STL Containers
(containers in STL)

containers are of 4 types

Class1

1) Sequence containers - Stores data in a sequential manner (Implement data structure which can be accessed in a sequential manner)

a) Vector
b) list (Doubly linked list)
c) deque (Doubly ended queue) (Insertion from both end)
d) array
e) forward list (Singly linked list)

Class2

2) container adaptors :- These containers although have a underlined data structure that is sequential but you cannot access all the elements of a sequence.

Eg Stack → stack is build using an array but the stack only gives the access to the top-most element.

(Provide a different interface for sequential containers)

a) queue
b) priority queue (heap (uses array))
c) stack

class3 Associative containers : That stores the data in a sorted manners.

Implement sorted data structure that can be quickly searched (O(log n)) complexity)

a) set

b) Multiset

c) map

d) multimap

class4 Unordered Associative Containers : where order doesn't matter when searching

(Implement unordered data structures that can be quickly searched).

a) Unordered set

b) unordered multiset

c) unordered map

d) unordered multimap

Algorithm (Inbuild header file and function)

STL for searching an element in an array (unsorted)

```
// #include <algorithm>
#include <iostream>
using namespace std;
int main()
{ int arr[] = {1, 10, 11, 9, 100};
  int n = sizeof(arr) / sizeof(int);
  // To search if an element is present or not in an array.
  int key;
  cin >> key;
  auto it = find(arr, arr+n, key);
  cout << it << endl; // returns the address of the key
  int index = it - arr; // key address - arrays base add gives the pos of the key
  cout << index; // returns the position
// if key is not present the index displays the size of array.
  if (index == n)
  { cout << key << "is not present"; }
  else
  return 0;
}
```

here we can also write `int* it` in place of `auto it`

STL for searching an element from sorted array

```
#include<algorithm>
#include<iostream>
using namespace std;
int main()
{ int a={1,2,3};
int n=3;
  int k;
  cin>>k;
```

To check if the element is present or not.

```
bool ypresent = binary_search(a, a+n, k);
if (ypresent) cout<<"present";
else cout<<"Absent";
```

// more function.

// 1. lowerbound.

```
auto it = lower_bound(a, a+n, k);
```

// This function will return the first address of the first
// element that is [>=] key.
// Eg a={1,2,2,2,2,3,4} (0,1,2,3,4,5,6) if the key is 2 Lowerbound
// will return the address of the first 2.

```
cout<<"\n"<<" lower bound of key "<<(it-arr);
auto it = upper_bound(a, a+n, k);
```

// this function returns strictly the no. which is strictly
// greater [>] than key.

// Eg a = { 1, 2, 2, 2, 2, 3 } , k = 2, ub = 4 (index)
          0 1 2 3 4 5
// Lb = 1st index). ub - Lb = no. of occurance of the element

cout << " upper bound of key is " << (it - a);

cout << " occurance of key = " << (ub - lb);

String Class (C++ STL)

