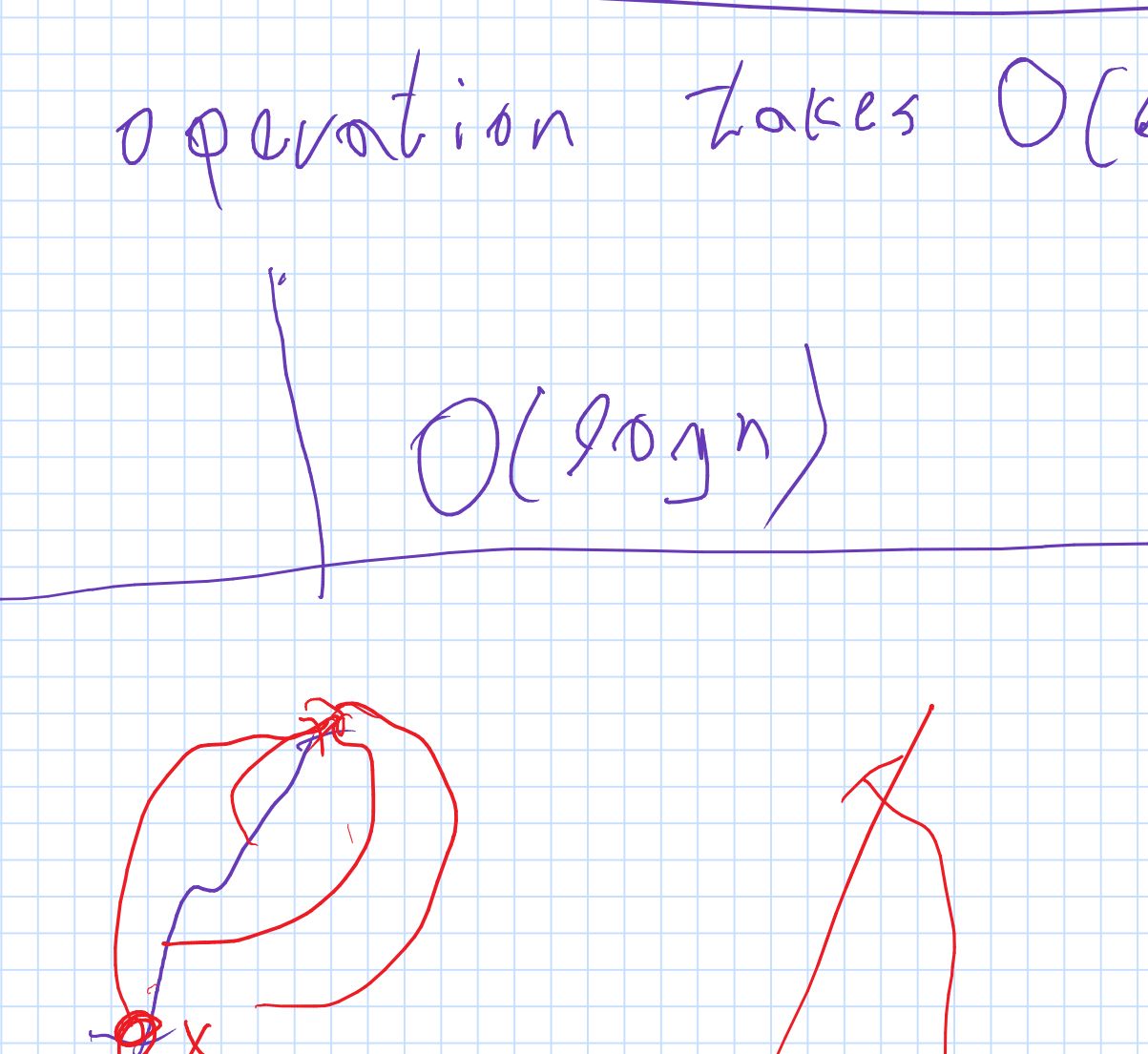
$\le n$	0
$n/2$	1
$n/4$	2
$n/2^i$	⋮

$\Rightarrow$ max rank $\log n$.

