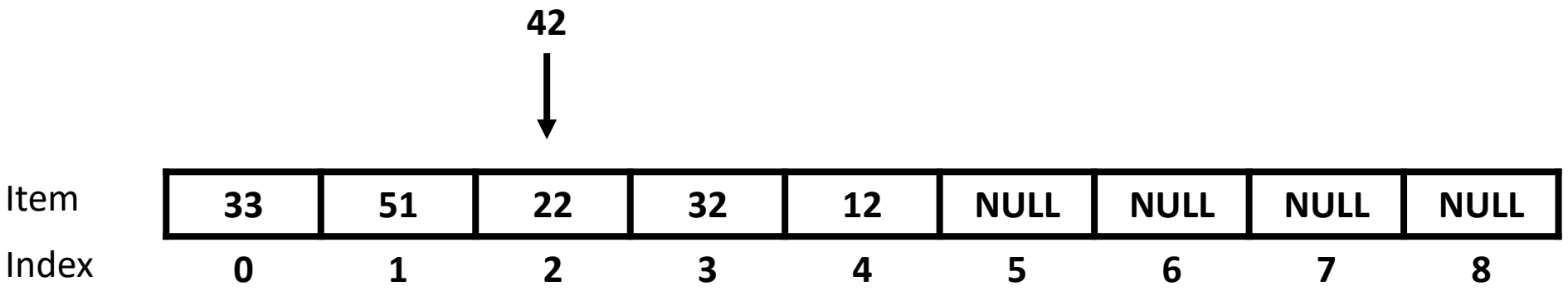
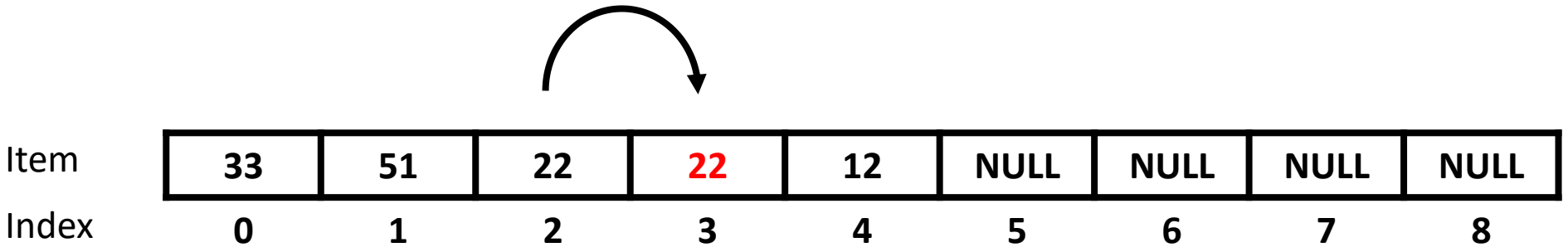
Insert when $n + 1 <= U$ (space left in array)

Wrong Way



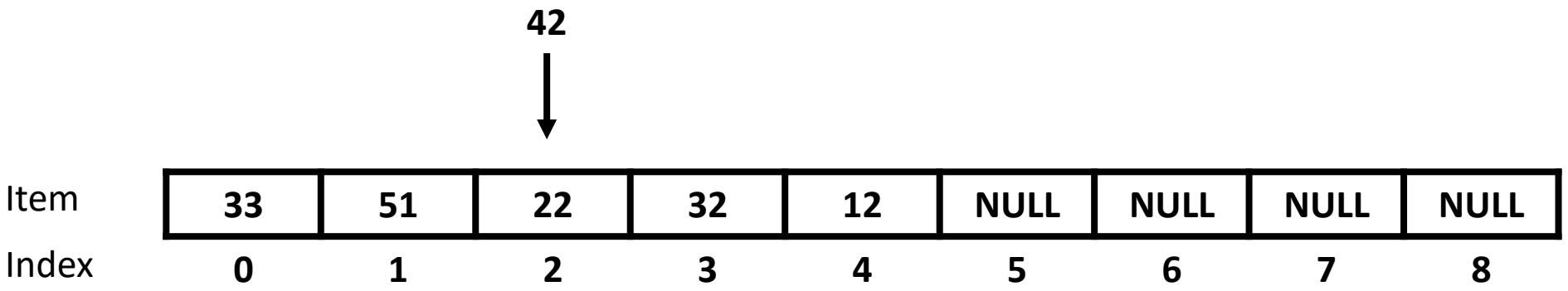
Wrong Way

- Forward right copy – Shift items to the right starting from the front.