Ly One major advantage is it provides an alternative for character arrays.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{  // Initialisizing A String.
1) string s0; // empty string
2) String s1("Helloco"); // O/P → Hello
3) string s2 = "Helloco world!"; // Hello co world!
4) string s3(s2); // Helloco coorld!
5) string s4 = s3; // Helloco woorld!
6) char a[] = {'a','b','c','\0'};
   string s5(a); // abc
// Empty funct^n
      if (s0.empty())
   { cout << "s0 is empty string";
   -3
```

// Append funct^n

1) s0.append("I love c++");
cout << s0 << endl; // I love c++

2) s0 += " and Python";
cout << s0 << "\n"; // I love c++ and Python

// length funct^n and clear funct^n

cout << s0.length() << "\n"; // 22
s0.clear(); // Erase the whole string.
cout << s0.length() << "\n"; // 0

// string compare function

s0 = ~~Mango~~ "Apple";
s1 = "Mango";
cout << s1.compare(s2) << "\n"; // -12
cout << s2.compare(s1) << "\n"; // 12
cout << s1.compare(s1) << "\n"; // 0

// compare a funct^n returns an integer == 0 if both the strings are equal

// if 1st string is lexological smaller than the other string then return some (-ve) value.

// if 1st string is larger then return (+ve) value

// Operator overloading

if (s1 > s0)
{ cout << "Mango is greater";
else cout << "Apple is greater";

// accessing the ith character of a string

cout << s[0] << "\n"; // A

// Find substring

string s = "I want to have apple juice";
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

int idx = s.find("apple");

cout << idx << "\n"; // o/p → 15 (returns starting index of the substring in the main string.

// Remove a word from the string

string word = "apple"

int len = word.length();

s.erase(idx, len+1);

// I want to have juice

// Iterate over all the character in the string

for (int i=0; i<s1.length(); i++)
{ cout << s1[i] << ":"; }

// o/p → M:a:n:g:o:

// Iterate with the iterator.

for (auto it = s1.begin(); it != s1.end(); it++)
(string::iterator it // auto can be replaced as.

{ cout << (*it) << ","; }

// o/p → M,a,n,g,o,

* it prints the address thus, * it prints the value in that address.

// For Each loop

// auto itself detect the datatype of the variable
// char, string, int, float etc.

```
for (auto c : s1)
{ cout << c << "."; } // M.a.n.g.o
```
(auto → char)

// This will iterate through loop and just like for loop.

String Sorting

```
#include <iostream>
#include <algorithm>
#include <string>
using namespace std;
int main()
{ int n;
  cin >> n;
  cin.get();
  string s[100];
  for (int i=0; i<n; i++)
  { getline(cin, s[i]); }
  for (int i=0; i<n; i++)
  { cout << s[i] << endl; }
  sort(s, s+n);
```

String Tokenization using strtok()

↳ It breaks the string into no. of tokens

Eg Today, is a raing, day

if delimeter is ','

O/P → Today
'is a raing
day

if delimeter is ' '

O/P → Today,
is
a
raing,
day

function declaration

char * strtok (char *s, , char *delimiters)

↙ the original string.

↓ It accept a string along which you want to break the particular string.

```
#include <iostream>
#include <cstring>
using namespace std;
// char * strtok ( char *s, char * delimeters)
// returns a token on every functh call
// On first call functh should be passed with string 's'
// On subsequent calls we should pass the string as NULL
int main ()
{ char s[100] = "Today is a raing day";
  char *ptr = strtok (s, " ");
  cout << ptr << endl; // Today
```

```
while (ptr != NULL)
{ ptr = strtok (NULL, " ");
  cout << ptr << endl;
}
  return 0;
}
```

Vector Introduction

(container)

↳ vector is a dynamic array

↳ It is an array that can grow and shrink in size automatically depending upon the requirements

↳ It grow by doubling its size

↳ We have to include header file `#include <vector>`

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{ // Initialising the vector also creating it
  // ty vector <datatype> vector_name
     vector <int> a;
```

2) vector <int> b(5, 10);
// there are 5 integer in a vector (array) each having
// value 10. ~~fri~~ used in initilise a vector with '0'.

3) vector <int> c(b.begin(), b.end());
// Here we created a vector c having all the element of
// b starting from begining to end.

4) vector <int> d {1, 2, 3, 10, 14, 17};
// create a vector d initialise it with given values

// How to iterate over the vector (array)

1) for (int i = 0; i < c.size(); i++)
{ cout << c[i] << ", "; }
cout << "\n";

2) // using iterators.
for (auto → vector<int>::iterator it = b.begin(); it != b.end(); it++)
{ cout << (*it) << ", "; }

3) // for each loop
for (int → auto x : d)
{ cout << x << ", "; } cout << "\n";

4) // sort the vector
sort(v.begin(), v.end());

// more function on vectors

// for accessing elements from the user and add them in

//the vector :

```
Vector <int> v;
int n;
cin>> n;
for( int i = 0; i <n; i++)
 { int num;
    cin >> num;
    v. push-back (num);
  }
```

// v. push-back add a no. at the end of the vector.

```
for (auto x: v)
{ cout <<x <<", ";
 }
```

