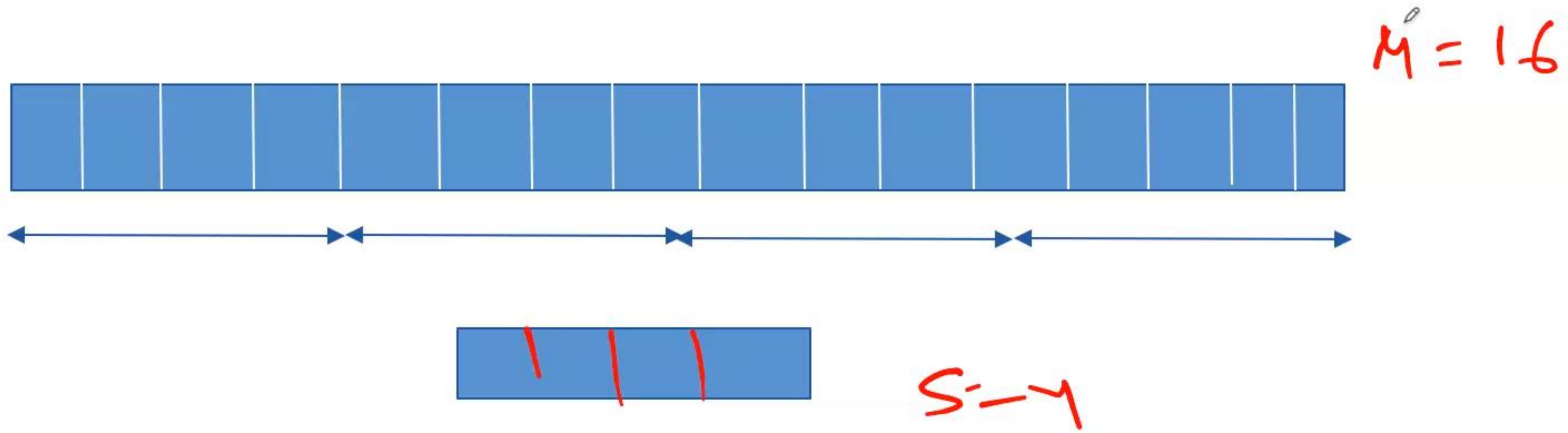
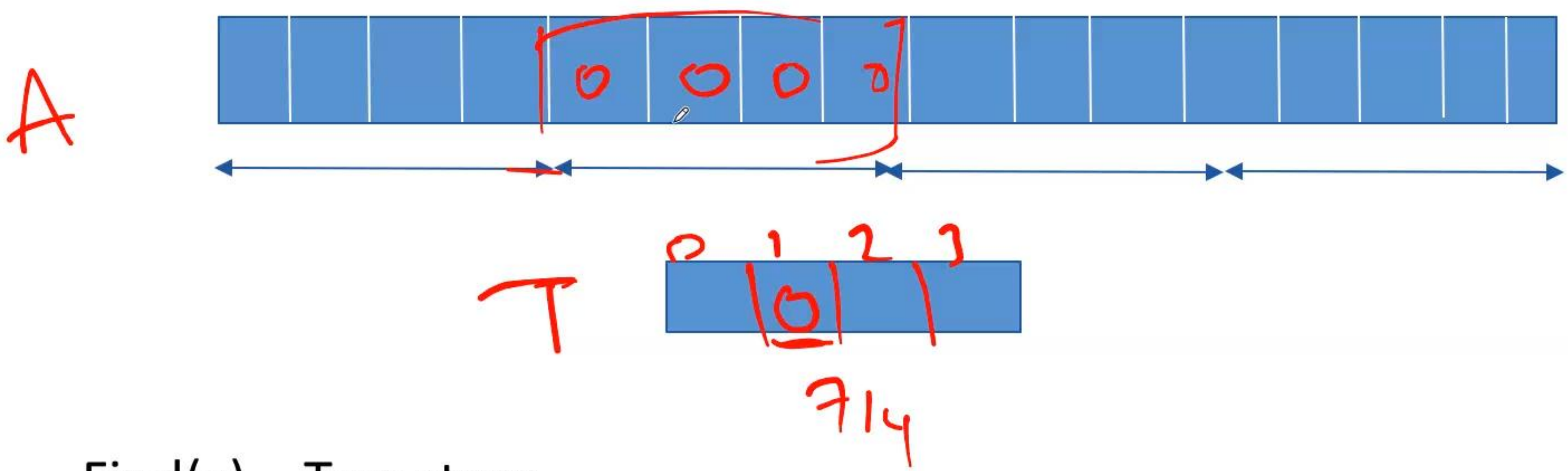
Tiered Bit Vectors



