Order maintenance problem

Dietz, Sleator 1987

Bender, Demaine, Farach, Zito 2001

Problem definition

Perform a sequence of the following operations on a list L

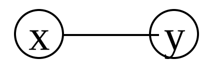
Insert(x,y) : Insert record y after x into the list

Delete(x) : delete record x from the list

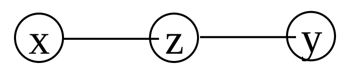
Order(x,y) : Return true if x is before y in the list. Otherwise return false

First attempt

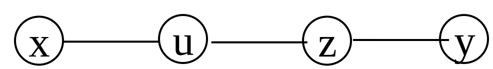
Insert(x,y)



Insert(x,z)



Insert(x,u)

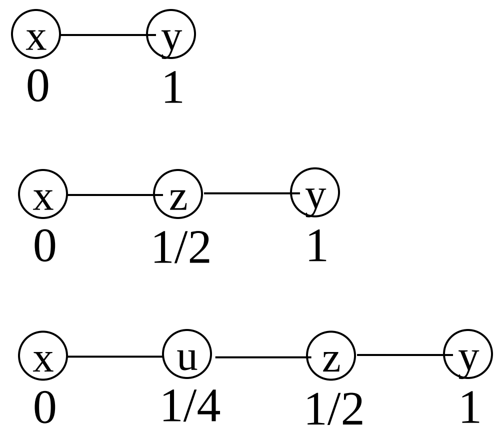


First attempt

Insert(x,y)

Insert(x,z)

Insert(x,u)



Bender et al's solution

Imagine your list items reside at the leaves of a complete binary tree. M leaves, item labeled i reside at the i^{th} leaf

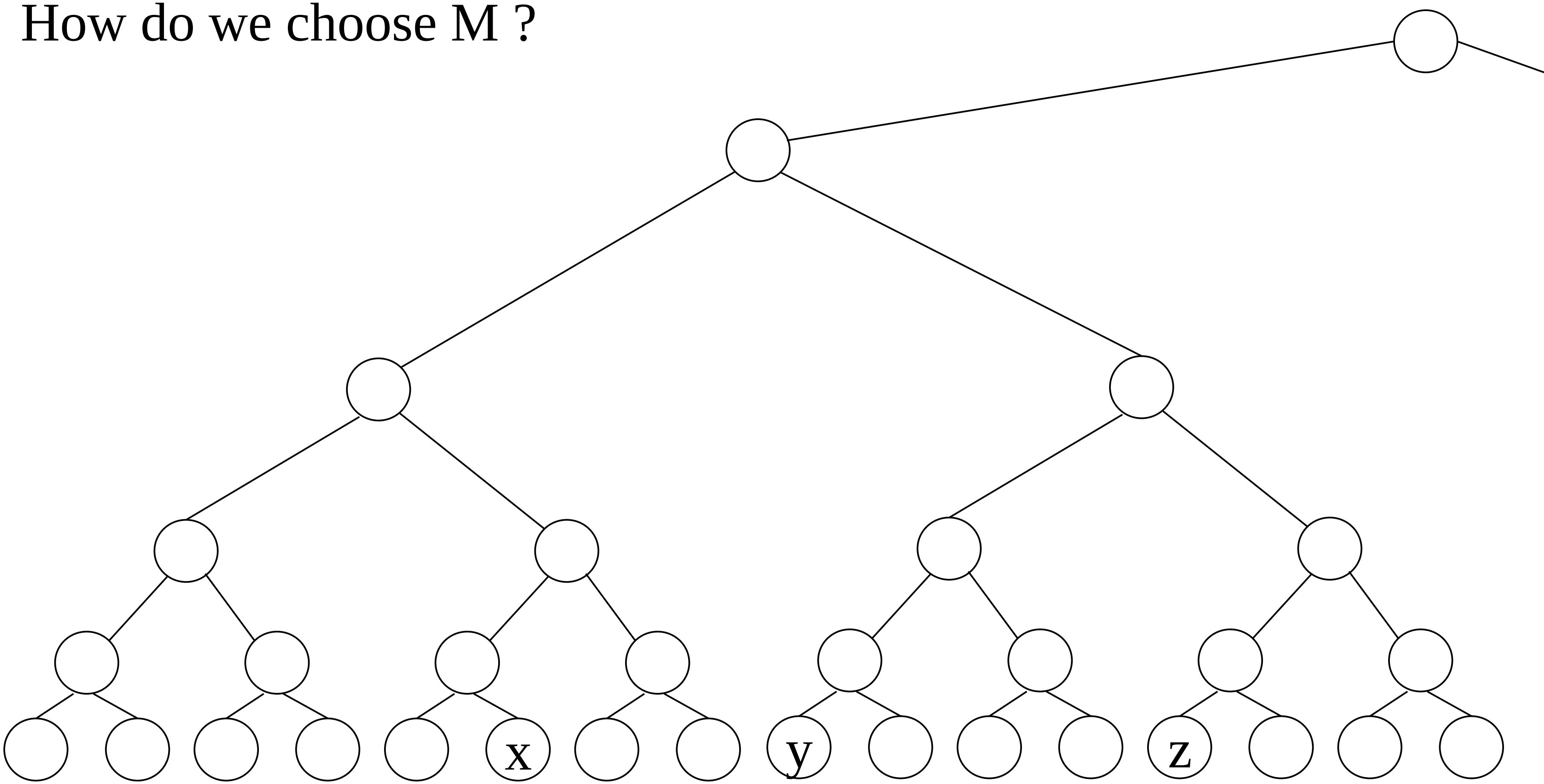
When a "phase" starts we relabel n items so they are equally spaced among the leaves

