

HCF / GCD

In [388]:

```
1  
2  
3
```

In [500]:

```
1 a = 15  
2 b = 12  
3  
4 n = 2  
5  
6 def gcd(a,b):  
7  
8     for i in range(min(a,b),0,-1):  
9         if a%i == 0 and b%i == 0:  
10             return (i)  
11  
12 gcd(a,b)
```

Out[500]: 3

In [471]:

```
1 gcd(32,64)
```

Out[471]: 32

In [479]:

```
1 math.gcd(16,64,36)
```

Out[479]: 4

In [478]:

```
1 a = 16  
2 b = 64  
3 c = 36  
4  
5  
6 def gcd(a,b,c):  
7  
8     for i in range(min(a,b,c),0,-1):  
9         if a%i == 0 and b%i == 0 and c%i == 0:  
10             return (i)  
11  
12 gcd(a,b,c)
```

Out[478]: 4

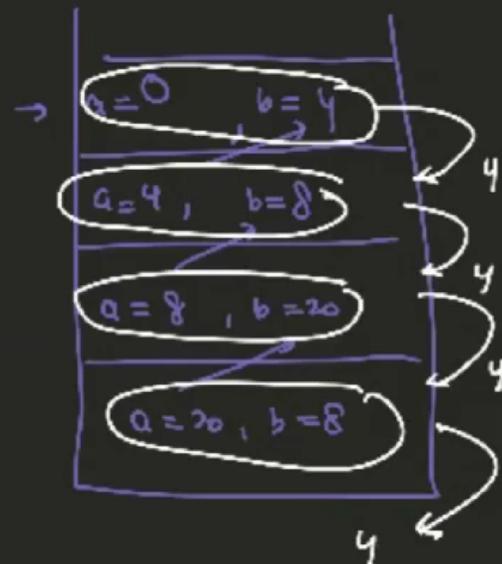
```

2 def gcd(a,b):
3     if a==0:
4         return b
5     return gcd(b%a,a)
6
7
8 print(gcd(11,19))

```

20, 8

8, 20,



In [481]:

```

1 def gcd(a,b):
2     if a == 0:
3         return b
4     return gcd(b%a,a)
5 gcd(50,64)

```

Out[481]: 2

In [489]:

```

1 A = [16,16,8,16]
2
3 def gcd_arr(A):
4     def gcd(a,b):
5         if a == 0:
6             return b
7         return gcd(b%a,a)
8     ans = gcd(A[0],A[1])
9     for i in range(2,len(A)):
10        ans = gcd(ans,A[i])
11    return ans
12
13 gcd_arr(A)

```

Out[489]: 8

In [492]:

```

1 math.gcd(16,16,8,16)
2
3

```

Out[492]: 8

In []:

1

```
In [494]: 1 def lcm(a,b):  
2     def gcd(a,b):  
3         if a == 0:  
4             return b  
5         return gcd(b%a,a)  
6     return (a*b)/gcd(a,b)  
7  
8  
9 lcm(15,12)
```

Out[494]: 60.0

```
In [504]: 1 a = 12  
2 b = 15  
3 m = 1  
4 while b*m % a != 0:  
5     m = m+1  
6 print(b*m)
```

60

In []: 1

In []: 1

In []: 1

isprime

```
In [25]: 1 def isprime(n):  
2     i = 1  
3     f = 1  
4  
5     while i*i <= n:  
6         if n % i == 0:  
7             f += 1  
8         i += 1  
9     return f == 2  
10
```

```
In [35]: 1 isprime(10**12+8)
```

Out[35]: False

```
In [498]: 1 def isprime2(n):  
2     i = 2  
3  
4     while i*i <= n:  
5         if n % i == 0:  
6             return False  
7         i+=1  
8     return True
```

```
In [56]: 1 isprime2(10**9+7)
```

Out[56]: True

```
In [499]: 1 isprime2(1)
```

```
Out[499]: True
```

```
In [ ]: 1
```

```
In [57]: 1 from math import *
```

```
In [59]: 1 sqrt(60)
```

```
Out[59]: 7.745966692414834
```

Q!

Q → given N find the square root.

49 is a perfect square because $7^*7 = 49$

29 is not a perfect square $i^*i = 29$

no such integers i

given a no N find $\lfloor \sqrt{N} \rfloor$ } floor
largest no

given a no N find $\lfloor \sqrt{N} \rfloor$ } floor
largest no
 \downarrow ignore the fractional part
round down.

$$N = 49 \rightarrow \text{sqrt}(49) = 7$$

$$\text{sqrt}(24) = 4$$

$$\text{sqrt}(25) = 5$$

$$\text{sqrt}(26) = 5$$

```
In [513]: 1 def perfect_square(n):
2     i = 1
3     for i in range(1,n+1):
4         if i*i == n:
5             return i
6
7 perfect_square(24)
```

```
In [506]: 1 def perfect_square(n):
2     i = 1
3     while i*i <= n:
4         i += 1
5     return i-1
```

```
In [507]: 1 perfect_square(17)
```

Out[507]: 4

```
def sqrt(n):
    i = 1
    while i * i <= n:
        i += 1
    return (i-1) ↪ ↪ ↪
```

$i=1$	$i^2 = 1 \leq 25$
2	$2^2 = 4 \leq 25$
3	$3^2 = 9 \leq 25$
4	$4^2 = 16 \leq 25$
5	$5^2 = 25 \leq 25$
6	$6^2 = 36 \not\leq 25$

$$m = 24$$

$i=1$	$i^2 = 1 \leq 24$
2	$4 \leq 24$
3	$9 \leq 24$
4	$16 \leq 24$
5	$25 \not\leq 24$

```
In [ ]: 1
```

```
In [161]: 1 def perfect_square(n):
2
3     low = 1
4     high = n
5     mid = (low + high)//2
6
7     while low <= high:
8         mid = (low + high)//2
9         if mid * mid > n:
10            high = mid - 1
11
12        elif mid * mid < n:
13            low = mid + 1
14        else:
15            return mid
16
17
18
19 perfect_square(26)
20
```

Out[161]: 5

In []:

1

$O(N) \rightarrow 217 \text{ years}$

$O(\sqrt{n}) \rightarrow 10 \text{ s}$

$O(\log_2 n) \rightarrow 0.6 \mu\text{s}$

In []:

1

In []:

1

$$b^a = y \xrightarrow{\log_b} \cancel{\log_b}(b^a) = \log_b(y)$$

$$\Rightarrow a = \log_b(y)$$

$$\log_b(b^x) = x$$

\log a power can increase
(if the base is same)

$$\log_b y = x \Rightarrow b^x = y$$

$$C = b^{\log_b C}$$

$$b^x = c^{x(\log_b C)}$$

$$5^{\log_2 3}$$

$$2^{\log_2 3}$$

$$\lg_3 1000 = \left(\lg_3 7 \right) \lg_7 1000$$

$$\lg_c a = \lg_b a * \lg_c b$$

In []:

1

In []:

1

$$2^{46} = 10^{\frac{46}{\lg 2}} = 10^{\frac{46}{2.322}} = 14$$

$$14+1 = 15 \text{ digits}$$

In []:

1

$$\lg(n!) = O(n \lg n) \Rightarrow \text{Stirling's approximation}$$

In []:

1

In []:

1

Identities

$$AP = a, a+d, a+2d, \dots, a+(n-1)d$$

$$S_m^{AP} = (\text{first} + \text{last}) \frac{n}{2} = \frac{n}{2} (2a + (n-1)d)$$

$$GP = a, ar, ar^2, \dots, ar^{n-1}$$

$$S_m^{GP} = a \cdot \frac{r^n - 1}{r - 1}$$

$$S_\infty^{GP} = a \cdot \left(\frac{1-r^n}{1-r} \right)$$

$r < 1$

In []:

1

$$1+2+3+\dots+n = \frac{n(n+1)}{2}$$

$$1^2 + 2^2 + 3^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6} = n \frac{(n+1)(2n+1)}{6}$$

$$1^3 + 2^3 + 3^3 + \dots + n^3 = \left(\frac{n(n+1)}{2} \right)^2 = (1+2+3+\dots+n)^2$$

$$1^k + 2^k + \dots + n^k = O\left(n^{k+1}\right)$$

In []:

1

In []:

1

In []:

1

append / prepend

insertion at end $O(1)$ in appeding

insertion at start position O(n) prepending

```
In [106]: 1 sqr = []
2
3 for i in range(1,11):
4     sqr.append(i*i)
5 print(sqr)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
In [110]: 1 sqr = []
2 index = []
3 idx = 0
4 for i in range(1,11):
5     index.append(idx)
6     sqr.insert(0,i*i)
7     idx += 1
8 print(sqr)
9 print(index)
```

```
[100, 81, 64, 49, 36, 25, 16, 9, 4, 1]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [108]: 1 sqr
```

```
Out[108]: [100, 81, 64, 49, 36, 25, 16, 9, 4, 1]
```

```
In [109]: 1 len(sqr)
```

```
Out[109]: 10
```

```
In [ ]: 1
```

```
In [111]: 1 n = len(sqr)
```

```
In [114]: 1 summation = 0
2
3 for i in range(n):
4     print(summation, " ",sqr[i], " ",i)
5     summation += sqr[i]
6
7 print(summation)
```

```
0   100   0
100  81    1
181  64    2
245  49    3
294  36    4
330  25    5
355  16    6
371  9     7
380  4     8
384  1     9
385
```

```
In [ ]: 1
```

Q!

Given n elements in a list.
Count the no of elements which have
at least 1 element *greater than itself*
present in the list

$[1, 3, 2, 6, 8, 9, 0, -30, -20, 5]$

```
In [117]: 1 A = [2,5,1,4,8,0,8,1,3,8]
2 n = len(A)
3
4 maxn = -float('inf')
5
6 for x in A:
7     if x > maxn:
8         maxn = x
9
10
11 ans = 0
12 for i in range(n):
13     if A[i] < maxn:
14         ans += 1
15 print(ans)
16
17
```

7

```
In [118]: 1 A = [2,5,1,4,8,0,8,1,3,8]
2 n = len(A)
3 c = 0
4 for v in A:
5     haslarger = False
6     for x in A:
7         if x > v:
8             haslarger = True
9     if haslarger :
10        c += 1
11 print(c)
12
13
```

7

```
In [120]: 1 def greaterthanitself(A):
2     n = len(A)
3
4     maxn = -float('inf')
5
6     for x in A:
7         if x > maxn:
8             maxn = x
9
10    c = 0
11    for v in A:
12        if v == maxn:
13            c += 1
14
15    return n - c
16 A = [2,5,1,4,8,0,8,1,3,8]
17 greaterthanitself(A)
```

Out[120]: 7

In []: 1

In []: 1

reverse the list

```
In [121]: 1
2 def revlist(A):
3     rev = []
4     n = len(A)
5     for i in range(n-1, -1, -1):
6         rev.append(A[i])
7     return rev
```

```
In [123]: 1 A = [1,10,3,6,8]
2 revlist(A)
```

Out[123]: [8, 6, 3, 10, 1]

In []: 1

```
In [134]: 1
2 def revlist(A):
3     n = len(A)
4
5     for i in range(n//2):
6         A[i], A[n-i-1] = A[n-i-1], A[i]
7     return A
8
9 A = [1,10,3,6,8,-99]
10 print(A)
11 ans = revlist(A)
12 print(ans)
```

[1, 10, 3, 6, 8, -99]
[-99, 8, 6, 3, 10, 1]

```
In [ ]: 1
```

```
In [ ]: 1
```

Problem Description

You are given **N** positive integers.

For each given integer **A**, you have to tell whether it is a **perfect number** or not.

Perfect number is a positive integer which is equal to the sum of its proper positive divisors.

```
In [139]: 1 def divisors(n):
2     ans = []
3     i = 1
4     while i < n:
5         if n % i == 0:
6             ans.append(i)
7         i += 1
8     return ans
9 divisors(6)
10
```

```
Out[139]: [1, 2, 3]
```

```
In [147]: 1 def isperfect(n):
2
3     ans = []
4     i = 1
5     sm = 0
6     while i < n:
7         if n % i == 0:
8             sm += i
9         i += 1
10
11     if sm == n:
12         return "perfect number"
13     return False
14
15 isperfect(6)
16
```

```
6
```

```
Out[147]: 'perfect number'
```

```
In [ ]: 1
```

```
In [ ]: 1
```

In []:

1

Given a number **A**. Return square root of the number if it is perfect square otherwise return -1.

In [515]:

```
1 def perfect_square(n):
2
3     low = 1
4     high = n
5     mid = (low + high)//2
6
7     while low <= high:
8         mid = (low + high)//2
9         if mid * mid > n:
10            high = mid - 1
11
12        elif mid * mid < n:
13            low = mid + 1
14        else:
15            return mid
16    return None
17
18 def sqrt_num_if_perfect_sq(n):
19     if perfect_square(n) != None:
20         return perfect_square(n)
21     return -1
22
23 sqrt_num_if_perfect_sq(25)
```

Out[515]: 5

In []:

1

Problem Description

You are given an integer **N** you need to print all the **Armstrong Numbers** between **1** to **N**.

If sum of cubes of each digit of the number is equal to the number itself, then the number is called an Armstrong number.

For example, $153 = (1 * 1 * 1) + (5 * 5 * 5) + (3 * 3 * 3)$.

In [174]:

```
1 def isarmstrong(n):
2     num = n
3     total = 0
4     while n > 0:
5         digit = n % 10
6         total += digit ** len(str(num))
7         n = n//10
8     return total == num
9
10 print(isarmstrong(407))
11
12 for i in range(1,10000):
13     if isarmstrong(i):
14         print(i)
```

True

```
1
2
3
4
5
6
7
8
9
153
370
371
407
1634
8208
9474
```

In []:

```
1
```

In []:

```
1
```

Problem Description

Given an integer array **A** and an integer **B**, you have to print the same array after rotating it **B** times towards the right.

NOTE: You can use extra memory.

```
[1,2,3,4] => [4,1,2,3] => [3,4,1,2]
```

```
In [189]: 1 A = [1,2,3,4]
2 B = 1560405406878
3 itr = B % len(A)
4 for i in range(itr):
5     A.insert(0,A[-1])
6     A.pop()
7 A
```

Out[189]: [3, 4, 1, 2]

```
In [187]: 1 1560405406878%4
```

Out[187]: 2

Little Ponny is given an array, **A**, of **N** integers. In a particular operation, he can set any element of the array equal to **-1**.

He wants your help in finding out the minimum number of operations required such that the maximum element of the resulting array is **B**. If it is not possible, then return **-1**.

```
In [192]: 1 A = [2,4,3,1,5]
2 B = 3
3 def ponny_and_max_ele(A,B):
4     c = 0
5     if B not in A:
6         return -1
7     for i in A:
8         if i > B:
9             c += 1
10    return(c)
11
12 ponny_and_max_ele(A,B)
```

Out[192]: 2

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

>> Prefix Array

Given an Array of n integers

we will get multiple Queries $\rightarrow Q$

↳ In each Query you will be given a start index & an end index
Start = 4 end = 10

print the sum of $a[\text{start}] + \dots + a[\text{end}]$

In [227]:

```
1
2
3 def getsm(A,start,end):
4
5     n = len(A)
6
7     prefixsm = [A[0]]
8     for x in A[1:]:
9         prefixsm.append(x+prefixsm[-1])
10
11    if start == 0:
12        return prefixsm[end]
13    else:
14        return prefixsm[end]-prefixsm[start-1]
15
16
17
18
19 def query_sm(A,Q):
20
21     while Q > 0:
22         Q -= 1
23         start = int(input("Start Index : "))
24         end = int(input("End Index : "))
25         if start < 0 or start > len(A) or end > len(A)-1 or start >= end:
26             print("Invalid Input")
27
28         print("Ans: ",getsm(A,start,end))
29
30
31 Q = int(input("Number of queries : "))
32 A = list(map(int,input("Enter Array : ").split()))
33 query_sm(A,Q)
```

Number of queries : 2

Enter Array : 1 1 3 4 5 8 5 8 2 1 4 5

Start Index : 1

End Index : 5

Ans: 21

Start Index : 2

End Index : 3

Ans: 7

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

equilibrium index in the array:

You are given an array **A** of integers of size **N**.

Your task is to find the equilibrium index of the given array

The equilibrium index of an array is an index such that the sum of elements at lower indexes is equal to the sum of elements at higher indexes.

NOTE:

Array indexing starts from 0.

If there is no equilibrium index then return -1.

If there are more than one equilibrium indexes then return the minimum index.

```
In [262]: 1
2
3 def equilibrium_index(A):
4
5     n = len(A)
6
7     prefix = [A[0]]
8     for x in A[1:]:
9         prefix.append(x+prefix[-1])
10
11    ans = []
12    for i in range(n):
13
14        if i == 0:
15            leftsum = 0
16        else:
17            leftsum = prefix[i-1]
18
19        rightsum = prefix[n-1]-prefix[i]
20
21        if leftsum == rightsum:
22            ans.append(i)
23    if len(ans) == 0:
24        return "NO Equilibrium Index"
25    return ans
26
27 A = [-7, 1, 5, 2, -4, 3, 0]
28 equilibrium_index(A)
```

Out[262]: [3, 6]

```
In [263]: 1 def equilibrium_index_optim(A):
2     n = len(A)
3     ans = []
4     total = 0
5     for x in A:
6         total += x
7     left = 0
8     for i in range(n):
9
10        right = total - left - A[i]
11
12        if left == right:
13            ans.append(i)
14            left += A[i]
15
16    return ans
17
18 A = [-7, 1, 5, 2, -4, 3, 0]
19 equilibrium_index_optim(A)
```

Out[263]: [3, 6]

In []: 1

Problem Description

Given an array, **arr[]** of size **N**, the task is to find the count of array indices such that removing an element from these indices makes the sum of even-indexed and odd-indexed array elements equal.

```
In [279]: 1 def odd_even_sum(A):
2
3     n = len(A)
4     oddprefix = []
5     odds_m = 0
6     for i in range(n):
7         if i % 2 != 0:
8             odds_m += A[i]
9             oddprefix.append(odds_m)
10
11    evenprefix = []
12    evens_m = 0
13    for i in range(n):
14        if i % 2 == 0:
15            evens_m += A[i]
16            evenprefix.append(evens_m)
17    print(oddprefix)
18    print(evenprefix)
19    c = 0
20    for i in range(n):
21        oddsumm = oddprefix[i-1] + evenprefix[n-1]-evenprefix[i]
22        evensumm = evenprefix[i-1] + oddprefix[n-1]-oddprefix[i]
23
24        if oddsumm == evensumm:
25            c += 1
26    return c
27
28
29 A = [2, 1, 6, 4]
30 odd_even_sum(A)
```

```
[0, 1, 1, 5]
[2, 2, 8, 8]
```

Out[279]: 1

In []: 1

You are given an integer array **A** of size **N**.

You have to pick **B** elements in total. Some (possibly 0) elements from left end of array **A** and some (possibly 0) from the right end of array **A** to get the maximum sum.

Find and return this **maximum possible sum**.

