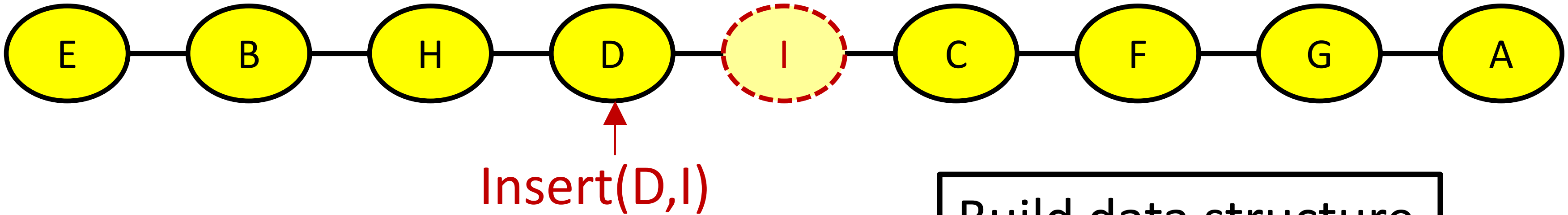
List Order Maintenance

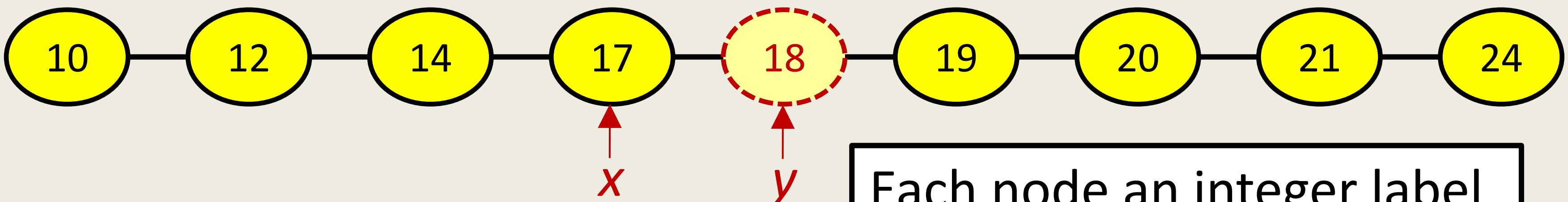


Build data structure

Insert(*x*,*y*) Insert *y* after *x*

Order(*x*,*y*) Returns if *x* is to the left of *y*

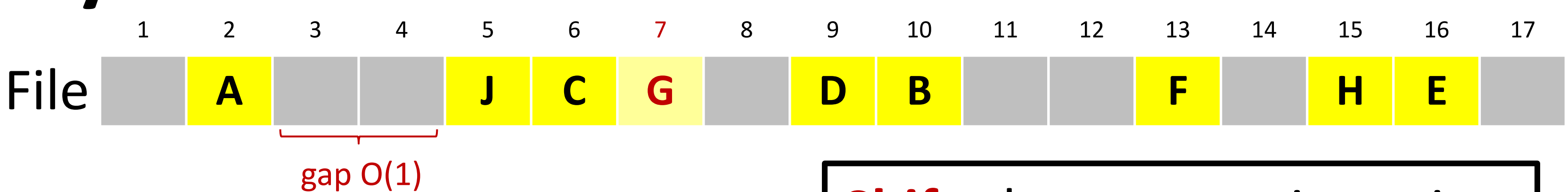
Monotonic List Labeling



Each node an integer label
Relabel nodes on insertion

Insert(*x*,*y*) Insert *y* after *x*

Density Maintenance



Insert(*i*,*x*) Insert *x* at postion *i*

Shift elements on insertion

List Order Maintenance

[P. Dietz, D. Sleator, *Two algorithms for maintaining order in a list*, ACM Conference on Theory of Computing, 365-372, 1987]