// size of vector (How many element the vector has)

```
cout << v. size()<< "\n";
```

// size of undelined array.

```
cout << v. capacity() <<"\n";
```

// How much the vector can expand in worst case
// according to available memory in system

```
cout << v. max_size () <"\n";
```

Vector 02 - Methods

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    // create and initialize a vector
    vector<int> d{1, 2, 3, 10, 14};
    // adding an element at the end. (T = O(1))
    d.push_back(16);
    // for accessing / displaying / printing all the vector elements
    for (int x : d)
    { cout << x << ", "; }
    // O/P -> 1, 2, 3, 10, 14, 16,
    // for deleting the last element from vector
    d.pop_back();
    // O/P -> 1, 2, 3, 10, 14, , T = O(1):
    // Inserting some element in the middle
    // vectorname.insert(vectorname.begin() + Pos, element)
    d.insert(d.begin() + 3, 100);
    // It will insert the element at 3rd pos starting from 0.
    // O/P -> 1, 2, 3, 100, 10, 14,
// also, we can add more than 1 elements.
// v.insert(d.begin() + 3, no.of elements, value)
```

```
d.insert(d.begin()+3, 4, 100);
// O/P → 1, 2, 3, 100, 100, 100, 100, 10, 14
// It will insert 4 elements after pos 3 each having value 100
// time comp → O(n)

// Erase some of the elements from the middle.
   d.erase(d.begin()+3);
   // O/P → 1, 2, 3, 100, 100, 100, 10, 14,
// we can erase a range of elements
   d.erase(d.begin()+2, d.begin()+5);
// O/P → 1, 2, 100, 10, 14  This will erase 3 elements
   // size
   cout << d.size() << "\n";
   // capacity
   cout << d.capacity() << "\n";
   // resize operation (to resize the vector)
   d.resize(8);
   cout << d.capacity() << "\n";
// 1, 2, 100, 10, 14, 0, 0, 0 → make the size of the vectors
   // remove the vector
   d.clear();
```

```
// getting the 1st element of the vector
cout << d.front();
// getting the last element of the vector
cout << d.back();
// To avoid doubling, we will use reserve functh
v.reserve(10000);
// its like predifing the size of array
return 0;
}
```

LIST STL

List → Doubly linked list
(where each element is connected with next element and also previous element

↳ useful for insertion & deletion of the data from middle

imp methods

1. push_back → Insert at end (tail)
2. push_front → Insert at head
3. pop_back → Remove from the tail.
4. pop_front → Remove from the head
5. insert → Insert at any place (middle etc)
6. erase (index) → Erase some elements
7. remove (x) → It will remove all occurance of x

Code

```
#include <iostream>
#include <list> // for list funct^n

using namespace std;

int main()
{
    // Initializing and creating
    // list <int> l;
    // list <int> l1{ 1, 2, 3, 10, 8, 5 }; // Create
    // 3) Different data type
// list <string> l2 {"apple", "guava", "mango", "banana"};
// Adding element in list
// l2.push_back("pinapple");
// Iterate over the list and print the data.
for(auto s : l2)
    { cout << s << " : "; }
// Sorting the list;
    l2.sort();
// reverse the list
    l2.reverse();
// remove 1st element
    l2.pop_front;
```

```
// print the front element
cout << l2.front();
// adding to the front of the list
l2.push_front("kiwi");
cout << l2.back();
l2.pop_back();
// iterate the list using iterator
for (auto it = l2.begin(); it != l2.end();
        it++)
{ cout << (*it) << "->"; }
cout << endl;
// some more function on the list
// adding more element to the list.
l2.push_back("orange");
l2.push_back("lemon");
// printing
for (auto x : l2)
{ cout << x << ":"; }
// remove a fruit (element)
string f;
cin >> f
l2.remove(f); // remove all occurance
                // of f.
```

// erase some element at a index

// for erasing 3rd element

```
auto it = l2.begin();
 it++;
 it++;
  l2.erase(it);
for (auto s:l2)
   cout << s << ":";
  // Now list removed 3rd elemnt
```

// insert at any position

Eg // Inserting at 2nd pos.