NOTE: Suppose $B = 4$, and array A contains 10 elements, then

- You can pick the first four elements or can pick the last four elements, or can pick 1 from front and 3 from the back, etc. You need to return the maximum possible sum of elements you can pick.

pick from both sides

In [30]:

```
1 def pick_from_both_side(A,B):
2     n = len(A)
3     maxsum = -float('inf')
4     for L in range(B+1):
5         R = B - L
6
7         leftsum = 0
8         for i in range(L):
9             leftsum += A[i]
10
11        rightsum = 0
12        for i in range(n-R,n):
13            rightsum += A[i]
14
15        total = leftsum + rightsum
16        maxsum = max(total , maxsum)
17    return maxsum
18
19
20
21 A = [1,2]
22 B = 1
23 pick_from_both_side(A,B)
```

Out[30]: 2

```
In [28]: 1 def pick_from_both_side(A,B):
2     n = len(A)
3
4     prefix = [A[0]]
5     for x in A[1:]:
6         prefix.append(x+prefix[-1])
7
8     maxsum = -float('inf')
9     for L in range(B):
10        R = B - L
11
12        if L == 0:
13            leftsum = 0
14        else:
15            leftsum = prefix[L-1]
16
17
18        if n-R == 0:
19            rightsum = prefix[n-1]
20        else:
21            rightsum = prefix[n-1]-prefix[n-R-1]
22
23
24        total = leftsum + rightsum
25        maxsum = max(total, maxsum)
26
27 return maxsum
28
29
30
31 A = [1,2]
32 B = 1
33 pick_from_both_side(A,B)
```

Out[28]: 2

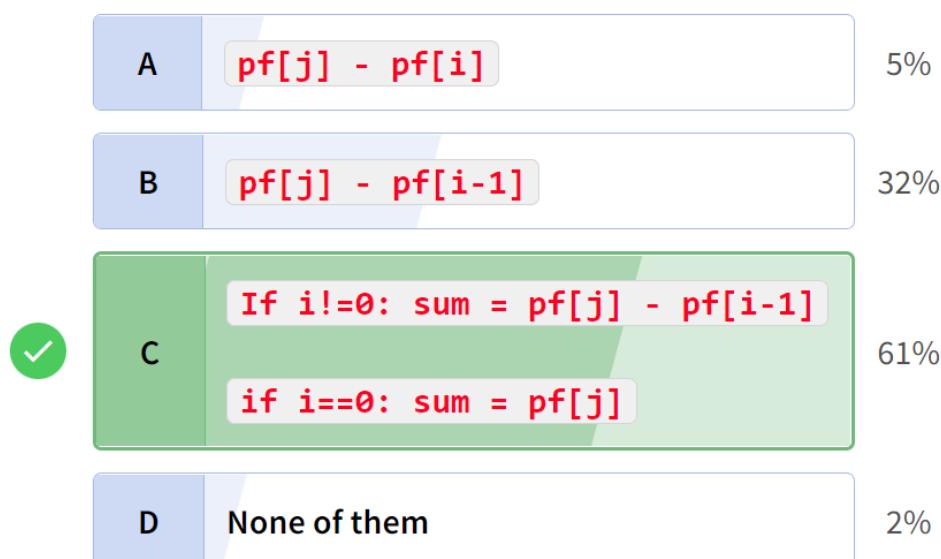
```
In [33]: 1 def pick_from_both_sides(A,B):
2     n = len(A)
3
4     maxsum = sum(A[:B])
5
6
7     rightpicks = n-1
8     total = maxsum
9     for i in range(B-1,-1,-1):
10
11        total = total + A[rightpicks] - A[i]
12
13        maxsum = max(total, maxsum)
14        rightpicks -= 1
15
16 return maxsum
17
18
19
20
21
22 A = [5, -2, 3, 1, 2]
23 B = 3
24 pick_from_both_sides(A,B)
```

Out[33]: 8

In []: 1

Sum of all elements from index i to j =

59 users have participated



Answered in 6.6 seconds

(Since you've already attempted the quiz before,
this attempt wouldn't update your score)

Problem Description

Given an integer array **A** of size **N**. In one second, you can increase the value of one element by 1.

Find the **minimum** time in seconds to make all elements of the array equal.

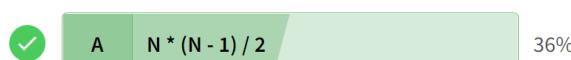
```
In [105]: 1 A = [2, 4, 1, 3, 2]
2
3 def time_to_equity(A):
4
5     n = len(A)
6     maxnum = max(A)
7     ans = 0
8     for i in range(n):
9         diff = maxnum - A[i]
10        ans += diff
11    return ans
12 time_to_equity(A)
```

Out[105]: 8

In []: 1

Given a string of length N, how many pairs of indices (i, j) are there, such that $i < j$?

64 users have participated



$$(0,1) / (0,2) / (0,3) / (0,4) / \dots / (0,n-1) \Rightarrow \binom{n-1}{n-1} + 1 = \binom{n}{n}$$

$$(1,2) / (1,3) / \dots / (1,n-1) \Rightarrow \binom{n-1}{n-2} + 1 = \binom{n}{n-1}$$

n chans



$$\begin{aligned} \text{Total no of pairs} &= (n-1) + (n-2) + (n-3) + \dots + 0 \\ &= \underbrace{(n-1)(n-1+1)}_{2} \quad \boxed{\frac{n(n+1)}{2}} \end{aligned}$$

Q1. Special Subsequences "AG" </>

Problem Description

You have given a string **A** having **Uppercase English letters**.

You have to find how many times subsequence "**AG**" is there in the given string.

NOTE: Return the answer modulo $10^9 + 7$ as the answer can be very large.

In [49]:

```

1 A = "asdagsdgaggg"
2 n = len(A)
3 count = 0
4 for i in range(n):
5     if A[i] == "a":
6         for j in range(i+1,n):
7             if A[j] == "g":
8                 count += 1
9 print(count)

```

```
In [48]: 1 def special_subsequences_AG(S):
2     n = len(S)
3
4     g_count = n*[0]
5     count = 0
6     for i in reversed(range(n)):
7         if S[i] == "g":
8             count += 1
9             g_count[i] = count
10    print(g_count)
11    ans = 0
12    for x in range(n):
13        if S[x] == "a":
14            ans += g_count[x]
15    return ans
16
17
18 S = "asdagsdgaggg"
19 special_subsequences_AG(S)
```

[5, 5, 5, 5, 5, 4, 4, 4, 3, 3, 2, 1]

Out[48]: 13

```
In [50]: 1 def special_subsequences_AG_optim(S):
2     n = len(S)
3     ans = 0
4     gc = 0
5     for i in reversed(range(n)):
6
7         if S[i] == "g":
8             gc += 1
9         if S[i] == "a":
10            ans += gc
11     return ans
12
13
14 S = "asdagsdgaggg"
15 special_subsequences_AG_optim(S)
```

Out[50]: 13

In []: 1

In []: 1

In []: 1

In []: 1

Q3. Leaders in an array </>

Problem Description

Given an integer array **A** containing **N** distinct integers, you have to find all the **leaders** in array **A**.

An element is a leader if it is strictly greater than all the elements to its right side.

NOTE: The rightmost element is always a leader.

```
In [60]: 1 A = [1,7,8,-6,0,3,-7,-10]           # O(n^2)O(1)
          2 n = len(A)
          3 c = 0
          4 for i in range(n):
          5     leader = True
          6     for j in range(i+1,n):
          7         if A[i] <= A[j]:
          8             leader = False
          9     if leader:
          10        c += 1
          11 print(c)
          12
          13
```

4

```
In [66]: 1 def leaders_in_array(A):           # most optimised O(n)O(1)
          2     n = len(A)
          3     count = 0
          4     maxnum = -float('inf')
          5     for i in range(n-1,-1,-1):
          6         if A[i] > maxnum :
          7             count +=1
          8             maxnum = A[i]
          9     return count
          10
          11
          12
          13 A = [1,7,8,-6,0,3,-7,-10]
          14 leaders_in_array(A)
```

Out[66]: 4

In []: 1

```
In [64]: 1 A = [1,7,8,-6,0,3,-7,-10]    # O(n)O(n)
          2 n = len(A)
          3
          4 postfixmax = [0]*n
          5 maxele = -float("inf")
          6 for i in range(n-1,-1,-1):
          7     maxele = max(A[i],maxele)
          8     postfixmax[i] = maxele
          9
         10 print("PostFix Max Array : ",postfixmax)
         11 print("ans : ", len(set(postfixmax)))
         12
```

```
PostFix Max Array :  [8, 8, 8, 3, 3, 3, -7, -10]
ans :  4
```

```
In [ ]: 1
```

Q2. Closest MinMax

</>

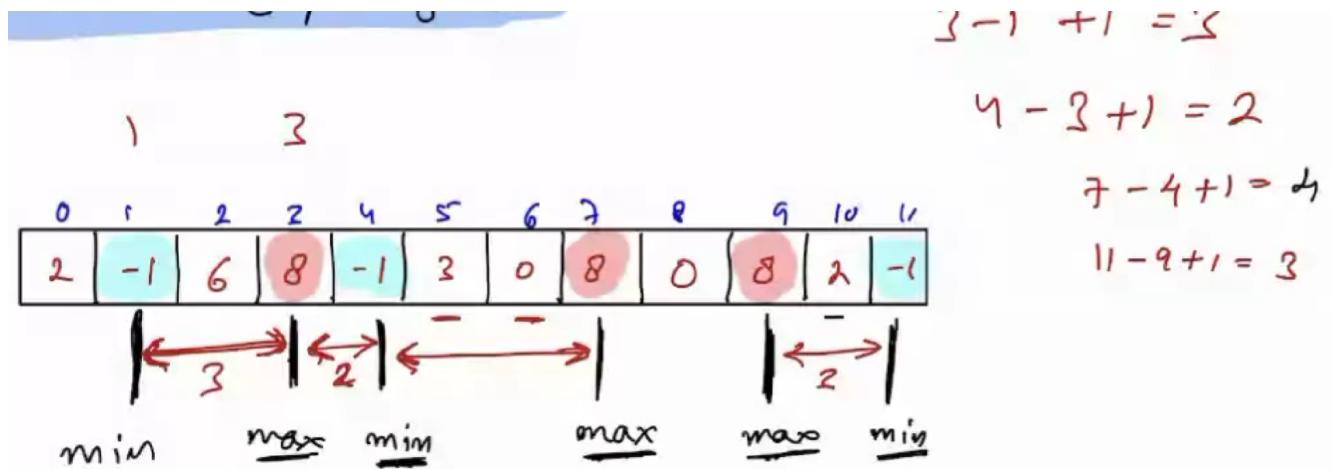
Problem Description

Given an array **A**, find the size of the smallest subarray such that it contains at least one occurrence of the maximum value of the array

and at least one occurrence of the minimum value of the array.

```
In [80]: 1 A = [2,2,6,4,5,1,5,2,6,4,1]
2 n = len(A)
3
4 minval = min(A)
5 maxval = max(A)
6 minlen = float('inf')
7 for s in range(n):
8     for e in range(s,n):
9         if minval in A[s:e+1] and maxval in A[s:e+1]:
10            l = e-s+1
11            if l < minlen:
12                minlen = l
13 print(minlen)
```

3



In [104]:

```
1 def closest_min_max(A):
2     n = len(A)
3     minval = min(A)
4     maxval = max(A)
5     minlen = float('inf')
6
7     imax = None # assume we have not found any min max
8     imin = None    # indexes for max and min updatation on the way
9
10    for i in range(n):
11
12        if A[i] == maxval:    # if we find max , update max index
13            imax = i
14        if imin != None:      # if we have previously found min
15            length = imax - imin + 1           # measure distance
16            minlen = min(length,minlen)
17
18        if A[i] == minval:      # if we find min , update min index
19            imin = i
20        if imax != None:      # if prevously found max val , measure distance
21            length = imin - imax + 1    # b-a+1
22            minlen = min(length,minlen)
23
24    return minlen
25
26 A = [2,2,6,4,5,1,5,2,6,4,1]
27 closest_min_max(A)
```

Out[104]: 3

In []:

1

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

Q3. Bulbs </> Solved

Problem Description

A wire connects **N** light bulbs.

Each bulb has a switch associated with it; however, due to faulty wiring, a switch also changes the state of all the bulbs to the right of the current bulb.

Given an initial state of all bulbs, find the **minimum number** of switches you have to press to turn on all the bulbs.

You can press the same switch multiple times.

Note: **0** represents the bulb is off and **1** represents the bulb is on.

```
In [96]: 1 A = [0,1,0,1]
2 n = len(A)
3 c = 0
4 for i in range(n):
5     if A[i] == 0:
6
7         for i in range(i,n):
8             if A[i] == 0:
9                 A[i] = 1
10            else:
11                A[i] = 0
12            c += 1
13
14
15 #     print(A)
16 print(c)
17
```

```
In [103]: 1 def bulb(A):
2     n = len(A)
3     initial = 0
4     c = 0
5
6     for i in range(n):
7         if A[i] == initial:
8             c += 1
9             initial = 1 - initial # change the initial state of bulb
10            A[i] = initial
11
12
13 A = [0,1,0,1]
14 bulb(A)
```

Out[103]: 4

In []: 1

In []: 1

subarrays

```
In [107]: 1 A = [1,2,3,4,5]
2
3 n = len(A)
4
5 for i in range(n):
6     for j in range(i,n):
7         for x in range(i,j+1):
8             print(A[x],end = " ")
9         print()
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
2
2 3
2 3 4
2 3 4 5
3
3 4
3 4 5
4
4 5
5
```

```
In [108]: 1 # total number of subarrays : (n*(n+1)) / 2
```

In []: 1

In []: 1

Q2. Even Subarrays </>

Solved

Problem Description

You are given an integer array **A**.

Decide whether it is possible to divide the array into one or more subarrays of **even length** such that the **first** and **last** element of all subarrays will be **even**.

Return "**YES**" if it is possible; otherwise, return "**NO**" (without quotes).

```
In [109]: 1 def even_subarray(A):
2     if len(A) % 2 == 0 and A[0] % 2 == 0 and A[-1] % 2 == 0:
3         return "YES"
4     return "NO"
```

```
In [112]: 1 A = [2, 4, 8, 6]
2 even_subarray(A)
```

Out[112]: 'YES'

```
In [111]: 1 A = [2, 4, 8, 7, 6]
2 even_subarray(A)
```

Out[111]: 'NO'

In []: 1

In []: 1

In []: 1

In []: 1

Q1. Amazing Subarrays </>

Solved



You are given a string **S**, and you have to find all the **amazing substrings** of **S**.

An amazing Substring is one that starts with a **vowel** (a, e, i, o, u, A, E, I, O, U).

In [117]:

```
1 def amazing_subarray_string(A):
2     n = len(A)
3
4     vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
5     index = len(A)
6     ans = 0
7     for i in range(n):
8         if A[i] in vowels:
9             ans += index
10            index -= 1
11    return ans
12
13
14 A = "ABEC"
15
16 amazing_subarray_string(A)
```

Out[117]: 6

In []:

1

In []:

1

In []:

1

Q2. Max Sum Contiguous Subarray </>

 Solved



Asked in:   

Problem Description

Find the **contiguous non-empty** subarray within an array, **A** of length **N**, with the **largest sum**.

In [145]:

```
1 A = [5, -10, 100, -20, 15, -10, 25, 30]
2
3 prefix = [A[0]]
4
5 for i in range(1, len(A)):
6     prefix.append(A[i]+prefix[i-1])
7 maxsum = -float('inf')
8 for start in range(len(A)):
9     for end in range(start, len(A)):
10        if start==0:
11            sum = prefix[end]
12        else:
13            sum = prefix[end]-prefix[start-1]
14        if sum > maxsum:
15            maxsum = sum
16 print(maxsum)
```

In [143]:

```
1 def max_sum_subarray(A):
2     n = len(A)
3     maxsum = -float("inf")
4
5     for i in range(n):
6         summ = 0
7         for j in range(i,n):
8             summ += A[j]
9             if summ > maxsum:
10                 maxsum = summ
11                 print(A[i:j+1],":",summ)
12
13
14     return maxsum
```

In [136]:

```
1 A = [3,-10,100,-20,15,-10,25,30]
2 max_sum_subarray(A)
3
```

```
[3] : 3
[3, -10, 100] : 93
[3, -10, 100, -20, 15, -10, 25] : 103
[3, -10, 100, -20, 15, -10, 25, 30] : 133
[100, -20, 15, -10, 25, 30] : 140
```

Out[136]: 140

In [41]:

```
1 def maxSubarraySum(A):                      # Kadane's Algo
2     n = len(A)
3     maximum = A[0]
4     currentMax = A[0]
5
6     for i in range(1,n):
7         #     print(maximum, currentMax,A[i], A[i]+currentMax)
8         currentMax = max( currentMax+A[i], A[i] )
9
10        maximum = max(maximum , currentMax)
11
12    return maximum
12 A = [1,2,3,-2,5]
13 maxSubarraySum(A)
```

Out[41]: 9

In []:

```
1
2 # max product subarray : kadane's algo
```

```
In [3]: 1 def maxproduct(arr,n):
2     n = len(arr)
3     maxproduct = -float("inf")
4
5
6     current_product = 1
7     for i in range(n):
8         current_product = current_product * arr[i]
9         maxproduct = max(current_product , maxproduct)
10        if current_product == 0:
11            current_product =1
12        current_product = 1
13        for i in reversed(range(n)):
14            current_product = current_product * arr[i]
15            maxproduct = max(current_product , maxproduct)
16            if current_product == 0:
17                current_product =1
18        return maxproduct
19 A = [3,-10,0,-20,15,0,25,30]
20 maxproduct(A,len(A))
```

Out[3]: 750

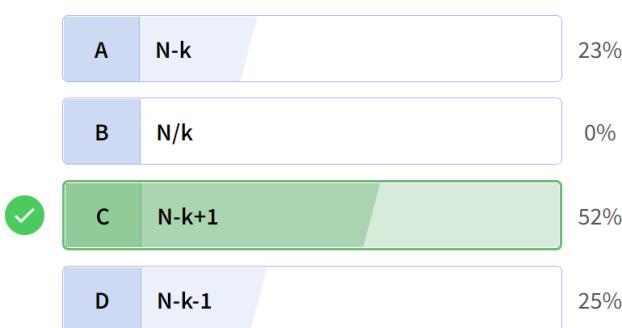
In []: 1

In []: 1

In []: 1

Given N array elements how many subarrays of len = k are there ?

60 users have participated



Given N array elements and len = k ?

Print last subarray start index and last index

58 users have participated



$$|A_{\text{array}}| = n$$

$k \rightarrow \text{subarray size}$

$s_{\text{start}}, e_{\text{end}}(n-k, n-1)$

$$\begin{aligned} l - s + 1 &= \text{length} \\ n-1 - s + 1 &= k \\ n-1 - k + 1 &= s \\ s &= n - k \end{aligned}$$

In []:	1

Q-2 Max subarray sum with length k

```

In [149]: 1
          2 def max_k_length_subarray_sum(A,k):
          3     n = len(A)
          4     maxsum = -float("inf")
          5     for s in range(n-k+1):
          6         e = k + s -1
          7
          8         summ = 0
          9         for i in range(s,e+1):
         10             summ += A[i]
         11
         12         maxsum = max(maxsum , summ)
         13
         14     return maxsum
         15
         16
         17 A = [5,-10,100,-20,15,-10,25,30]
         18 k = 6
         19
         20 max_k_length_subarray_sum(A,k)
         21
         22
         23
    
```

Out[149]: 140

In []:	1
In []:	1

In [151]:

```
1 def max_k_length_subarray_sum(A,k):
2     n = len(A)
3     maxsum = -float("inf")
4
5     prefixsum = [A[0]]
6     for x in A[1:]:
7         prefixsum.append(x+prefixsum[-1])
8     for s in range(n-k+1):
9         e = k + s -1
10
11     if s == 0:
12         summ = prefixsum[e]
13     else:
14         summ = prefixsum[e]-prefixsum[s-1]
15
16     maxsum = max(maxsum , summ)
17
18 return maxsum
19
20
21
22 A = [5,-10,100,-20,15,-10,25,30]
23 k = 6
24
25 max_k_length_subarray_sum(A,k)
26
27
28
```

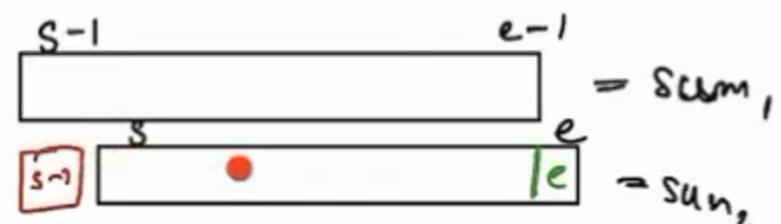
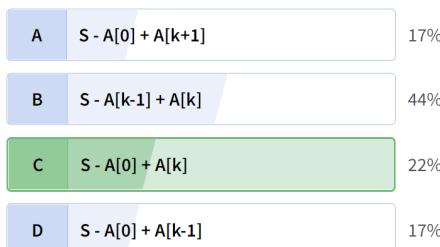
Out[151]: 140

In []:

1

In an array A, if the sum of the first subarray of size k is S, what will be the sum of the second subarray of size k?