Lemma Find operation takes $O(\log n)$ time.

$O(\log n)$

Progress



$Tower(i) = 2^{2^{\cdot^{\cdot^{2}}}}$ i times

$= \begin{cases} 1 & i=0 \\ 2^{Tower(i-1)} & i>0 \end{cases}$

$\log^* tower(i) = i$.

$Block(i) = B(i)$

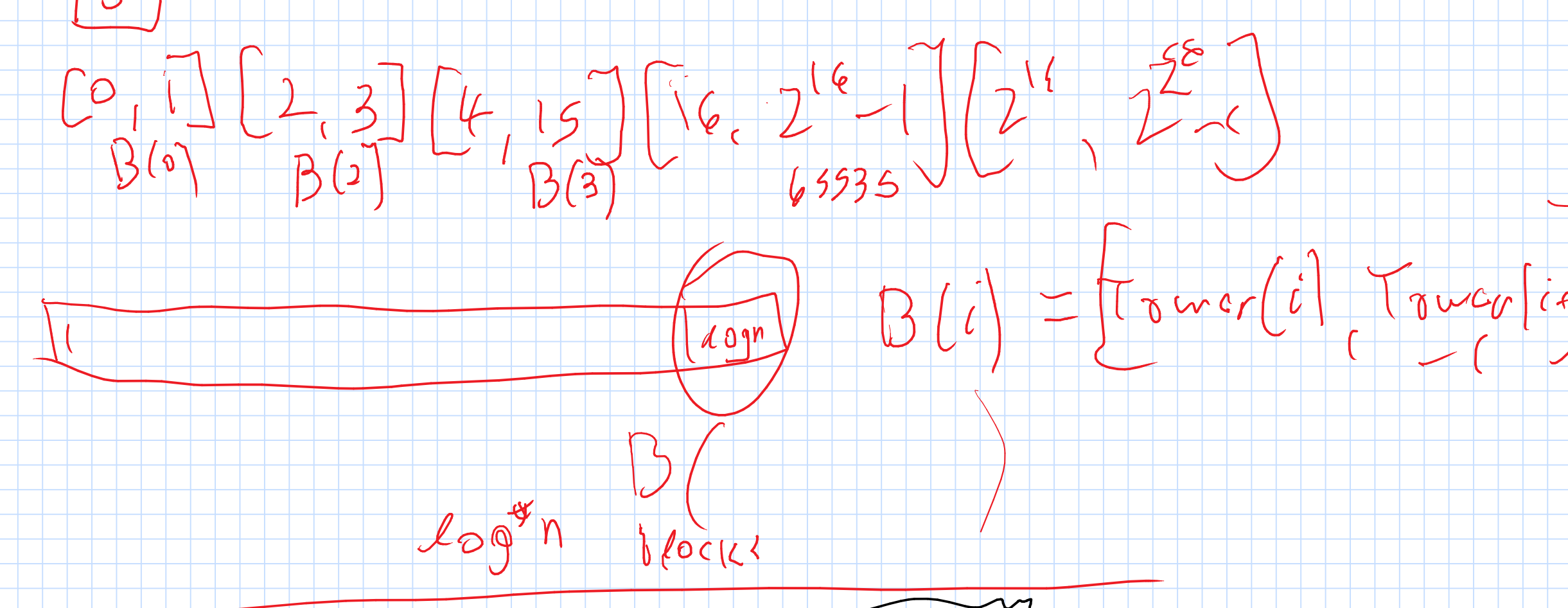
$= [Tower(i), Tower(i+1)-1]$

$K = Tower(i)$

$[K, 2^K-1]$

