Ques Ugly number

↳ factors are only 2, 3, 5

n^{th}

1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15

two

1 2 3

1st th

next mul 2 = ~~4~~ 4

next mul 3 = ~~6~~ 6

next mul 5 = 5

```java
public int nthUglyNumber(int n) {
    int ugly[] = new int[n];
    int idx2 = 0, idx3 = 0, idx5 = 0;
    int nextMultipleOf2 = 2;
    int nextMultipleOf3 = 3;
    int nextMultipleOf5 = 5;
    ugly[0] = 1;

    for(int i = 1; i < ugly.length; i++) {
        int nextUglyNo = Math.min(nextMultipleOf2, Math.min(nextMultipleOf3, nextMultipleOf5));

        ugly[i] = nextUglyNo;
        if(nextUglyNo == nextMultipleOf2) {
            nextMultipleOf2 = ugly[++idx2] * 2;
        }
        if(nextUglyNo == nextMultipleOf3) {
            nextMultipleOf3 = ugly[++idx3] * 3;
        }
        if(nextUglyNo == nextMultipleOf5) {
            nextMultipleOf5 = ugly[++idx5] * 5;
        }
    }
    return ugly[n-1];
}
```

TC → O(n) SC → O(n)

Ques K diff pair

3 1 4 1 5

K = 2

Brute force TC → O(n³) SC → O(n)

2 pointer ① Sort

1 1 3 4 5

l r

c = 0 → 1 → 2

K = 0

1 1 3 3 4 5

l r

c = 0 → 1 → 2

```java
public int findPairs(int[] nums, int k) {
    Arrays.sort(nums);
    int count = 0;
    int l = 0, r = 1;
    while(r < nums.length) {
        if(l == r) {
            r++;
            continue;
        }
        int diff = nums[r] - nums[l];
        if(diff == k) {
            count++;
            r++;
            l++;
            while(r < nums.length && nums[r-1] == nums[r])
                r++;
            while(l < r && nums[l-1] == nums[l])
                l++;
        }
        else if(diff < k) r++;
        else l++;
    }
    return count;

}
```

n log n

TC → O(n log n)

Ques 3 arrays A, B, C

minimize → (max(A[i], B[j], C[k]) − min(A[i], B[j], C[k]))

A	1	4	5	8	10	m
B	6	9	15			n
C	2	3	6	6		o

diff = ~~9~~ ~~4~~ ~~3~~ ~~2~~ 1

Brute force → 3 nested loop

TC → O(m × n × o)

Three pointers TC → O(m + n + o)