







Simple Sieve

iven a number n, print all primes smaller than or equal to n. It is also given that n is a small number.

**Example:**

***Input :****n =10****Output :****2 3 5 7*

***Input :****n = 20****Output:****2 3 5 7 11 13 17 19*

**Sieve of Eratosthenes in 0(n) time complexity**

Auxiliary Space: O(1)

Euler's phi Algorithm

Euler’s Totient function Φ (n) for an input n is the count of numbers in {1, 2, 3, …, n} that are relatively prime to n, i.e., the numbers whose GCD (Greatest Common Divisor) with n is 1.

**Examples :**

Φ(1) = 1

gcd(1, 1) is 1

Φ(2) = 1

gcd(1, 2) is 1, but gcd(2, 2) is 2.

Φ(3) = 2

gcd(1, 3) is 1 and gcd(2, 3) is 1

Φ(4) = 2

gcd(1, 4) is 1 and gcd(3, 4) is 1

Φ(5) = 4

gcd(1, 5) is 1, gcd(2, 5) is 1,

gcd(3, 5) is 1 and gcd(4, 5) is 1

Φ(6) = 2

gcd(1, 6) is 1 and gcd(5, 6) is 1,

time complexity

O(N log N)

Space complexity

***Auxiliary Space:*** O(log N)

# Strobogrammatic number

**Strobogrammatic Number** is a number whose numeral is rotationally symmetric so that it appears the same when rotated 180 degrees. In other words, Strobogrammatic Number appears the same right-side up and upside down.

*0 after 180° rotation : (0 → 0)  
1 after 180° rotation : (1 → 1)  
8 after 180° rotation : (8 → 8)  
6 after 180° rotation : (6 →****9****)  
9 after 180° rotation : (9 →****6****)*

**Examples :**

Input : n = 2

Output : 88 11 96 69

Input : n = 4

Output : 8008 1001 9006 6009 8888 1881 9886 6889 8118 1111

9116 6119 8968 1961 9966 6969 8698 1691 9696 6699

1. Time complexity O(n)
2. Space complexity O(1), since we only put 5 key-value pairs

# Chinese Remainder Theorem

We are given two arrays num[0..k-1] and rem[0..k-1]. In num[0..k-1], every pair is coprime (gcd for every pair is 1). We need to find minimum positive number x such that:

x % num[0] = rem[0],

x % num[1] = rem[1],

.......................

x % num[k-1] = rem[k-1]

Basically, we are given k numbers which are pairwise coprime, and given remainders of these numbers when an unknown number x is divided by them. We need to find the minimum possible value of x that produces given remainders.

**Time Complexity  :**O(M), M is the product of all elements of num[] array.

**Auxiliary Space :**O(1)

Toggle the switch & Alice Apple tree

Given an array of n integers containing only 0 and 1. Find the minimum toggles (switch from 0 to 1 or vice-versa) required such the array become partitioned, i.e., it has first 0s than 1s. There should be at least one 0 in the beginning, and there can be zero or more 1s in the end.

Input: arr[] = {1, 0, 1, 1, 0}

Output: 2

Toggle the first and last element i.e.,

1 -> 0

0 -> 1

Final array will become:

arr[] = {0 0 1 1 1}

Since first two consecutive elements are all 0s

and rest three consecutive elements are all 1s.

Therefore minimum two toggles are required.

Input: arr[] = {0, 1, 0, 0, 1, 1, 1}

Output: 1

Input: arr[] = {1, 1}

Output: 1

There should be at least one 0.

Input: arr[] = {0, 0}

Output: 0

There can zero 1s.

**Time complexity:**O(n)   
**Auxiliary space:**O(n)

Binary Palindrome

a numbers with binary representation as 10..01 is palindrome and number with binary representation as 10..00 is not palindrome.

Time Complexity: O(x)

Auxiliary Space: O(1)

[Euclidean Algorithm](http://geeksforgeeks.org/euclidean-algorithms-basic-and-extended/)

 time complexity ***O(log(min(a, b))***

Karatsuba algorithm

 time complexity of **O(n^{1.59})** and a space complexity of O(n)

Block Swap Algorithm. Array rotation

It is used to rotate an array by any number of positions with a

**Time Complexity of O(N)**

**Space Complexity of O(1)**.

**Maximum Product Subarray**

Given an array that contains both positive and negative integers, find the product of the maximum product subarray. Expected

**Time Complexity:** O(N2)  
**Auxiliary Space:** O(1)

**Leaders in an array**

Write a program to print all the LEADERS in the array. An element is leader if it is greater than all the elements to its right side. And the rightmost element is always a leader. For example in the array {16, 17, 4, 3, 5, 2}, leaders are 17, 5 and 2.

**Time Complexity:** O(n\*n)

**Auxiliary Space:**O(1)

**Majority Element**

Write a function which takes an array and prints the majority element (if it exists), otherwise prints “No Majority Element”. A ***majority element*** in an array A[] of size n is an element that appears more than n/2 times (and hence there is at most one such element).

* **Time Complexity:** O(n\*n).   
  A nested loop is needed where both the loops traverse the array from start to end, so the time complexity is O(n^2).
* **Auxiliary Space:** O(1).

**Lexicographically first palindromic string**

* Difficulty Level : [Easy](https://www.geeksforgeeks.org/easy/)
* Last Updated : 21 May, 2019

Rearrange the characters of the given string to form a lexicographically first palindromic string. If no such string exists display message “no palindromic string”.

**Time Complexity :** O(n)

The average time complexity of quick sort is O(N log(N)).

[Selection Sort](http://geeksquiz.com/selection-sort/) θ(n^2)

[Quick Sort](http://geeksquiz.com/quick-sort/) θ(n log(n))

# Count all sub-strings with weight of characters atmost K

Given a string **P** consisting of small English letters and a string **Q** consisting of weight of all characters of English alphabet such that for all ‘i’, 0 ≤ Q[i] ≤ 9. The task is to find the total numbers of unique substring with sum of weights atmost **K**.  
**Examples:**

***Input:****P = “ababab”, Q = “12345678912345678912345678”, K = 5****Output:****7****Explanation:****The substrings with the sum of weights of individual characters ≤ 5 are:   
“a”, “ab”, “b”, “bc”, “c”, “d”, “e”****Input:****P = “acbacbacaa”, Q = “12300045600078900012345000”, K = 2****Output:****3****Explanation:****The substrings with the sum of weights of individual characters ≤ 2 are:   
“a”, “b”, “aa”*

**Time Complexity:** O(N2)

Manacher's algorithm is used to find the longest palindromic substring in any string.

* **Time complexity - O(N)*O*(*N*).**