[A. Tsakalidis, *Maintaining Order in a Generalized Linked List*. Acta Informatica 21: 101-112, 1984]

Query and Insert O(1)

Monotonic List Labeling

[P. Dietz, *Maintaining Order in a Linked List*, ACM Conference on Theory of Computing, 122-127, 1982]

[P. Dietz, J. Seiferas, J. Zhang: *A Tight Lower Bound for On-line Monotonic List Labeling*. Scandinavian Workshop on Algorithm Theory, 131-142, 1994]

Max label $O(n^k)$, $k>1+\varepsilon$ $\Theta(\log n)$ relabelings

[D. Willard, *Maintaining Dense Sequential Files in a Dynamic Environment*, ACM Conference on Theory of Computing, 114-121, 1982]

[P. Dietz, J. Zhang: *Lower Bounds for Monotonic List Labeling*. Scandinavian Workshop on Algorithm Theory, 173-180, 1990]

Max label $O(n)$ $\Theta(\log^2 n)$ relabelings

Applications

[G. Brodal, R. Fagerberg, R. Jacob, *Cache-Oblivious Search Trees via Binary Trees of Small Height,* ACM-SIAM Symposium on Discrete Algorithms, pages 39-48, 2002]

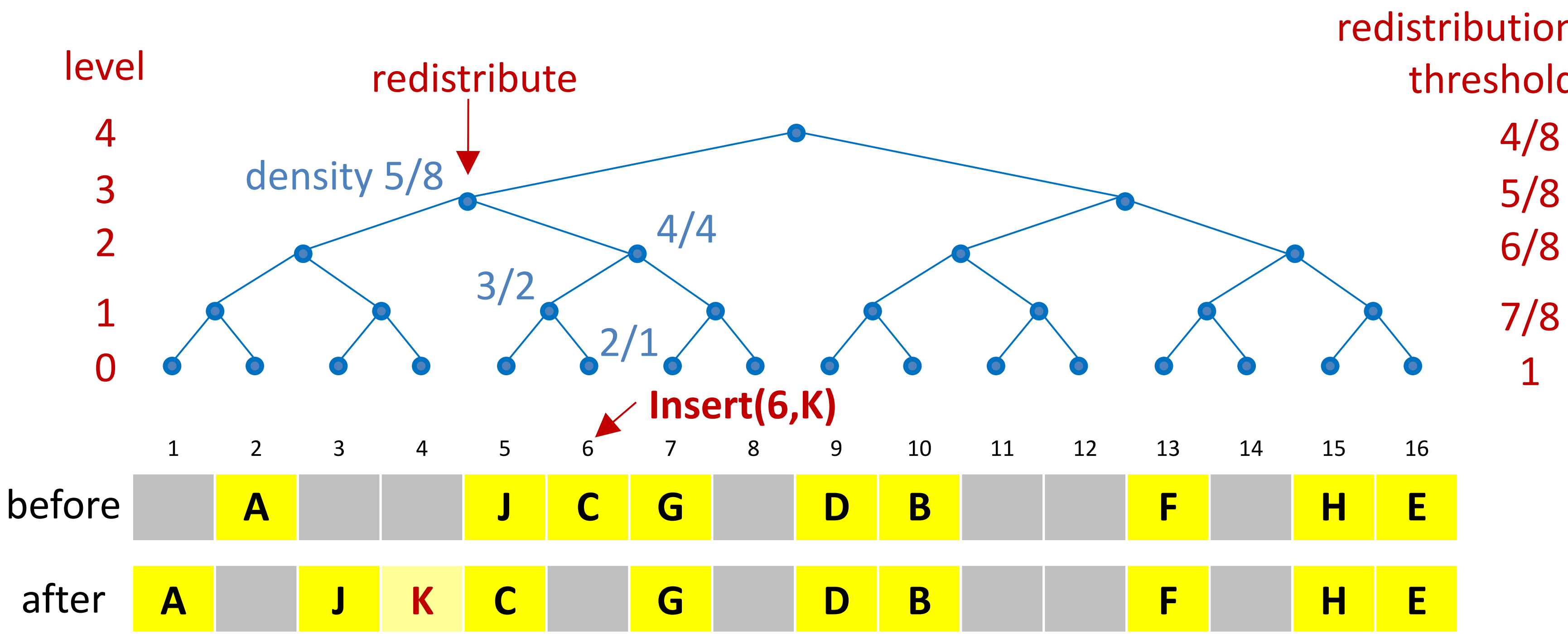
[J. Driscoll, N. Sarnak, D. Sleator, R. Tarjan, *Making Data Structures Persistent*, Journal of Computer and System Sciences, 38(1), 86-124, 1989]