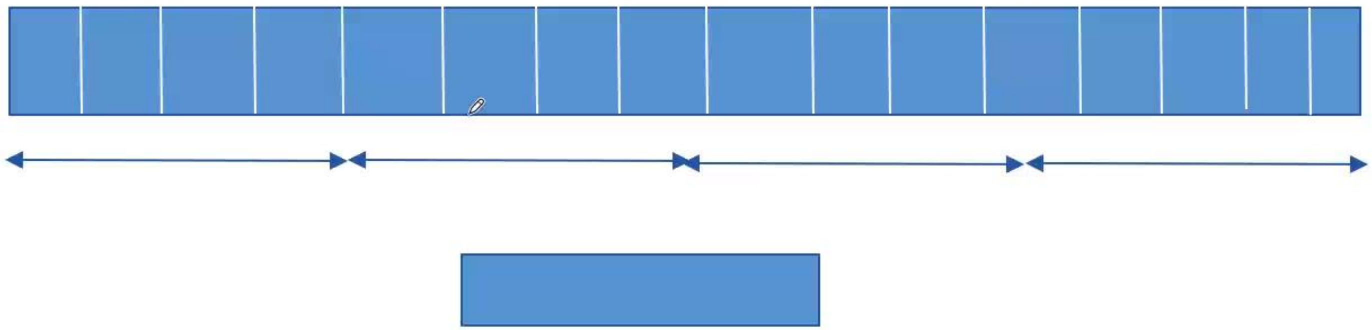
- Formally, let s be a parameter.
- The array A is partitioned into A/s pieces and the array T is of size s.

Operations on Tiered Bit Vectors



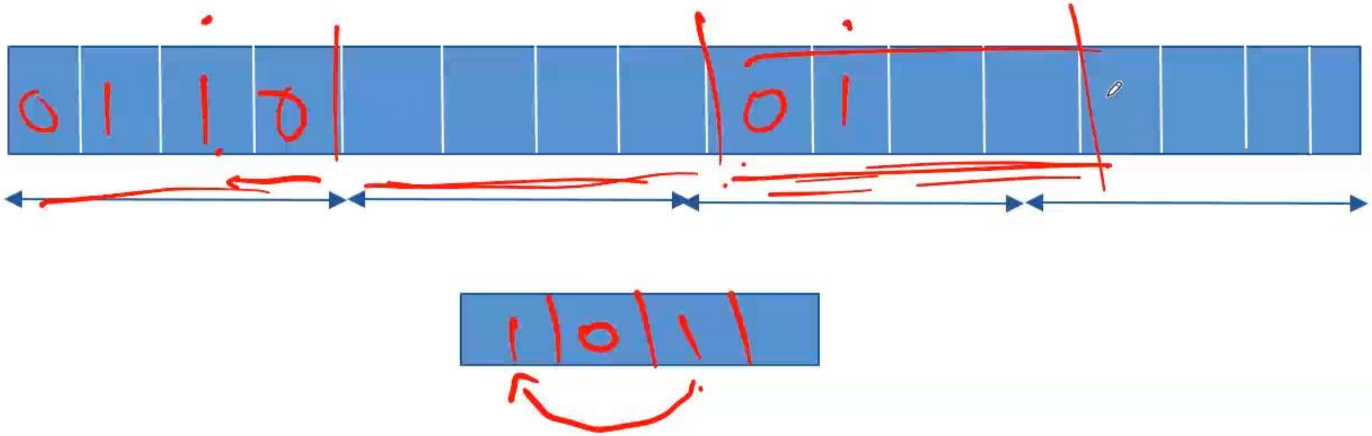
- Find(x) – Two steps
  - Otherwise, check for x in array $A_{x/s}$.
- Each Find(x) is therefore translated to one isempty() query and one find on a small array.