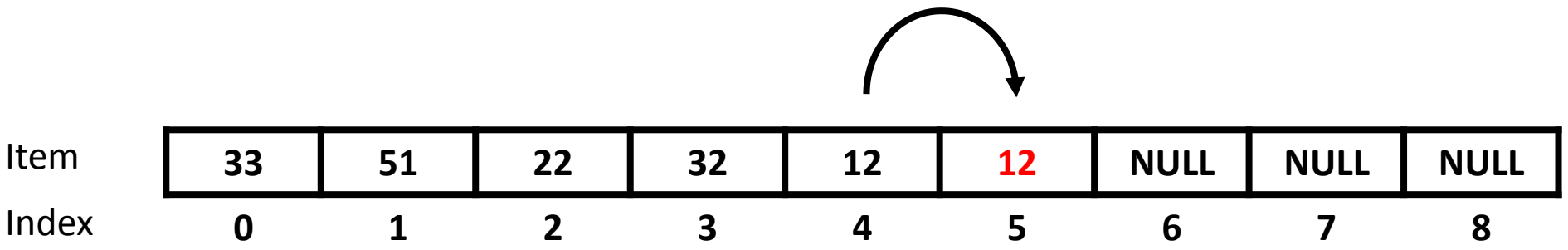
We have just written over 32. Oops!

Right Way

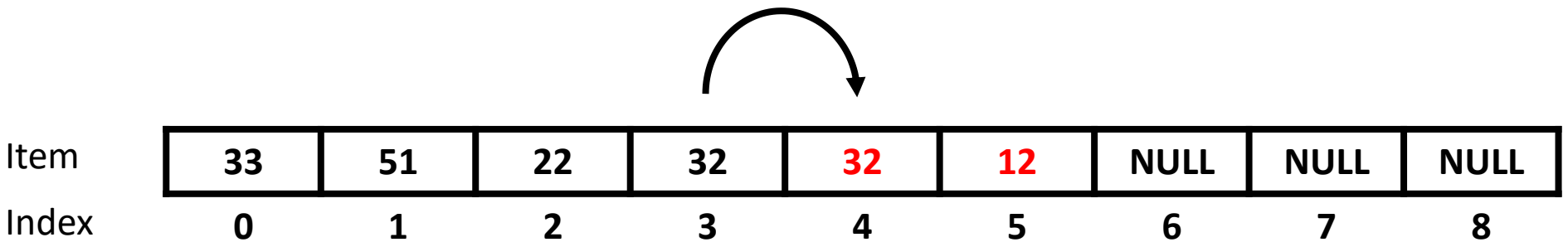


Right Way

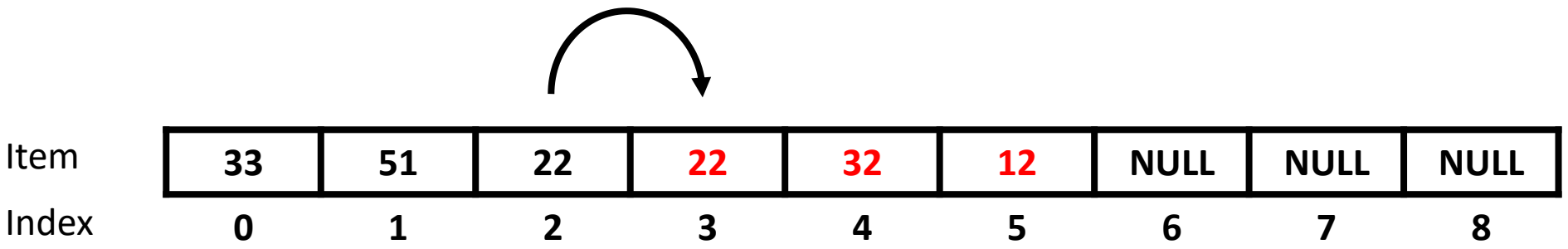
- Backward right copy – Shift items to the right starting from the back.