Amortized O(log² *n*) Density Maintenance

- Threshold $\tau = 1/(2\log n)$
- Level *i* node overflows if density $> 1 - i \cdot \tau$
- **Insert** redistribute lowest non-overflowing ancestor

$\Rightarrow$ a child requires τ fraction insertions before next overflow

$\Rightarrow$ amortized insertion cost = #levels $\cdot\ 1 / \tau = O(\log^2 n)$

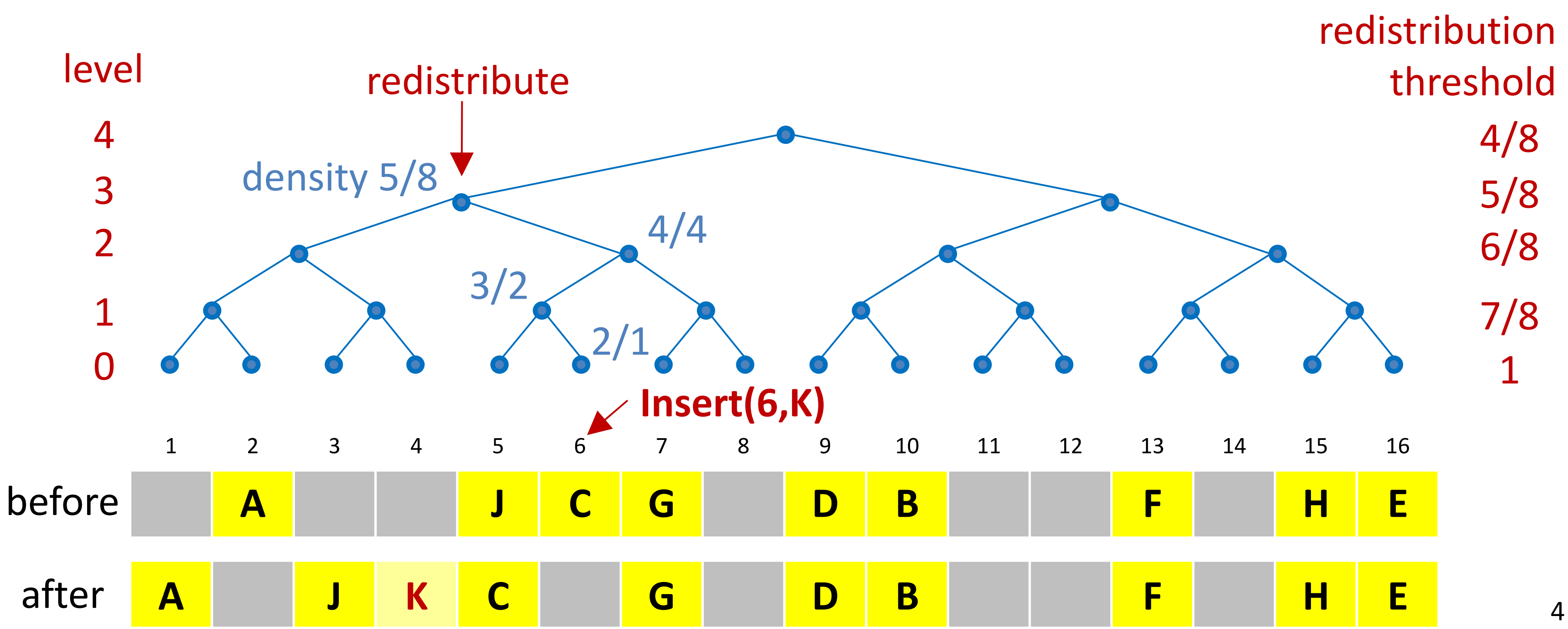


Amortized O($\log^2 n$) Density Maintenance

- Threshold $\tau = 1/(2\log n)$
- Level i node overflows if density $> 1 - i \cdot \tau$