Operations on Tiered Bit Vectors



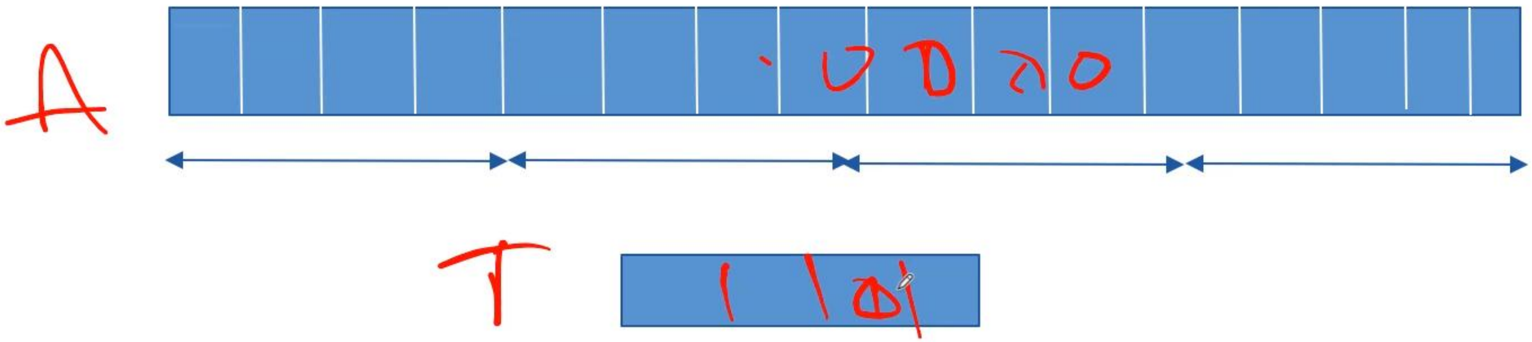
- Min() should return the value of the smallest present element.
- Find the minimum value in the smaller vector, say j.
  - Suggests that T[j] is 1 and all other T[i] if i < j are 0.
- Find minimum in A_j.
- In essence, Min() translates to two Min() operations on smaller vectors.

Operations on Tiered Bit Vectors



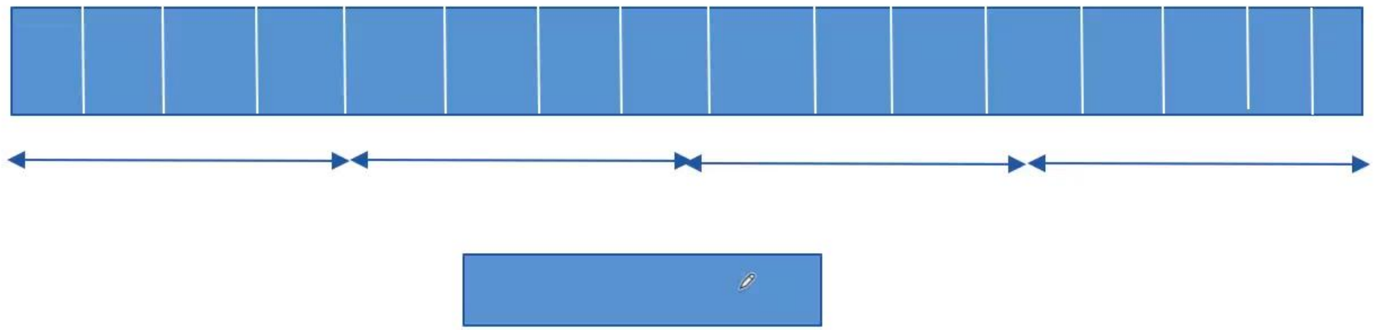
- Prev(x) – Again multiple steps on smaller vectors.
  - Find Min() in the vector $A_{x/s}$.
  - If the answer is different from x, return.
  - Otherwise, find j = Prev(x/s) in the smaller vector.
  - Find Max() in A_j.
- Each Prev(x) is translated to two Prev() queries. Same with Next(x).