Phase goes on as long as

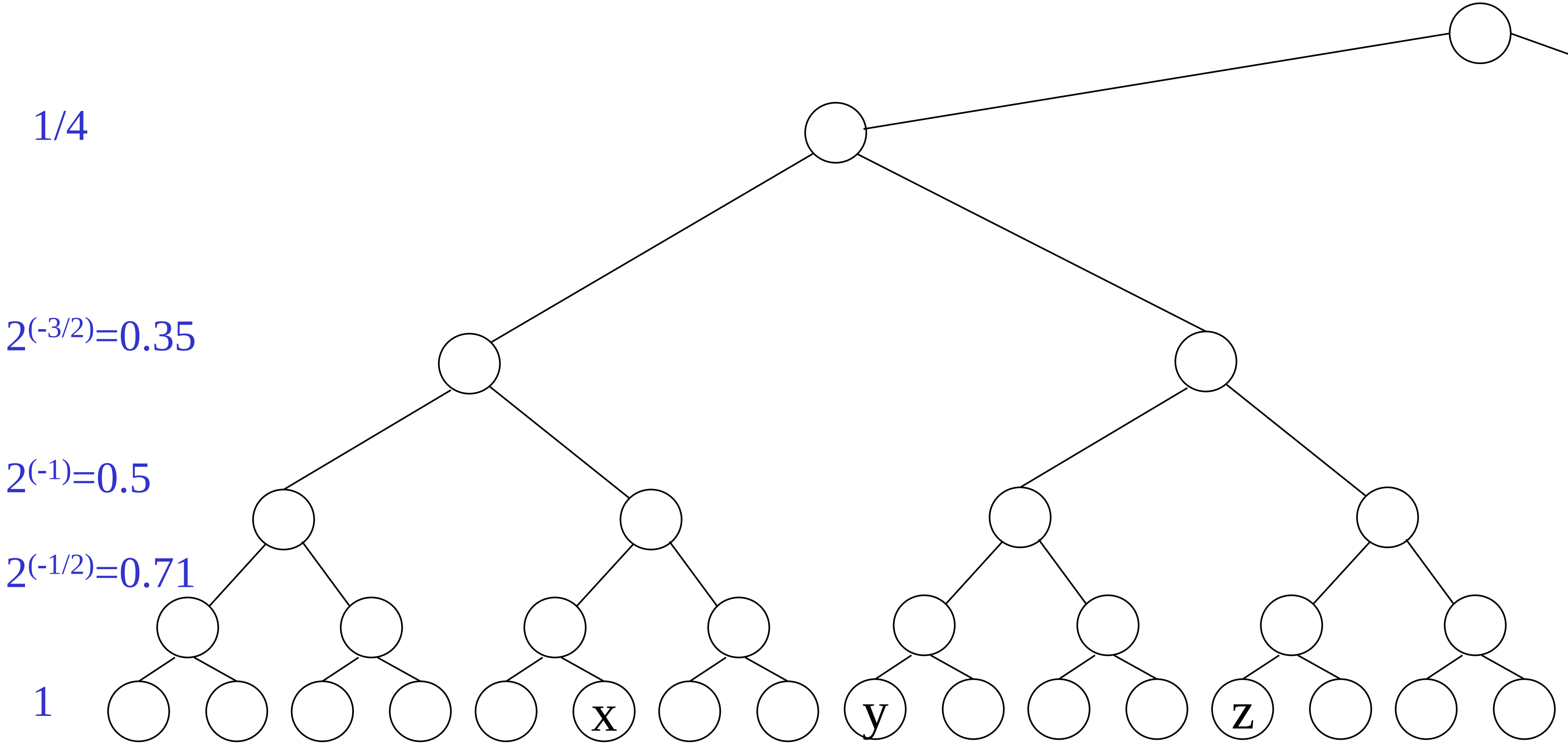
$$n/2 \leq \#\text{items} \leq 2n$$

Remember the data structure is only a linked list

How do we choose M ?



Associate maximum density with the nodes, 1, T^{-1}, T^{-2} . . .
$T \in (1,2)$



We pick the tree large enough such that density of the root breaks only when the # of items grow by a factor of 2.

Size of the labels

Lemma1:

We use O(log n) bits per tag.