54 users have participated



sliding window

```
In [19]: 1 def max_k_length_subarray_(A,k):
2     n = len(A)
3
4     maxsum = -float("inf")
5     total = sum(A[:k])
6
7     for s in range(1,n-k+1):
8         e = k + s - 1
9         total = total - A[s-1] + A[e] # current end
10        # previous start
11     maxsum = max(maxsum, total)
12     return maxsum
13
14 A = [5,-10,100,-20,15,-10,25,30]
15 k = 6
16 max_k_length_subarray_(A,k)
17
```

Out[19]: 140

In []:

Q3. Subarray with least average </>

Solved



Problem Description

Given an array of size **N**, find the subarray of size **K** with the least average.

```
In [38]: 1 def k_size_subarray_with_least_average(A,k):
2     n = len(A)
3
4     leastavg = float('inf')
5     total = sum(A[:k])
6
7
8     for s in range(n-k+1):
9         e = k + s - 1
10        length = e-s+1
11
12        total = total - A[s-1] + A[e]
13
14        average = total / length
15
16        if average < leastavg:
17            leastavg = average
18            starting_index = s
19
20
21
22
23
```

```
In [39]: 1 A = [3, 7, 5, 20, -10, 0, 12]
2 k = 2
3 k_size_subarray_with_least_average(A,k)
```

Out[39]: 4

```
In [40]: 1 def k_size_subarray_with_least_average(A,k):      # we dont have to find avg
2     n = len(A)
3
4     leastsum = float('inf')
5     total = sum(A[:k])
6
7
8     for s in range(n-k+1):
9         e = k + s - 1
10        length = e-s+1
11
12        total = total - A[s-1] + A[e]
13
14
15        if total < leastsum:
16            leastsum = total
17            starting_index = s
18
19        return starting_index
20
21
22 A = [3, 7, 90, 20, 10, 50, 40]
23 k = 3
24 k_size_subarray_with_least_average(A,k)
```

Out[40]: 3

In []: 1

In []: 1

inverting the problem

Q1. Sum of All Subarrays </> Solved



Problem Description

You are given an integer array **A** of length **N**.

You have to find the sum of all subarray sums of A.

More formally, a subarray is defined as a contiguous part of an array which we can obtain by deleting zero or more elements from either end of the array.

A subarray sum denotes the sum of all the elements of that subarray.

```
In [22]: 1 A = [2,1,3]      # carry forward #O(n^2)O(1)
2
3 n = len(A)
4 all_subarray_total = 0
5 for s in range(n):
6     subarraysum = 0
7     for e in range(s,n):
8         subarraysum += A[e]
9         all_subarray_total += subarraysum
10
11 print(all_subarray_total)
```

19

```
In [25]: 1 def sum_of_all_subarrays(A):          # O(n)O(1)
2     n = len(A)
3     total = 0
4     for i in range(n):
5         total += A[i] * ((n-i)*(i+1))
6         # element * subarray count containing element
7     return total
8 A = [2,1,3]
9 sum_of_all_subarrays(A)
```

Out[25]: 19

In []: 1

In []: 1

In []: 1

Q4. Maximum Subarray Easy </>  Solved



Problem Description

You are given an integer array **C** of size **A**. Now you need to find a subarray (contiguous elements) so that the sum of contiguous elements is maximum.

But the sum must not exceed **B**.

In [46]:

```
1 def max_subarray_Easy(A,B):
2     n = len(A)
3     ans = 0
4     maxsum = -float("inf")
5     for s in range(n):
6         subarray_sum = 0
7         for e in range(s,n):
8             subarray_sum += A[e]
9             if subarray_sum > maxsum and subarray_sum <=B:
10                 maxsum = subarray_sum
11                 ans = maxsum
12
13     return ans
14 A = [2,1,3,4,5]
15 B = 12
16 max_subarray_Easy(A,B)
```

Out[46]: 12

In [47]:

```
1 A = [2,2,2]
2 B = 1
3 max_subarray_Easy(A,B)
```

Out[47]: 0

In []:

Q1. Counting Subarrays! </>

 Solved



Problem Description

Given an array **A** of **N** non-negative numbers and you are also given non-negative number **B**.

You need to find the **number of subarrays in A having sum less than B**. We may assume that there is no overflow.

In [70]:

```
1 def counting_subarrays(A,B):
2
3     n = len(A)
4     count = 0
5
6     for s in range(n):
7         subsum = 0
8         for e in range(s,n):
9             subsum += A[e]
10            if subsum < B:
11                print(A[s:e+1])
12                count += 1
13
14 return count
```

In [71]:

```
1 A = [2,5,6]
2 B = 10
3 counting_subarrays(A,B)
```

```
[2]
[2, 5]
[5]
[6]
```

Out[71]: 4

In [72]:

```
1 A = [1,11,2,3,15]
2 B = 10
3 counting_subarrays(A,B)
```

```
[1]
[2]
[2, 3]
[3]
```

Out[72]: 4

In []:

```
1
```

Q2. Good Subarrays Easy </> ✓ Solved



Problem Description

Given an array of integers **A**, a subarray of an array is said to be good if it fulfills any one of the criteria:

1. Length of the subarray is even, and the sum of all the elements of the subarray must be less than **B**.

2. Length of the subarray is odd, and the sum of all the elements of the subarray must be greater than **B**.

Your task is to find the count of good subarrays in **A**.

In [74]:

```
1 def good_subarray_easy(A,B):
2
3     n = len(A)
4     count = 0
5
6     for s in range(n):
7         subsum = 0
8         for e in range(s,n):
9             length = e-s+1
10            subsum += A[e]
11            if length % 2 == 0 and subsum < B:
12                count += 1
13            if length % 2 != 0 and subsum > B:
14                count += 1
15
16    return count
```

In []:

```
1
```

```
In [75]: 1 A = [13, 16, 16, 15, 9, 16, 2, 7, 6, 17, 3, 9]
2 B = 65
3 good_subarray_easy(A,B)
```

Out[75]: 36

```
In [76]: 1 A = [1, 2, 3, 4, 5]
2 B = 4
3 good_subarray_easy(A,B)
```

Out[76]: 6

```
In [ ]: 1
```

```
In [ ]: 1
```

Q3. Alternating Subarrays Easy </> (✓) Solved

You are given an integer array **A** of length **N** comprising of **0's & 1's**, and an integer **B**.

You have to tell all the indices of array **A** that can act as a center of $2 * B + 1$ length 0-1 alternating subarray.

A **0-1 alternating array** is an array containing only **0's & 1's**, and having no adjacent **0's or 1's**.
For e.g. arrays **[0, 1, 0, 1]**, **[1, 0]** and **[1]** are 0-1 alternating, while **[1, 1]** and **[0, 1, 0, 0, 1]** are not.

```
In [155]: 1 def alternating_subarray_easy(A,B):
2
3     n = len(A)
4     l = 2*B + 1
5     centers = []
6
7     for s in range(n-l+1):
8         e = l + s -1
9
10        is_alternating = True
11        previous = None
12
13        for i in range(s,e+1):
14            if A[i] == previous:
15                is_alternating = False
16                break
17            previous = A[i]
18
19        if is_alternating:
20            centers.append((e+s)//2)
21
22    return centers
23
```

```
In [158]: 1 A = [1,0,0,0,1,0,1,0,1,1,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,]
2 B = 2
3 alternating_subarray_easy(A,B)
```

Out[158]: [5, 6, 14, 15, 16, 17, 18]

In []: 1

```
In [147]: 1 def alternating_subarray_easy_optim(A,B):
2     string = ""
3     for x in range(len(A)):
4         string += str(A[x])
5
6     l = 2*B+1
7     n = len(string)
8     ans = []
9
10
11    for i in range(B,n-B):
12
13        if "00" not in string[i-B:i+B+1] and "11" not in string[i-B:i+B+1]:
14            ans.append(i)
15
16
17 return ans
```

```
In [157]: 1 A = [1,0,0,0,1,0,1,0,1,1,1,0,0,1,0,1,0,1,0,1,0,0,0,]
2 B = 2
3 alternating_subarray_easy_optim(A,B)
```

Out[157]: [5, 6, 14, 15, 16, 17, 18]

In []: 1

In []: 1

2d Matrix

In []: 1

$$M = \left[\begin{array}{ccccccc} [0, 1, -6, 8, 3, 2, 5], & & & & & & \\ [10, 10, 9, 7, 0, 1, 3], & & & & & & \\ [1, 6, 8, 2, 4, 9, 2], & & & & & & \\ [3, -1, 7, 2, 8, 9, 0] \end{array} \right]$$

$\boxed{122
454
789}$

$$M = \left[[-], [-], [-], [-] \right] \quad \text{len}(M)$$

Indexing a Matrix

$$M[2][4]$$

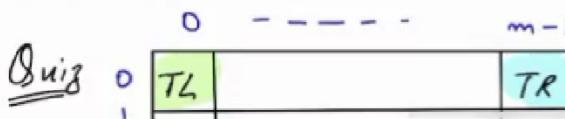
$$M[\text{row}][\text{col}] \neq M[\text{row}, \text{col}] \neq M[\text{row}:\text{col}]$$

numpy /
pandas

$M \rightarrow$ find n, m

$n = \text{len}(M)$ # num Rows

$m = \text{len}(M[0])$ # Num Cols



Bottom right $\rightarrow M[m-1][m-1]$

Top right $\rightarrow M[0][m-1]$

Bottom left $\rightarrow M[m-1][0]$

```
In [243]: 1 # row sum / column sum / traversal row wise and column wise
```

You are given a 2D integer matrix A, return a 1D integer vector containing row-wise sums of original matrix.

```
In [242]: 1 M = [[0,1,2,3,4],  
2     [9,8,7,4,5],  
3     [1,5,9,8,4],  
4     [3,6,2,5,9]]  
5  
6 n = len(M)  
7 m = len(M[0])  
8  
9 for i in range(n):  
10    rowsum = 0  
11    for j in range(m):  
12        rowsum += M[i][j]  
13        print(M[i][j], end = " ")  
14    print(": ", rowsum)  
15
```

```
0 1 2 3 4 : 10  
9 8 7 4 5 : 33  
1 5 9 8 4 : 27  
3 6 2 5 9 : 25
```

You are given a 2D integer matrix A, return a 1D integer vector containing column-wise sums of original matrix.

In [178]:

```
1 for j in range(m):
2     colsum = 0
3     for i in range(n):
4         colsum += M[i][j]
5         print(M[i][j],end = " ")
6     print(": ",colsum)
7
```

```
0 9 1 3 : 13
1 8 5 6 : 20
2 7 9 2 : 20
3 4 8 5 : 20
4 5 4 9 : 22
```

In []:

```
1
```

In []:

```
1
```

PRIMARY DIAGOAL

You are given a **N X N** integer matrix. You have to find the sum of all the main diagonal elements of **A**.

Main diagonal of a matrix **A** is a collection of elements **A[i, j]** such that **i = j**.

In [248]:

```
1 M = [[0,1,2,3,4],
2     [9,8,7,4,5],
3     [1,5,9,8,4],
4     [3,6,2,5,9]]
5
6 n = len(M)
7 m = len(M[0])
8 ans = 0
9 for i in range(n):
10    print(M[i][i], end = " ")      # PRIMARY DIAGOAL
11    ans += M[i][i]
12
13 print()
14 print(ans)
```

```
0 8 9 5
22
```

In []:

```
1
```

SECONDARY DIAGONAL

In [219]:

```
1 M = [[0,1,2,3,4],  
2     [9,8,7,4,5],  
3     [1,5,9,8,4],  
4     [3,6,2,5,9]]  
5  
6 n = len(M)  
7 m = len(M[0])  
8  
9 r = 0  
10 c = m-1  
11  
12 while c >= 0 and r < n:      # secondary diagonal  
13     print(M[r][c])  
14     r += 1  
15     c -= 1  
16
```

```
4  
4  
9  
6
```

In [226]:

```
1 M = [[0,1,2,3,4],  
2     [9,8,7,4,5],  
3     [1,5,9,8,4],  
4     [3,6,2,5,9]]  
5  
6 n = len(M)  
7 m = len(M[0])  
8  
9 for r in range(n):  
10    print(M[r][n-r],end = " ")  
11
```

```
4 4 9 6
```

You are given a $N \times N$ integer matrix. You have to find the sum of all the minor diagonal elements of \mathbf{A} .

Minor diagonal of a $M \times M$ matrix \mathbf{A} is a collection of elements $\mathbf{A}[i, j]$ such that $i + j = M + 1$ (where i, j are 1-based).

```
In [249]: 1 M = [[0,1,2,3,4],
2     [9,8,7,4,5],
3     [1,5,9,8,4],
4     [3,6,2,5,9]]
5
6 n = len(M)
7 m = len(M[0])
8 ans = 0
9 for r in range(n):
10    #   print(M[r][n-r],end = " ")
11    ans += M[r][n-r]
12 print()
13 print(ans)
14
```

23

```
In [ ]: 1
```



```
In [ ]: 1
```



```
In [ ]: 1
```



```
In [ ]: 1
```

CREATE EMPTY 2d MATRIX OF n * m SIZE

ASSIGN VALUE TO M

```
In [213]: 1 M = []
2 n = 4
3 m = 8
4 val = 0
5 for i in range(n):
6     M.append([])
7     for j in (range(m)):
8         val += 1
9         M[i].append(val)
10 M
```

```
Out[213]: [[1, 2, 3, 4, 5, 6, 7, 8],
[9, 10, 11, 12, 13, 14, 15, 16],
[17, 18, 19, 20, 21, 22, 23, 24],
[25, 26, 27, 28, 29, 30, 31, 32]]
```

```
In [217]: 1 M = [[0]*m for _ in range(n)]
2 val = 0
3 for i in range(n):
4     for j in range(m):
5         val += 1
6         M[i][j] = val
7
8 M
```

Out[217]: `[[1, 2, 3, 4, 5, 6, 7, 8],
 [9, 10, 11, 12, 13, 14, 15, 16],
 [17, 18, 19, 20, 21, 22, 23, 24],
 [25, 26, 27, 28, 29, 30, 31, 32]]`

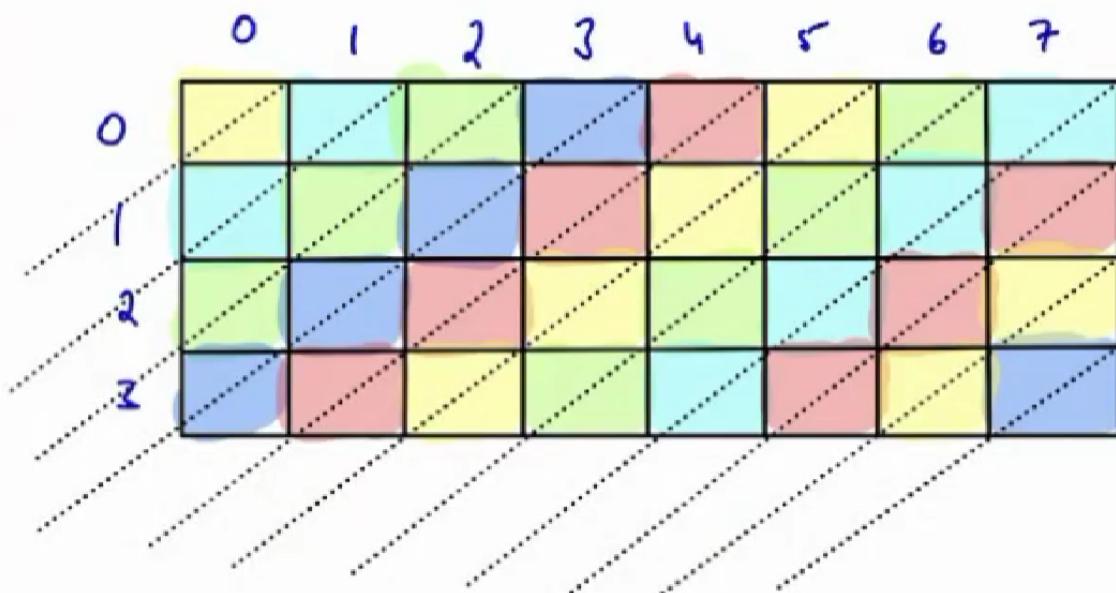
In []:

1

In []:

1

Given a rectangular Matrix , N * M. print all the right -->-- left diagonals



Given a matrix of size NxM, how many diagonals will be there?

59 users have participated

A	N*M	2%
B	N*M-1	8%
C	N+M-1	90%
D	N+M	0%

```
In [240]: 1 M =[[1 , 2, 3, 4, 5, 6, 7, 8],
2 [9, 10, 11, 12, 13, 14, 15, 16],
3 [17, 18, 19, 20, 21, 22, 23, 24],
4 [25, 26, 27, 28, 29, 30, 31, 32]]
5 n = len(M)
6 m = len(M[0])
7
8
9 for c in range(0,m):
10
11     r = 0
12     while c >= 0 and r < n:
13         print(M[r][c],end = " ")
14         r += 1
15         c -= 1
16     print()
17 for r in range(1,n):
18     c = m-1
19
20     while c >= 0 and r < n:
21         print(M[r][c],end = " ")
22         r += 1
23         c -= 1
24     print()
```

```
1
2 9
3 10 17
4 11 18 25
5 12 19 26
6 13 20 27
7 14 21 28
8 15 22 29
16 23 30
24 31
32
```

In []:

1
2

Q5. Matrix Transpose </> ✓ Solved



Problem Description

You are given a matrix A, you have to return another matrix which is the transpose of A.

NOTE: Transpose of a matrix A is defined as - $A^T[i][j] = A[j][i]$; Where $1 \leq i \leq \text{col}$ and $1 \leq j \leq \text{row}$

```
In [255]: 1 def TransposeMatrix(M):
2     n = len(M)
3     m = len(M[0])
4
5     Mt = [[0]*n for _ in range(m)]
6
7     for r in range(n):
8         for c in range(m):
9             Mt[c][r] = M[r][c]
10
11    return Mt
12
13
14
15 M =[[1 , 2, 3, 4, 5, 6, 7, 8],
16   [9, 10, 11, 12, 13, 14, 15, 16],
17   [17, 18, 19, 20, 21, 22, 23, 24],
18   [25, 26, 27, 28, 29, 30, 31, 32]]
19 n = len(M)
20 m = len(M[0])
21 TransposeMatrix(M)
```

Out[255]: [[1, 9, 17, 25],
[2, 10, 18, 26],
[3, 11, 19, 27],
[4, 12, 20, 28],
[5, 13, 21, 29],
[6, 14, 22, 30],
[7, 15, 23, 31],
[8, 16, 24, 32]]

In []:

1

```
In [259]: 1 M = [[1,2,3], # only for square matrix
2   [4,5,6],
3   [7,8,9]]
4 n = len(M)
5
6 for i in range(n):
7     for j in range(i,n):
8         M[i][j],M[j][i] = M[j][i],M[i][j]
9 print(M)
```

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

In []:

1

In []:

1

In []:

1

In []:

1

Rotate 90degree clockwise

Right rotate a given Matrix $M_{m \times n}$

1	2	3	4
5	6	7	8
9	10	11	12

3×4

Rotate 90° clockwise
→

9	5	1
10	6	2
11	7	3
12	8	4

4×3

M rotated $90 \neq M^T$

M rotated $90 \neq M^T$

9	5	1
10	6	2
11	7	3
12	8	4

4×3

Horizontal
flip

1	5	9
2	6	10
3	7	11
4	8	12

$$M^R = (M^T)^H$$

In [284]:

```

1 def FlipHorizontal(M):
2     n = len(M)
3     m = len(M[0])
4
5     Mh = [[0]*m for _ in range(n)]
6
7     for i in range(n):
8         for j in range(m):
9             Mh[i][m-j-1] = M[i][j]
10    return Mh
11

```

In [285]:

```

1
2 M = [[1, 2, 3, 4, 5, 6, 7, 8],
3   [9, 10, 11, 12, 13, 14, 15, 16],
4   [17, 18, 19, 20, 21, 22, 23, 24],
5   [25, 26, 27, 28, 29, 30, 31, 32]]
6 FlipHorizontal(M)