- **Insert** redistribute lowest non-overflowing ancestor

  $\Rightarrow$ a child requires τ fraction insertions before next overflow

  $\Rightarrow$ amortized insertion cost = #levels $\cdot$ $1 / \tau = O(\log^2 n)$



Amortized $O(\log^2 n)$ Density Maintenance

- Threshold $\tau = 1/(2\log n)$
- Level i node overflows if density $> 1 - i \cdot \tau$
- Insert ⇒ redistribute lowest non-overflowing ancestor

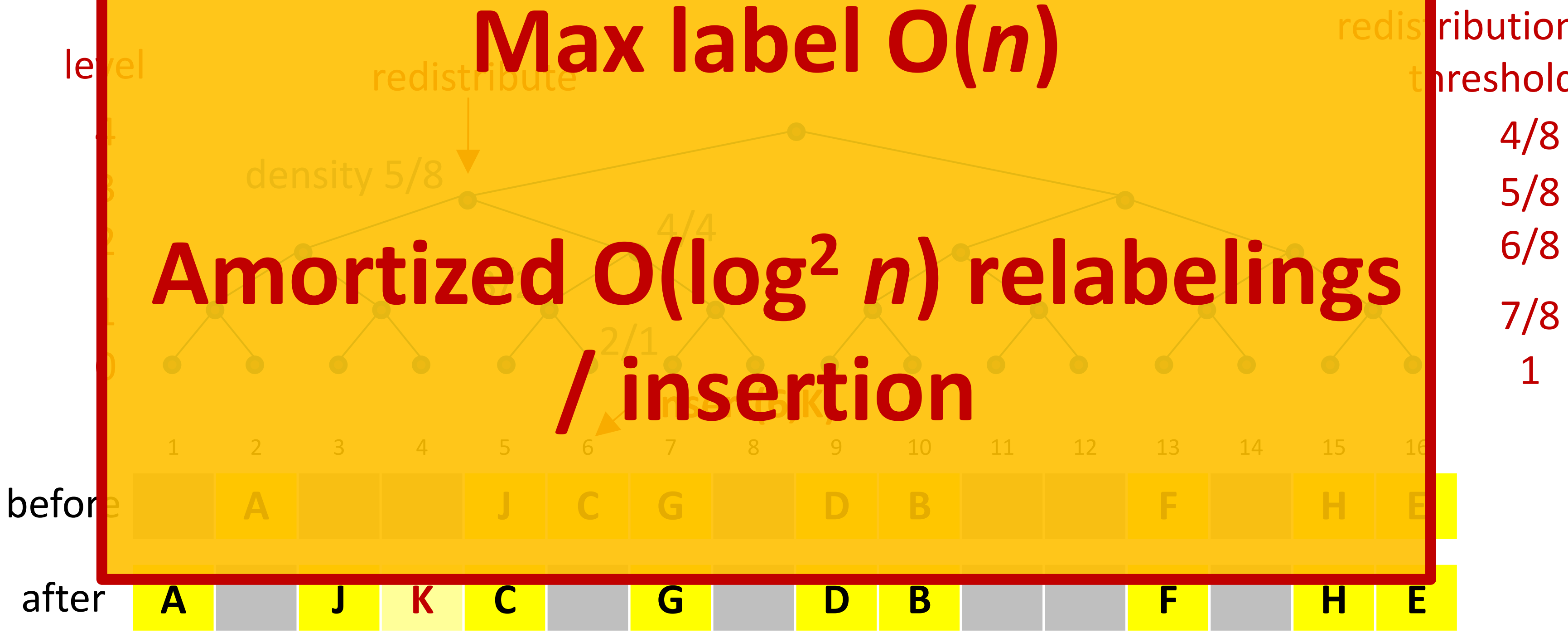
  ⇒ a child requires τ fraction insertions before next overflow