```
it = l2.begin();
it++;
 l2.insert(it, "flute");
for (auto s: l2)
  cout<<s;
```

// Insert flute at second pos.

```
return 0;
}
```

Stack STL (LIFO)

↳ Important functn
(i) Push (d) (ii) Pop () , (iii) topel , (iv) empty()

```
#include <iostream>
#include <stack>
using namespace std;

int main ()
{
    // Initialize
    stack <int> s;
    // Inserting in stack
for (int i=0; i<5 ; i++)
 {  s.push (i); }

    // removing and displaying
    while (!s.empty())
  { cout << s.top() << " ";
    s.pop();
  }

  return 0;
}
```

Queue STL (FIFO)

```
#include <iostream>
#include <queue>
using namespace std;
int main()
{
    queue<int> q;
    // Initialize
    // Adding in the queue
    for (int i = 1; i <= 5; i++)
    { q.push(i); }
    // finding the 1st member of queue & remove it
    while (!q.empty())
    { cout << q.front() << " :";
      q.pop();
    }
return 0;
}
```

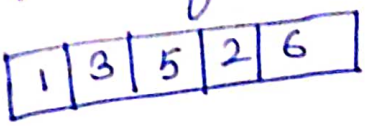
// Imp funct^n

- ↳ Push
- ↳ Pop
- ↳ front
- ↳ empty

Priority Queue STL - 01

↳ Underlined Data structure (Heap)

↳ Here we can push data & Pop data acc to a certain priority.

Eg: 

Priority statement : Higher the value, Higher the no.

When we perform pop operation

↳ 6, 5, 3, 2, 1

To push Time ↳ O(log n)

(Insert in heap)

To pop Time = O(log n)

(Deletion a node from heap)

Top → To see the topmost element -

Time - O(1)

Empty or not → O(1)

code

This is also a part of queue header file.

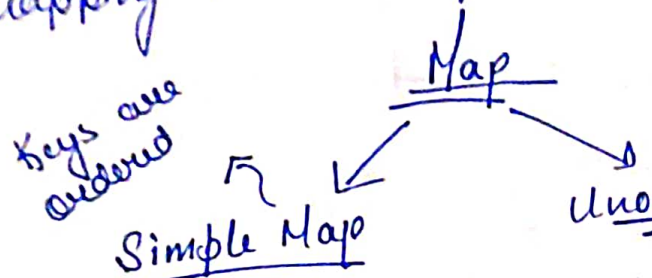
code

```
#include <iostream>
#include <queue>
using namespace std;

int main()
{
    // Initialize max heap
    priority_queue<int> pq;

    // Inserting element user defined
    for (int i=0; i<7; i++)
    {   int no;
        cin>> no;
        pq.push(no); // O(log N)
    }
    // Remove the element from heap
    while (!pq.empty())
    {   cout<< pq.top()<<" :";
        pq.pop();
    }
    // Initialize min heap
priority_queue<int, vector<int>, greater<int>> pq;
    return 0;
}
```

Priority Queue STL 02 Comparator Class (Doc)

MAP STL (Map is {[Key ↔ Value] pair})

↳ A map is an associative containers that stores value mapping between key and the value pairs



↳ Uses self balancing bst to do the work

Eg map
When we go to clg canteen & we ask for menu card there for every item there is a price.

	I/P	O/P
Eg	Maggi	₹20
	Chocolate	₹45
	Pepsi	₹35.
	Key	↔ Value

Main Operations

STL

1) Insert() insert ((K, V)) → value, ↳ Key

2) find() query (K) ↳ for a given key wat's the value.

3) erase() delete (*) → delete a particular key and value pair

MAP IMPLEMENTATION

```
#include <iostream>
#include <map>
#include <string>
using namespace std;
int main()
{ //Initializing
  map<string, int> m;
  //Inserting method 1
  m.insert(make_pair("Mango",100));

