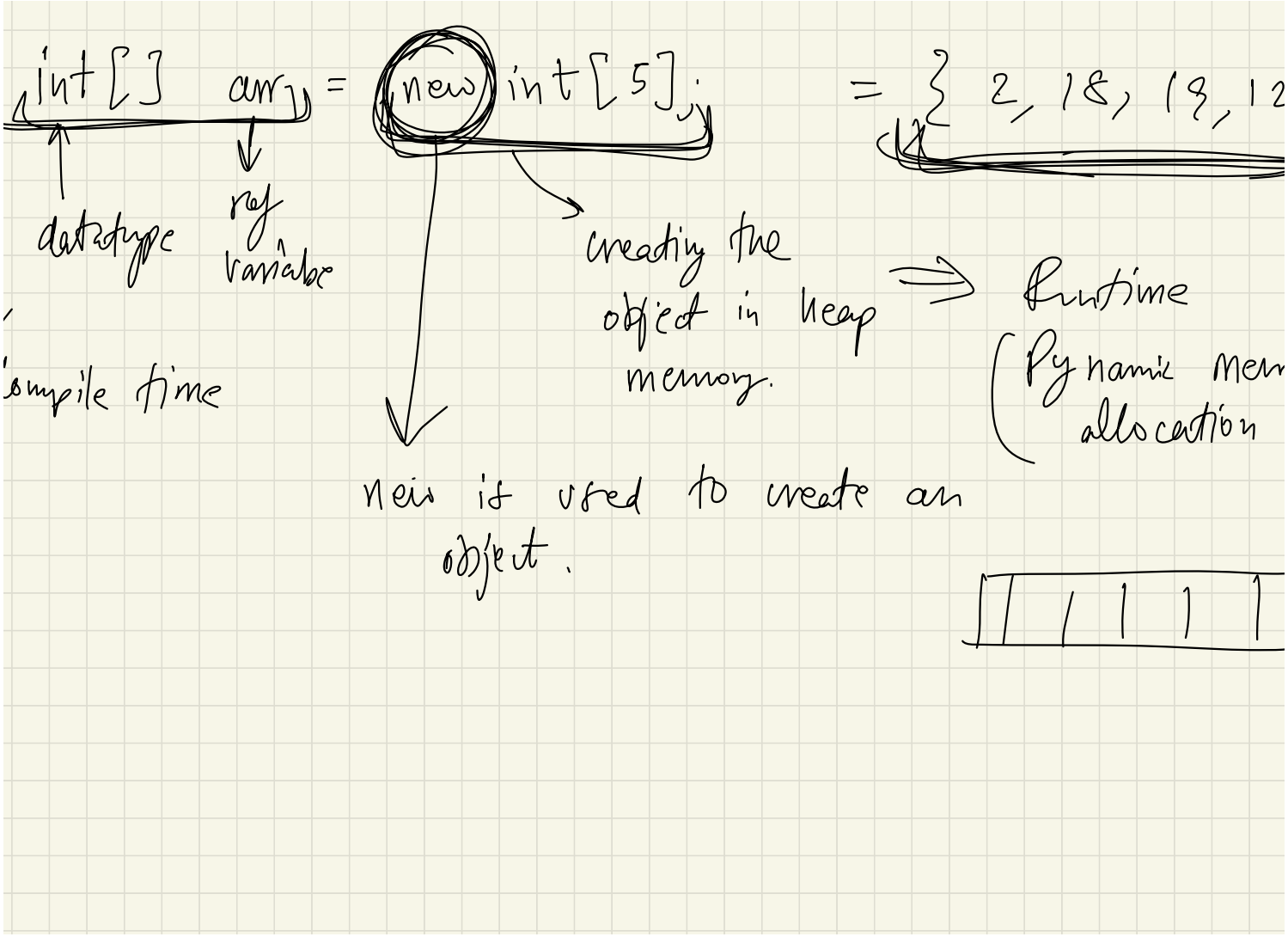
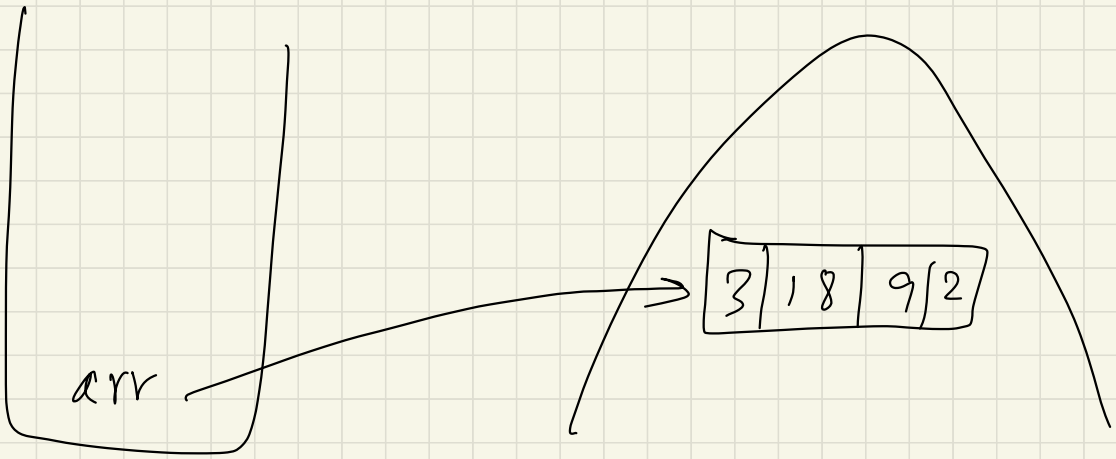

int[] arr = new int[5];
= { 2, 18, 19, 12
datatype
ref variable
creating the object in heap memory.
⇒ Runtime
(Dynamic mem allocation
ompile time
new is used to create an object.