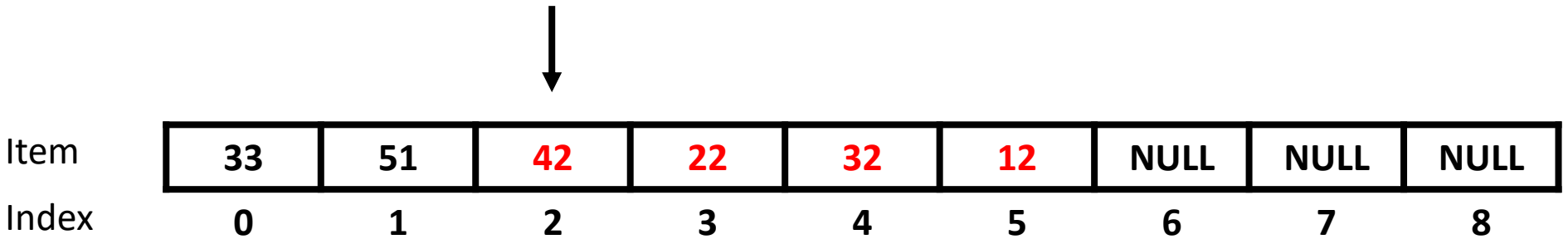
- Backward right copy – Shift items to the right starting from the back.

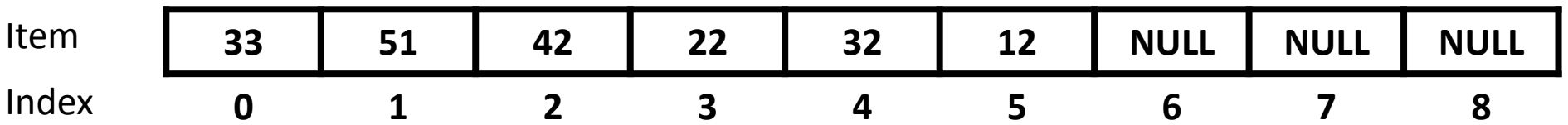


- Backward right copy – Shift items to the right starting from the back.