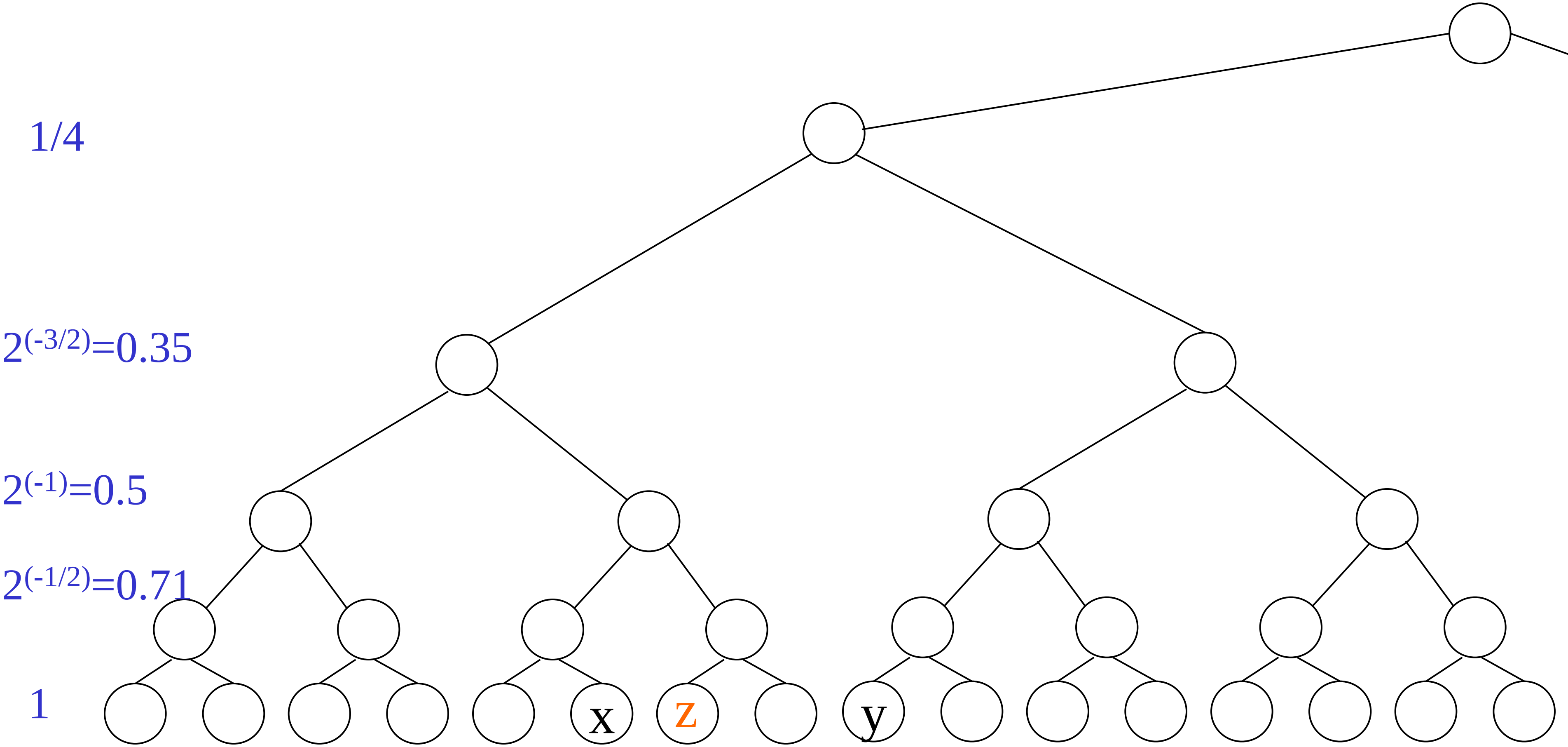
Proof.

At the start of a phase, we choose M such that:

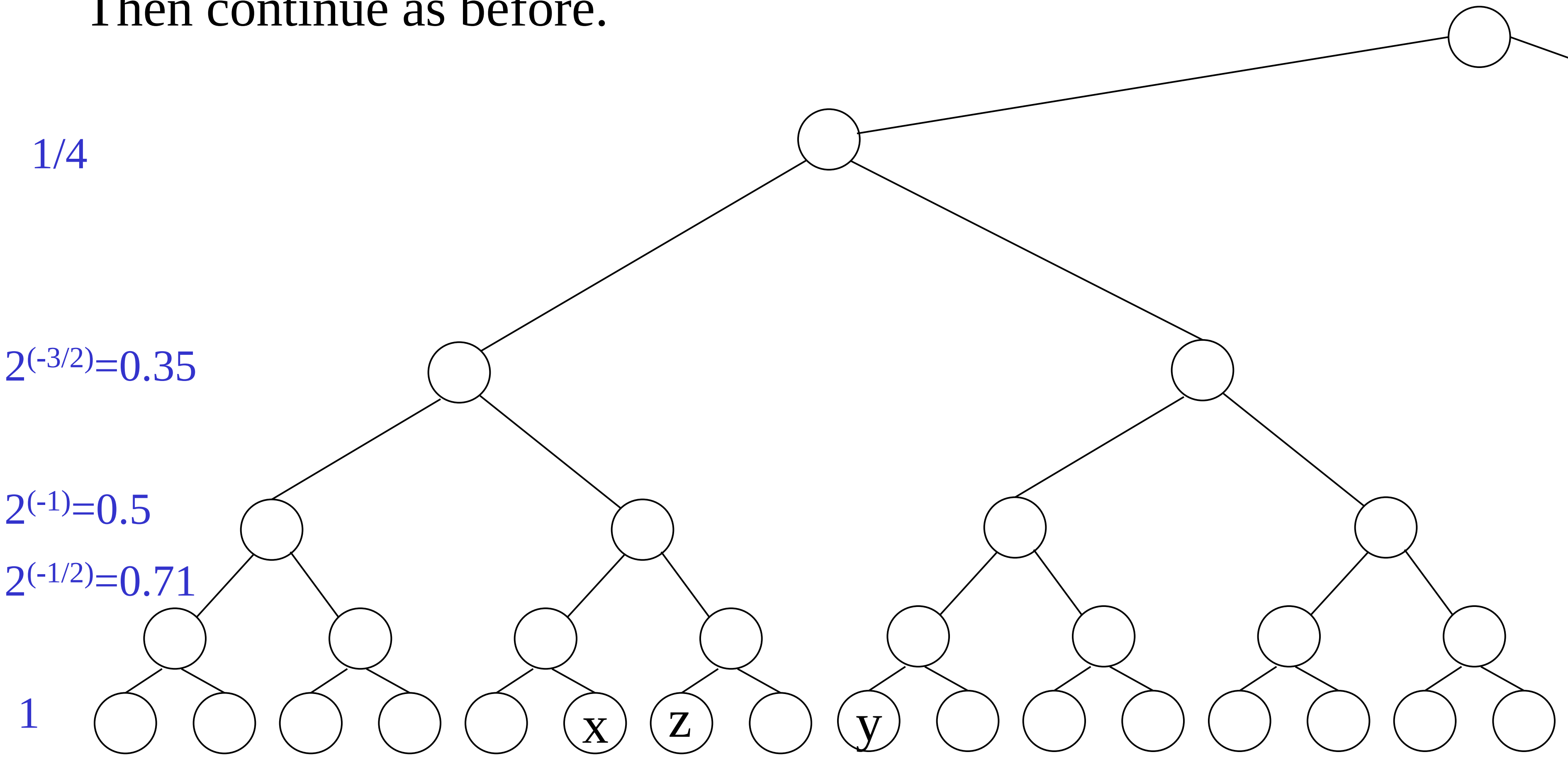
$T^{-\log(M)} = 2n/M$

So we use $\log(M) = \log(2n)/(1-\log T)$ bits $\square$