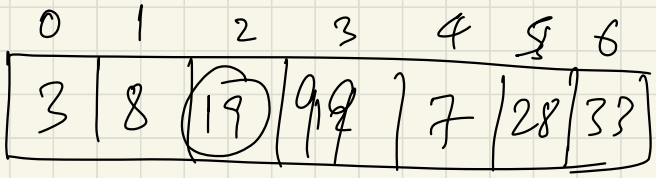
Stack Heap

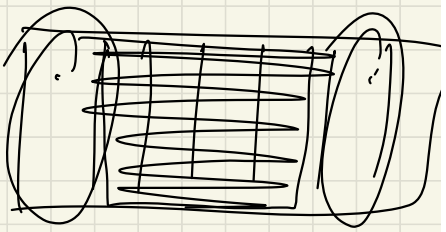
① array objects are in heap

② heap objects are, not continuous

③ DMA

Leave! may not be continuous → Depends on JVM





print (arr [0]) ⟹ 3

arr [2] ⟹ 19

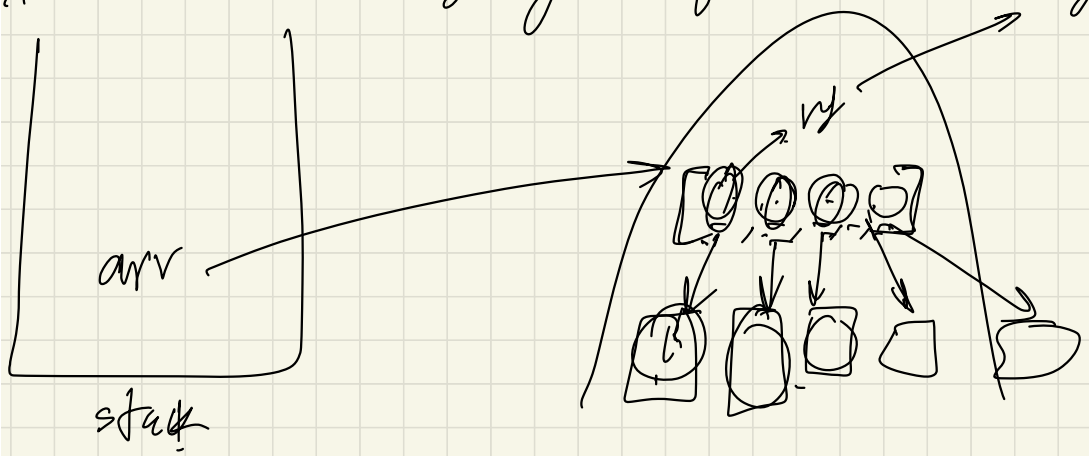
arr [3] = 99

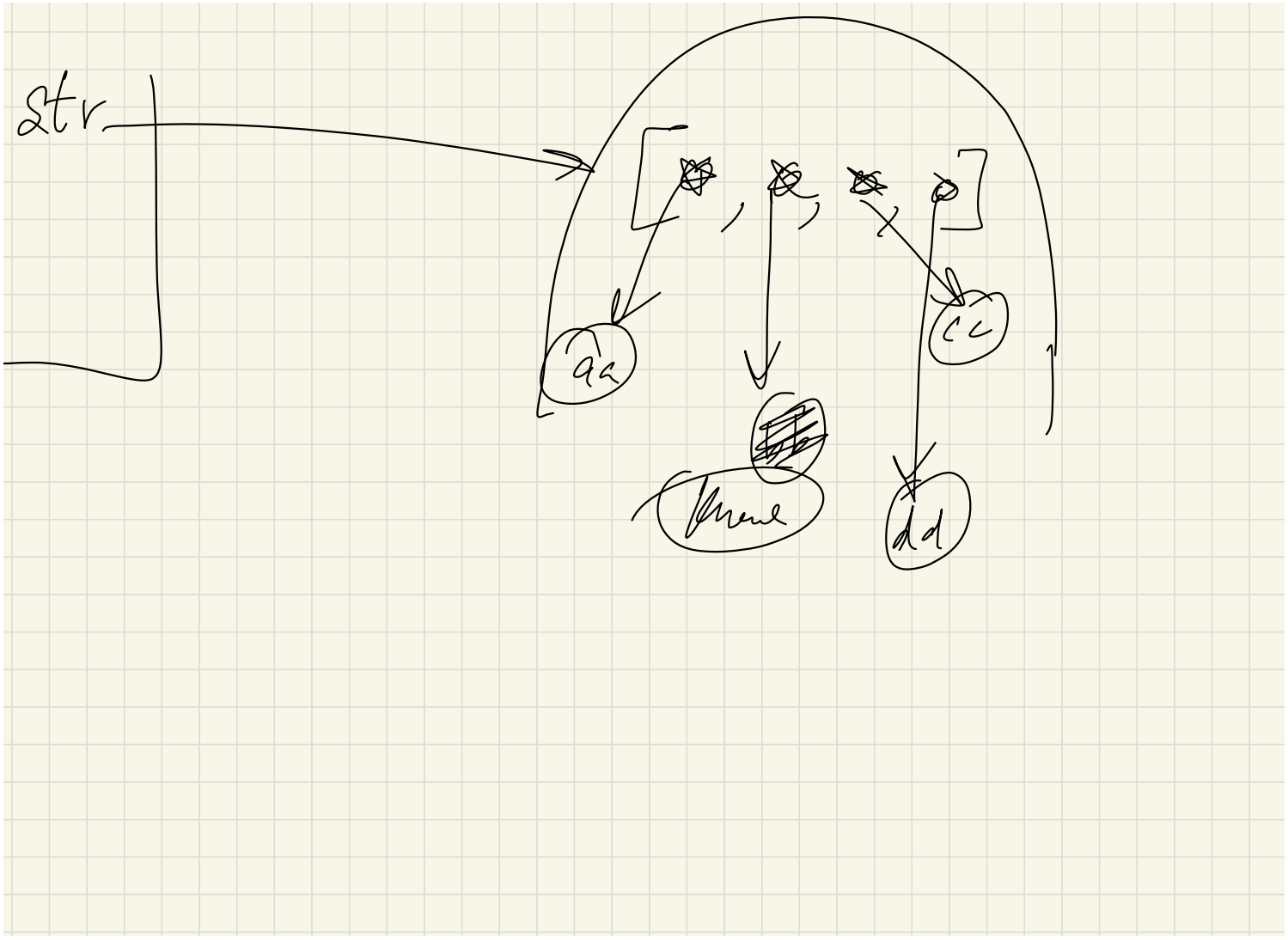
String [] arr = new String [5];

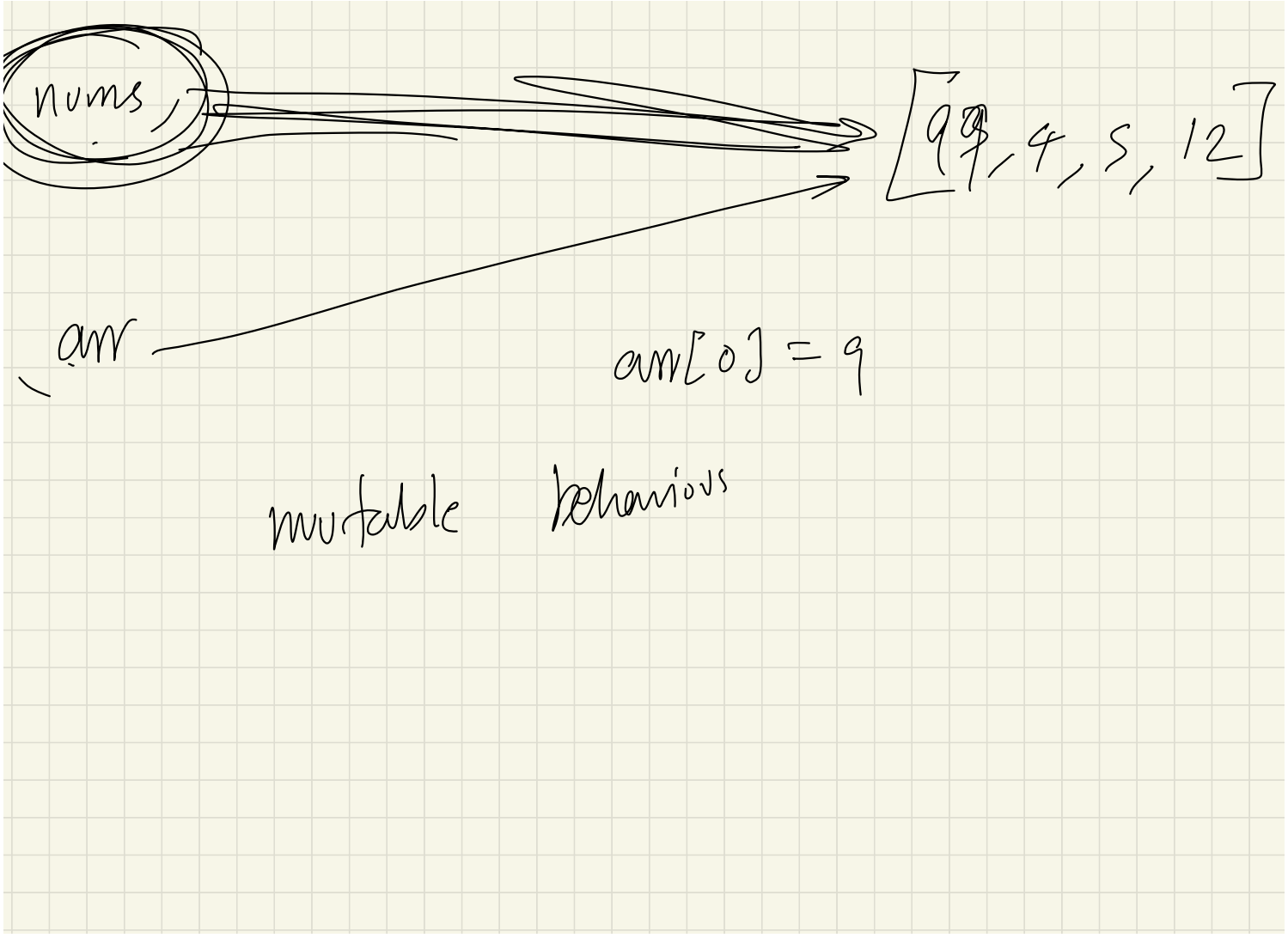
arr[0] = "3"

