Solution

(i) An unsorted array

$T(n) = O(n^2)$

- O(n) iterations through the outer loop (array A)
- O(n) iterations for each check for duplicates / add element

(ii) A sorted linked list: Same answer as (i)

Note: If we consider a sorted array, then T(n) = O(nlogn)

- O(n) iterations through the outer loop (array A)
- O(logn) iterations for each check for duplicates / add element

(iii) A hashmap

T(n) = O(n)

- O(n) iterations through the outer loop (array A)
- O(1) iterations for each check for duplicates / add element

Hashmap is the fastest

Possible Pseudocode

```
item firstDuplicate(item[] A)
{
Map m = new HashMap
for (item i : A)
  if (m.contains(i))  // in this case i is the key and value
    return i;
  else
    m.put(i);
return null;
}
```