**Competative programming**

**Time complexity:** vector.size()->O(1)

Element insertion in map -> O(logn).

While declaring an array or vector size limitation : 105 in function and 107 for global declaration.

1.Laguage: c and c++;

2.STL: Vector,stack,queue,list,set,multiset,unordered set,map,unordered,multi map,pair,nested container,iterator,algorithms,

3.Algorithms(upper\_bound,lower\_bound,sort(comparator),max\_element,min\_element,accumulate,revere,count,find,next\_permutation,prev\_permutations,functor)

Algorithms in STL:

4.Data structure.

5.Bit manipulation.

6.Sorting and searching Algorithms.

7.Number theory and combinatorics.

8.Recursion and Dynamic programming.

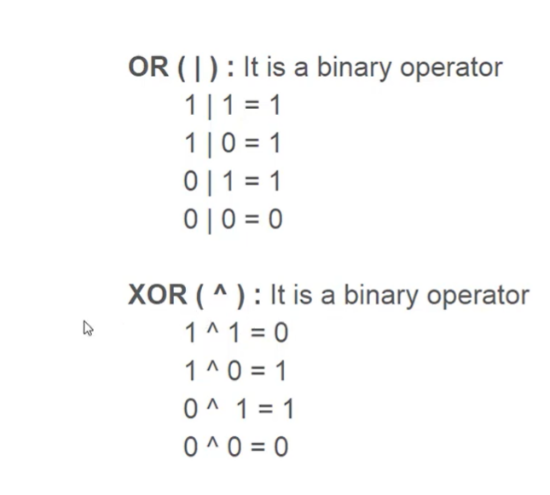
9.Implementation.

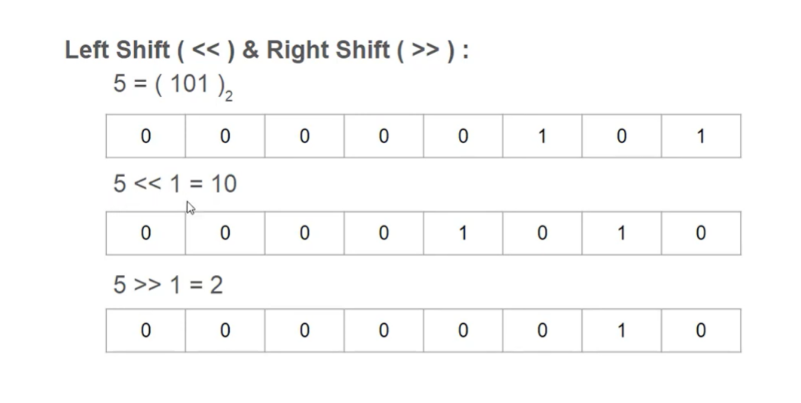
10.Algorithms(sieve of Eratosthenes,).

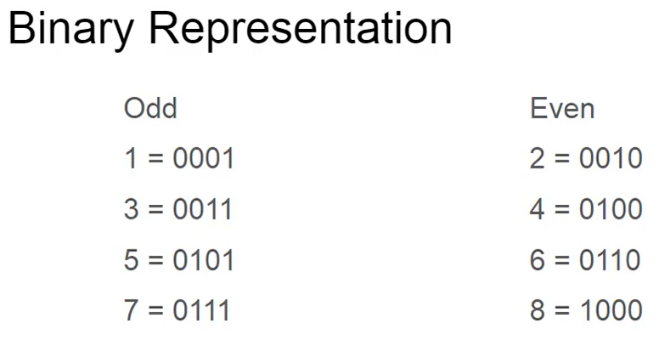
String operation: string s; s.push\_back(‘character’);

Vector operation:

* copying one vector into another in time complexity O(n).
* vector call by value and call by reference.

**Bit manipulation:**

****

**Cause of runtime error:** Accessing invalid memory in array or another.

Property of prime factor:

1.minumum divisor except one of any number is a prime number.

\*\*pow(a,b) function return double.It should not be used in competitive Programing.

* Filling something in vector->fill(vect1.begin(), vect1.end(), value);

Initialization vector with data-> **int** arr[] = { 10, 20, 30 };

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

    vector<**int**> vect(arr, arr + n);

Binary search in STL->binary\_search(**startaddress, endaddress, valuetofind**)

**Parameters** :

**startaddress**: the address of the first element of the array.

**endaddress:** the address of the next contiguous location of the last element of the array.

**valuetofind:** the target value which we have to search for.

**Returns** :true if an element equal to valuetofind is found, else false.

cout<<"By Other Method:"<<endl;

**for**(**int** i=0;i<6;i++)

        cout<<i[arr]<<" ";

String return error while checking s[i]==’\’;

If we find any x using Lower\_bound it return the address of x that occurs fisrt int the array. if it present int the array otherwise it returns the next location.if it doesn’t exist int the array it returns -1 or next address of the last element in the vector return v.end();

On the other hand upper\_bound return the address of the element that is greater than x.