Operations on Tiered Bit Vectors



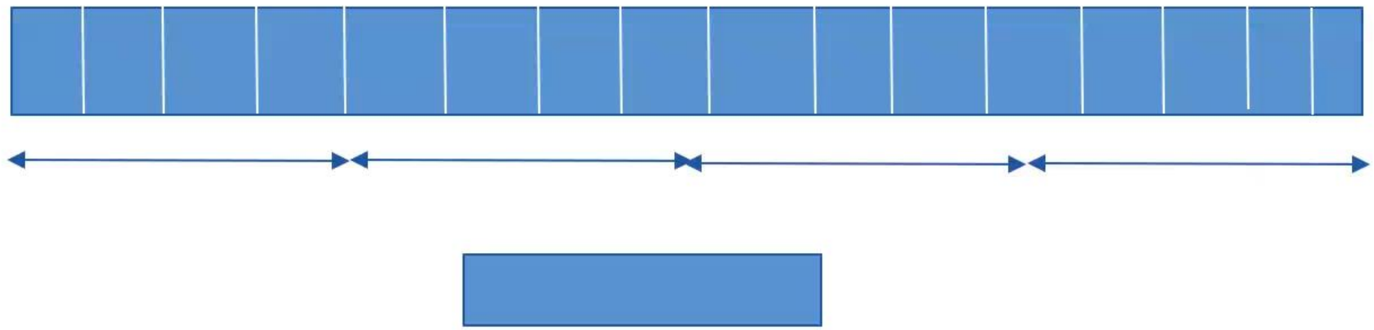
- Delete(x) – Again two steps
  - Mark A[x] in $A_{x/s}$ as 0.
  - If all entries in $A_{x/s}$ are 0, then set T[x/s] to 0.