```

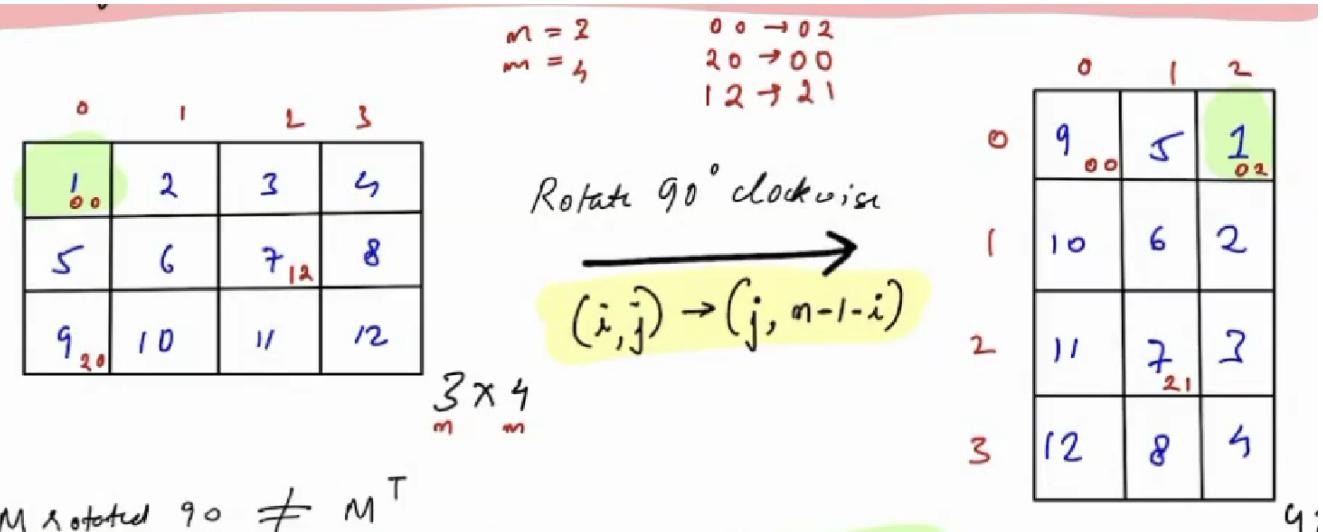
Out[285]: [[8, 7, 6, 5, 4, 3, 2, 1],
[16, 15, 14, 13, 12, 11, 10, 9],
[24, 23, 22, 21, 20, 19, 18, 17],
[32, 31, 30, 29, 28, 27, 26, 25]]

In [290]:

```
1 def rotate90degClockwise(M):
2
3     def TransposeMatrix(M):
4         n = len(M)
5         m = len(M[0])
6         Mt = [[0]*n for _ in range(m)]
7         for r in range(n):
8             for c in range(m):
9                 Mt[c][r] = M[r][c]
10        return Mt
11    def FlipHorizontal(M):
12        n = len(M)
13        m = len(M[0])
14        Mh = [[0]*m for _ in range(n)]
15        for i in range(n):
16            for j in range(m):
17                Mh[i][m-j-1] = M[i][j]
18        return Mh
19
20    return FlipHorizontal(TransposeMatrix(M))
21
22 rotate90degClockwise(M)
```

Out[290]: [[25, 17, 9, 1],
[26, 18, 10, 2],
[27, 19, 11, 3],
[28, 20, 12, 4],
[29, 21, 13, 5],
[30, 22, 14, 6],
[31, 23, 15, 7],
[32, 24, 16, 8]]

In []:



In [9]:

```
1 def rotate90degClockwise(M):
2     n = len(M)
3     m = len(M[0])
4
5     Mr = [[0]*n for _ in range(m)]
6
7     for i in range(n):
8         for j in range(m):
9             Mr[j][n-i-1] = M[i][j]
10    return Mr
```

In [10]:

```
1 M=[[1 , 2, 3, 4, 5, 6, 7, 8],  
2 [9, 10, 11, 12, 13, 14, 15, 16],  
3 [17, 18, 19, 20, 21, 22, 23, 24],  
4 [25, 26, 27, 28, 29, 30, 31, 32]]  
5  
6 rotate90degClockwise(M)
```

Out[10]: [[25, 17, 9, 1],
[26, 18, 10, 2],
[27, 19, 11, 3],
[28, 20, 12, 4],
[29, 21, 13, 5],
[30, 22, 14, 6],
[31, 23, 15, 7],
[32, 24, 16, 8]]

In []:

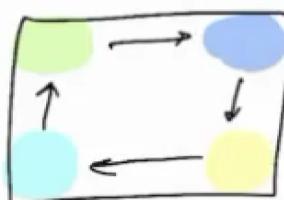
1

$$(i, j) \rightarrow (j, m-i)$$

$$(j, m-i) \rightarrow (m-i, m-j)$$

$$(n-i, m-j) \rightarrow (m-j, n-i)$$

$$(m-j, i) \rightarrow (i, m-(n-i))$$



In []:

1

In []:

1

In []:

1

Q3. Matrix Multiplication </>

Solved

You are given two integer matrices **A**(having **M X N** size) and **B**(having **N X P**). You have to multiply matrix **A** with **B** and return the resultant matrix. (i.e. return the matrix **AB**).

If **A** is an $m \times n$ matrix and **B** is an $n \times p$ matrix,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

the matrix product **C** = **AB** (denoted without multiplication signs or dots) is defined to be the $m \times p$ matrix

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

such that

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^n a_{ik}b_{kj},$$

for $i = 1, \dots, m$ and $j = 1, \dots, p$.

```
In [17]: 1 A = [[0,1,2],
2      [3,2,0],
3      [1,0,2],
4      [0,3,1]]
5
6 B = [[1,0,2,3,0],
7      [0,1,0,2,3],
8      [3,0,1,0,2]]
9
10 def multiplyMatrixes(A,B):
11     m = len(A)
12     n = len(A[0])
13     p = len(B)
14     q = len(B[0])
15
16     C = [[0]*p for _ in range(m)]
17
18     for i in range(m):
19         for j in range(p):
20             s = 0
21             for k in range(n):
22                 C[i][j] += A[i][k]*B[k][j]
23
24
25
26     return C
27
28
29 multiplyMatrixes(A,B)
```

Out[17]: [[6, 1, 2, 2, 7], [3, 2, 6, 13, 6], [7, 0, 4, 3, 4], [3, 3, 1, 6, 11]]

In []: 1

In [20]: 1 A = [[1, 2],
2 [3, 4]]

```
In [22]: 1 B = [[5, 6], [7, 8]]  
2 multiplyMarixes(A,B)
```

```
Out[22]: [[19, 22], [43, 50]]
```

```
In [ ]:
```

```
1
```

Print Matrix Boundary

```
In [63]: 1 n = 5  
2 m = 10  
3 M = [[0]*m for _ in range(n)]  
4  
5 x = 0  
6 for c in range(m):  
7     for r in range(n):  
8         M[r][c] = x  
9         x = x+1  
10 M
```

```
Out[63]: [[0, 5, 10, 15, 20, 25, 30, 35, 40, 45],  
[1, 6, 11, 16, 21, 26, 31, 36, 41, 46],  
[2, 7, 12, 17, 22, 27, 32, 37, 42, 47],  
[3, 8, 13, 18, 23, 28, 33, 38, 43, 48],  
[4, 9, 14, 19, 24, 29, 34, 39, 44, 49]]
```

```
In [65]:
```

```
1 for c in range(m-1):  
2     print(M[0][c], end= " ")
```

```
0 5 10 15 20 25 30 35 40
```

```
In [66]:
```

```
1 for r in range(n-1):  
2     print(M[r][m-1], end= " ")
```

```
45 46 47 48
```

```
In [69]:
```

```
1 for c in reversed(range(1,m)):  
2     print(M[n-1][c], end= " ")
```

```
49 44 39 34 29 24 19 14 9
```

```
In [70]: 1 for r in reversed(range(1,n)):
2     print(M[r][0], end= " ")
```

4 3 2 1

In []:

1

In []:

1

In [72]:

1 M

```
Out[72]: [[0, 5, 10, 15, 20, 25, 30, 35, 40, 45],
[1, 6, 11, 16, 21, 26, 31, 36, 41, 46],
[2, 7, 12, 17, 22, 27, 32, 37, 42, 47],
[3, 8, 13, 18, 23, 28, 33, 38, 43, 48],
[4, 9, 14, 19, 24, 29, 34, 39, 44, 49]]
```

In [89]:

```
1 def print_boundary(M):
2
3     n = len(M)
4     m = len(M[0])
5
6     r = 0
7     c = 0
8     direction = "right"
9
10    while True:
11
12        print(M[r][c], end = " ")
13        if direction == "right":
14            if c == m-1:
15                direction = "down"
16                r = r+1
17            else:
18                c = c +1
19
20        elif direction == "down":
21            if r == n-1:
22                direction = "left"
23                c = c-1
24            else:
25                r = r+1
26
27        elif direction == "left":
28            if c == 0:
29                direction = "up"
30                r = r-1
31            else:
32                c = c-1
33
34        elif direction == "up":
35
36            if r == 1:
37                break
38            else:
39                r = r-1
40
41 M = [[0, 5, 10, 15, 20, 25, 30, 35, 40, 45],
42       [1, 6, 11, 16, 21, 26, 31, 36, 41, 46],
43       [2, 7, 12, 17, 22, 27, 32, 37, 42, 47],
44       [3, 8, 13, 18, 23, 28, 33, 38, 43, 48],
45       [4, 9, 14, 19, 24, 29, 34, 39, 44, 49]]
46
47 print_boundary(M)
```

0 5 10 15 20 25 30 35 40 45 46 47 48 49 44 39 34 29 24 19 14 9 4 3 2 1

In []:

1

In [95]:

```
1 def print_boundary_r_c(n , m):
2
3     M = [[0]*m for _ in range(n)]
4
5     r = 0
6     c = 0
7     direction = "right"
8     number = 0
9     while True:
10         number += 1
11         M[r][c] = number
12         if direction == "right":
13             if c == m-1:
14                 direction = "down"
15                 r = r+1
16             else:
17                 c = c +1
18
19         elif direction == "down":
20             if r == n-1:
21                 direction = "left"
22                 c = c-1
23             else:
24                 r = r+1
25
26         elif direction == "left":
27             if c == 0:
28                 direction = "up"
29                 r = r-1
30             else:
31                 c = c-1
32
33         elif direction == "up":
34
35             if r == 1:
36                 break
37             else:
38                 r = r-1
39
40
41
42 print_boundary_r_c(5,7)
```

Out[95]: [[1, 2, 3, 4, 5, 6, 7],
[20, 0, 0, 0, 0, 0, 8],
[19, 0, 0, 0, 0, 0, 9],
[18, 0, 0, 0, 0, 0, 10],
[17, 16, 15, 14, 13, 12, 11]]

In []:

1

Print Spiral

In [105]:

```
1 def print_spiral(M):
2
3
4     n = len(M)
5     m = len(M[0])
6
7     r = 0
8     c = 0
9     direction = "right"
10    boundary = 0
11
12
13    for _ in range(n*m):
14
15
16        print(M[r][c],end = " ")
17        if direction == "right":
18            if c == m-1 - boundary:
19                direction = "down"
20                r = r+1
21            else:
22                c = c +1
23
24        elif direction == "down":
25            if r == n-1 - boundary:
26                direction = "left"
27                c = c-1
28            else:
29                r = r+1
30
31        elif direction == "left":
32            if c == 0 + boundary:
33                direction = "up"
34                r = r-1
35            else:
36                c = c-1
37
38        elif direction == "up":
39            if r == 1 + boundary:
40                direction = "right"
41                boundary += 1
42                c = c+1
43            else:
44                r = r-1
45
46
47
48 M = [[0, 5, 10, 15, 20, 25, 30, 35, 40, 45],
49      [1, 6, 11, 16, 21, 26, 31, 36, 41, 46],
50      [2, 7, 12, 17, 22, 27, 32, 37, 42, 47],
51      [3, 8, 13, 18, 23, 28, 33, 38, 43, 48],
52      [4, 9, 14, 19, 24, 29, 34, 39, 44, 49]]
53 print_spiral(M)
54
```

```
0 5 10 15 20 25 30 35 40 45 46 47 48 49 44 39 34 29 24 19 14 9 4 3 2 1 6 11 16 21 26 3
1 36 41 42 43 38 33 28 23 18 13 8 7 12 17 22 27 32 37
```

```
In [4]: 1 def print_spiral(n , m):
2
3     M = [[0]*m for _ in range(n)]
4
5     r = 0
6     c = 0
7     direction = "right"
8     number = 0
9     boundry = 0
10
11    for _ in range(n*m):
12
13        number += 1
14        M[r][c] = number
15        if direction == "right":
16            if c == m-1 - boundry:
17                direction = "down"
18                r = r+1
19            else:
20                c = c +1
21
22        elif direction == "down":
23            if r == n-1 - boundry:
24                direction = "left"
25                c = c-1
26            else:
27                r = r+1
28
29        elif direction == "left":
30            if c == 0 + boundry:
31                direction = "up"
32                r = r-1
33            else:
34                c = c-1
35
36        elif direction == "up":
37            if r == 1 + boundry:
38                direction = "right"
39                boundry += 1
40                c = c+1
41            else:
42                r = r-1
43
44    return M
45
46 print_spiral(5,5)
```

Out[4]: [[1, 2, 3, 4, 5],
[16, 17, 18, 19, 6],
[15, 24, 25, 20, 7],
[14, 23, 22, 21, 8],
[13, 12, 11, 10, 9]]

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

In []: 1

In []: 1

```
In [ ]: 1
```

Q2. Christmas Trees </> Solved



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

You are given an array **A** consisting of heights of Christmas trees and an array **B** of the same size consisting of the cost of each of the trees (B_i is the cost of tree A_i , where $1 \leq i \leq \text{size}(A)$), and you are supposed to choose **3** trees (let's say, indices p , q , and r), such that $A_p < A_q < A_r$, where $p < q < r$. The cost of these trees is $B_p + B_q + B_r$.

You are to choose **3** trees such that their total cost is minimum. Return that cost.

If it is not possible to choose 3 such trees return **-1**.

Example Input

Input 1:

```
A = [1, 3, 5]
B = [1, 2, 3]
```

Input 2:

```
A = [1, 6, 4, 2, 6, 9]
B = [2, 5, 7, 3, 2, 7]
```

6, 7

```
In [ ]: 1
```

```
In [372]: 1
```

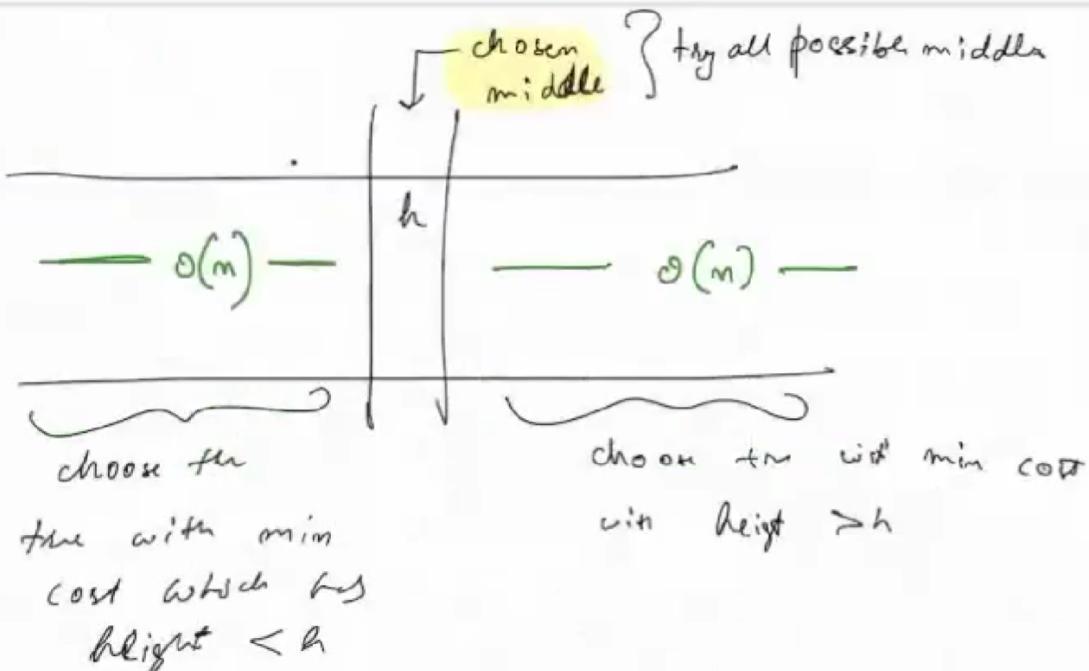
```

In [390]: 1 def chistrmas_trees(H,C):    # O(n**3)O(1)
2
3     n = len(C)
4
5     mincost = float("inf")
6
7     for t1 in range(n):
8         for t2 in range(t1+1,n):
9             for t3 in range(t2+1,n):
10                if H[t1] < H[t2] < H[t3]:
11                    cost = C[t1] + C[t2] + C[t3]
12                    if cost < mincost:
13                        mincost = cost
14
15     return mincost
16
17 H = [9,7,15,3,6,1,2, 8,2,11,5,13,15,6]
18 C = [6,4, 1,5,8,5,9,11,6, 3,3, 2, 1,4]
19 chistrmas_trees(H,C)

```

Out[390]: 6

Select one → try to optimize the rest
 try all possibilities $\Theta(n)$ try to do something clever $< \Theta(n^2)$
 all possible for next $\rightarrow \Theta(n^2)$



```

In [402]: 1 def chistrmas_trees_opt(H,C):
2
3     n = len(C)
4
5     mincost = float("inf")
6
7     for t2 in range(n):
8         cost_t2 = C[t2]
9         height_t2 = H[t2]
10
11        cost_t1 = float('inf')
12        for i in range(0,t2):
13            if H[i] < height_t2:
14                cost_t1 = min(cost_t1,C[i])
15
16        cost_t3 = float('inf')
17        for i in range(t2+1,n):
18            if H[i] > height_t2:
19                cost_t3 = min(cost_t3,C[i])
20
21        cost = cost_t1 + cost_t2 + cost_t3
22
23        mincost = min(cost,mincost)
24
25    return mincost
26
27
28 H = [9,7,15,3,6,1,2, 8,2,11,5,13,15,6]
29 C = [6,4, 1,5,8,5,9,11,6, 3,3, 2, 1,4]
30 chistrmas_trees_opt(H,C)

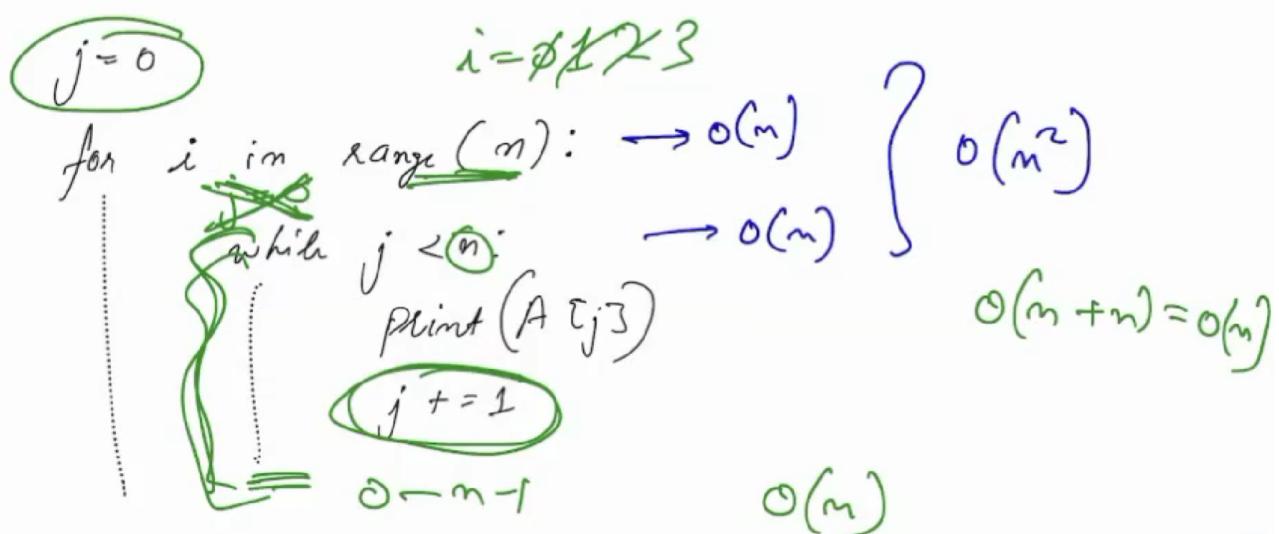
```

Out[402]: 6

In []: 1

In []: 1

In [403]: 1 # time complexity nice problem



```
In [ ]: 1
```

```
In [ ]: 1
```