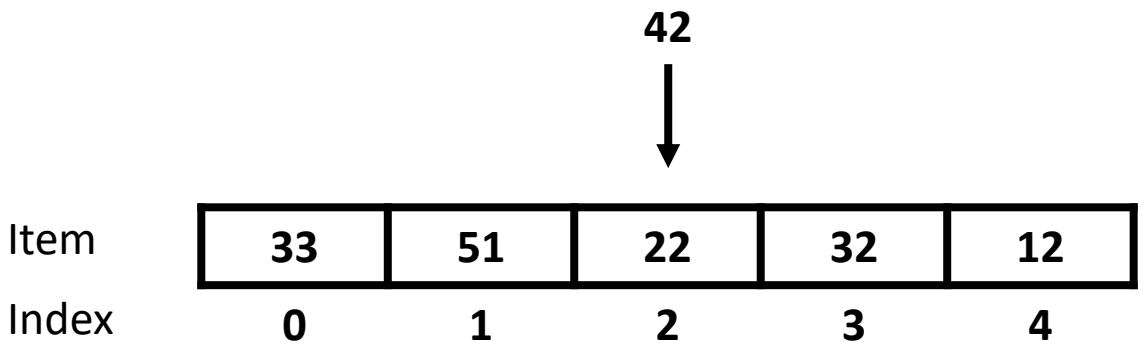
- Perform Insert

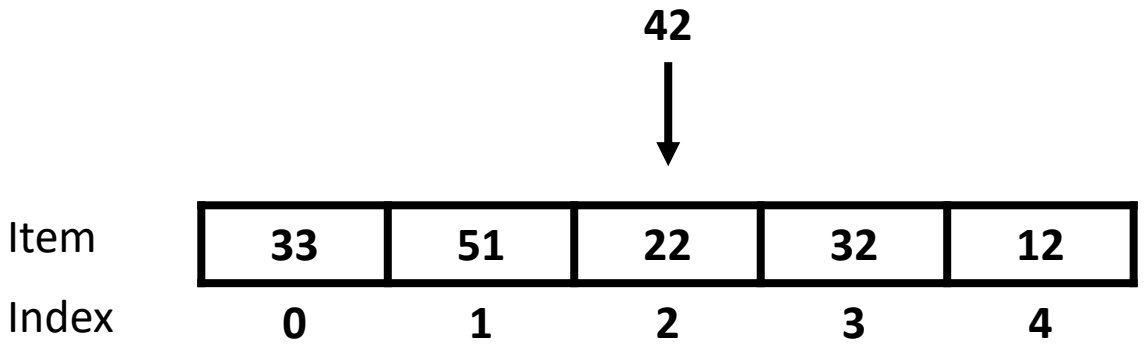




Insert when $n + 1 > U$ (array full)

- Resize and Forward right copy. We could use a backward right copy as well.

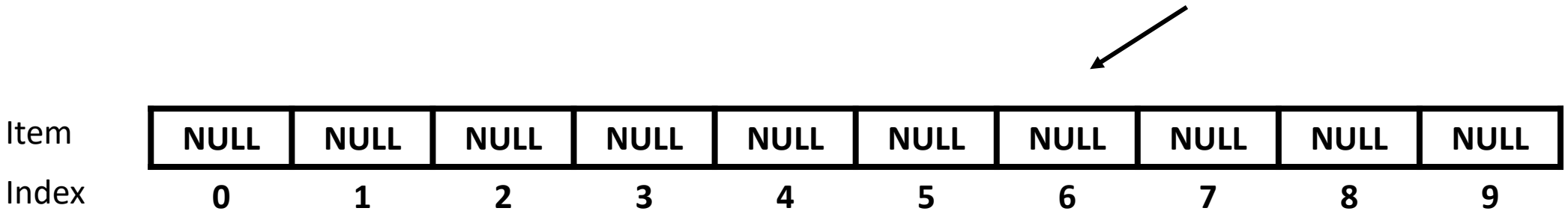




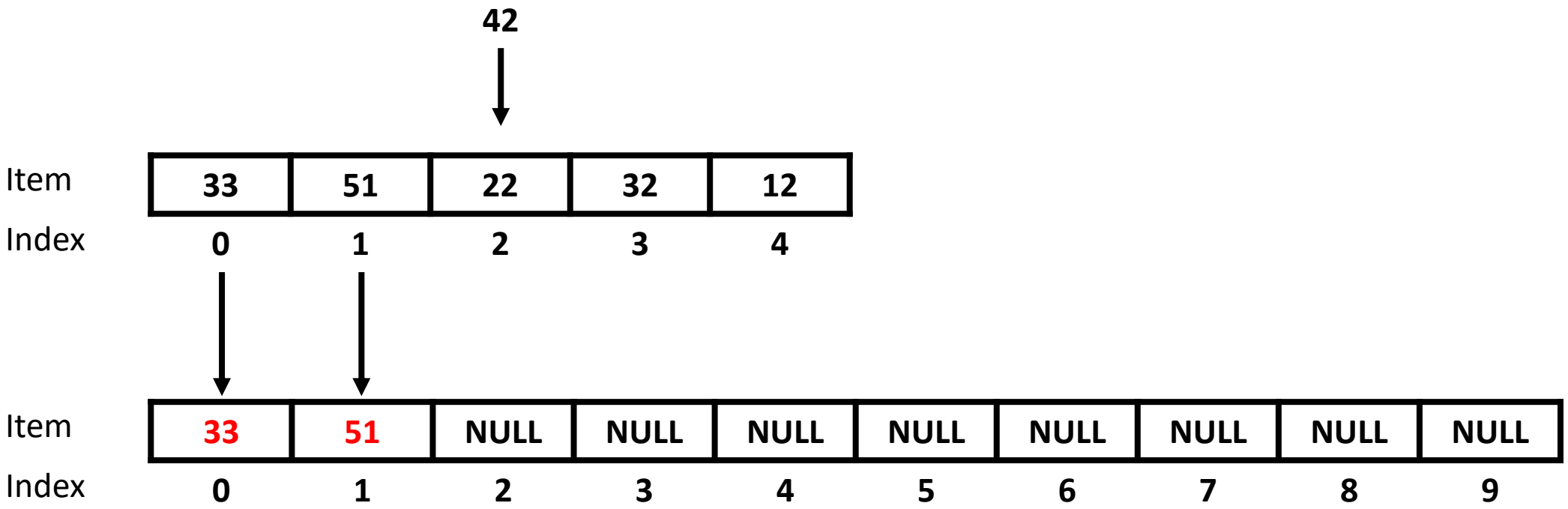
We assume this is constant time as we are just asking for a chunk of memory.