Operations on Tiered Bit Vectors



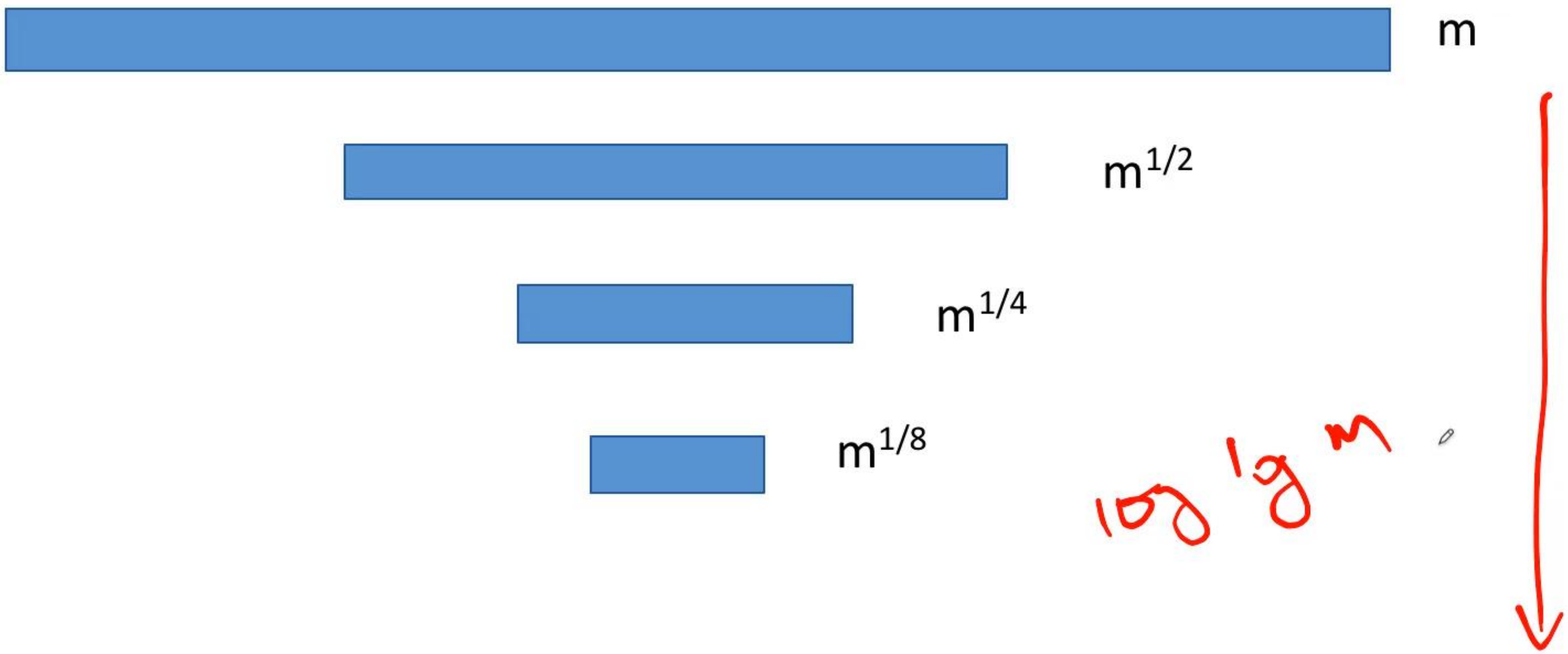
- Each operation turns into operations on smaller arrays as follows. Time taken is:
  - insert: 2x insert – O(1)
  - find : 1x lookup – O(1)
  - is-empty: 1x is-empty – O(1)
  - min: 2x min – O(s + m/s)
  - next: 1x next, 1x max, 1x min – O(s+m/s)
  - delete: 2x delete, 1x is-empty – O(s+m/s)

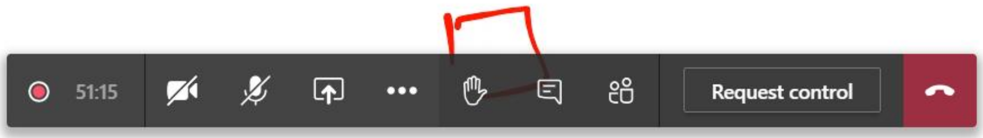
Operations on Tiered Bit Vectors



- Each operation turns into operations on smaller arrays as follows. Time taken is:
  - The run time of O(s+m/s) is seen to be minimized when s = sqrt{m}.
  - In other words, all operations finish in time O(sqrt{m}).
    - Some operations are much faster.
    - Not a good solution – BST much better.