Bit Manipulation

```
In [ ]: 1
```

```
1 10 base
2
3 xbase10 = [0-1-2-3-4-5-6-7-8-9]
4
5
6
7
8
9
10 81 base 10 = (a b c d e f g)base2
11      = a*2^6 + b*2^5 + c*2^4 + d*2^3 + e*2^2 + f*2^1 + g*2^0
12          a*64    b*32    c*16    d*8     e*4     f*2     g*1
13
14      >-----< |---|
15                      g is the only odd value in the total sum
16
17 because 81 is odd : g value is 1 >> g = 81%2 = 1
18 rest all a,b,c,d,e,f wil be even
19
20 80 = a*2^6 + b*2^5 + c*2^4 + d*2^3 + e*2^2 + f*2^1 + g*2^0
21 g == 1
22
23 40 = a*2^5 + b*2^4 + c*2^3 + d*2^2 + e*2^1 + f*2^0
24 f == 0,
25
26 20 = a*2^4 + b*2^3 + c*2^2 + d*2^1 + e*2^0
27 e == 0
28
29 10 = a*2^3 + b*2^2 + c*2^1 + d*2^0
30 d == 0
31
32 5 = a*2^2 + b*2^1 + c*2^0
33 c == 1
34
35 2 = a*2^1 + b*2^0
36 b == 0
37
38 1 = a*2^0
39 a == 1
40
41
42 ans : 1010001
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [415]: 1 nn = 81
2
3 anss = ""
4 while nn > 0 :
5
6     digit = nn % 2
7     anss += str(digit)
8     nn = nn // 2
9
10 print(int(anss[::-1]))
11
```

```
1010001
```

```
In [467]: 1 def dectobin(nn):
2
3     anss = ""
4     while nn > 0 :
5
6         digit = nn % 2
7         anss = str(digit) + anss
8         nn = nn // 2
9
10    return ((anss))
11 dectobin(81)
```

```
Out[467]: '1010001'
```

```
In [418]: 1 bin(81)
```

```
Out[418]: '0b1010001'
```

```
In [424]: 1 int(math.log(81,2)+1) # number of digits in binary string
```

```
Out[424]: 7
```

```
In [ ]: 1
```

```
In [434]: 1 def deci_to_oct(nn):
2
3     anss = ""
4     while nn > 0 :
5
6         digit = nn % 8
7         anss += str(digit)
8         nn = nn // 8
9
10    return ((anss[::-1]))
11 deci_to_oct(98)
```

```
Out[434]: '142'
```

```
In [437]: 1 (98)%8
```

```
Out[437]: 2
```

```
In [432]: 1 ((98//8)%8)
```

```
Out[432]: 4
```

In [433]: 1 (98//8)//8

Out[433]: 1

In []: 1

In []: 1

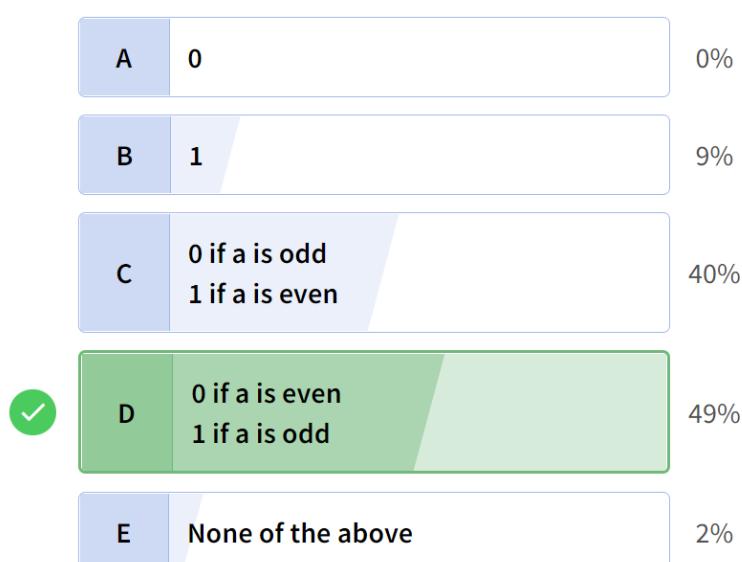
```
In [454]: 1 def deci_to_oct(nn,base):
2     if base in [0,1]:
3         return "base 0 or 1 not valid"
4     anss = ""
5     while nn > 0 :
6
7         digit = nn % base
8         anss += str(digit)
9         nn = nn // base
10
11    return ((anss[::-1]))
12 deci_to_oct(98,5)
```

Out[454]: '343'

In []: 1

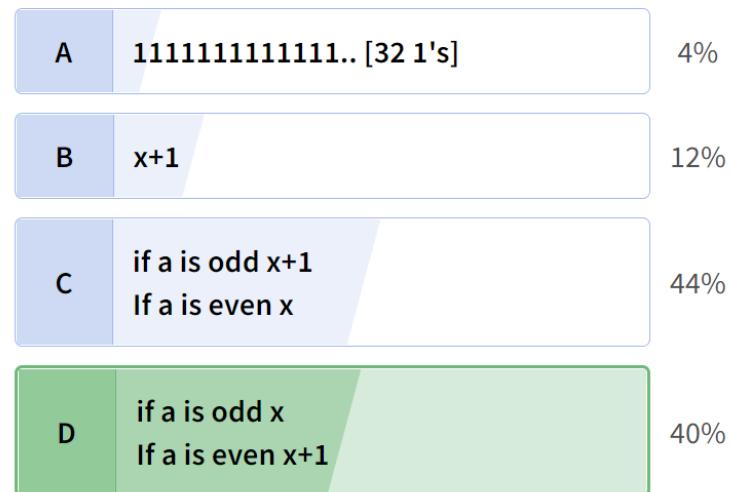
**int a = x // random value x is assigned to a
print(a&1)**

55 users have participated



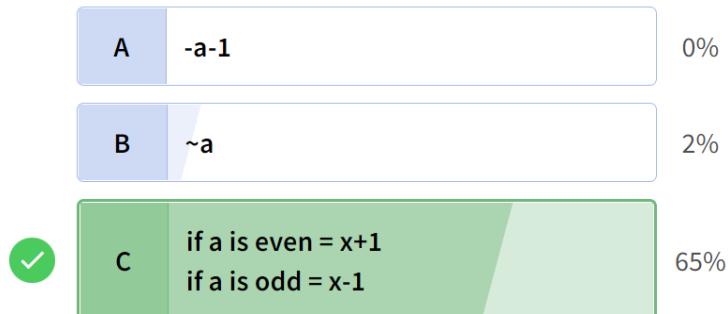
**int a = x // random value x is assigned to a
print(a|1)**

52 users have participated



**int a = x // random value x is assigned to a
print(a^1)**

49 users have participated



$O(1)$

$$= \begin{cases} a+1 & \rightarrow \text{if } a \\ & \text{no. } \rightarrow \text{even} \\ & \text{or} \\ & \underline{\text{odd}} \\ a+1 & \cdot \frac{x}{x+1} \\ a^1 & \rightarrow \frac{x+1}{x-1} \end{cases}$$

In []:

1

In [9]:

```
1 print(bin(52)[2:])
2 print(" ",bin(4)[2:])
```

110100
100

In [12]:

```
1 # if we want to turn on/off the bit :
2
3 # 52 & 4 = 4 # will check the bit on or off
4
5
```

In [15]:

```
1 binarytodecimal(9)
```

Out[15]: 9

In []:

```
1 110100 52
2 110000 48
3 110000 48
```

In []:

1

In [16]:

```
1 bin(9)
```

Out[16]: '0b1001'

```
In [13]: 1 def binarytodecimal(s):    # binary sting
2     s = str(s)
3     i = 0
4     result = 0
5     for d in s[::-1]:
6         result = result + int(d)*(2**i)
7         i = i+1
8     return result
9 x = str(binarytodecimal(10100))
10 print(x)
```

20

In []:

1

Q1. Single Number </> Solved



Asked in:



Problem Description

Given an array of integers **A**, every element appears twice except for one. Find that integer that occurs once.

NOTE: Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

```
In [20]: 1 A = [1, 2, 2, 3, 1]
2
3 def single_number(A):
4     n = len(A)
5     ans = A[0]
6     for i in range(1,n):
7         ans = ans ^ A[i]
8     return ans
9 single_number(A)
```

Out[20]: 3

In []:

1

In []:

1

In []:

1

```
In [21]: 1 bin(44)[2:]
```

```
Out[21]: '101100'
```

```
In [22]: 1 # check if last bit is on/off :  
2  
3 44&1
```

```
Out[22]: 0
```

```
In [23]: 1 # 2nd bit is on or off ? 101100  
2 44 &2
```

```
Out[23]: 0
```

```
In [24]: 1 # 3rd bit is on or off ? 101100  
2 44&4      # ans is not 0 , means its on
```

```
Out[24]: 4
```

$$(44)_{10} \rightarrow (101100)_2$$

check if 1st bit is ON or OFF $\Rightarrow 44 \& 1$ 0000

2nd bit is ON or OFF $\Rightarrow 44 \& 2$ 0000

3rd bit is ON or OFF $\Rightarrow 44 \& 4$ 000100

K^{th} bit is ON or OFF $\Rightarrow 44 \& (2^{K-1})$

```
In [ ]: 1
```

```
In [27]: 1 print(bin(11)[2:])  
2 print(bin(22)[2:])  
3 print(bin(44)[2:])  
4
```

```
1011  
10110  
101100
```

```
In [28]: 1 print(bin(31)[2:])  
2 print(bin(62)[2:])  
3 print(bin(124)[2:])  
4
```

```
11111  
111110  
1111100
```

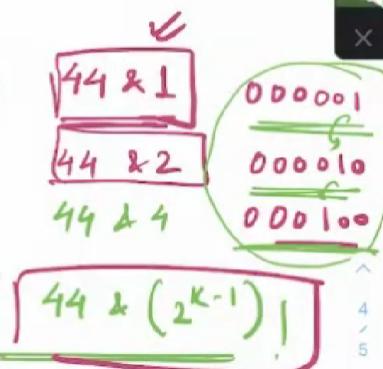
```
In [ ]: 1 # shifting all bits towards left
2
```

$$11 = 8 + 2 + 1 = 2^3 + 2^1 + 2^0 = 1011$$

$11 = 2(\dots) + 2^3 + 2^1 + 2^0 = 2^4 + 2^2 + 2^1$

$$(44)_{10} \rightarrow (101100)_2$$

check if 1st bit is ON or OFF $\Rightarrow 44 \& 1$
 2nd bit is ON or OFF $\Rightarrow 44 \& 2$
 3rd bit is ON or OFF $\Rightarrow 44 \& 4$
 Kth bit is ON or OFF $\Rightarrow 44 \& (2^{K-1})$



```
In [35]: 1 # shifting operation is very useful and it
2 #has some correlation with multiplication by 2
3
4
5 print(bin(1)[2:])
6 print(bin(2)[2:])
7 print(bin(4)[2:])
```

```
1
10
100
```

```
In [31]: 1 1<<1
```

```
Out[31]: (2, '1')
```

```
In [41]: 1 3<<1, bin(3)[2:], bin(3<<1)[2:]
```

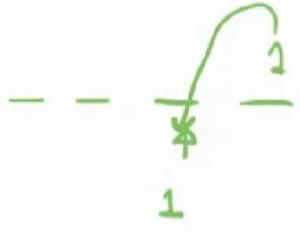
```
Out[41]: (6, '11', '110')
```

```
In [42]: 1 3<<2, bin(3)[2:], bin(3<<2)[2:]
```

```
Out[42]: (12, '11', '1100')
```

```
In [ ]: 1
```

```
In [ ]: 1 # a << x == a*(2^x)
```

$a \geq 1$ \Rightarrow 1st bit ON/OFF

 $a \& (\underline{\underline{1}} \ll 1)$ \leftarrow 2nd bit ON or OFF

$\underline{\underline{a \& (1 \ll 2)}}$ \rightarrow 3rd bit ON or OFF

k^{th} bit ON / OFF

$a \& (\underline{\underline{1 \ll (k-1)}})$
 $\cancel{a \& (2^{k-1})}$

(1)

$a \ll 2$

$a \ll k$

$(15) \ll 2$

$= = a * 2^k$

In [45]: 1 15<<2, 15*(2**2) # a = 15, k = 2

Out[45]: (60, 60)

In [46]: 1 2<<2

Out[46]: 8

In [47]: 1 binarytodecimal(1101)

Out[47]: 13

In [48]: 1 13&(1<<3)

Out[48]: 8

In []:

```
In [51]: 1 # check if ith bit is on or off !
2
3 def check_i_bit(num,i):
4
5     if num & (1 << (i-1)) == 0:
6         return "OFF"
7     return "ON"
```

```
In [54]: 1 check_i_bit(7,4)
```

```
Out[54]: 'OFF'
```

```
In [55]: 1 bin(7)
```

```
Out[55]: '0b111'
```

```
In [ ]: 1
```

```
In [56]: 1 44>>2
```

```
Out[56]: 11
```

```
In [ ]: 1 # 101100 ==> 1011
```

```
In [ ]: 1 # << Left shifting is multiplication by *2
2 # >> right shifting is integer division //2
```

```
In [ ]: 1
```

```
In [57]: 1 44 >> 3, 44//(2**3)
```

```
Out[57]: (5, 5)
```

```
In [59]: 1 29>>2, 29//(2**2)
2
```

```
Out[59]: (7, 7)
```

```
In [ ]: 1
```

```
In [60]: 1 binarytodecimal(1011)
```

```
Out[60]: 11
```

```
In [62]: 1 11//(2**3),11>>3
```

```
Out[62]: (1, 1)
```

```
In [ ]: 1
```

define a function to calculate number of set bits in binary representation of any number

```
# count of bits = log2(x) + 1

01001010100101
|0000000000000000 | I

# x & (1 << 0)
# x & (1 << 1)
# ...
# x & (1 << k)
```

In [72]:

```
1 def count_set_bits(num):
2
3     k = 0
4     count = 0
5     while True:
6         if num & (1 << k) != 0:
7             count += 1
8             k = k+1
9             if num < (1<<k):
10                 break
11     return count
12 count_set_bits(13)
```

Out[72]: 3

In [83]:

```
1 def count_set_bitsX(num):
2
3     k = 0
4     count = 0
5     while k < len(bin(num)[2:]):
6         if num & (1 << k) != 0:
7             count += 1
8             k = k+1
9     return count
10 count_set_bitsX(654)
```

Out[83]: 5

In [81]:

```
1 bin(654)[2:]
```

Out[81]: '1010001110'

In []:

```
1
```

In [85]:

```
1 def count_set_bit_with_rightshift_optr(n):
2     cnt = 0
3     while n > 0:
4         cnt += n & 1
5         n = n >> 1
6     return cnt
7 count_set_bit_with_rightshift_optr(654)
```

Out[85]: 5

```
17] # 0000000000000001  
  
# 100001010100101 >> 1  
# 10000101010010  
# 0000000000000001
```

keep checking the last bit if its 0 or not : using & operator : if its 1 add in count and shift right the given number

In []:

```
1
```

In []:

```
1
```

Type *Markdown* and *LaTeX*: α^2

In []:

```
1
```

In [160]:

```
1 def people(num):  
2  
3     a =[i for i in range(1,num+1)]  
4  
5     p = 0  
6  
7     while len(a) > 1:  
8         p += 1  
9  
10        p = p % len(a)  
11        del a[p]  
12  
13  
14    return a[0]  
15  
16  
17
```

In [161]:

```
1 people(1000)
```

Out[161]: 977

In []:

```
1
```

	Remaining Count	Starting Place
	13	1
1	12	2
2	11	5
3	10	7
4	9	9
5	8	11

$N = 13$

$2? = 8$

```
In [178]: 1 total_people = 11
2
3 def josephus_prob1m(total_people):
4     power = 1
5     while power <= total_people:
6         power = power * 2
7     power = power // 2
8
9     number_of_people_killed = total_people - (power)
10
11    person_left_alive = 2 * (number_of_people_killed) + 1
12
13    return person_left_alive
```

```
In [180]: 1 josephus_prob1m(1000)
```

Out[180]: 977

```
In [177]: 1 def josephusProblem(N):
2
3     i = 1
4     while i <= N :
5         i = i*2
6     power = i//2
7     C = N - power
8
9     return (2*C+1)
```

```
In [ ]: 1
```

```
In [186]: 1 N = 15
2 i = 1
3 while i <= N :
4     i = i*2
5 power = i//2
6 print(power)
```

8

```
In [ ]: 1
```

Q1. Majority Element </>

 Solved

Asked in:



Problem Description

Given an array of size **N**, find the majority element. The majority element is the element that appears more than $\text{floor}(n/2)$ times.

You may assume that the array is non-empty and the majority element always exists in the array.

majority elements

In [194]:

```
1 A = [1,2,3,5,5,1,5,5,5,2,3,4,4,5,5,5,5,5,6,7,8,9,5,5]
2 n = len(A)
3 freq_required = n//2
4 ans = None
5 for i in A:
6     c = 0
7     for j in A:
8         if i == j :
9             c += 1
10    if c >= freq_required:
11        ans = i
12 print(ans)
```

5

In []:

1

In [198]:

```
1 A = [1,2,3,5,5,1,5,5,5,2,3,4,4,5,5,5,5,5,6,7,8,9,5,5]
2 n = len(A)
3 A = sorted(A)
4 freq_required = n//2
5 ans = None
6 curr = None
7 freq = 0
8 for x in A:
9     if x == curr:
10        freq += 1
11        if freq > freq_required:
12            ans = curr
13    else:
14        curr = x
15        freq = 1
16 print(ans)
```

5

In []:

1

In [200]:

```
1 from collections import Counter
```

```
In [209]: 1 A = [1,2,3,5,5,1,5,5,5,2,3,4,4,5,5,5,5,5,6,7,8,9,5,5]
2 n = len(A)
3 freq_required = n//2
4
5 hmap = Counter(A)
6 ans = None
7 for x in A:
8     if hmap[x] > freq_required:
9         ans = x
10 print(x)
11
```

5

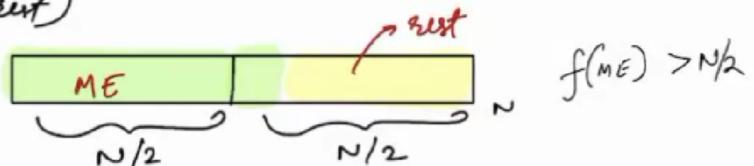
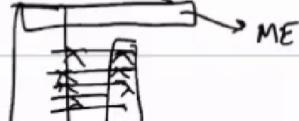
```
In [210]: 1 hmap
```

Out[210]: Counter({1: 2, 2: 2, 3: 2, 5: 12, 4: 2, 6: 1, 7: 1, 8: 1, 9: 1})

~ ~ space

Observation

- ① there can be at-most 1 majority element
- ② $f(\text{ME}) > \text{freq}(\text{rest})$



Majority Element = None & None & None & None & None
 count = 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0

$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 4 & 2 & 6 & 1 & 3 & 2 & 5 & 3 & 3 & 3 \end{bmatrix}$

3 is the ans ✓

In [293]:

```
1 A = [5,5,3,4,5,5,3,5,6,5,5,5,1,3,3,3,5,5,5,3,3,5,5,3,3,5,2,5,3,3,3,5,5,5]
2 n = len(A)
3 ans = None
4 freq_required = n//2
5
6
7 ME = None
8 count = 0
9
10 for i in range(n):
11
12     if ME == A[i]:
13         count += 1
14         print(A[i], "ME=", ME, " count = ", count)
15
16     elif count == 0:
17         ME = A[i]
18         count += 1
19         print(A[i], "ME=", ME, " count = ", count)
20     else:
21         count -= 1
22         print(A[i], " count = ", count)
23 print()
24 print()
25 print("Majority Element", ME)
26 f = 0
27 for i in A:
28     if i == ME:
29         f += 1
30     if f > freq_required:
31         ans = i
32 print()
33 print(ans)
```

```
5 ME= 5    count = 1
5 ME= 5    count = 2
3      count = 1
4      count = 0
5 ME= 5    count = 1
5 ME= 5    count = 2
3      count = 1
5 ME= 5    count = 2
6      count = 1
5 ME= 5    count = 2
5 ME= 5    count = 3
5 ME= 5    count = 4
1      count = 3
3      count = 2
3      count = 1
3      count = 0
5 ME= 5    count = 1
5 ME= 5    count = 2
5 ME= 5    count = 3
3      count = 2
3      count = 1
5 ME= 5    count = 2
5 ME= 5    count = 3
3      count = 2
3      count = 1
5 ME= 5    count = 2
2      count = 1
5 ME= 5    count = 2
3      count = 1
3      count = 0
```

```
3 ME= 3    count = 1  
5           count = 0  
5 ME= 5    count = 1  
5 ME= 5    count = 2
```

Majority Element 5

5

$$A = [1 \ 6 \ 1 \ 1 \ 2 \ 1]$$

$$ME = X \ 1$$

$$F = X \ O \ X \ X \ X \ 2$$

$$A = [4 \ 6 \ 5 \ 3 \ 4 \ 5 \ 6 \ 3 \ 3 \ 4]$$

$$M = X \ X \ X \ X \ X \ 5$$

$$F = X \ O \ X \ O \ X \ O \ X \ O \ X \ 2$$

Edge Case — Morris's algo works only if $f > n/2 \rightarrow$ a majority element exists

In []:

1

Q4. N/3 Repeat Number </> ✓ Solved

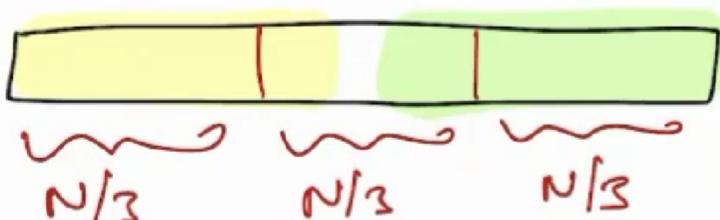
Problem Description

You're given a read-only array of **N** integers. Find out if any integer occurs more than $N/3$ times in the array in linear time and constant additional space.

