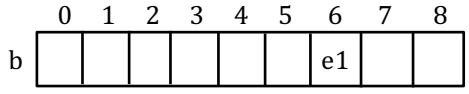
Hashing with linear probing (part 1)

The main advantage of hashing with linear probing instead of linked lists is a large reduction in space requirements. There are no linked lists; instead the elements of the set are kept directly in an array b. We show the array for an empty set —empty array elements are assumed to contain **null**.

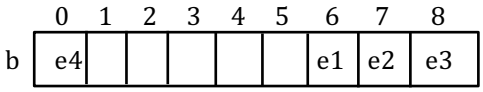
As usual, an element e1 is hashed to a bucket, and if the bucket contains **null**, e1 is placed there.



(assume e1 hashes to 6)

Handling collisions

If another element e2 hashes to an occupied bucket, a "collision" is said to have occurred, and e2 must be placed elsewhere. In linear probing, it is placed in the next bucket (with wraparound) that is null. We show the placement of two more values that hash to bucket 6 or 7.



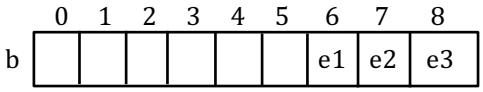
(assume e2, e3, e4 hash to 6 or 7)

In summary: If an element hashes to bucket h, the element is placed in the first of the following buckets that contains null.

h, (h+1) % b. length, (h+2) % b. length, (h+3) % b. length, …

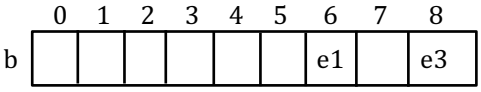
Searching for a value

Searching for an element involves hashing it to a bucket and looking at that bucket and the following buckets in wraparound fashion until either the element is found or **null** is found, in which case the element is not in the set. Suppose e1, e2, and e3 all hashed to 6 and were added as shown. A test of whether e3 is in the set is successful: Linear probing looks in buckets 6, 7, and 8, and e3 is there. A test whether e4 is in the set is not successful.

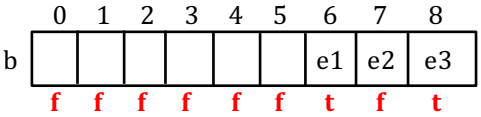


Operation remove

Suppose e2 is removed. Now, a test of whether e3 is in the set is unsuccessful, because linear probing stops as soon as a null bucket is found.



Clearly, removing e2 cannot be done simply by setting the bucket to ***null***. Instead, we use a boolean value for each bucket to indicate whether it contains a value. Removing e2 then consists only in setting its boolean to **false**.



Now, testing for the presence of a value in the set consists of testing not only the bucket but the boolean that indicates whether the value is in the set. We give an example. Again, suppose e1, e2, and e3 all hash to 6 and were added as shown. Let's remove e2, leaving the bucket alone but setting the boolean to **false** (as shown above). Now, a test whether e3 is in the set still says yes because the search stops with a negative answer only when a null bucket is encountered.