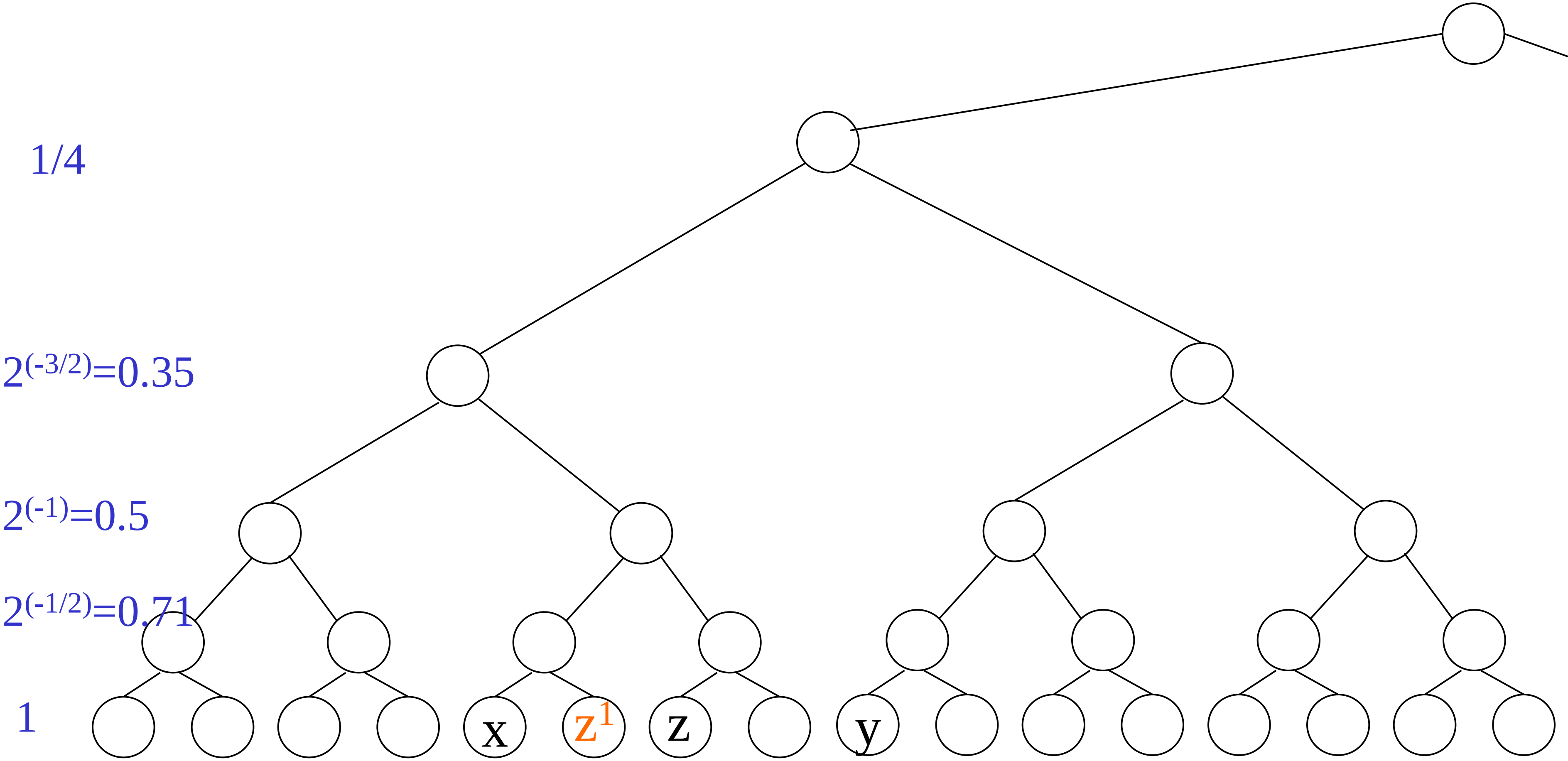
Insert(x,z) : Choose any label between l(x) and l(s(x)) if they are not consecutive