If so, return the integer. If not, return -1.

If there are multiple solutions, return any one.

0



```
In [254]: 1 A = [5,5,3,4,5,5,3,5,6,5,5,5,1,3,3,3,5,5,5,3,3,5,5,3,3,5,2,5,3,3,3,5,5,5]
2 n = len(A)
3 ans = None
4 freq_required = n//3
5
6
7 ME1 = None
8 ME2 = None
9
10 count1 = 0
11 count2 = 0
12
13 for i in range(n):
14
15     if ME1 == A[i]:
16         count1 += 1
17     elif ME2 == A[i]:
18         count2 += 1
19
20     elif count1 == 0:
21         ME1 = A[i]
22         count1 += 1
23     elif count2 == 0:
24         ME2 = A[i]
25         count2 += 1
26
27     else:
28         count1 -= 1
29         count2 -= 1
30 print()
31 print()
32 print("Majority Element",ME1,ME2)
33
34 f1 = 0
35 f2 = 0
36
37 for i in range(n):
38     if A[i]== ME1:
39         f1 += 1
40     elif A[i] == ME2:
41         f2+= 1
42 ans1 = None
43 if f1 > freq_required:
44     ans1 = ME1
45 ans2 = None
46 if f2 > freq_required:
47     ans2 = ME2
48 print(ans1,ans2)
49
50
```

Majority Element 5 3
5 3

In []:

1

In []:

1

In []:

1



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

There are 100 doors, all closed. In a nearby cage are 100 monkeys.

The first monkey is let out and runs along the doors opening every one. The second monkey is then let out and runs along the doors closing the 2nd, 4th, 6th,... - **all the even-numbered doors**. The third monkey is let out. He attends only to the 3rd, 6th, 9th,... doors (**every third door, in other words**), closing any that is open and opening any that is closed, and so on. After all 100 monkeys have done their work in this way, what state are the doors in after the last pass?

Answer with the number of open doors.

*Answer is an integer. Just put the number without any decimal places if it's an integer. If the answer is Infinity, output **Infinity**.*

Feel free to get in touch with us if you have any questions

In [281]:

```

1 n = 100
2
3 def open_doors_count(n):
4
5     A = [False]*n
6
7     for i in range(1,n+1):
8         for j in range(i,n+1,i):
9             A[j-1] = not A[j-1]
10    #     print(i,A)
11    c = 0
12    for x in A:
13        if x == True:
14            c += 1
15    return c
16 open_doors_count(100)

```

Out[281]: 10

In [288]:

```
1 open_doors_count(150)
```

Out[288]: 12

In []:

```
1
```

In [290]:

```

1 c = 0
2 i = 1
3 N = 150
4
5
6 while i*i <= N:
7     c += 1
8     i = i+1
9 print(c)

```

```
In [291]: 1 def perfect_square(n):
2
3     low = 1
4     high = n
5     mid = (low + high)//2
6
7     while low <= high:
8         mid = (low + high)//2
9         if mid * mid > n:
10            high = mid - 1
11
12        elif mid * mid < n:
13            low = mid + 1
14        else:
15            return mid
16    return high
17
18
19 perfect_square(150)
20
```

Out[291]: 12

In []: 1

In []: 1

Q5. Find nth Magic Number </> ✓ Solved



Asked in: DS AC a



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given an integer **A**, find and return the **Ath** magic number.

A magic number is defined as a number that can be expressed as a power of 5 or a sum of **unique** powers of 5.

First few magic numbers are 5, 25, 30($5 + 25$), 125, 130($125 + 5$),

In [298]: 1 bin(10)[2:][::-1]

Out[298]: '0101'

In [301]: 1 (0*(5**0))+((5**2)*1)+((5**3)*0)+((5**4)*1)

Out[301]: 650

In []: 1

```
In [317]: 1 def nth_magic_number(n):
2     s = bin(n)[2:][::-1]
3
4     p = 1
5     ans = 0
6     for i in s:
7         ans += int(i)*(5**p)
8         p += 1
9
10    return ans
```

In [319]: 1 nth_magic_number(3)

```
1
1
```

Out[319]: 30

In []: 1

rearrange elements of A array of n integers containing elements from 0 to n-1 , such that A[i]= A[A[i]]

	0	1	2	3	4	5	6
$\text{arr}[7] :=$	$\frac{3}{7}$	$\frac{1}{7}$	$\frac{4}{7}$	$\frac{6}{7}$	$\frac{5}{7}$	$\frac{0}{7}$	$\frac{2}{7}$
$\text{arr}[7] :=$	$21+6$	$7+1$	$28+5$	$42+2$	$35+0$	$0+3$	$14+4$

<u>pre</u>	<u>val</u>	update
$\text{arr}[2] = \text{arr}[2]/7 = 4$	$\text{arr}[4]/7 = 5$	$\text{arr}[2] = \text{arr}[2] + \text{val}$
$\text{arr}[3] = \text{arr}[3]/7 = 6$	$\text{arr}[6]/7 = 2$	$\text{arr}[3] = \text{arr}[3] + \text{val}$
$\text{arr}[4] = \text{arr}[4]/7 = 5$	$\text{arr}[5]/7 = 0$	$\text{arr}[4] = \text{arr}[4] + \text{val}$
$\text{arr}[5] = \text{arr}[5]/7 = 0$	$\text{arr}[0]/7 = 3$	$\text{arr}[5] = \text{arr}[5] + \text{val}$
$\text{arr}[6] = \text{arr}[6]/7 = 2$	$\text{arr}[5]/7 = 4$	$\text{arr}[6] = \text{arr}[6] + \text{val}$

In [357]:

```
1 A = [3, 1, 4, 7, 6, 5, 0, 2]
2 n = len(A)
```

In [358]:

```
1 for i in range(n):
2     A[i] = n*A[i]
3
```

In [359]:

```
1 A
```

Out[359]:

```
[24, 8, 32, 56, 48, 40, 0, 16]
```

In []:

```
1
```

In [360]:

```
1 for i in range(n):
2
3     A[i] += A[A[i]//n]//n
```

In [361]:

```
1 A
```

Out[361]:

```
[31, 9, 38, 58, 48, 45, 3, 20]
```

In [362]:

```
1 for i in range(n):
2     A[i] = A[i] % n
```

In [363]:

```
1 A
```

Out[363]:

```
[7, 1, 6, 2, 0, 5, 3, 4]
```

In [364]: 1 -

Out[364]: [7, 1, 6, 2, 0, 5, 3, 4]

In []: 1

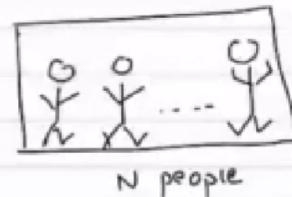
In []: 1

In []: 1

in a hall of N people, what should be value of N (howmany people should be in the hall), such that probability of atleast two people havng the same birthday is $\geq 50\%$.

Prob that
atleast 2
ppl have
same
birthday
on
same date

In a hall of N ppl, what
should be the value of
N so that the
prob of atleast
2 ppl having
the same birthday is 50%.



365 days

23 March

27 March

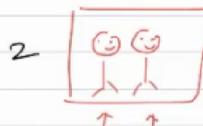
SCALER \Rightarrow

N

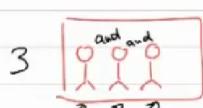


$$\frac{365 \times 1}{365} = \frac{365}{365} = 1 = 100\%$$

More
ppl
more
close



$$\frac{365}{365} \cdot \frac{364}{365} = 0.9972 = 99.72\%$$



$$\frac{365}{365} \cdot \frac{364}{365} \cdot \left(\frac{363}{365} \right) = 0.9917 = 99.17\%$$

4

⋮
⋮
⋮
⋮

- - - x - - - x
nth person

$$* \frac{362}{365}$$



$$\frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdots \frac{365-(n-1)}{365} =$$

6.50 ↓ drops
below
50%.

$P(\text{at least 2 ppl same bday}) = p$

$P(\text{all ppl have diff bday}) = 1 - p$

Prob	1.0	People	1
Prob	0.9972602739726028	People	2
Prob	0.9917958341152185	People	3
Prob	0.9836440875334497	People	4
Prob	0.9728644263002063	People	5
Prob	0.9595375163508884	People	6
Prob	0.9437642969040244	People	7
Prob	0.9256647076483308	People	8
Prob	0.905376166110833	People	9
Prob	0.8830518222889221	People	10
Prob	0.8588586216782667	People	11
Prob	0.8329752111619353	People	12
Prob	0.8055897247675702	People	13
Prob	0.7768974879950266	People	14
Prob	0.7470986802363133	People	15
Prob	0.7163959947471498	People	16
Prob	0.6849923347034391	People	17
Prob	0.6530885821282105	People	18
Prob	0.6208814739684632	People	19
Prob	0.5885616164194198	People	20
Prob	0.5563116648347941	People	21
Prob	0.5243046923374498	People	22
Prob	0.49270276567601445	People	23
Ans	23		

```
N = 0
num = 365
denom = 365
prod = 1

while prod > 0.5:
    prod = prod*num/denom
    num = num - 1
    N = N + 1
    print("Prob ", prod, "People ",N)

print("Ans ",N)
```

```
In [370]: 1 number_of_people = 0
2 choose_out_of_days = 365
3 days_of_year = 365
4
5 probability = 1
6
7 while probability > 0.5:
8     probability = (probability * choose_out_of_days)/days_of_year
9     choose_out_of_days -= 1
10    number_of_people += 1
11    print(number_of_people)
```

In [377]:

```
1 n = 0
2 nCr = 365
3 total = 365
4 probability = 1
5
6 while probability > 0.5:
7     probability = (probability * nCr) / total
8     nCr -= 1
9     n += 1
10    print("Probability:", round(probability, 4), " People : ", n)
11 print(n)
```

```
Probability: 1.0  People : 1
Probability: 0.9973  People : 2
Probability: 0.9918  People : 3
Probability: 0.9836  People : 4
Probability: 0.9729  People : 5
Probability: 0.9595  People : 6
Probability: 0.9438  People : 7
Probability: 0.9257  People : 8
Probability: 0.9054  People : 9
Probability: 0.8831  People : 10
Probability: 0.8589  People : 11
Probability: 0.833  People : 12
Probability: 0.8056  People : 13
Probability: 0.7769  People : 14
Probability: 0.7471  People : 15
Probability: 0.7164  People : 16
Probability: 0.685  People : 17
Probability: 0.6531  People : 18
Probability: 0.6209  People : 19
Probability: 0.5886  People : 20
Probability: 0.5563  People : 21
Probability: 0.5243  People : 22
Probability: 0.4927  People : 23
23
```

In []:

1

In [373]:

1

In []:

1



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given an integer array **A** of size **N**. You can **remove** any element from the array in one operation.

The cost of this operation is the **sum of all elements** in the array present **before this operation**.

Find the **minimum cost** to remove all elements from the array.

In [322]:

```

1 A = [3,5,1,-3]
2
3 def elements_removal(A):
4     n = len(A)
5
6     A = sorted(A,reverse = True)
7     x = 1
8     cost = 0
9     for i in range(n):
10         cost += (x*A[i])
11         x += 1
12     return cost
13 elements_removal(A)

```

Out[322]: 2

In []:

1

Given N elements, count the no of noble integers present.

Noble Integer : n is noble iff \rightarrow if & only if

No of elements $< n = n$

In [331]:

1

Out[331]: 3

```
In [361]: 1 A = [-10,1,1,2,2,4,4,4,8,10]
2
3 def nobel_elements(A):
4     n = len(A)
5     c = 0
6     for x in A:
7         less_number_counts = 0
8         for y in A:
9             if x > y:
10                 less_number_counts += 1
11         if less_number_counts == x:
12             c += 1
13     return c
14 print(nobel_elements([1,-5,3,5,5,-10,4]))
15 print(nobel_elements([-10, -5, 1, 3, 4, 5, 10]))
```

```
4
3
```

```
In [380]: 1 A = [-10,1,1,2,4,4,4,8,10]
2
3 nobel_elements(A)
```

```
Out[380]: 5
```

```
In [342]: 1 # if list has no duplicates :
2 # sort in ascending order : the indexes == less numbers counts
3
4 A = [-10, -5, 1, 3, 4, 5, 10]
5 x = sorted(A)
6 c = 0
7 for i in range(len(x)):
8     if i == x[i]:
9         c += 1
10 print(c)
```

```
3
```

```
In [ ]:
```

```
1
```

```
In [366]: 1 #      0 1 2 3 4 5 6 7 8 9
2 A = [-10,1,1,2,2,4,4,4,8,10] # already sorted
3     # 0 1 1 3 3 5 5 5 8 9
4 # in case of duplicates
5 n = len(A)
6
7 count = 0
8 prev = None
9 prev_i = None
10
11 for i,x in enumerate(A):
12     if x == prev:
13         lesser_elements = prev_i
14     else:
15         lesser_elements = i # if x is not equal to previous element:
16                         # then Lesser than x elements will be i
17         prev = x           # update previous element
18         prev_i = i        # update prev_i for in case of duplicate
19     if lesser_elements == x: # if Lesser ele == x : add 1 in count
20         count += 1
21 print(count)
```

3

```
In [376]: 1 # easy code
```

```
In [379]: 1 X = [-10,1,1,2,4,4,4,8,10]
2 A = sorted(X)
3
4 n = len(A)
5 nobel = 0
6 lesser = 0
7 for i in range(n):
8     if i > 0 and A[i] == A[i-1]:
9         nobel += 1
10        nobel -= 1 # do not increase
11    else:
12        lesser = i
13    if A[i] == lesser:
14        nobel += 1
15
16 print(nobel)
```

5

```
In [ ]: 1
```

custom sorting

```
In [381]: 1 # return sorting array by abs values
2
3 nums = [-10,3,5,6,10,4,-5,-44]
4
5 from functools import cmp_to_key
6
```

```
In [383]: 1 def cmpr_abs(a,b):
2     if abs(a) < abs(b):
3         return -1
4     elif abs(a) > abs(b):
5         return 1
6     else:
7         return 0
```

```
In [386]: 1 n = [-10,3,5,6,10,4,-5,-44]
2
3 sorted_by_abs_values = sorted(n,key = cmp_to_key(cmpr_abs))
```

```
In [388]: 1 sorted_by_abs_values
```

```
Out[388]: [3, 4, 5, -5, 6, -10, 10, -44]
```

```
In [ ]:
```

```
1 # return string array after sorting by Length of each strings
2
3 strings = ['aaaa','aa','a','aaaaaaaaa','aaaa','bbaabb','aaasaaaa','bab']
4
5
6 absorted = sorted( strings , key = len )
7
8 print(absorted)
```

```
['a', 'aa', 'bab', 'aaaa', 'aaaa', 'bbaabb', 'aaasaaaa', 'aaaaaaaaa']
```

```
In [ ]:
```

```
1
2 strings = ['apple','boy','do','sunny','bay','god','elephant','scaler','interviewbit']
3 print("original strings:\n",strings)
4 print()
5 lex = sorted(strings)
6 print("sorted lexicographically: \n",lex)
7 print()
8 absorted = sorted( strings , key = len )
9 print("sorted by length of string :\n ",absorted)
```

```
original strings:
['apple', 'boy', 'do', 'sunny', 'bay', 'god', 'elephant', 'scaler', 'interviewbit ']

sorted lexicographically:
['apple', 'bay', 'boy', 'do', 'elephant', 'god', 'interviewbit ', 'scaler', 'sunny']

sorted by length of string :
['do', 'boy', 'bay', 'god', 'apple', 'sunny', 'scaler', 'elephant', 'interviewbit ']
```

```
In [ ]:
```

```
In [391]: 1 strings = ['apple', 'boy', 'do', 'sunny', 'bay',
2             'god', 'elephant', 'scaler', 'interviewbit ']
3
4
5 def ascii_val_of_string(s):
6     ans = 0
7     for x in s:
8         ans += ord(x)
9     return ans
```

```
In [392]: 1 ascii_val_of_string("apple")
```

```
Out[392]: 530
```

```
In [394]: 1 sorted_by_ascii_val_strings = sorted(strings, key = ascii_val_of_string)
2 sorted_by_ascii_val_strings
```

```
Out[394]: ['do',
'god',
'bay',
'boy',
'apple',
'sunny',
'scaler',
'elephant',
'interviewbit ']
```

```
In [ ]: 1
```

```
In [433]: 1 def factor_count(n):
2     c = 0
3     i = 1
4     while i*i <=n:
5         if n % i == 0:
6             if i*i == n: c += 1
7             else: c += 2
8         i += 1
9     return (c)
10 factor_count(152)
```

```
Out[433]: 8
```

```
In [435]: 1 items = [1, 2, 5, 10, 15, 16, 6]
2 from functools import cmp_to_key
3
4 sorted_item = sorted(items, key = factor_count)
5 sorted_item
```

```
Out[435]: [1, 2, 5, 10, 15, 6, 16]
```

```
In [436]: 1 def compare_by_factors(a,b):
2     a_count = factor_count(a)
3     b_count = factor_count(b)
4
5     if a_count < b_count:
6         return -1
7     elif a_count == b_count:
8         return 0
9     else:
10        return 1
11
12 sortedlist = sorted(items,key = cmp_to_key(compare_by_factors))
13 print(sortedlist)
14
```

```
[1, 2, 5, 10, 15, 6, 16]
```

```
In [437]: 1 sorted(items, key= factor_count, reverse = True)
```

```
Out[437]: [16, 10, 15, 6, 2, 5, 1]
```

```
In [440]: 1 def compare_by_factors(a,b):
2     a_count = factor_count(a)
3     b_count = factor_count(b)
4
5     if a_count < b_count: return 1
6     elif a_count > b_count: return -1
7     else:
8         if a < b: return -1
9         elif a == b: return 0
10        else: return 1
11
12 X = list(range(20))
13 print(sorted(X, key = cmp_to_key(compare_by_factors)))
```

```
[12, 18, 16, 6, 8, 10, 14, 15, 4, 9, 2, 3, 5, 7, 11, 13, 17, 19, 1, 0]
```

```
In [439]: 1 X = list(range(20))
2
3
4 def custom_key(n):
5     return (-factor_count(n),n)
6 print(sorted(X,key = custom_key))
```

```
[12, 18, 16, 6, 8, 10, 14, 15, 4, 9, 2, 3, 5, 7, 11, 13, 17, 19, 1, 0]
```

```
In [ ]: 1
```

Asked in:

**Stuck somewhere?**

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given an array A of non-negative integers, arrange them such that they form the largest number.

Note: The result may be very large, so you need to return a string instead of an integer.

In []:

1

In [446]:

```
1 A = [3,30,34,5,9]
2
3 from functools import cmp_to_key
4
5 def largest_number(A):
6     n = len(A)
7
8     def comparator(a,b):
9         x = int(str(a)+str(b))
10        y = int(str(b)+str(a))
11
12        if x > y :
13            return -1
14        elif x < y :
15            return 1
16        else:
17            return 0
18
19    x = sorted(A,key = cmp_to_key(comparator))
20    ans = ""
21    for l in x:
22        ans += str(l)
23    return int(ans)
24
25 largest_number(A)
```

Out[446]: 9534330

In []:

1

In []:

1

In []:

1

```
In [ ]: 1
```

. Toggle Case </>

 Solved



Problem Description

You are given a character string **A** having length **N**, consisting of only lowercase and uppercase latin letters.