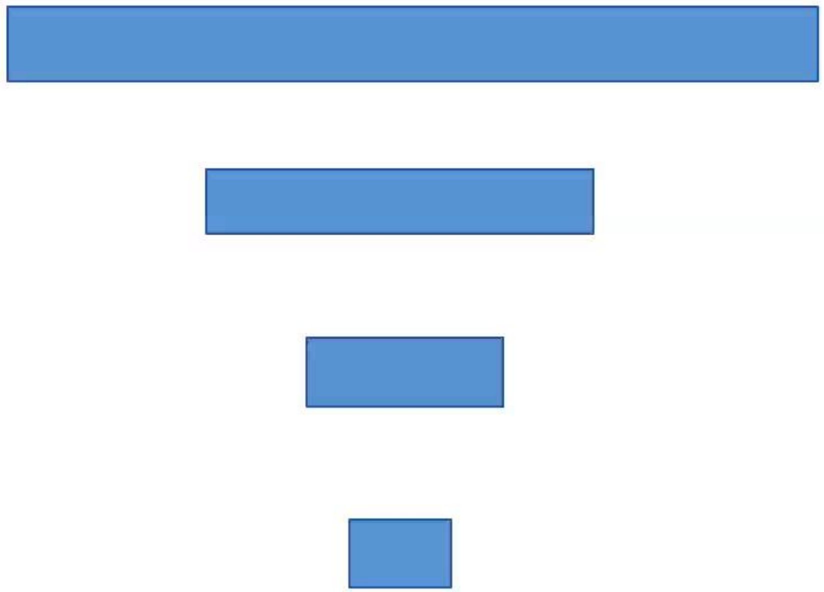
Extending Tiered Bit Vectors





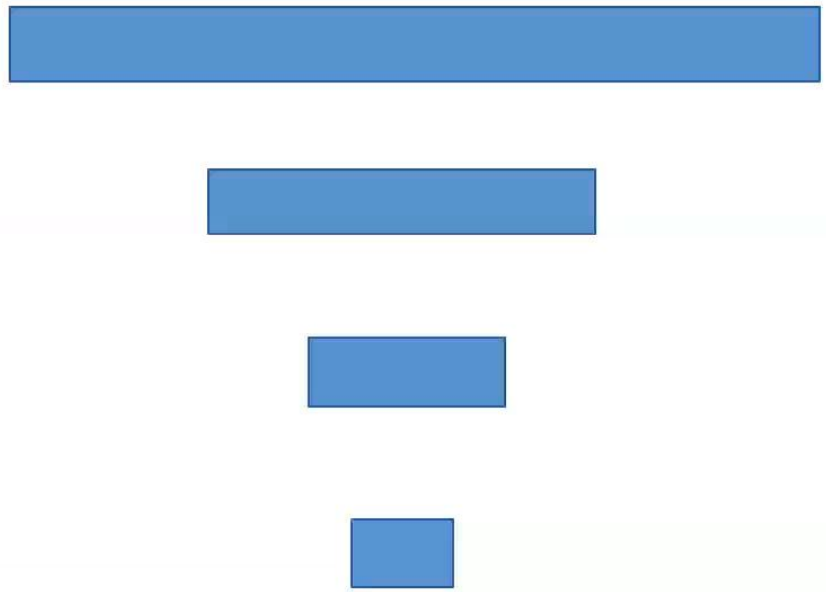
Extending Tiered Bit Vectors

- A helpful view is to see the two levels of the hierarchy as forming $1+m^{1/2^i}$ smaller vectors at level i.



Operations on Extended Tiered Bit Vectors

- Consider the set of operations again.
  - Insert: 2x insert, at each recursive level.
  - Find : 1x lookup – but can say lookup in a smaller bit vector.
  - is-empty: 1x is-empty, but can say on a smaller bit vector
  - min: 2x min, one each at a smaller level
  - next: 1x next, 1x max, 1x min, at a smaller level
  - delete: 2x delete at a smaller level, 1x is-empty,