  //Inserting Method 2
  pair<string, int> p;
  p.first = "Apple";
  p.second = 120;
  m.insert(p);
  //Inserting Method 3
   m["Banana"] = 20;
  //Searching in the map
  string key;
  cin>> key;
  auto it = m.find(key);
  if(it != m.end())
  { cout<<"price of "<<key<<m[key]; }
  else
   { cout<<key<<"is not present";
```

map<string, int>::iterator it

```
// another way to find a key
// It stores unique key only once
if (m.count (key))
{ cout << "Price " << m[key]; }
else
{ cout << "fruit is not available";

// erase an particular key.
    m.erase (key);

// update the price.
    m[key] += 22;

// iterate over all key value
    m["Litchi"] = 60;
    m["pineapple"] = 100;
for (auto it = m.begin(); it != m.end(); it++)
{ cout << it -> first << " and " << it -> second;

    // for each loop

for (auto p : m)
{ cout << p.first << " : " << p.second; }

return 0;
}
```

Multimap

↳ Similar to map except
↳ it can contain elements where multiple keys can have the same values

Multimap	Map
↓	↓
It can have multiple map key	unique key elements

Eg b = bat
b = ball

```
#include <iostream>
#include <map>
#include <cstring>
using namespace std;
int main
{
    // Initialize
multimap <char, string> m;
// Inserting into map
/*  7
b banana
a apple
b ball
a angry
c cat
d dog
e elephant */
```

```
int n;
cin>>n;
for (int i=0; i<n; i++)
{ char ch;
  string s;
  cin>>ch>>s;
  m.insert(make_pair(ch,s));
}
// Try to print the entire map
for (auto p:m)
{ cout<<p.first<<" ->"<<p.second<<endl;
}
// erase a particular key
  auto it=m.begin();
  m.erase(it);
// lower bound will return key>=b.
auto it2= m.lower_bound('b');
auto it3 = m.upper_bound('d');
for (auto it4=it2; it4!=it3; it4++)
{ cout<<it4->second<<", "<<endl; }
  // search for cat
  auto f=m.find('c'); if(f->second=="cat")
{ m.erase(f); }
```

Unordered Map STL

↳ Implementation of Hash table

↳ Insertion, Deletion, find → O(1)

Implementation

```
#include <iostream>
#include <unordered_map>
using namespace std;
int main()
{
    // Initialisation
    unordered_map<string, int> um;
```

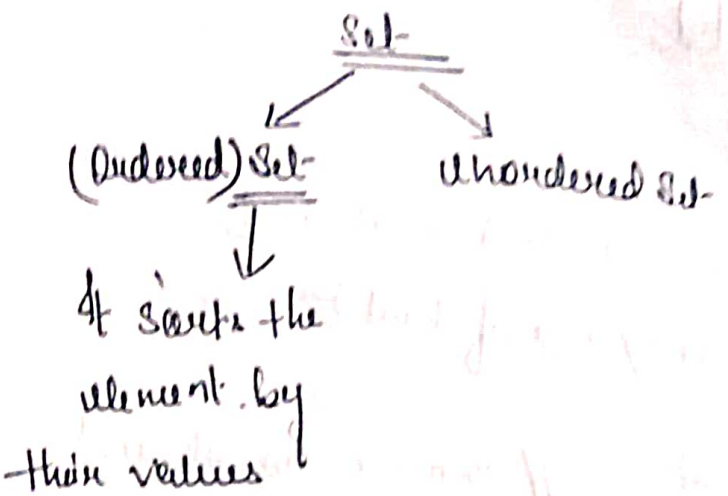
rest all the program is ditto same as map. We can exactly do the same thing with unordered map.

↳ The only difference is we will not get all element in sorted order.

```
}
```

Set (STL)

↳ It is a container that is use to store a unique collection of elements.



Eg Set {1, 2, 3, 4, 5, 1, 1, 2, 3}

O/P = {1, 2, 3, 4, 5}

↳ It will remove all duplicates and sorts all elements

Eg2 Finding Out Permutation of A number.

★ Set is implement using BST or Red Black Tree.

★ Most of the operation takes log N time

★ Once you inserted element into a set you cannot modify the element

★ If at all you want to modify then remove an element and then insert another element.

★ ↳ Insert ↳ erase ↳ Remove ↳ Iterate over element

Implementation of Set...