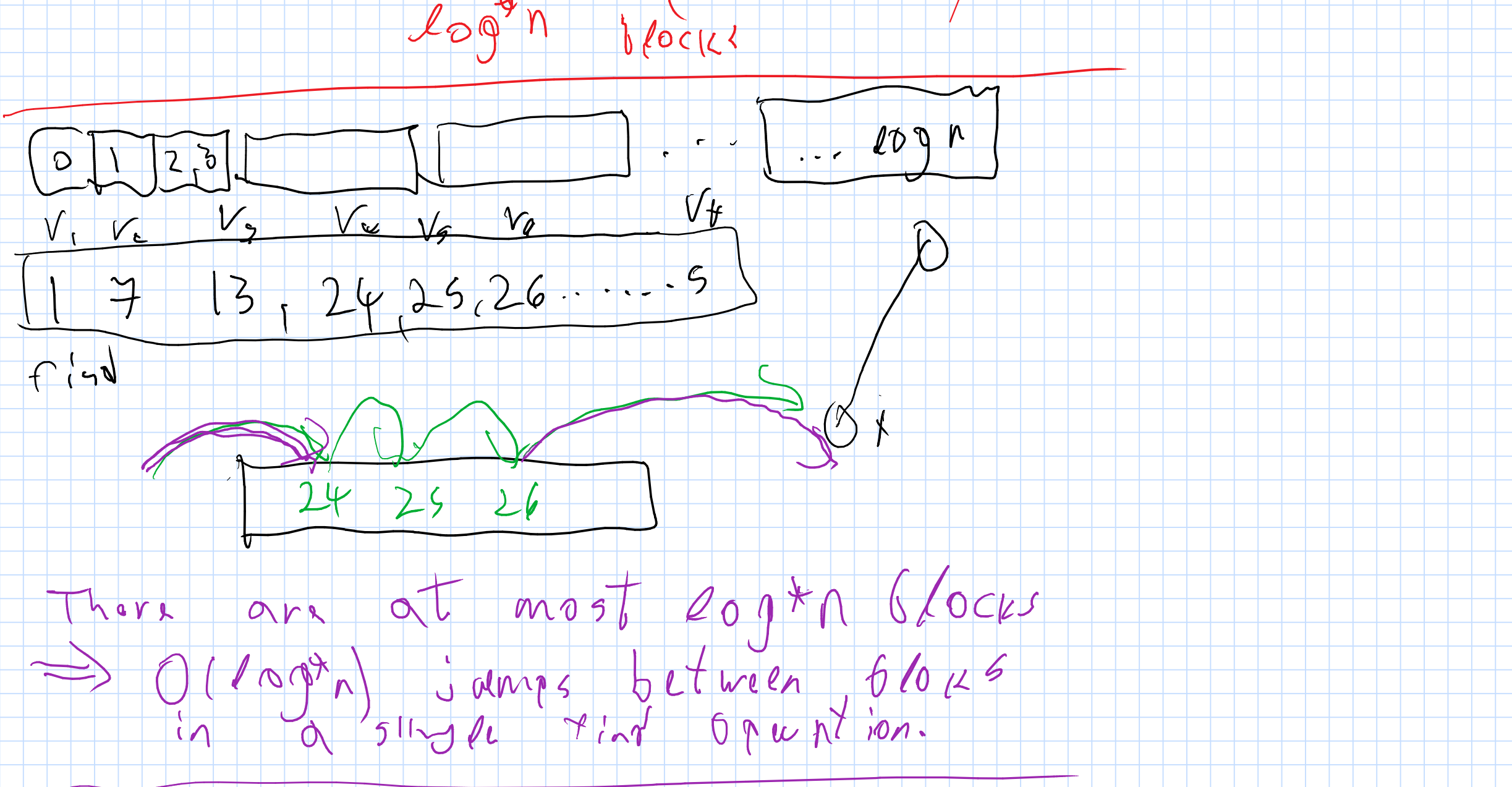
$\log n$

$[0,1]$ $[2,3]$ $[4,15]$ $[16, 2^{16}-1]$ $[2^{16}, 2^{2^{16}}-1]$
B(0) B(2) B(3) 65535

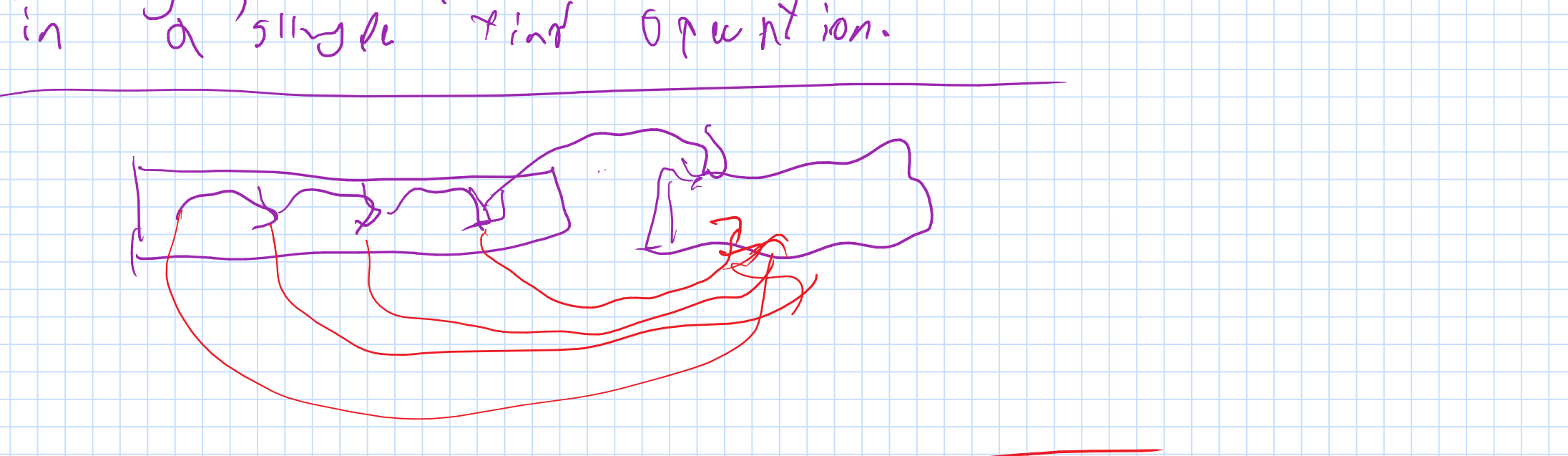


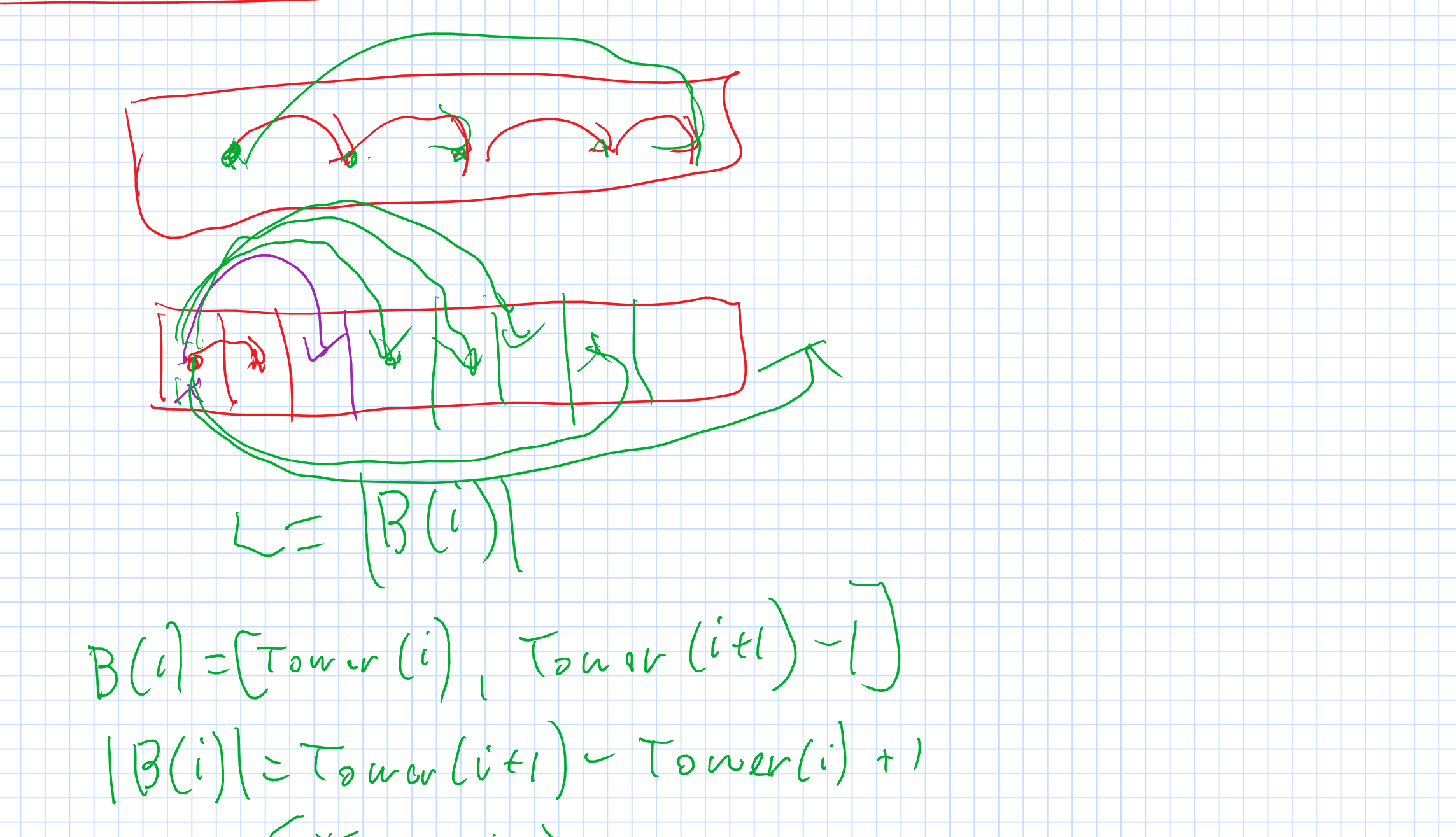
$B(i) = [Tower(i), Tower(i+1)-1]$

$\log^* n$ blocks



There are at most $\log^* n$ blocks
$\Rightarrow O(\log^* n)$ jumps between blocks in a single find operation.





$L = |B(i)|$

$B(i) = [Tower(i), Tower(i+1)-1]$

$|B(i)| = Tower(i+1) - Tower(i) + 1$
$\le Tower(i+1)$

$K = Tower(i)$

$B(i) = [K, 2^K - 1]$

Q: How many elements have rank k?

$\frac{n}{2^k} = \frac{n}{Tower(i+1)}$

Total number of elements that got rank $\ge k$ through out the alg execution

$\sum_{i=k}^{\infty} \frac{n}{2^i} = \frac{n}{2^k}\cdot 2$

Total number of internal jumps during the alg execution in the ith block.

(# elements that has at some point rank in the ith block) $\le 2\cdot\frac{n}{2^k}$

(# many times an element with rank in ith block can have an internal jump in this block) $\le |B(i)| \le Tower(i+1) = 2^{k+1}$