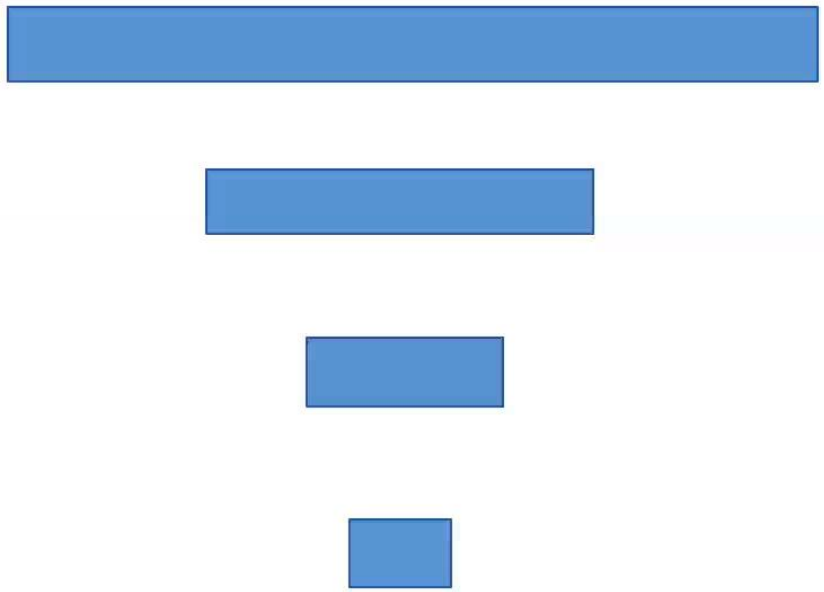


Operations on Extended Tiered Bit Vectors

- Consider the set of operations again.
  - Find : 1x lookup – but can say lookup in a smaller bit vector.

    Time : T(m) = T(sqrt{m}) + O(1)

    Solution: T(m) = O(loglog m).



Extending Tiered Bit Vectors

Proposition 1. $T(U) = T(\sqrt{U}) + O(1) = O(\log \log U)$.

Proof. Let $m = \log U \Rightarrow U = 2^m$. Our recurrence relation is now: $T(2^m) = T(2^{\frac{m}{2}}) + O(1)$. Let $S(m) = T(2^m)$, then we have that: $S(m) = S(\frac{m}{2}) + O(1)$. By case 2 of the master method, $S(m) = O(\log m)$. Therefore, $T(U) = T(2^m) = S(m) = O(\log m) = O(\log \log U)$. [**?**] $\square$

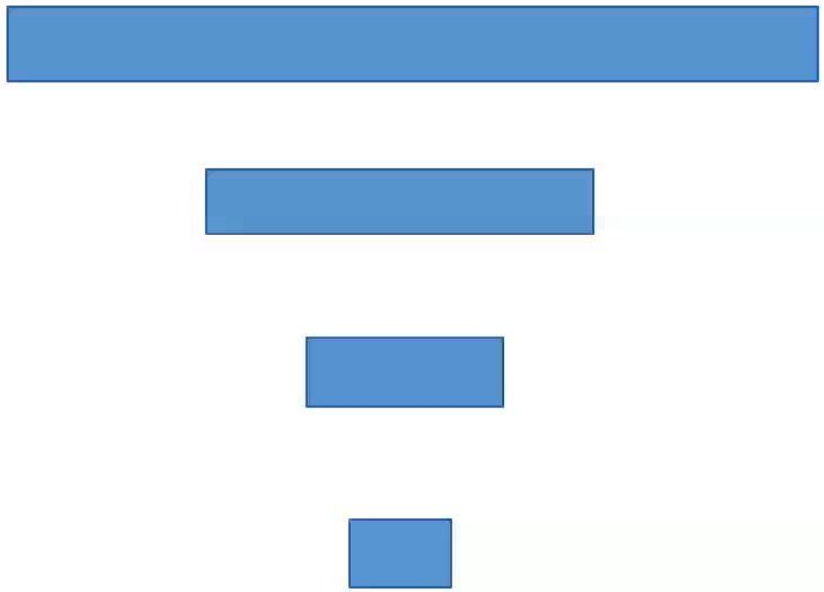
59:18 Request control

AS

Advanced Data Structures ...