```
#include <iostream>
#include <set>
using namespace std;
int main()
{ int arr[] = {10, 20, 11, 9, 8, 11, 10};
  int n = sizeof(arr)/sizeof(int);
  // Initialize the set
  set<int> s;
  // Inserting into set
  for(int i=0; i<n; i++)
    { s.insert(arr[i]); }
  // Iterate over the set and print the element
for(set<int>::iterator it=s.begin(); it!=s.end(); it++)
{ cout << *(it) << ", ";
return 0;
}
```

// remove an element

// s.erase(10);

// auto it = s.find(11);
s.erase(it);

* Output will be ordered (Sorted)
and it will store only unique elements.

Multiset

↳ It is a container in STL
↳ It's a variation of set
↳ It can store multiple elements having same value
↳ It is stored in a specific order → sorted according to Internal comparision object.
↳ Values that's inserted in a multiset cannot be modified
↳ Associative container → elements are reffered by their values and not by their index.
↳ underlined implementation uses BST to implement.

Implementation

```
#include<iostream>
#include<set> // It's used for both set & multiset
Using namespace std;
→ typedef multiset<int>:: iterator It;
int main()
{
    int arr[7] = {10, 20, 30, 20, 10, 10};
    // inserting into multiset.
    multiset<int> m(arr, arr+6);
    //Iterate over all element
    for(int x: m)
    { cout << x << ","; }
    // O/P → 10, 10, 10, 20, 20, 30.
    // erase a particular element eg-20.
    m.erase(20); // all occurance of 20 are erased.
```

// for Inserting an element

m.insert(80)

// How many times an element occurs
// check the occurance of a particular element

cout << m.count(10);

// to get the iterator at a particular element

auto it = m.find(30);
cout << (*it) << endl;

// prints the first occurance of 30 ↑

// To get all the element which is equal to 30.

pair<It, It> range = m.equal_range(30);

for (auto it = range.first; it != range.second; it++)
{ cout << *it << "-"; }

// O/P => 30-30-30-30-30.

// We have to define ~~typ~~ iterator before main ⇒.

// typedef multiset<int>::iterator It;

// Lower bound and ~~two~~ upper bound.

⇓ >= ⇓ >

for (auto it = m.lower_bound(10); it != m.upper_bound(77); it++)
{ cout << *it << "-"; } // 10,10,30,30,30,30

return 0;

Policy based Data Structure

↳ Only available in c++

Topic we'll cover :

(i) Declaration (ii) Header & namespace are required

(iii) Important funct^n (diff) (iv) code (v) Window & Ubuntu

(vi) Ques in codeforces (Inversion Count).

code

```
#// Header files needed to include in cpp files

#include <ext/pb_ds/assoc_container.hpp>
```

↳ extension of policybased data structure and associative containers

```
#include <ext/pb_ds/tree_policy.hpp>
```

↳ extension of policy based datastructure and tree policy to use,

↳ this is different because it contains order statistics node object

↳ including tree_order_statistics_node_update↑

```
Using namespace __gnu_pbds;
```

↳ using namespace for GNU (operating system) for pbds (Policy based DS).

PBDS

↳ It works like a set (Has all the functionality of a set

It has 2 more unique functionality.

1) find_by_order(k) → It return the iterator to the k^{th} largest element (starting from 0). Time O(log(N))

2) order_of_key(k) → It returns the number of elements in the set which are strictly less than value k. Time O(log N)

N using binary search find the nth largest element. it takes nlog(N) time but here its possible only with O(log N) time.

Eg

```
neo_data_set S;
find_by_order(3);  → Index ⇒ 10
```

15	4
10	3
4	2
3	1
1	0

↳ this function will gives the o/p = 10

order_of_key(10); → value = 3 → returns the index.

↳ this function returns the count value of how many nos are smaller than the given value present in the set. O/P = 3.

↳ That is there are 3 elements less than 10.

If it will return the index.

↳ If the value is not present in the set then it will return of the index of next higher no. that the value.

Eg order_of_key(11); ⇒ 4 (index of 15 ∵ 11 is not present).

Policy Based Data Structure - 2

```
#include<bits/stdc++.h>

// header files for policy based data structure
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>

using namespace std;
// header files for Pb_ds
using namespace __gnu_pbds;

// defining the tree
typedef tree<int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> PBDS;
```

↓ name of set.