At a low level this will be dependent on the OS and programming language.

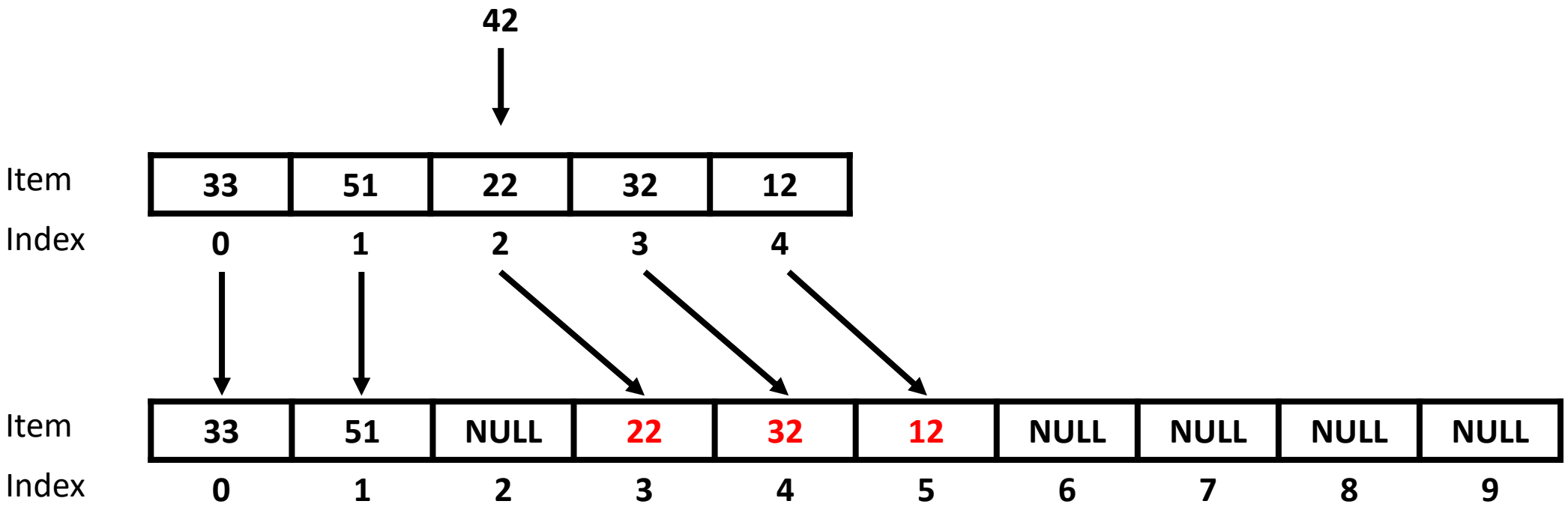
Our assumption is we ignore it which is fine for our analysis.