# of internal jumps through the ith block

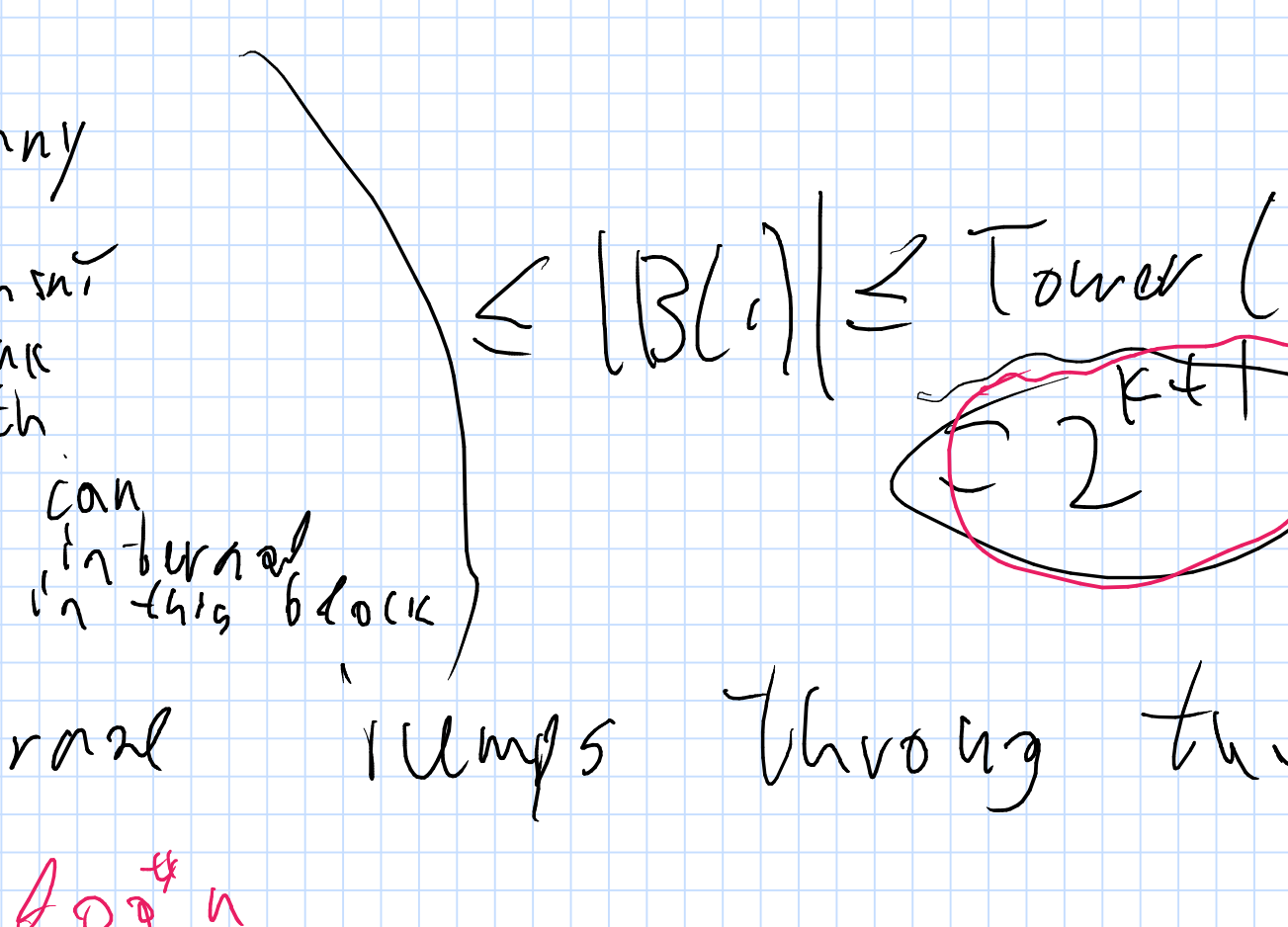
$4n$ $\log^* n$ blocks

overall: the alg performs $O((n+m)\log^* n)$

$O(n\log^* n)$

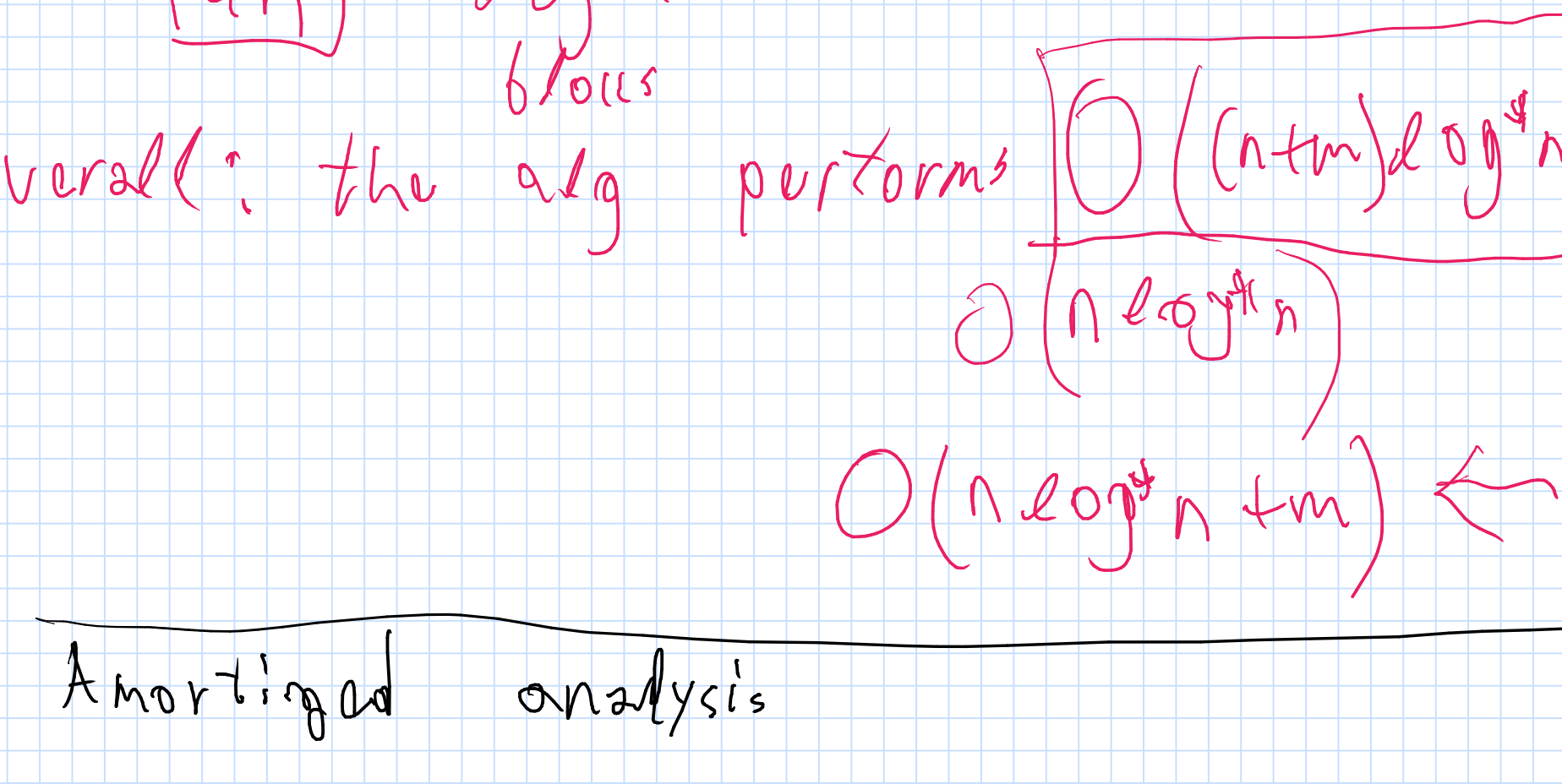
$O(n\log^* n + m)$ ←

Amortized analysis



$\log n$

$O\left(\frac{n\log^* n+m}{n+m}\right)$: average cost of operation