  ⇒ amortized insertion cost = #levels $\cdot\ 1/\tau = O(\log^2 n)$

⇒ List Order Maintenance

Max label $O(n)$

Amortized $O(\log^2 n)$ relabelings / insertion



Amortized O(log n) List Relabelings

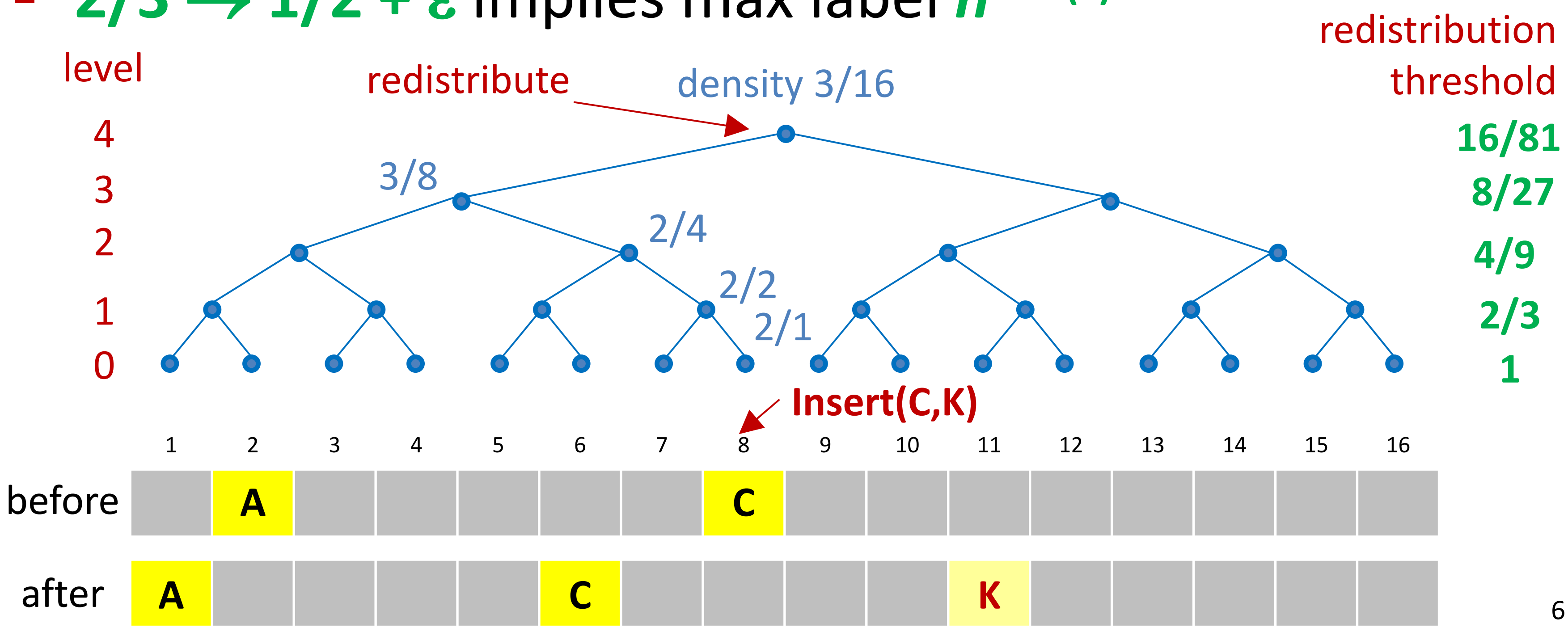
- Level i node overflows if density > $(2/3)^i$
- **Insert** redistribute lowest non-overflowing ancestor