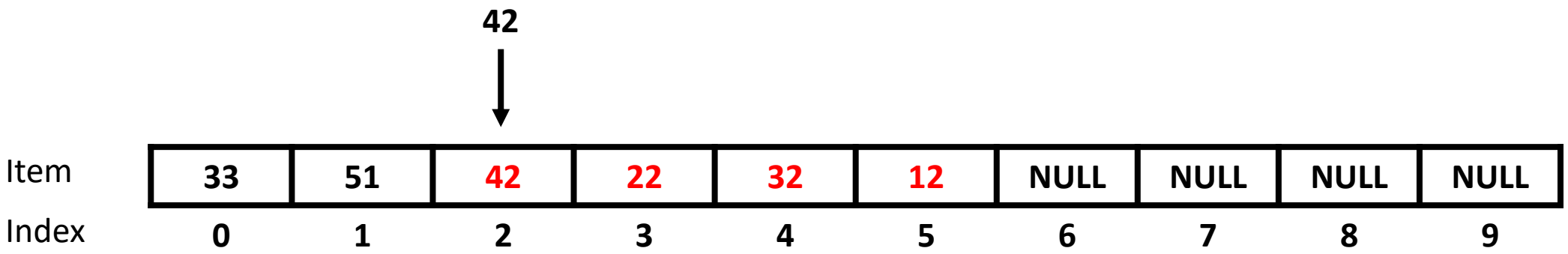
- Copy across before insert

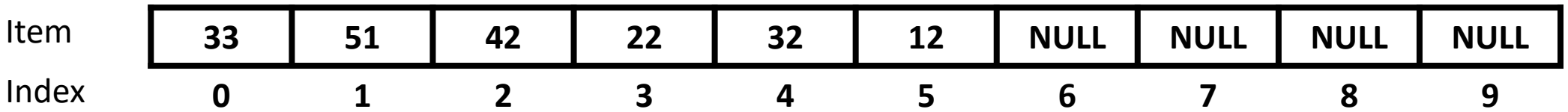


- Copy across after insert



- Perform Insert

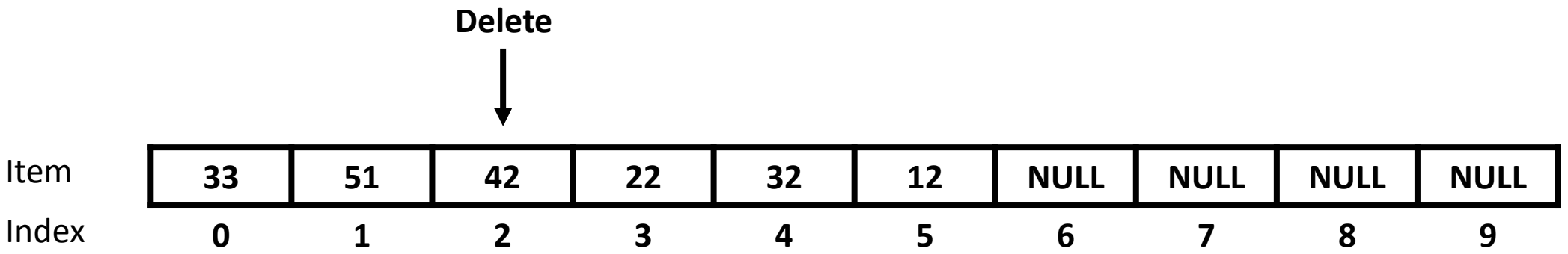




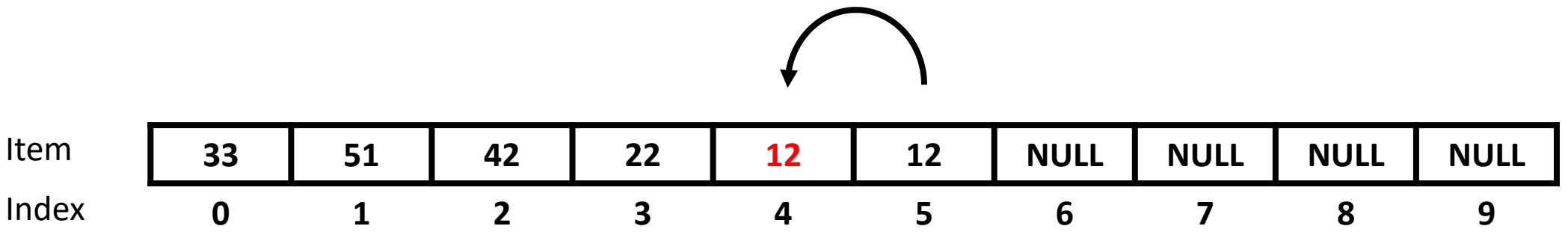
Delete when $n - 1 > L$ (above threshold)