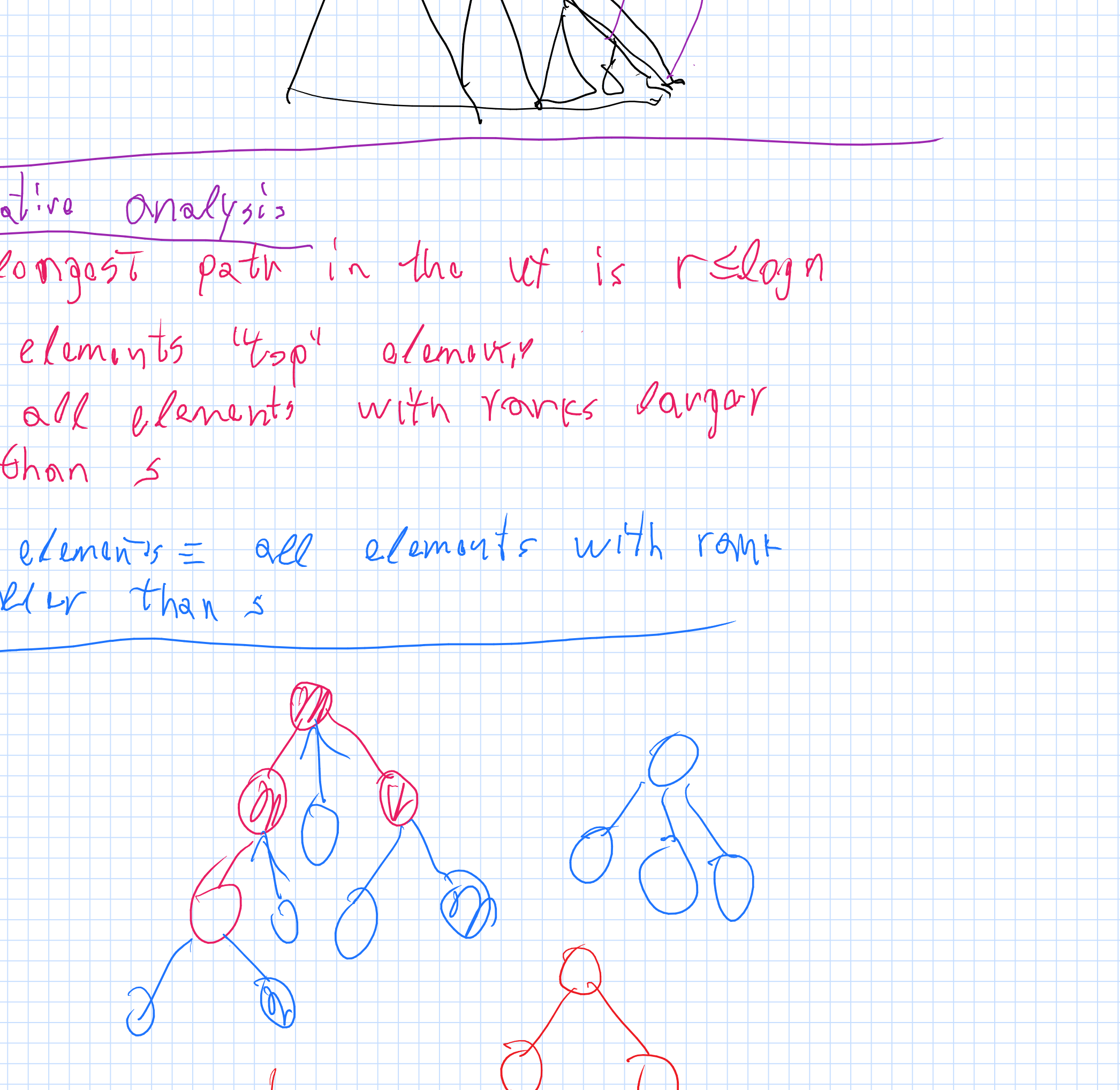


Alternative analysis

The longest path in the uf is $r \le \log n$

red elements "top" elements
$\equiv$ all elements with ranks larger than s

blue elements $\equiv$ all elements with rank smaller than s



$T(m,n,r) \equiv$ total work in m find operations where max length of path is r

$T(m,n,r) \le (m+n)r$ $O(n)$

$T(m,n,r) \le T(m, n_{red}) + T(m, n_{blue}) + m$

$n_{red} = |$red elements$|$ = all element of rank larger than $\log\log n$

$\frac{n}{2^{\log\log n}} = \frac{n}{\log n}$

$T(n,m,r) \le O(n+m) + T(\frac{n}{2}, m, r)$

$O((m+n)\log^* n)$. Hand wavy.