  $\Rightarrow$ $\leq \log_{4/3}$ n levels $\Rightarrow$ max label $2^{\log_{4/3} n} \leq n^{2.41}$

  $\Rightarrow$ a child requires **1/2** fraction insertions before next overflow

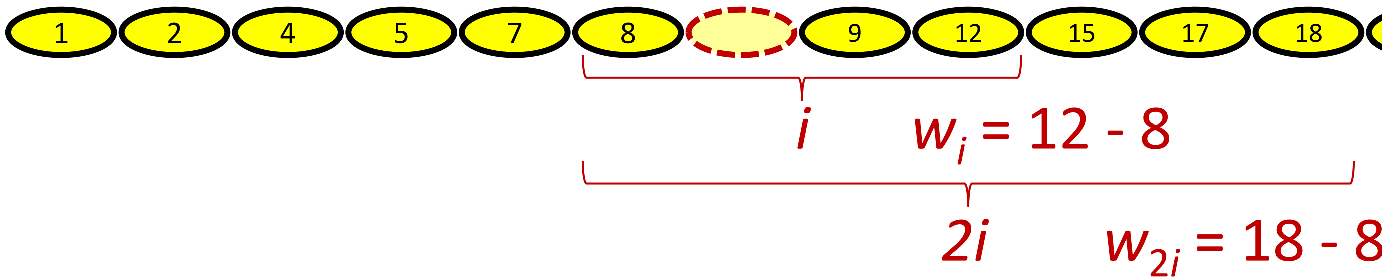
  $\Rightarrow$ amortized insertion cost = **#levels** · **3** = O(log n)

- **2/3 → 1/2 + ε** implies max label $n^{1+O(\varepsilon)}$



Amortized O(log *n*) List Relabelings

[P. Dietz, D. Sleator, *Two algorithms for maintaining order in a list*, ACM Conference on Theory of Computing, 365-372, 1987]



i = 1

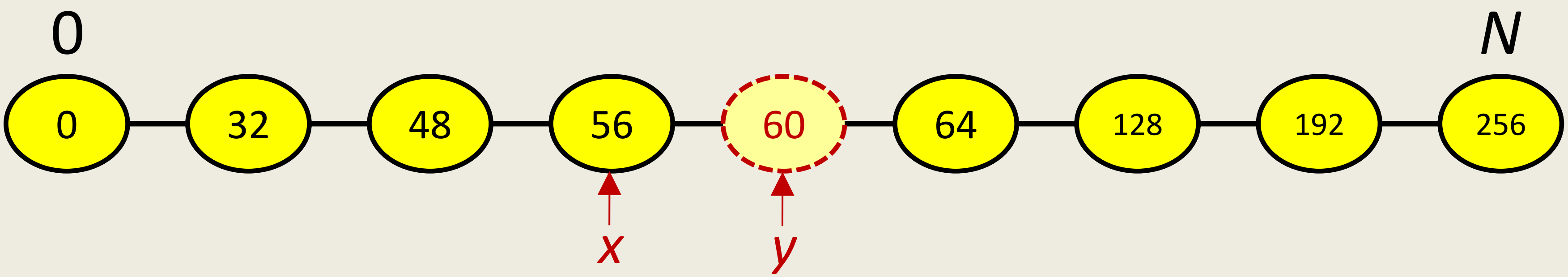
while $w_{2i} \leq 4 \cdot w_i$ **do**

  $i = i + 1$

Relabel uniformly "2*i* area"