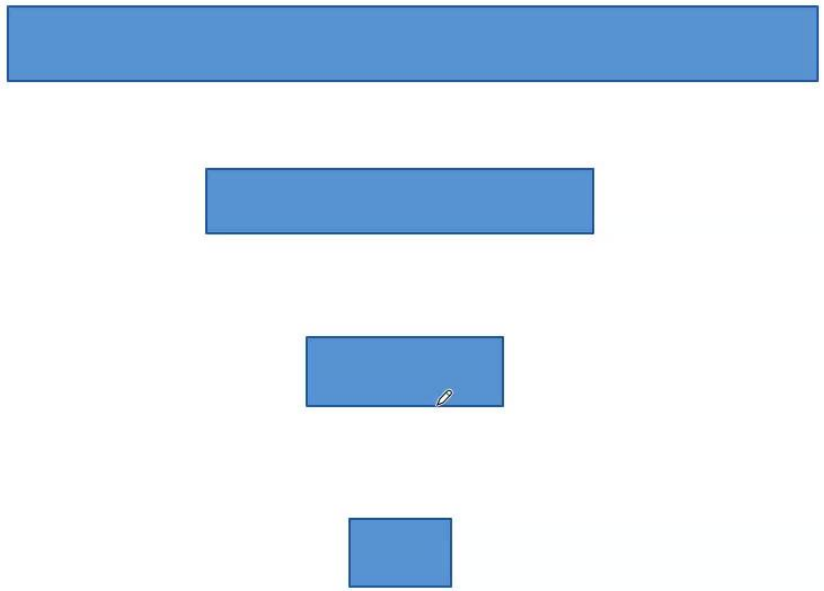
Operations on Extended Tiered Bit Vectors

- Consider the set of operations again.
  - Insert: 2x insert, at each recursive level.
  - Find : 1x lookup – but can say lookup in a smaller bit vector.
  - is-empty: 1x is-empty, but can say on a smaller bit vector
  - min: 2x min, one each at a smaller level
  - next: 1x next, 1x max, 1x min, at a smaller level
  - delete: 2x delete at a smaller level, 1x is-empty,



Operations on Extended Tiered Bit Vectors

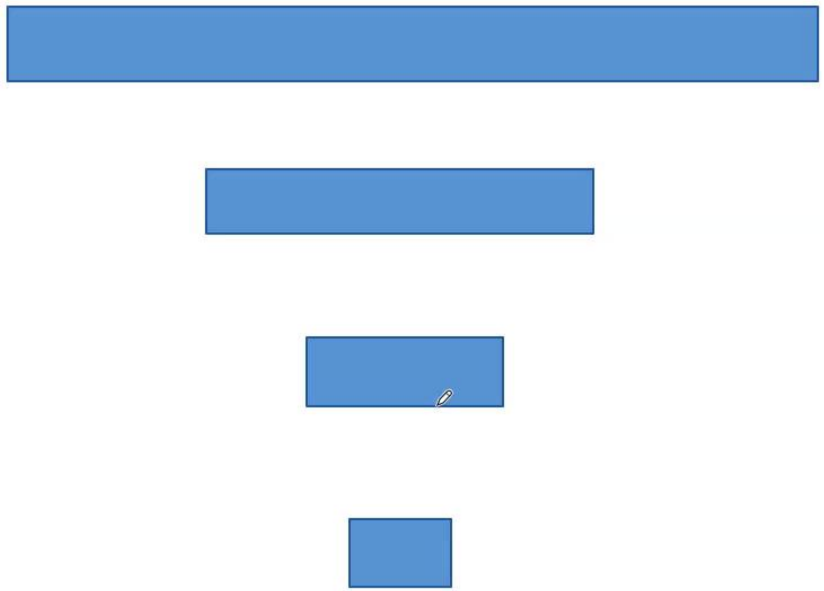
- Consider the set of operations again.
  - next: 1x next, 1x max, 1x min, at a smaller level
  - delete: 2x delete at a smaller level, 1x is-empty,



T(m) = T(sqrt{m}) + T(max) + T(min)
= T(sqrt{m}) + O(log m)
= O(log m)

Operations on Extended Tiered Bit Vectors

- Consider the set of operations again.
  - is-empty: 1x is-empty, but can say on a smaller bit vector
    Time: T(m) = T(sqrt{m}) + O(1).
  - min: 2x min, one each at a smaller level
    Time: T(m) = 2T(sqrt{m}) + O(1)
    Solution: From earlier, T(m) = log(m)



Our Solution So Far

- We have a data structure, the tiered bit vector, where some operations are O(loglog m).
  - These are exponentially faster than using BSTs.
- Other operations are in O(log m).
  - These are worse than BST.
- Can we make all operations faster?
  - Let us identify some potential places for optimization.

01:02:50 Request control

AS
Advanced Data Structures ...