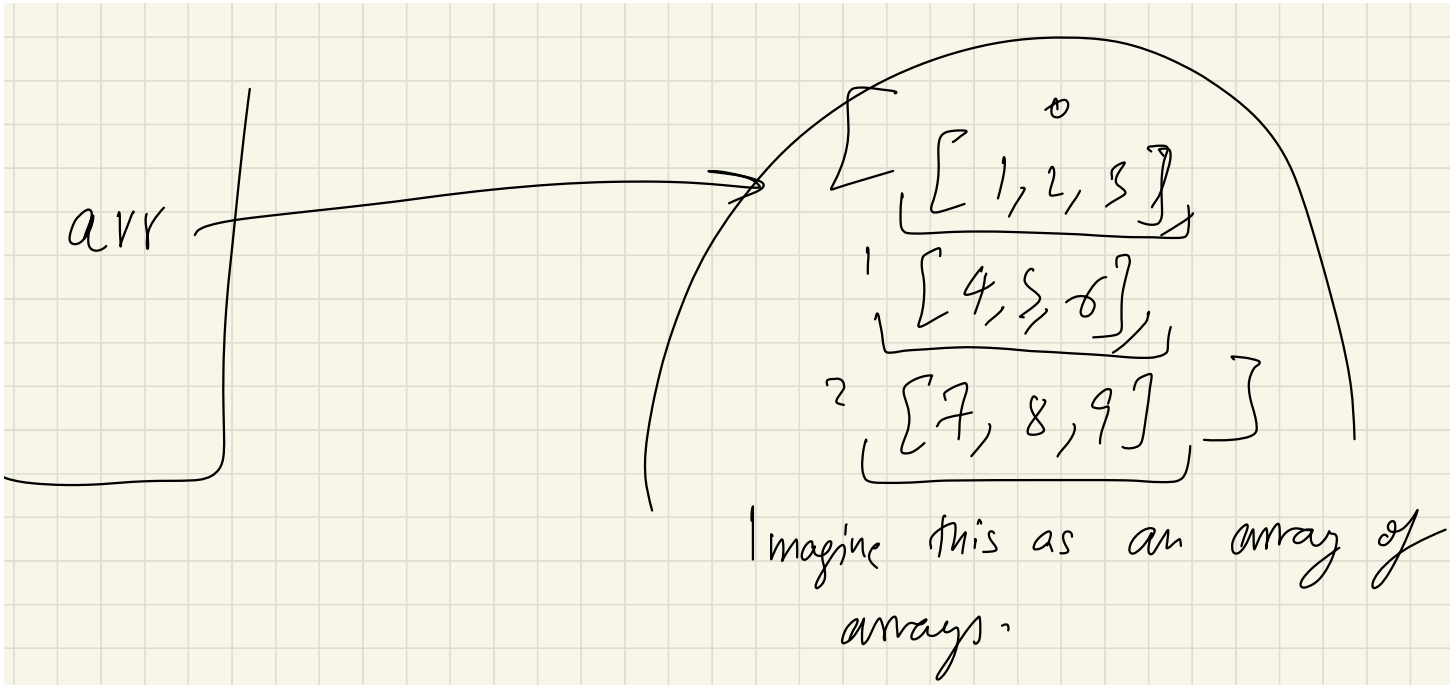
// internal working of object

arr[0] = null

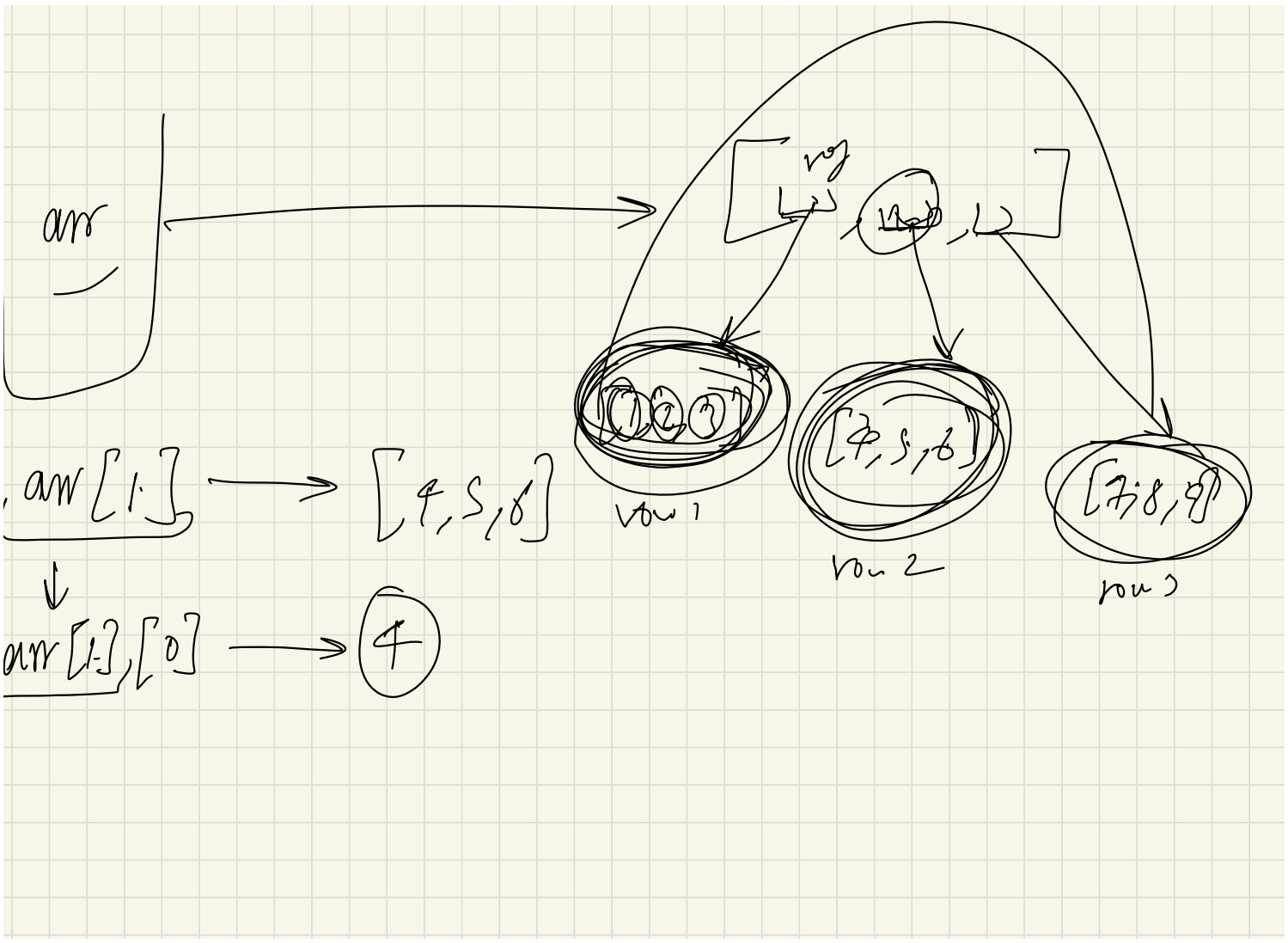


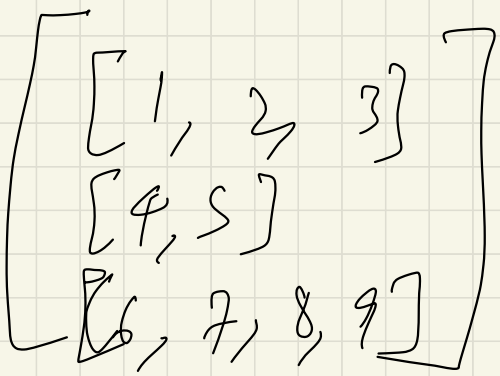

str
aa
cc
dd

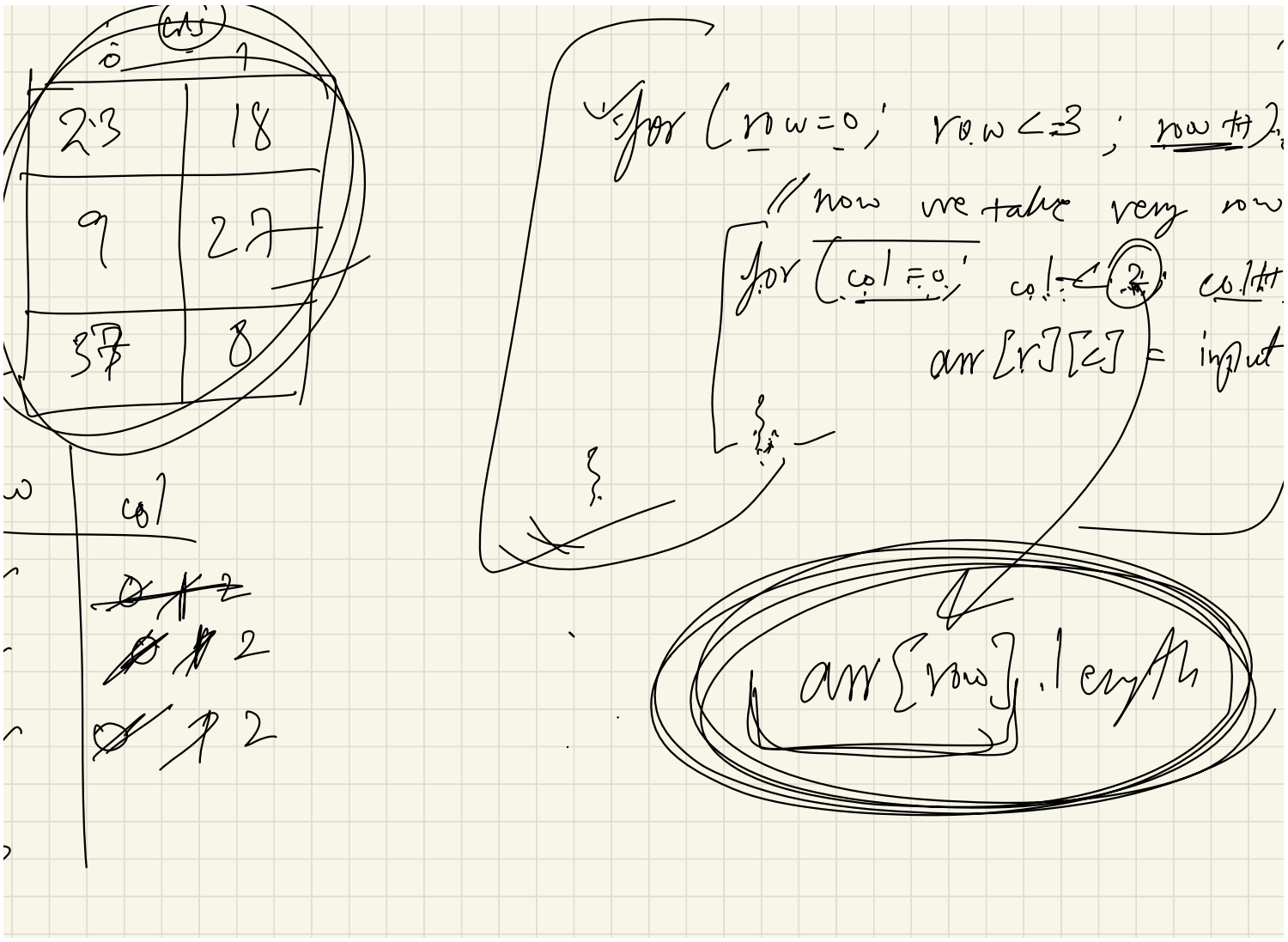

nums
[9, 4, 5, 12]
arr
arr[0] = 9
mutable behaviors

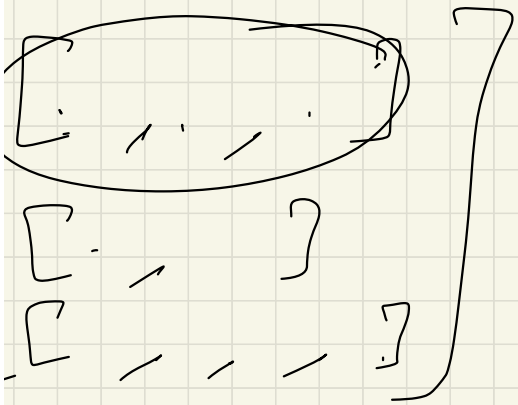


Imagine this as an array of arrays.


arr
arr[1]
[4,5,6]
arr[1][0]
4
row 1
row 2
row 3


[1, 2, 3]
[4, 5]
[6, 7, 8, 9]


0
1
23
18
9
27
3
8
col
0
1
2
0
1
2
0
1
2
for (row=0; row<=3; row++)
// now we take very row
for (col=0; col<2; col++)
arr[r][c] = input
arr[row].length