=>4 MB ~ array of size 10^6 . Or 2-d array of size 10^3\*10^3

Vector initialization with value:

vector<**int**> vect;

vect.push\_back(10);

// Create a vector of size n with

// all values as 10.

vector<**int**> vect(n, 10);

vector<**int**> vect{ 10, 20, 30 };

**int** arr[] = { 10, 20, 30 };

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

vector<**int**> vect(arr, arr + n);

vector<**int**> vect1{ 10, 20, 30 };

vector<**int**> vect2(vect1.begin(), vect1.end());

vector<**int**> vect1(10);

**int** value = 5;

    fill(vect1.begin(), vect1.end(), value);

Vector\_note : while declaring vector with size.its took place in the memory with that size and with default intial value 0 . if we push back it size will be n + number of element that pushedback.

**Frequency of character or digit using vector:**

Vector<int> v(9);

Cin >> v[digit-‘0’]++;

Vector<int> v(26);

Cin >> v[charter-‘a’]++;

**Nice process to prevent the integer overflow**

1. **int mid = left + (right - left)/2;**
2. **mid>x/mid instead of mid\*mid>x**

return (ans<INT\_MIN || ans> INT\_MAX) ? 0 : ans;

static bool comp(vector<int> x,vector<int> y){

if(x[1]==y[1])

return x[0]> y[0];

else

return x[1]>y[1];

}

Cin.ignore();

getline(cin,s);

=>> condition in if() condition work from left to right;

If(a<b && c<d) first work a<b then c<d;

Min interger-> INT\_MIN;

max interger-> INT\_MAX;

Min interger->LLONG\_MIN;

Max interger-> LLONG\_MAX;

How Are Submissions Evaluated

Verdicts

Accepted

The verdict “Accepted” is what you will be aiming for. It indicates that the submission has passed all of the available test cases.

Wrong Answer

The submission failed to produce the correct output.

Compilation Error

The source code could not be compiled, possibly because of syntax error in the code.

Runtime Error

CPU Limit Exceeded

CPU Limit Exceeded is awarded when any of the test cases takes more time than the CPU limit.

Output Limit Exceeded

Output Limit Exceeded indicates that your program tried to print way too much data, more than what is necessary to solve this problem.

Memory Limit Exceeded

Memory Limit Exceeded is awarded when any of the test cases takes more memory than the memory limit.

The log2() function takes a single argument and returns a value of type double, float or long double.

The log2() function returns the base-2 logarithm of a number.

| log2() return value | |
| --- | --- |
| Parameter (x) | Return Value |
| x > 1 | Positive |
| x = 1 | Zero |
| 0 > x > 1 | Negative |
| x = 0 | -∞ (- infinity) |
| x < 0 | NaN (Not a Number) |

**Converting Strings to Numbers**

There are 3 major methods to convert a number to a string, which are as follows:

* **Using string Stream**
* **Using stoi()**
* **Using atoi()**

Integer to string conversion:

String str =to\_string(x);

class Solution {

public boolean checkIfPangram(String sentence) {

int seen = 0;

for(char c : sentence.toCharArray()) {

int ci = c - 'a';

seen = seen | (1 << ci);

}

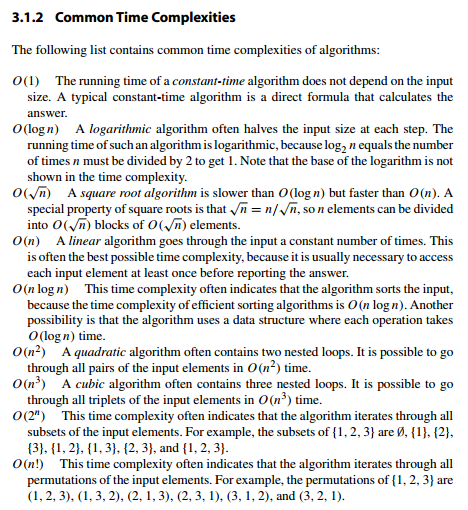
return seen == ((1 << 26) - 1);