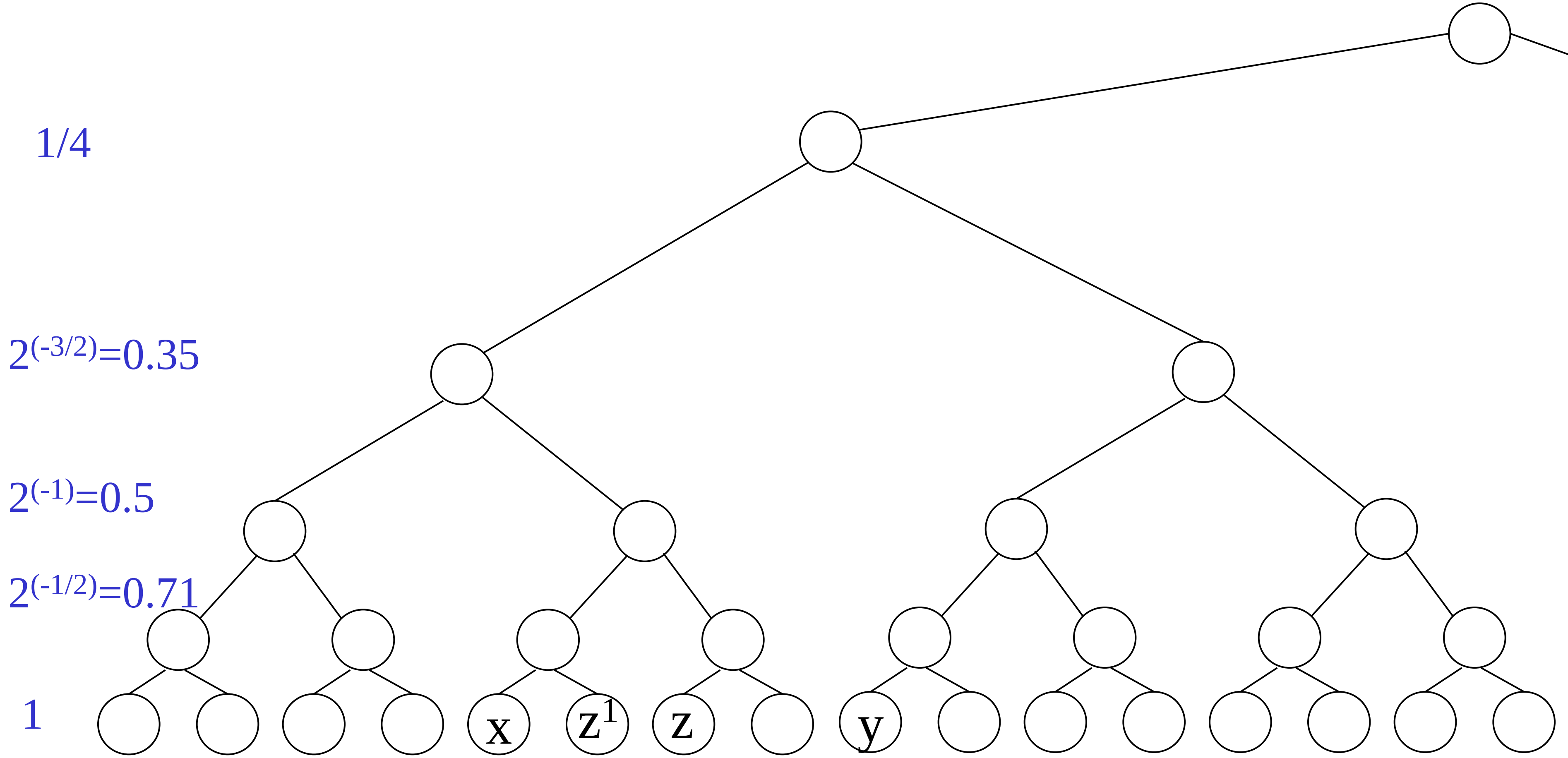
Insert(x,*z'*) : Choose any label between l(x) and l(s(x)) if they are not consecutive. Otherwise, find the lowest ancestor of x such that the density of items in its subtree is below the threshold and relabel them evenly. Then continue as before.