Wrong Way

- Delete, backward left copy. Threshold might be 4 items. 6-1 = 5 > 4



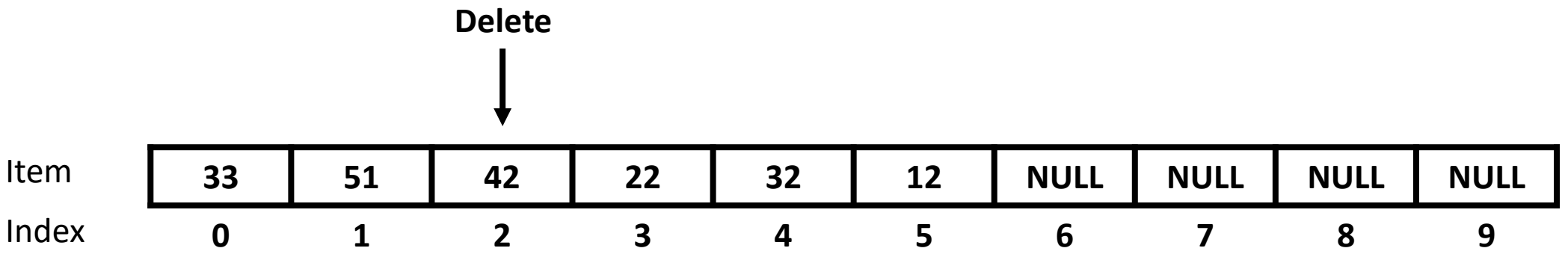
- Delete, Backward left copy. Threshold might be 4 items. 6-1 = 5 > 4



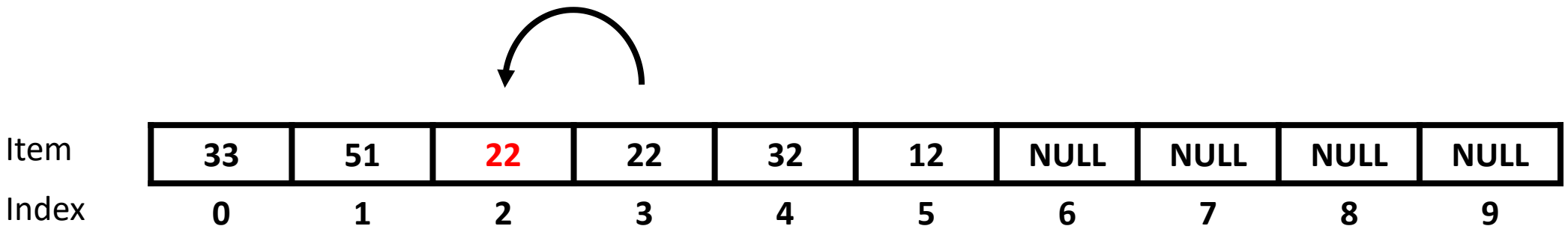
We have just written over 32. Oops!

Right Way

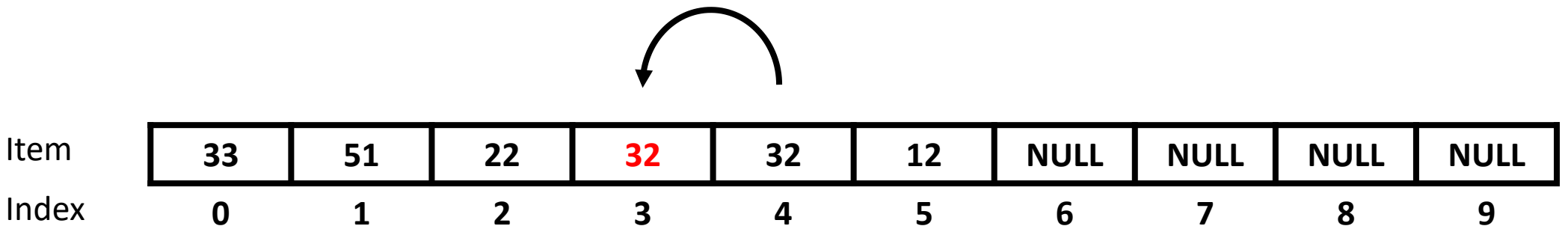
- Delete, Forward left copy. Threshold might be 4 items. 6-1 = 5 > 4