You have to toggle case of each character of string **A**.
For e.g 'A' is changed to 'a', 'e' is changed to 'E', etc.

```
In [10]: 1 # toggle the case
2
3 def toggle_case(s):
4     listring = list(s)
5
6     n = len(listring)
7
8     for i in range(n):
9
10         if 65 <= ord(listring[i]) <= 90:
11             listring[i] = chr(ord(listring[i]) + 32)
12
13         elif 97 <= ord(listring[i]) <= 122:
14             listring[i] = chr(ord(listring[i]) - 32)
15
16
17     return "".join(listring)
18
19
20 s = "Python Is Awesome !! codE?"
21 toggle_case(s)
```

Out[10]: 'pYTHON iS aWESOME !! CODe?'

In []:

1

In [12]:

```
1 # toggle the case
2
3 def toggle_case(s):
4     listring = list(s)
5
6     n = len(listring)
7
8     for i in range(n):
9
10         if (65 <= ord(listring[i]) <= 90) or (97 <= ord(listring[i]) <= 122): # as
11             listring[i] = chr(ord(listring[i]) ^ 32) # we can toggle
12
13
14     return "".join(listring)
15
16
17 s = "Python Is Awesome !! codE?"
18 toggle_case(s)
```

Out[12]: 'pYTHON iS aWESOME !! CODe?'

In []:

1

```
In [ ]: 1
```

Problem Description

Given a string A of size N, find and return the **longest palindromic substring** in A.

Substring of string A is $A[i \dots j]$ where $0 \leq i \leq j < \text{len}(A)$

Palindrome string:

A string which reads the same backwards. More formally, A is palindrome if $\text{reverse}(A) = A$.

Incase of conflict, return the substring which occurs first (with the least starting index).

```
In [65]: 1 def longestPalindrome(A):
2     maxlen = -float('inf')
3     n = len(A)
4     ans = None
5     for s in range(n):
6         for e in range(s,n):
7             x = A[s:e+1]
8             if x == x[::-1]:
9                 length = e-s+1
10                if length > maxlen:
11                    maxlen = length
12                    ans = x
13    return (ans)
```

```
In [66]: 1 longestPalindrome("abacaabaxa")
```

```
Out[66]: 'aba'
```

`longest = ""`
 for i in range(n):
 $\quad l = 0$ \nearrow out of bounds check palindrome
 \quad while $i - l \geq 0$ and $i + l < n$ and $A[i-l] == A[i+l]$:
 $\quad \quad l += 1$
 \quad if $2 * l + 1 > \text{len}(\text{longest})$:
 $\quad \quad \quad \text{longest} = A[i-l:i+l+1]$

 \quad if $i > 0$ and $A[i] == A[i-1]$:
 $\quad \quad \quad \# (i-1, i)$ are the mid
 $\quad \quad l = 0$
 $\quad \quad$ while $i - l - 1 \geq 0$ and $i + l < n$ and
 $\quad \quad \quad A[i-1-l] == A[i+l]$:
 $\quad \quad \quad l += 1$
 $\quad \quad$ if $2 * l + 2 > \text{len}(\text{longest})$
 $\quad \quad \quad \quad \quad \text{longest} = A[i-1-l:i+l+1]$

```
In [155]: 1 def longest_palindromic_substring(A):
2     longest = ""
3     n = len(A)
4     ans = []
5     for i in range(n):
6
7         l = 0
8         while i - l >= 0 and i + l < n and A[i-l] == A[i+l]:
9             l += 1
10        if 2*l + 1 > len(longest):
11            longest = A[i-l+1:i+l]
12
13
14        elif i > 0 and A[i] == A[i-1]:
15            l = 0
16            while i-1-l >= 0 and i+l < n and A[i-1-l] == A[i+l]:
17                l += 1
18                if 2*l + 2 > len(longest):
19                    longest = A[i-1 : i+l+1]
20
21 return longest
```

```
In [156]: 1 longest_palindromic_substring("abcdcbxaaxbcd")
```

```
Out[156]: 'dcbxaaxbcd'
```

```
In [ ]: 1
```

```
In [157]: 1 longest_palindromic_substring("abbcccbbcaaccbababcabcabca")
```

```
Out[157]: 'cbababc'
```

```
In [158]: 1 longest_palindromic_substring("aaaabaaa")
```

```
Out[158]: 'aaabaaa'
```

```
In [ ]: 1
```

Q1. Amazing Subarrays </> ✔ Solved



You are given a string **S**, and you have to find all the **amazing substrings** of **S**.

An amazing Substring is one that starts with a **vowel** (a, e, i, o, u, A, E, I, O, U).

```
In [67]: 1 def amazing_subarray(A):
2     n = len(A)
3
4     count = 0
5     x = n
6     for i in range(n):
7         if A[i] in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:
8             count += x
9             x = x-1
10    return count
```

```
In [68]: 1 amazing_subarray("ABEC")
```

```
Out[68]: 6
```

```
In [ ]: 1
```

Problem Description

Find the number of occurrences of **bob** in string **A** consisting of lowercase English alphabets.

```
In [69]: 1 def count_occurrences(A):
2     n = len(A)
3     count = 0
4     for i in range(n-2):
5         if A[i] == "b" and A[i+1] == "o" and A[i+2] == "b":
6             count += 1
7     return count
```

```
In [70]: 1 count_occurrences("bobejhshajboajbobobo")
```

Out[70]: 3

```
In [ ]: 1
```

```
In [73]: 1 S = "aabbaa"
2 def isPalindrom(S):
3     i = 0
4     j = len(S)-1
5     palindrom = True
6     while i < j:
7         if S[i] != S[j]:
8             palindrom = False
9             break
10        i += 1
11        j -= 1
12
13    return palindrom
14
15 isPalindrom(S)
```

Out[73]: True

```
In [75]: 1 isPalindrom("aasd")
```

Out[75]: False

```
In [ ]: 1
```

```
In [ ]: 1
```

Q5. Longest Common Prefix </>

 Solved

Problem Description

Given the array of strings **A**, you need to find the longest string **S**, which is the prefix of **ALL** the strings in the array.

The longest common prefix for a pair of strings **S1** and **S2** is the longest string **S** which is the prefix of both **S1** and **S2**.

Example: the longest common prefix of "abcdefgh" and "abcefgh" is "abc".

```
A = ["abcdefgh", "aefghijk", "abcefgh"]
```

```
In [159]: 1 # ans : a
2
3
4 # A = ["abab",
5   # "ab",
6   # "abcd"]
7
8 # ans = ab
```

```
In [179]: 1 def longestCommonPrefix(A):
2
3     def find_index(A):
4         n = len(A)
5         m = len(A[0])
6         ans = None
7         for j in range(len(A[0])):
8
9             ch = A[0][j]
10
11            for i in range(n):
12                if j >= len(A[i]):
13                    return j
14                if A[i][j] != ch:
15                    return j
16
17
18    return A[0][:find_index(A)]
```

```
In [180]: 1 longestCommonPrefix([
2           "abab",
3           "ab",
4           "abcd"])
```

```
Out[180]: 'ab'
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

hashmaps

```
In [184]: 1 A = [1,2,3,1,2,3,4,5,6,4,5,6,7,8,9,7,8,9,9,8,7,  
2     6,9,5,4,3,2,1,6,5,4,8,7,9,4,8,4,7,4,6,1,3,5,4,6,5,7,4,9,8]  
3  
4 freq = {}  
5
```

```
In [185]: 1  
2 for i in A:  
3     if i in freq:  
4         freq[i] += 1  
5     else:  
6         freq[i] = 1  
7 print(freq)  
8  
9
```

```
{1: 4, 2: 3, 3: 4, 4: 9, 5: 6, 6: 6, 7: 6, 8: 6, 9: 6}
```

```
In [186]: 1 for k in freq:  
2     print(k,freq[k])  
3  
4
```

```
1 4  
2 3  
3 4  
4 9  
5 6  
6 6  
7 6  
8 6  
9 6
```

```
In [187]: 1 for k,v in freq.items():  
2     print(k,v)  
3  
4
```

```
1 4  
2 3  
3 4  
4 9  
5 6  
6 6  
7 6  
8 6  
9 6
```

```
In [188]: 1 sq = []
2
3 for i in range(1,16):
4     sq[i] = i*i
5
6
7
```

```
Out[188]: {1: 1,
2: 4,
3: 9,
4: 16,
5: 25,
6: 36,
7: 49,
8: 64,
9: 81,
10: 100,
11: 121,
12: 144,
13: 169,
14: 196,
15: 225}
```

```
In [189]: 1 sq[10]
2
3
```

```
Out[189]: 100
```

```
In [190]: 1 asd = {i:i*i for i in range(1,16)}
2
3
```

```
In [191]: 1 asd
2
3
```

```
Out[191]: {1: 1,
2: 4,
3: 9,
4: 16,
5: 25,
6: 36,
7: 49,
8: 64,
9: 81,
10: 100,
11: 121,
12: 144,
13: 169,
14: 196,
15: 225}
```

```
In [192]: 1 list(asd.values())
```

```
Out[192]: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225]
```

```
In [ ]: 1
```

In []: 1

ans the queries

In [193]: 1 A = [1,3,4,-9,1,0,3,5,6,3,3,1,9]
2 Q = [3,4,0,-9,3]

In [194]: 1 freq = {}
2
3 for x in A:
4 freq[x] = freq.get(x,0)+1
5 print(freq)

{1: 3, 3: 4, 4: 1, -9: 1, 0: 1, 5: 1, 6: 1, 9: 1}

In [196]: 1 for q in Q:
2 print(q,freq[q])

3 4
4 1
0 1
-9 1
3 4

better ways to generate dictionaries

In [197]: 1 from collections import defaultdict
2
3 freq = defaultdict(int)
4 for x in A:
5 freq[x] += 1
6 freq

Out[197]: defaultdict(int, {1: 3, 3: 4, 4: 1, -9: 1, 0: 1, 5: 1, 6: 1, 9: 1})

In []: 1

In [198]: 1 from collections import Counter
2
3 freq = Counter(A)
4 freq

Out[198]: Counter({1: 3, 3: 4, 4: 1, -9: 1, 0: 1, 5: 1, 6: 1, 9: 1})

In []: 1

In []: 1

```
In [202]: 1 names = ["Sunny", "Abhi", "Uttam", "Sunny",
2           "Srikar", "Mehul", "Sunny", "Urvi", "Abhi", "Sunny", "Prithvi", "Bhavya"]
3
4 freq = defaultdict(set)
5
6 for x in names:
7     s = x[0]
8     freq[s].add(x)
9 freq
```

```
Out[202]: defaultdict(set,
{'S': {'Srikar', 'Sunny'},
 'A': {'Abhi'},
 'U': {'Urvi', 'Uttam'},
 'M': {'Mehul'},
 'P': {'Prithvi'},
 'B': {'Bhavya'}})
```

```
In [215]: 1 names = ["Sunny", "Abhi", "Uttam", "Sunny",
2           "Srikar", "Mehul", "Sunny", "Urvi", "Abhi", "Sunny", "Prithvi", "Bhavya"]
3
4 freq = defaultdict(list)
5
6 for x in names:
7     s = x[0]
8     freq[s].append(x)
9 freq
```

```
Out[215]: defaultdict(list,
{'S': ['Sunny', 'Sunny', 'Srikar', 'Sunny', 'Sunny'],
 'A': ['Abhi', 'Abhi'],
 'U': ['Uttam', 'Urvi'],
 'M': ['Mehul'],
 'P': ['Prithvi'],
 'B': ['Bhavya']})
```

In []: 1

In []: 1

count the distinct elements in array

```
In [232]: 1 A = [1, 3, 4, -9, 1, 55, -77, 0, 3, 5, 6, 3, 3, 1, 9, 91]
2 n = len(A)
3 count = 0
4 for i in range(n):
5     is_first = True
6     for j in range(i):
7         if A[j] == A[i]:
8             is_first = False
9             break
10    if is_first:
11        count += 1
12 print(count)
13
```

```
In [233]: 1 print(len(set(A)))
```

```
11
```

```
In [236]: 1 freq = Counter(A)
```

```
In [235]: 1 len(freq)
```

```
Out[235]: 11
```

```
In [ ]: 1
```

Non repeating elements

return first non repeating

```
In [248]: 1 A= [1,3,4,-9,1,0,3,5,6,3,3,1,9,4,-9,0,5,6,9]
```

```
In [250]: 1 def return_first_non_repeating_element(A):
2     ans = []
3     for x in A:
4         c = 0
5         for y in A:
6             if x == y:
7                 c+= 1
8             if c == 1:
9                 return x
10    return -1
11
```

```
In [254]: 1 A= [1,3,2,3,2,1]
2 return_first_non_repeating_element(A)
```

```
Out[254]: -1
```

```
In [253]: 1 A = [1,3,4,-9,1,55,-77,0,3,5,6,3,3,1,9,91]
2
3 return_first_non_repeating_element(A)
```

```
Out[253]: 4
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [256]: 1 def return_first_non_repeating_element(A):
2     freq = Counter(A)
3     for x in A:
4         if freq[x] == 1:
5             return x
6     return -1
```

```
In [257]: 1 A = [1,3,4,-9,1,55,-77,0,3,5,6,3,3,1,9,91]
2
3 return_first_non_repeating_element(A)
```

Out[257]: 4

```
In [258]: 1 A= [1,3,2,3,2,1]
2 return_first_non_repeating_element(A)
```

Out[258]: -1

first repeating array

Q2. First Repeating element </> Solved

Asked in: DS GS



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given an integer array **A** of size **N**, find the first repeating element in it.

We need to find the element that occurs **more than once** and whose index of the first occurrence is the **smallest**.

If there is no repeating element, return -1.

```
In [263]: 1 def first_repeating_elemanet(A):
2     freq = Counter(A)
3     for x in A:
4         if freq[x] > 1:
5             return x
6     return -1
```

```
In [264]: 1 A = [10, 7, 3, 4, 3, 5, 6,7]
2 first_repeating_elemanet(A)
```

Out[264]: 7

In []: 1

Q4 - Subarray Sum

Given an array of integers, check if there exists a subarray with sum 0.

Q3. Sub-array with 0 sum </> Solved

Asked in: MA PA



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given an array of integers **A**, find and return whether the given array contains a non-empty subarray with a sum equal to 0.

If the given array contains a sub-array with sum zero return 1, else return 0.

In [269]:

```
1
2
3 def subarray_sum_0(A):
4     n = len(A)
5     ans = []
6     for s in range(n):
7         sm = 0
8         for e in range(s,n):
9             sm += A[e]
10            if sm == 0:
11                ans.append(A[s:e+1])
12
13
14
```

In [270]:

```
1 A = [2,2,1,-3,4,3,1,-2,-3]
2 subarray_sum_0(A)
```

Out[270]: [[2, 1, -3], [-3, 4, 3, 1, -2, -3]]

In []:

```
1
```

```
In [311]: 1 def subarray_sum_0_(A):
2     for i in A:
3         if i == 0:
4             return 1
5
6     if sum(A) == 0:
7         return 1
8
9     seen = set()
10
11    prefix_sum = 0
12    seen.add(prefix_sum)
13    for x in A:
14        prefix_sum += x
15        if prefix_sum in seen:
16            return 1
17        seen.add(prefix_sum)
18
19    return 0
```

```
In [312]: 1 subarray_sum_0_([1,2,3,4,5])
```

```
Out[312]: 0
```

```
In [315]: 1 subarray_sum_0_([0,5])
```

```
Out[315]: 1
```

```
In [314]: 1 subarray_sum_0_([-5,5])
```

```
Out[314]: 1
```

```
In [323]: 1 def subarray_with_sum_0(A):
2     n = len(A)
3
4     current_sum = 0
5     prefixmap = set()
6
7     for i in range(n):
8         current_sum += A[i]
9         if A[i] == 0 or current_sum == 0 or current_sum in prefixmap:
10            return 1
11        else:
12            prefixmap.add(current_sum)
13
14
15
16 A = [2,2,1,3,4,-3,-1,2,3]
17 subarray_with_sum_0(A)
```

```
Out[323]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

In []:

1

Q1. Common Elements </>

Solved

**Problem Description**

Given two integer arrays, **A** and **B** of size **N** and **M**, respectively. Your task is to find all the **common elements** in both the array.

NOTE:

- Each element in the result should appear as many times as it appears in both arrays.
- The result can be in any order.

In [350]:

```

1 A = [1, 2, 2, 1]
2 B = [2, 3, 1, 2]
3
4 freqA = Counter(A)
5 freqB = Counter(B)
6
7 intersection = list(set(A).intersection(set(B)))
8 intersection

```

In [345]:

```
1 freqA, freqB
```

Out[345]: (Counter({1: 2, 2: 2}), Counter({2: 2, 3: 1, 1: 1}))

In []:

1

In [349]:

```

1 counts = -1
2 finalans = []
3 for e in intersection:
4     counts = (min(freqA[e], freqB[e]))
5     for _ in range(counts):
6         finalans.append(e)
7
8 print(intersection)
9 print(finalans)

```

```
[1, 2]
[1, 2, 2]
```

neat code

```
In [525]: 1 from collections import Counter
2 def common_elements(A,B):
3     freqA = Counter(A)
4     freqB = Counter(B)
5     intersection = list(set(A).intersection(set(B)))
6     print(intersection)
7     counts = -1
8     finalans = []
9     for e in intersection:
10         counts = (min(freqA[e],freqB[e]))
11         for _ in range(counts):
12             finalans.append(e)
13
14 return (finalans)
```

```
In [526]: 1 A = [1, 2, 2, 1, 2, 3, 5]
2 B = [2, 3, 1, 2, 2]
3 common_elements(A,B)
```

[1, 2, 3]

Out[526]: [1, 2, 2, 2, 3]

In []:

1

Q4. Shaggy and distances </> Solved



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Shaggy has an array **A** consisting of N elements. We call a pair of distinct indices in that array a special if elements at those indices in the array are equal.

Shaggy wants you to find a special pair such that the distance between that pair is minimum. Distance between two indices is defined as $|i-j|$. If there is no special pair in the array, then return -1.

```
In [467]: 1 A = [7,1,3,4,1,7,3]
2
3 indexmap = defaultdict(list)
4 for i in range(len(A)):
5     lst = indexmap[A[i]]
6     lst.append(i)
7 print(indexmap)
8
9 ans = []
10 for k,v in indexmap.items():
11     if len(v) == 2:
12         ans.append(v)
13 print(ans)
14 final = []
15 for i in range(len(ans)):
16     for j in range(len(ans[i])):
17         x = ans[i][0]
18         y = ans[i][-1]
19         final.append(abs(x-y))
20 print(min(final))
21
22
```

```
defaultdict(<class 'list'>, {7: [0, 5], 1: [1, 4], 3: [2, 6], 4: [3]})  
[[0, 5], [1, 4], [2, 6]]  
3
```

In []:

1

In [425]:

1
2

In []:

1

In [468]:

```
1 A = [7,1,3,4,1,7,3]
2
3 indexmap = defaultdict(list)
4
5 for i in range(len(A)):
6     lst = indexmap[A[i]]
7     lst.append(i)
8 print(indexmap)
9 minlen = float('inf')
10
11 for x in indexmap:
12
13     for j in range(len(indexmap[x])):
14         if len(indexmap[x]) == 2 :
15             minlen = min(minlen, abs(indexmap[x][0]-indexmap[x][1]))
16     print(x,minlen)
17
18 print()
19 print(minlen)
```

```
defaultdict(<class 'list'>, {7: [0, 5], 1: [1, 4], 3: [2, 6], 4: [3]})  
7 5  
1 3  
3 3  
4 3
```

```
3
```

In []:

```
1
```

In []:

```
1
```

Q5. Largest Continuous Sequence Zero Sum </>

 Solved



Asked in:  



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given an array **A** of **N** integers.