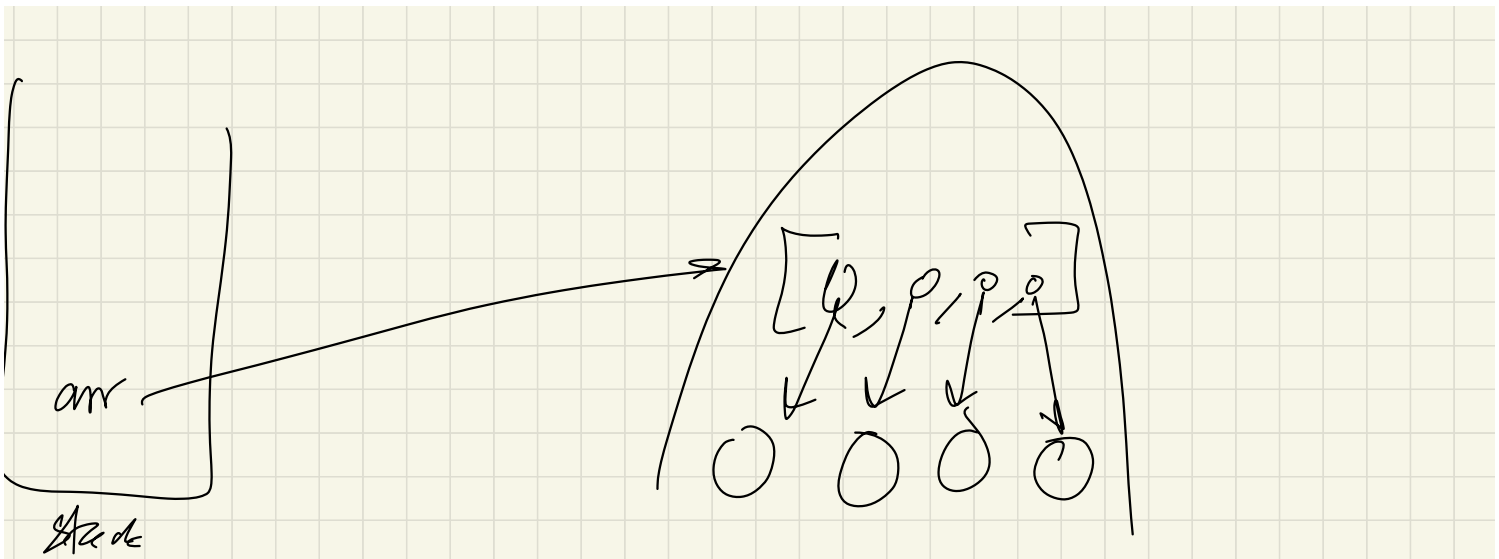


r	c
0	0
1	

```
for(r=0; r<3; r
    for(c=0; c<size
        c++)
            input
    }
}
```

size(arr[0])

arr[1]