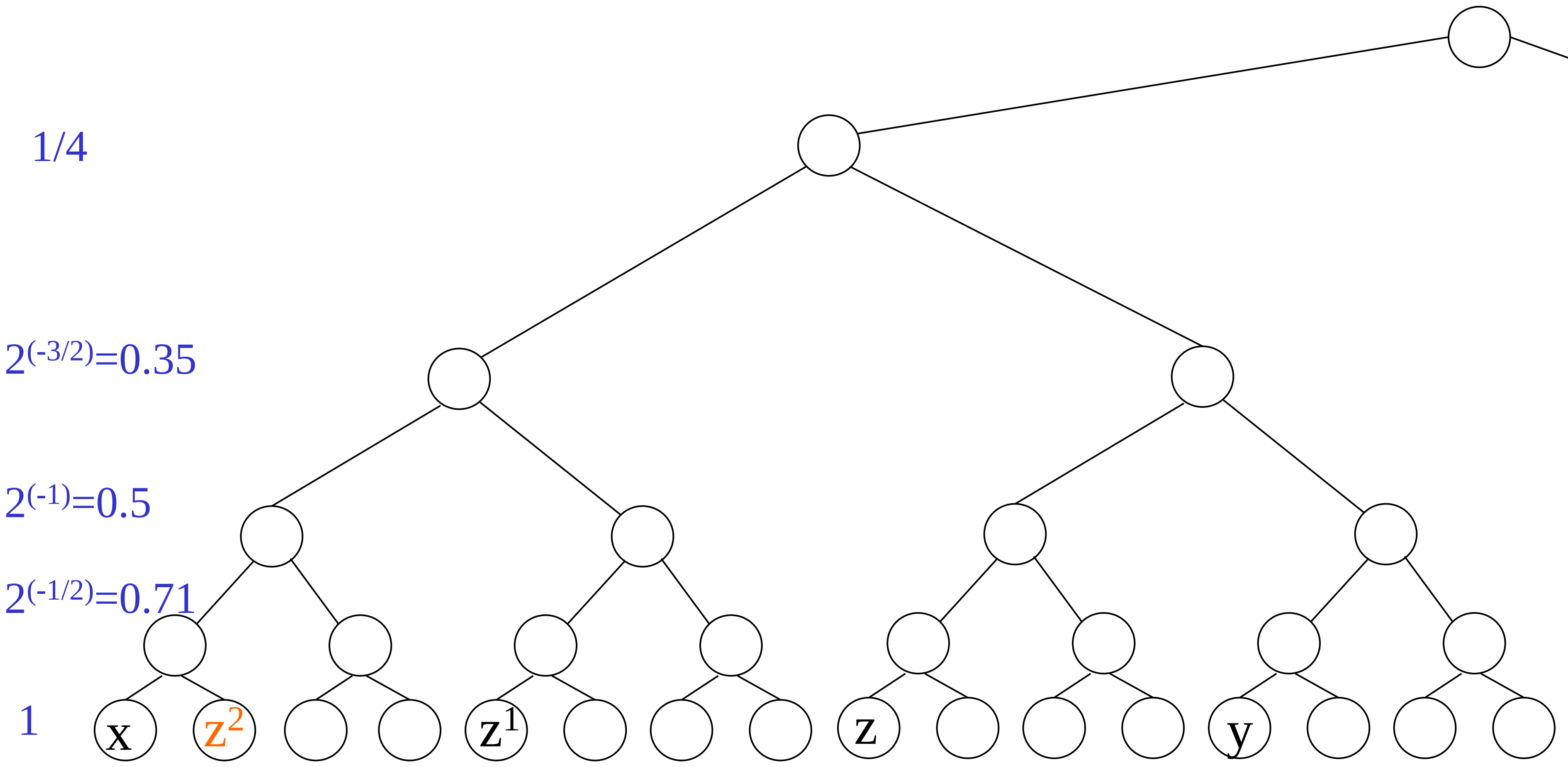
Insert(x,z[1]) : Choose any label between l(x) and l(s(x)) if they are not consecutive. Otherwise, find the least ancestor of x such that the density of items in its subtree is below the threshold and relabel them evenly. Then continue as before.



Insert(x,z^2)



Insert(x,z^2)



Few important observations:

- The tree does not exist! We do the density calculations as we go up just by going to the right and to the left in the list.

- Once the density of the root overflows the phase ends and we “create” a new tree twice as large. We relabel all nodes evenly in the larger space.

- We do the same if the number of nodes decrease by a factor of 2 since the last time we fixed the tree.

Complexity analysis

- How much time it takes to relabel a subtree rooted at x with 2^i leaves ?

$O(2^i / T^i)$

- What is the density of the children of x after the relabeling ?

At most $1/ T^i$

- Before we relabel x again the density of one of its children must grow to ?

At least $1/ T^{i-1}$

- So between relabeling we have inserted

$(1/ T^{i-1} - 1/ T^i)\ 2^{i-1}$ nodes.

Complexity analysis

- Charge the relabeling work to the new nodes inserted since the last relabeling. How much do we charge to each node ?

$$\frac{2^i / T^i}{(1/T^{i-1} - 1/T^i)\, 2^{i-1}}$$

$= 2/(T-1)$

- How many relabelings charge a node x ?

At most log M = O(log n)