- Only relabels to the **right**
- Max label ***M*=4*n*^{2}**
- Requires labels **mod *M*+1**

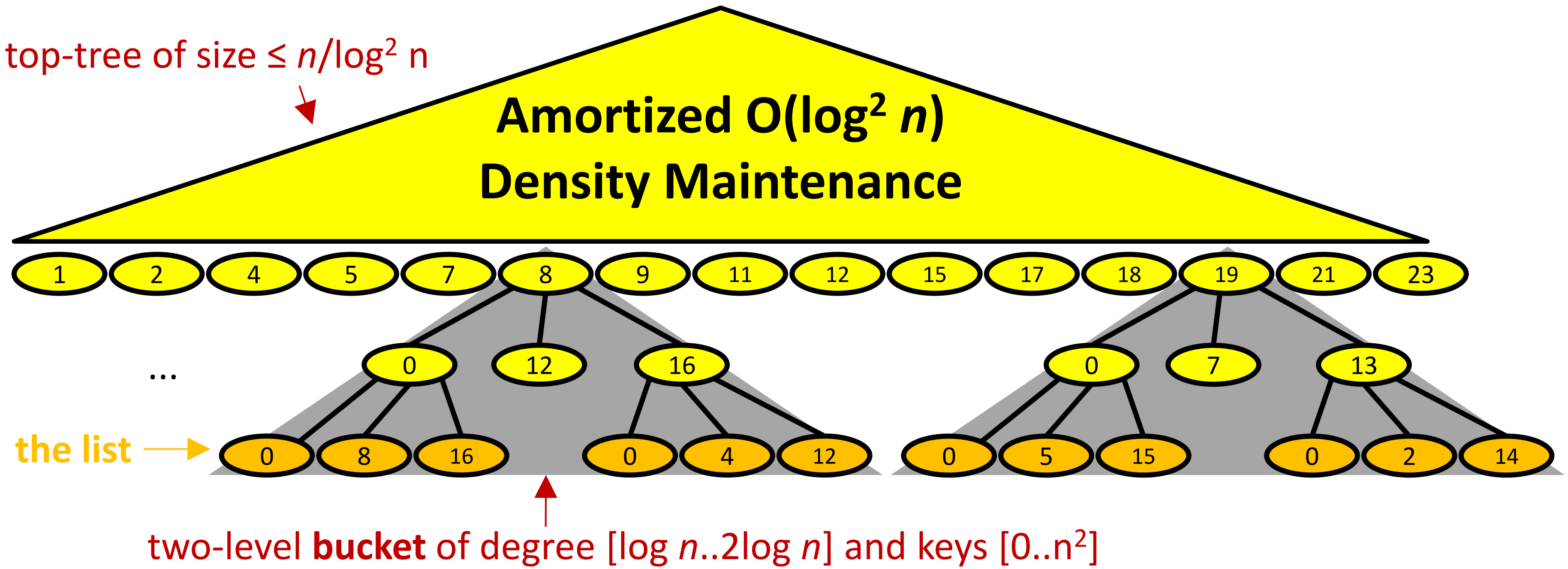
Monotonic List Labeling
O(log *N*) easy insertions