Find the largest continuous sequence in a array which sums to zero.

```
In [492]: 1 def longest_con_sq_zero_sm(A):
2     n = len(A)
3     istart = None
4     iend = None
5     maxlen = -float('inf')
6     for s in range(n):
7         sum = 0
8         for e in range(s,n):
9             sum += A[e]
10            length = e-s+1
11            if sum == 0:
12                if length > maxlen:
13                    maxlen = length
14                    istart = s
15                    iend = e
16
17        if istart == None:
18            return []
19
20    return (A[istart:iend+1])
```

```
In [494]: 1 longest_con_sq_zero_sm([1,2,-2,4,-4])
```

```
Out[494]: [2, -2, 4, -4]
```

```
In [ ]: 1
```

```
In [44]: 1 def lszero(A):
2     N = len(A)
3     pre = [None]*N
4     #     print(pre)
5     curr = 0
6     for i in range(N):
7         curr += A[i]
8         pre[i] = curr
9     #     print(pre)
10
11    from collections import defaultdict
12    seen_table = defaultdict(list)
13
14    seen_table[0]=[-1]
15    longest = -1
16    indices = None
17    for i in range(N):
18        s = pre[i]
19    #     print(s)
20        seen_table[s].append(i)
21    #     print(seen_table)
22        gap = seen_table[s][-1] - seen_table[s][0]
23        if gap > longest:
24            longest = gap
25            indices = (seen_table[s][0]+1,seen_table[s][-1]+1)
26
27    if indices:
28        return (A[indices[0]:indices[1]])
29    else:
30        return ([])
```

```
In [45]: 1 lszero([1,2,-2,4,-4])
```

Out[45]: [2, -2, 4, -4]

```
In [ ]:
```

Q1. K Occurrences </> Solved



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Groot has N trees lined up in front of him where the height of the i'th tree is denoted by H[i].

He wants to select some trees to replace his broken branches.

But he wants uniformity in his selection of trees.

So he picks only those trees whose heights have frequency K.

He then sums up the heights that occur K times. (He adds the height only once to the sum and not K times).

But the sum he ended up getting was huge so he prints it modulo 10^{9+7} .

In case no such cluster exists, Groot becomes sad and prints -1.

```
In [498]: 1 A = [1,2,2,3,3]
2 k = 2
3 freq = Counter(A)
```

```
In [499]: 1 freq
```

Out[499]: Counter({1: 1, 2: 2, 3: 2})

```
In [500]: 1 ans = 0
2 for i in freq:
3     if freq[i] == k:
4         ans += i
5 ans
```

Out[500]: 5

```
In [504]: 1 def getSum( k, A):
2
3     from collections import Counter
4
5     freq = Counter(A)
6
7     lists = freq.values()
8
9     if k not in lists:
10        return -1
11
12     ans = 0
13     for x in freq:
14         if freq[x] == k:
15             ans += x
16     return ans
17
18 getSum(2,[1,2,2,3,3,3,2,5,5,6,6])
19
20
```

Out[504]: 11

In []: 1

In []: 1

Q2. Check Palindrome - II </> Solved



Asked in: MS



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given a string **A** consisting of lowercase characters.

Check if characters of the given string can be rearranged to form a **palindrome**.

Return 1 if it is possible to rearrange the characters of the string **A** such that it becomes a palindrome else return 0.

```
In [513]: 1 def check_palindorm(A):
2     x = list(A)
3     x
4
5     freq = Counter(x)
6     print(freq)
7
8     oddfreq = 0
9
10    for i in freq:
11        if freq[i] % 2 != 0 :
12            oddfreq += 1
13    print(oddfreq)
14    if oddfreq <= 1:
15        return 1
16    return 0
```

```
In [516]: 1 A= "abbaee"
2
3 check_palindorm(A)
```

Counter({'a': 2, 'b': 2, 'e': 2, 'A': 1})
1

Out[516]: 1

```
In [529]: 1 A= "asdffffwasd"
2 check_palindorm(A)
```

Counter({'f': 3, 'a': 2, 's': 2, 'd': 2, 'w': 1})
2

Out[529]: 0

In []: 1

Q3. Colorful Number </> ✓ Solved



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given a number **A**, find if it is **COLORFUL** number or not.

If number **A** is a **COLORFUL** number return **1** else, return **0**.

What is a **COLORFUL** Number:

A number can be broken into different contiguous sub-subsequence parts. Suppose, a number 3245 can be broken into parts like 3 2 4 5 32 24 45 324 245. And this number is a COLORFUL number, since product of every digit of a contiguous subsequence is different.

In [538]:

```
1 Ax = 3245
2
3 def colorful(Ax):
4     A = []
5     while Ax > 0:
6         d = Ax % 10
7         A.append(d)
8         Ax //= 10
9     A.reverse()
10    n = len(A)
11    seen = set()
12    for s in range(n):
13        product = 1
14        for e in range(s,n):
15            product *= A[e]
16            if product in seen:
17                return 0
18            seen.add(product)
19    return 1
20
21
22 colorful(Ax)
```

Out[538]: 1

In []:

1

Q1. Subarray with given sum </> ✓ Solved



Asked in:



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

Given an array of positive integers A and an integer B, find and return first continuous subarray which adds to B.

If the answer does not exist return an array with a single element "-1".

First sub-array means the sub-array for which starting index is minimum.

```
In [38]: 1 A = [1, 2, 3, 4, 5]          # two pointer technique , starting from index 0
2 B = 5
3
4
5 def subarray_with_given_sum(A,B):
6     n = len(A)
7     p1 = 0
8     p2 = 0
9
10    ans_array = []
11    total = A[0]
12    xx = []
13    flag = False
14
15    while p1 < n and p2 < n:
16
17        if total == B:
18            flag = True
19            break
20
21        elif total < B:
22            if p2 +1 == n:
23                break
24            p2 += 1
25            total += A[p2]
26
27        else:
28            if p1 + 1 == n:
29                break
30            p1 += 1
31            total = total - A[p1-1]
32
33    if flag == False:
34        return -1
35
36    # xx.append(p1+1)
37    # xx.append(p2+1)
38    # return xx
39    for x in range(p1,p2+1):
40        ans_array.append(A[x])
41    return ans_array
42
43
44
45
46
47 subarray_with_given_sum(A,B)
```

Out[38]: [2, 3]

In []:

1

```
In [39]: 1 arr = [1,2,3,4,5,6,7,8,9,10]
2 n = len(arr)
3 s = 15
4 subarray_with_given_sum(arr,s)
```

Out[39]: [1, 2, 3, 4, 5]

```
In [40]: 1 arr = [1,2,3,7,5]
2 n = len(arr)
3 s = 12
4 subarray_with_given_sum(arr,s)
```

```
Out[40]: [2, 3, 7]
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [547]: 1 def subarray_with_given_sum_brute_force(A, k):
2
3     n = len(A)
4     x = None
5     y = None
6     for s in range(n):
7         sum = 0
8         for e in range(s,n):
9             sum += A[e]
10            if sum == k:
11                x = s
12                y = e+1
13
14            return A[x:y]
15        return -1
16 A = [1, 2, 3, 4, 5]
17 B = 5
18
19 subarray_with_given_sum_brute_force(A, B)
```

```
Out[547]: [2, 3]
```

```
In [548]: 1 def subarysm_k_uttam(A,k):
2
3     if sum(A) == k:
4         return A
5
6     prefix = [0]
7     for x in A:
8         prefix.append(x+prefix[-1])
9     print(prefix)
10
11     for b in prefix:
12         a = b-k
13         if a in prefix:
14             index1 = prefix.index(a)
15             index2 = prefix.index(b)
16             #         print(b-a, prefix.index(a)+1, prefix.index(b))
17             return (A[index1:index2])
18
19 subarysm_k_uttam(A, B)
```

```
[0, 1, 3, 6, 10, 15]
```

```
Out[548]: [2, 3]
```

```
In [ ]: 1
```

```
In [ ]: 1
```

In []: 1

Q4. Distinct Numbers in Window </> Solved



Asked in:



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Problem Description

You are given an array of **N** integers, A_1, A_2, \dots, A_N and an integer **B**. Return the count of distinct numbers in all windows of size **B**.

Formally, return an array of size **N-B+1** where **i'th** element in this array contains number of distinct elements in sequence $A_i, A_{i+1}, \dots, A_{i+B-1}$.

NOTE: if **B > N**, return an empty array.

```
In [465]: 1 A = [1,2,1,3,4,5,1,2,3,4,1]
2 k = 5
3
4
5 def distinct_number_in_window(A,k):
6     n = len(A)
7
8     freq = Counter(A[:k])
9     print(freq)
10
11    ans = []
12    ans.append(len(freq))
13
14    for i in range(k,n):
15
16        freq[A[i]] += 1
17        freq[A[i-k]] -= 1
18
19        if freq[A[i-k]] == 0:
20            del freq[A[i-k]]
21
22    ans.append(len(freq))
23
24
25
26 return ans
27
28
29
30
31
32 distinct_number_in_window(A,k)
```

Counter({1: 2, 2: 1, 3: 1, 4: 1})

Out[465]: [4, 5, 4, 5, 5, 4]

```
In [466]: 1 A = [1, 2, 1, 3, 4, 3]
2 B = 3
3 distinct_number_in_window(A,B)
```

Counter({1: 2, 2: 1})

Out[466]: [2, 3, 3, 2]

In []: 1

given an array , and integer k

pick any k elements such that difference of the sum of k elements and

the sum of remaining elements is minimum

In [138]:

```
1 def find_min_diff(A,k):
2     ...
3     given an array , and integer k
4     pick any k elements such that difference of the sum of k elements and
5     the sum of remaining elements is minimum
6     ...
7
8     def binarytodecimal(s):    # binary string
9         s = str(s)
10        i = 0
11        result = 0
12        for d in s[::-1]:
13            result = result + int(d)*(2**i)
14            i = i+1
15        return result
16    def count_set_bits_right_shift(x):
17        cnt = 0
18        while x > 0:
19            cnt += x&1
20            x = x>>1
21        return cnt
22
23    n = len(A)
24    totalsbset_of_k = []
25    for i in range(2**n):
26        subset = []
27
28        s = bin(i)[2:]
29        s = s.rjust(n,"0")
30        x = 0
31        if count_set_bits_right_shift(binarytodecimal(s)) == k:
32
33            for j in s:
34                if j == "1":
35                    subset.append(A[x])
36
37                x += 1
38        totalsbset_of_k.append(subset)
39
40    mindiff = float("inf")
41    for x in totalsbset_of_k:
42        y = []
43
44        for i in A:
45            if i not in x:
46                y.append(i)
47        diff = abs(sum(x) - sum(y))
48
49        if diff < mindiff:
50            mindiff = diff
51            print(x,y)
52
53    return (mindiff)
54
55
56 A = [3,-2,4,-1,6,7,2,0]
57 k = 3
58 find_min_diff(A,k)
```

[7, 2, 0] [3, -2, 4, -1, 6]

Out[138]: 1

In []:

1

given an array , and integer k

pick any k elements such that difference of the sum of k elements and

the sum of remaining elements is maximum

In [133]:

```
1 def max_difference_k_size_subsequences(Ax,k):
2     # maximise the differnce
3
4     # one side 2 elemenrts , 6 on other
5
6     A = sorted(Ax)
7     n = len(A)
8     #print(A)
9
10    if k < n/2:
11
12        K_side = A[:k]
13        print(K_side)
14
15        rest_side = A[k:]
16        print(rest_side)
17
18        ans = abs(sum(K_side) - sum(rest_side))
19        return (ans)
20
21    else:
22        left = A[k:]
23        print(left)
24
25        right = A[:k]
26        print(right)
27
28        ans = abs(sum(left) - sum(right))
29        return (ans)
30
31
32
33
34
35
36
```

In [134]:

```
1 Ax = [-1,3,6,0,8,10,12,100,-60]
2 k = 7
3 max_difference_k_size_subsequences(Ax,k)
```

```
[12, 100]
[-60, -1, 0, 3, 6, 8, 10]
```

Out[134]: 146

```
In [135]: 1 Ax = [1,2,3,4,5,6,7,8]
2 k = 2
3 max_difference_k_size_subsequences(Ax,k)
```

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

Out[135]: 30

```
In [136]: 1 Ax = [1,2,3,4,5,6,7,8]
2 k = 6
3 max_difference_k_size_subsequences(Ax,k)
```

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

Out[136]: 6

In []: 1

In []: 1

Q! All subset sum

```
In [144]: 1
2
3
4
5 A = [3,-6,7,0]
6 n = len(A)
7 def all_subsets_sum(A):
8     n = len(A)
9     total = 0
10    for i in range(2**n):
11        subset = []
12        s = (bin(i)[2:])
13        s = s.rjust(n,"0")
14        x = 0
15        summ = 0
16        for j in s:
17            if j == "1":
18                summ += A[x]
19                x += 1
20            total += summ
21    return total
22
23 all_subsets_sum(A)
24
25
```

Out[144]: 32

```
In [146]: 1 A = [3,-6,7,0]
2 n = len(A)
3 summ = 0
4
5 for i in A:
6     summ += (i*(2**(n-1)))
7 print(summ)
8
```

32

```
In [ ]: 1
```

```
In [ ]: 1
```

find the sum of max of all subsequence

```
In [140]: 1 A = [3,0,6,9]
2
3
4 def sum_of_max_of_subsets(A):
5     n = len(A)
6     totalsum = 0
7     for i in range(1,2**n):
8         subset = []
9         s = bin(i)[2:]
10        s = s.rjust(n,"0")
11        x = 0
12        maxel = -float('inf')
13        for j in s:
14            if j == "1":
15                maxel = max(maxel,A[x])
16
17
18        x += 1
19        totalsum += maxel
20    return totalsum
21 sum_of_max_of_subsets(A)
22
```

Out[140]: 102

```
In [149]: 1 Ax = [9,3,6,0]
2
3
4 A = sorted(Ax)
5
6 total_sum = 0
7
8 for i in range(len(A)):
9     total_sum += A[i]*2**i
10
11 print("sum of all maximum number from subsequence : ",total_sum)
12
```

sum of all maximum number from subsequence : 102

find the sum of minimum of all subsequence

```
In [154]: 1 A = [3,0,6,9]
2
3
4 def sum_of_min_of_subsets(A):
5     n = len(A)
6     totalsum = 0
7     for i in range(1,2**n):
8         subset = []
9         s = bin(i)[2:]
10        s = s.rjust(n,"0")
11        x = 0
12        minele = float('inf')
13        for j in s:
14            if j == "1":
15                minele = min(minele,A[x])
16
17
18        x += 1
19        totalsum += minele
20    return totalsum
21 sum_of_min_of_subsets(A)
22
```

Out[154]: 33

```
In [148]: 1 # for min :
2 # 9 6 6 0 0 0 3 3 3 0 0 0 0
3
4 # 0 3 6 9
5 # 8 4 2 1
6
7 Ax = [0,3,6,9]
8
9 A = sorted(Ax)
10
11 total_sum = 0
12 x = len(A)-1
13 for i in range(len(A)):
14     # print(i,2**x,A[i])
15     total_sum += A[i]*(2**x)
16
17     x = x-1
18 print("sum of all minimum number from subsequence : ",total_sum)
19
20
```

sum of all minimum number from subsequence : 33

In []: 1

In []: 1

Q3. Longest Consecutive Sequence </>  Solved

Asked in:   

Problem Description

Given an unsorted integer array **A** of size N.

Find the length of the longest set of consecutive elements from array A.

In []:

1

In [218]:

```
1 def longestConsecutive(Ax):
2
3     A = sorted(Ax)
4     n = len(A)
5     ans = []
6     maxlen = -float('inf')
7     for i in range(n-1):
8         if A[i+1] == A[i]:
9             ans.append(A[i])
10
11        elif A[i+1] == A[i]+1:
12            ans.append(A[i])
13            ans.append(A[i]+1)
14        elif A[i+1] != A[i]:
15            ans.clear()
16            maxlen = max(len(set(ans)),maxlen)
17
18    if maxlen == -float("inf"):
19        return 1
20    if maxlen == 0:
21        return 1
22    return maxlen
23
24 Ax = [50,41]
25 longestConsecutive(Ax)
```

Out[218]: 1

In [289]:

1

```
longestConsecutive([ 97, 86, 67, 19, 107, 9, 8, 49, 23, 46, -4, 22, 72, 4, 57, 111]
```

Out[289]: 6

In []:

1

```
In [283]: 1 def longestConsecutive_optimised(A):
2     freqmap = {}
3     Amax = max(A)
4     Amin = min(A)
5
6     maplist = list(range(Amin,Amax+1))
7     for i in maplist:
8         if i in A:
9             freqmap[i]= True
10        else:
11            freqmap[i]= False
12
13    anss =[]
14    maxlen = -float("inf")
15    for i in freqmap:
16        if freqmap[i]:
17            anss.append(i)
18        else:
19            anss.clear()
20
21    maxlen = max(maxlen,len(anss))
22    return maxlen
```

```
In [284]: 1 longestConsecutive_optimised([100,200,5,7,9,8,4,625,3,6,10,12,23,14,13,14,15,16,17,
```

Out[284]: 9

```
In [285]: 1 A = [ 96, 10, 19, -2, 52, 8, 79, 93, 71, 62, 48, 13, 23 ]
2 longestConsecutive_optimised(A)
```

Out[285]: 1

```
In [287]: 1 longestConsecutive_optimised([ 97, 86, 67, 19, 107, 9, 8, 49, 23, 46, -4, 22, 72,
```

Out[287]: 6

In []:

Q2. Diffk II </> Solved



Stuck somewhere?

Ask for help from a TA and get it resolved.

[Get help from TA.](#)

Given an array A of integers and another non negative integer k, find if there exists 2 indices i and j such that $A[i] - A[j] = k$, $i \neq j$.

```
In [301]: 1 A = [1,5,4,1,2]
2 k = 0
3 ans = "NO"
4 n = len(A)
5 for i in range(n):
6     target = A[i]-k
7
8     if target in A:
9         ans = "YES"
10 print(ans)
11
```

YES

```
In [ ]: 1 i - j = k
2 j = i-k
```

```
In [309]: 1 def diffpossible(A,k):
2     from collections import Counter
3
4     ans = 0
5     freq = Counter(A)
6     print(freq)
7     for x in A:
8         if k+x != x:
9             if k+x in freq:
10                 ans = 1
11                 break
12         else:
13             if freq[x] > 1:
14                 ans = 1
15                 break
16     return ans
17
18
19 A = [1,5,4,1,2]
20 k = 0
21
22 diffpossible(A,k)
```

Counter({1: 2, 5: 1, 4: 1, 2: 1})

Out[309]: 1

First Missing Integer

```
In [16]: 1 A = [3,4,-1,1]                      # O(n**2),O(1)
2 def first_missing_int(A):
3     n = len(A)
4
5     for i in range(1,n+1):
6         if i not in A:
7             return i
8
9
10
11
12 A = [3,4,-1,1,2,6,5]
13 first_missing_int(A)
```

Out[16]: 7

```
In [215]: 1 A = [3,4,-1,1]                      # O(n),O(n)
2 def first_missing_int(A):
3     n = len(A)
4     from collections import Counter
5     freq = Counter(A)
6     print(freq)
7     for i in range(1,n+1):
8         if i not in freq:
9             return i
10
11
12
13 A = [3,4,-1,1,2]
14 first_missing_int(A)
```

Counter({3: 1, 4: 1, -1: 1, 1: 1, 2: 1})

Out[215]: 5

```
In [216]: 1 A = [2,3,-7,6,8,1,-10,15]
2 first_missing_int(A)
```

Counter({2: 1, 3: 1, -7: 1, 6: 1, 8: 1, 1: 1, -10: 1, 15: 1})

Out[216]: 4

```
In [217]: 1 first_missing_int([1])
2
```

Counter({1: 1})

In [212]:

```
1 def firstMissingPositive(A):
2     j = 0
3     n = len(A)
4     for i in range(n):
5         if A[i] <= 0:
6             A[i],A[j] = A[j],A[i]
7             j+=1
8     print(A)
9     for i in range(j,n):
10        print(i)
11        if abs(A[i])-1+j < n and A[abs(A[i])-1+j] > 0:
12            A[abs(A[i])-1+j] = -A[abs(A[i])-1+j]
13    print(A)
14    for i in range(j,n):
15        if A[i] > 0:
16            return i-j+1
17
18 return n - j + 1
```

In [197]: 1 firstMissingPositive([2,3,-7,6,8,1,-10,15])

```
[-7, -10, 2, 6, 8, 1, 3, 15]
2
3
4
5
6
7
[-7, -10, -2, -6, -8, 1, 3, -15]
```

Out[197]: 4

In []: 1

In [161]: 1

In []: 1