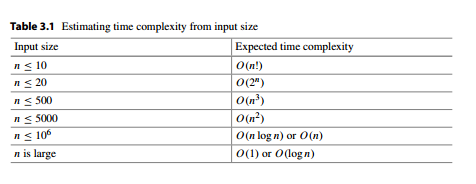
}

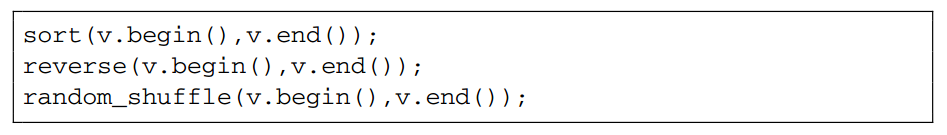
}

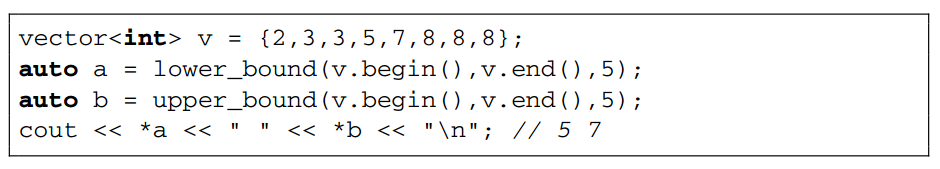
Using the and operation, we can check if a number x is even because x & 1 = 0  
if x is even, and x & 1 = 1 if x is odd. More generally, x is divisible by 2k exactly  
when x & (2k - 1) = 0.

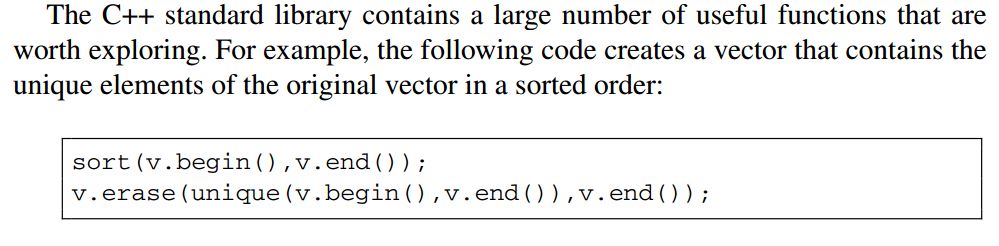
**Additional Functions** The g++ compiler also provides the following functions for  
counting bits:  
• \_\_builtin\_clz*(x)*: the number of zeros at the beginning of the bit representation  
• \_\_builtin\_ctz*(x)*: the number of zeros at the end of the bit representation  
• \_\_builtin\_popcount*(x)*: the number of ones in the bit representation  
• \_\_builtin\_parity*(x)*: the parity (even or odd) of the number of ones in the  
bit representation

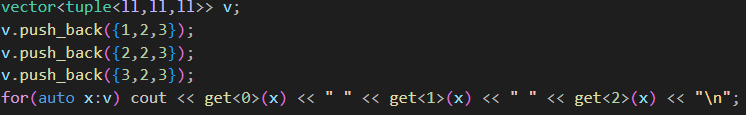


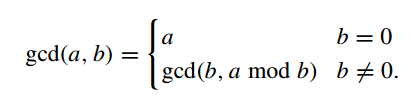


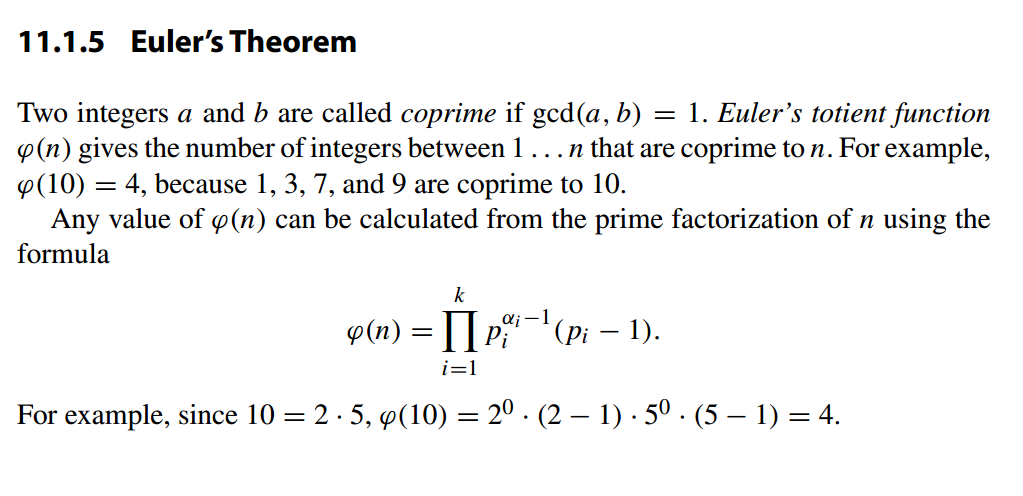












  vector<vector<**int**>> vec( n , vector<**int**> (m));