Insert(x,y) Label y = (left + right)/2

⇒ Can perform log *N* insertions without relabeling

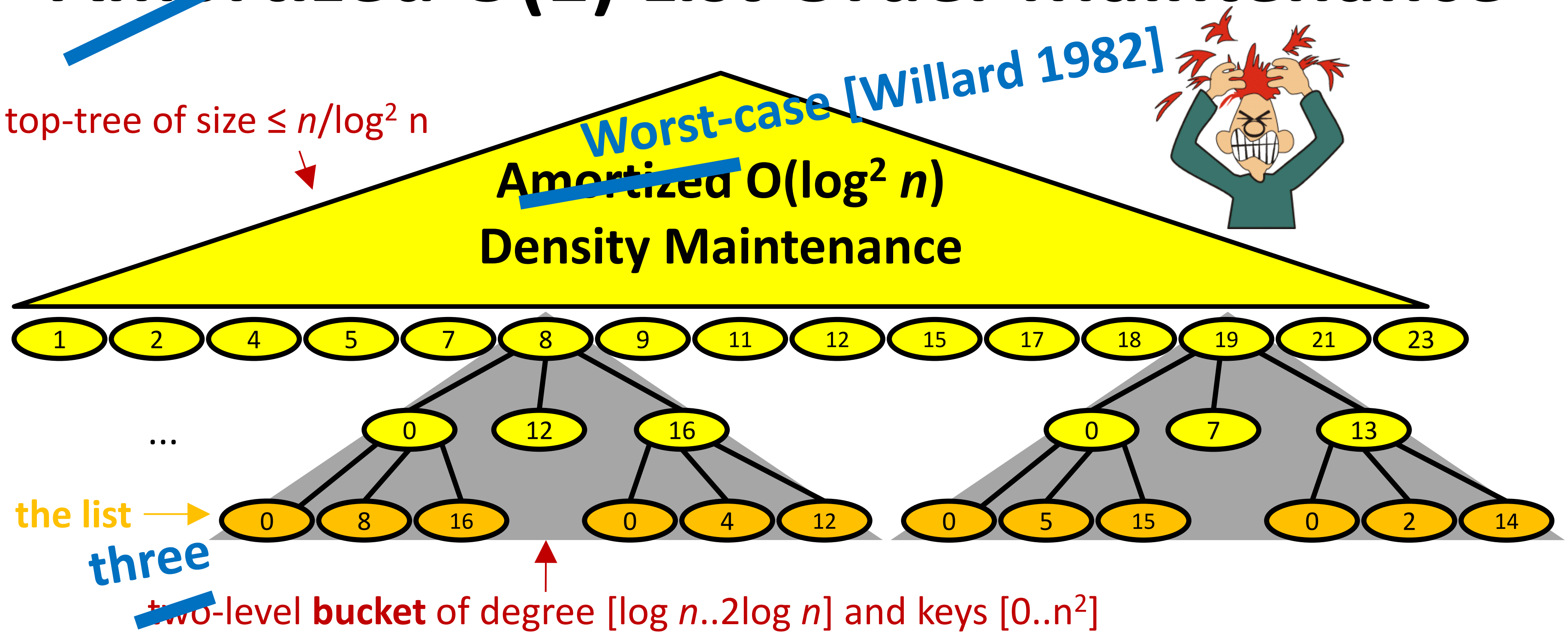
Amortized O(1) List Order Maintenance



Insertion

- create and label new **leaf**
- split nodes of degree > 2log n and relabel with gap n
- insert in top tree

~~Amortized~~ Worst-case O(1) List Order Maintenance



Insertion

– create and label new leaf

– split nodes of degree > $2\log n$ and relabel with gap n

– insert in top tree

+ incremental insertion into top tree

+ incremental splitting of bucket nodes

+ every $O(\log^2 n)$'th operation split largest bucket

$\Rightarrow$ largest bucket size $O(\log^3 n)$

Theorem 5 *Let $x_1, \ldots, x_n$ be n real valued variables, all initially zero. Repeatedly perform the following procedure:*

1. *Find an i, $1 \le i \le n$, such that $x_i = \max_j \{x_j\}$. Set x_i to zero.*
2. *Pick n nonnegative reals $a_1, \ldots, a_n$ such that $\sum_{i=1}^{n} a_i = 1$.*
3. *For $i = 1, \ldots, n$, set x_i to $x_i + a_i$.*

No x_i will ever exceed $H_{n-1} + 1$, where $H_k = \sum_{i=1}^{k} i^{-1}$, the kth harmonic number.