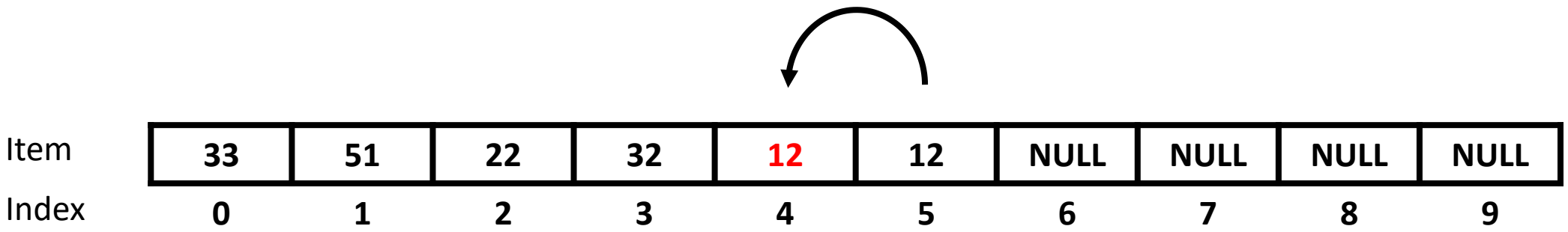
- Delete, Forward left copy. Threshold might be 4 items. 6-1 = 5 > 4



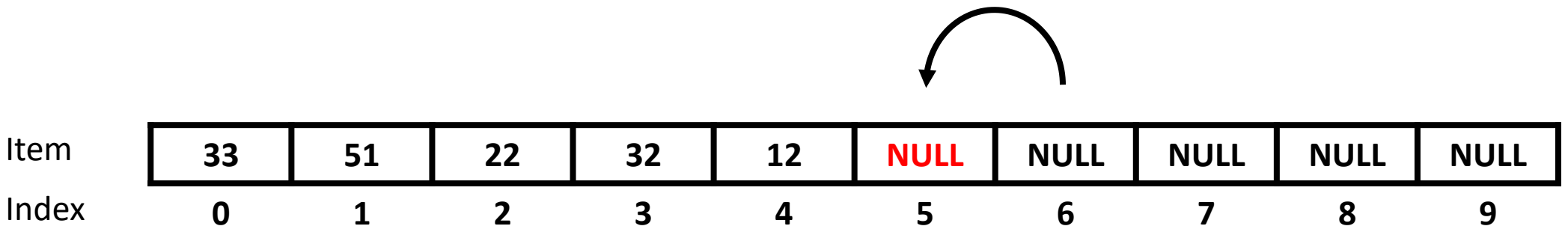
- Delete, Forward left copy. Threshold might be 4 items. 6-1 = 5 > 4

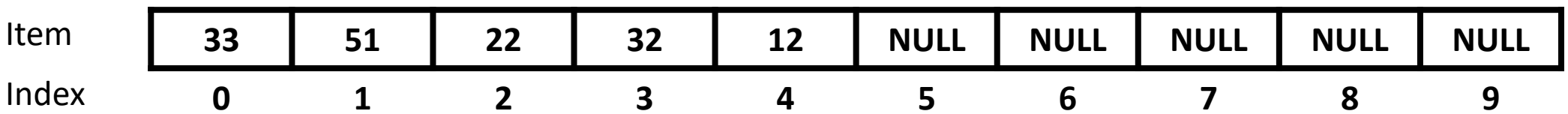


- Delete, Forward left copy. Threshold might be 4 items. 6-1 = 5 > 4



- Delete, Forward left copy. Threshold might be 4 items. 6-1 = 5 > 4





Delete when $n - 1 \leq L$ (at threshold)

- Resize, Forward left copy, $n - 1 = 4 - 1 = 3 \leq 3$. Here our threshold is $L = 3$