arr

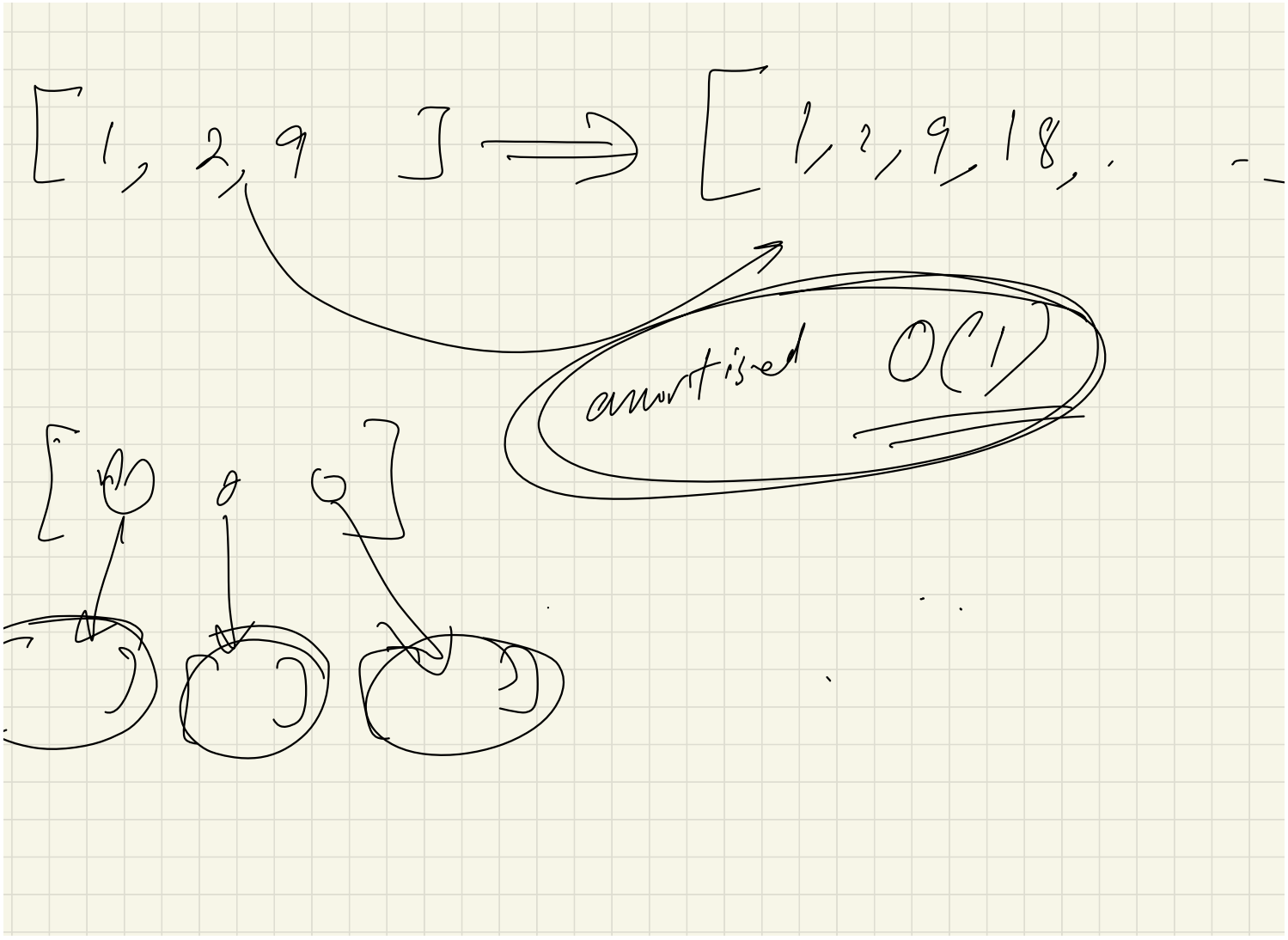
) Size is fixed Internally

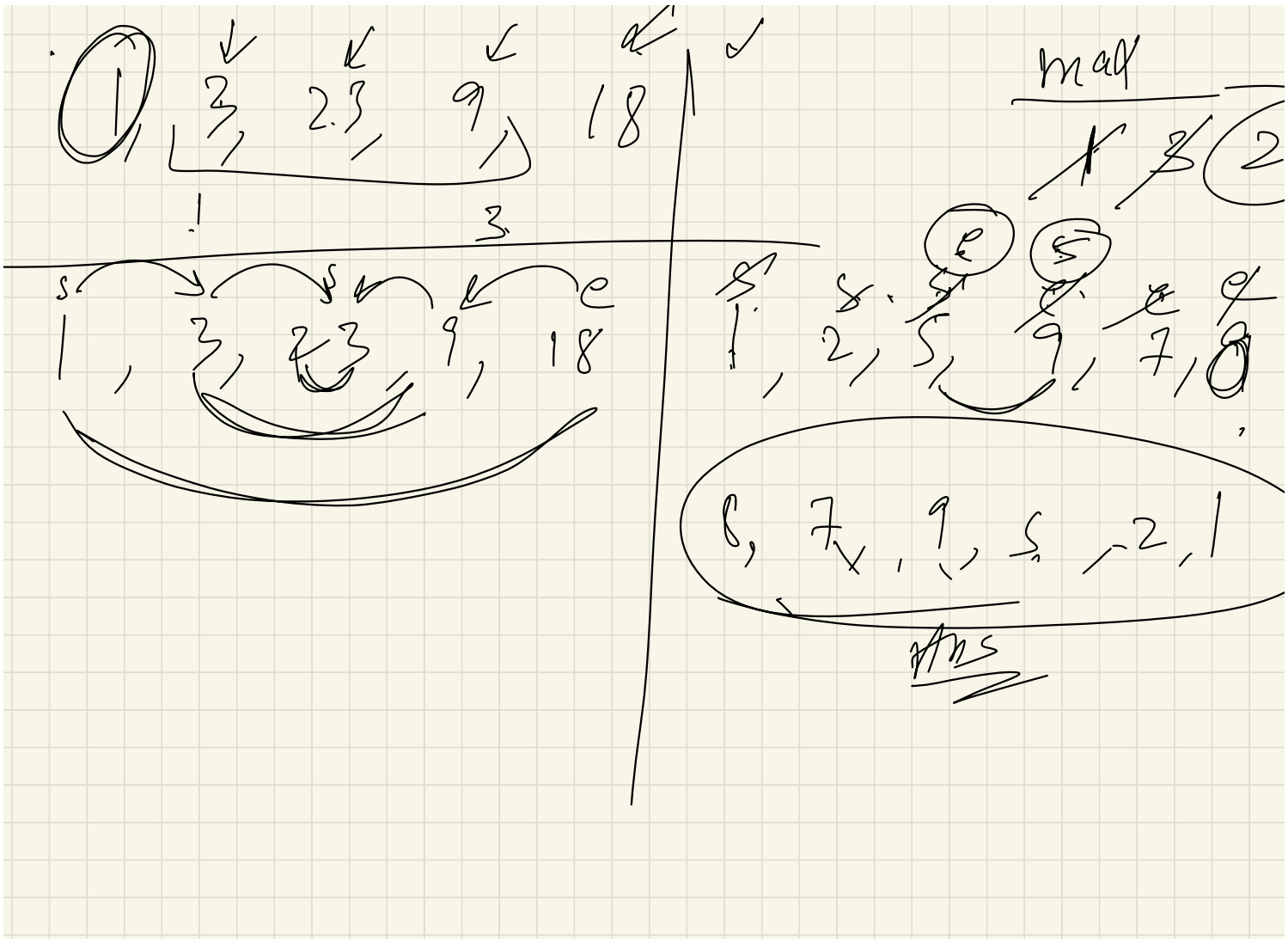
) Say arraylist fills by some amount

⇒ It will a new arraylist of say, double the

⇒ old elements are copied in new one

⇒ old one is deleted


[1, 2, 9] ⟹ [1, 2, 9, 18, . . .
amortised O(1)
[0 0 0]


1
3
2.3
9
18
max
2
s
e
1, 3, 2.3, 9, 18
1, 2, 5, 9, 7, 8
6, 7, 9, 5, 2, 1
Ans