```
int main()
{  // Ignore => for input and output

#ifndef ONLINE_JUDGE
   freopen("input.txt", "r", stdin);
   freopen("output.txt", "w", stdout);
#endif
  // ignore

      PBDS st;
      st.insert(1);
      st.insert(13);
      st.insert(4);
      st.insert(10);
      st.insert(15);
      st.insert(6);

    for(int i=0; i<st.size(); i++)
   { cout<<i<<" "<<*st.find_by_order(i)<<"\n"; }
// returns the index and value uniquely and in sorted order.
  // Kth largest element at till now (Time O(log(N))).

     cout << st.order_of_key(5) << '\n';
// returns how many no. are smaller than the given no.

return 0;
}
```

Policy Based Datastructures - 03

INVERSION COUNT Using PBDS

↳ They count all pairs of no. in an array such that

↳ all pairs of indexes such that $i < j$, then $a[i] > a[j]$

Eg A = {0, 1} ⇒ Yes - (1)

Logic first we will traversed through all the elements that are in the left of j and also $a[i] > a[j]$;

Approaches To solve the Problems

↳ Merge sort Technique.

↳ By using Policy Based Data Structure (More Efficient).

Eg	1	5	2	3	6	4	10	no. of Inversion
Ind	0	0	1	1	0	2	0	⇒ 1+1+2 = 4

↳ Standing at a particular index looking for all index to its left if any value associated to that index is greater than the particular index.

Implementing based on PBDS

first we'll insert values till the index we want to find then by using order_of_key() function we'll know the no. of smaller element. & then

We will subtract index − order of key value.
Thus we get our required result.

Greater element = st.size() − order_of_key

↓ Total element inserted till now.

(Index)

Pseudo code

```
PBDS st;
int inversion_count = 0;
for(int i=0; i<n; i++)
{ inversion_count += (st.size() - st.order_of_key(a[i]));
  st.insert(a[i]);
}
cout << inversion_count; // o/p = 4
```

codeforces (Pair of Topics)

5
4 8 2 6 2
4 5 4 1 3

1 + 1 + 1 + 1 + 1 + 1 + 1 = 7

4
1 3 2 4 = 0
1 3 2 4

i < j , a[i] + a[j] > b[i] + b[j]

first 13 nearest Hotel.

Q -> Query , 13 -> 13 nearest hotel.

Bit-wise operator & shift operators

& AND

| OR

~ NOT

^ XOR

AND

0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1

OR

0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1

Eg a = 5, b = 7

a & b = 5

a | b = 7

a ^ b = 2

$2^2 \; 2^1 \; 2^0$

a = 5 = 101   4 2 1   101

b = 7 = 111   111

a & b = 101
& 111

101

a | b = 101
| 111

111

a ^ b = 101
^ 111

010

XOR -> (Exclusive OR)

1 ^ 1 = 0
0 ^ 0 = 0
1 ^ 0 = 1
0 ^ 1 = 1

* Zero XOR with any no. is the num. itself

7 ^ 0 = 7
8 ^ 0 = 8

5 ^ 5 = 0

5 = 101
^ 101

000

* XOR of same no. is Zero

Eg 5 ^ 5 = 0
6 ^ 6 = 0

Eg ~~5~~ ^ 7 ^ ~~5~~ = 7

~ not → flip all the bits

Eg 5 = 101

~5 = 010 = 2

Shift operators: (Binary operators)
needed 2 operator

2 types

1) Left shift (<<) 2) Right shift (>>)

1) Left shift

Here 5 << 1 → ~~value~~

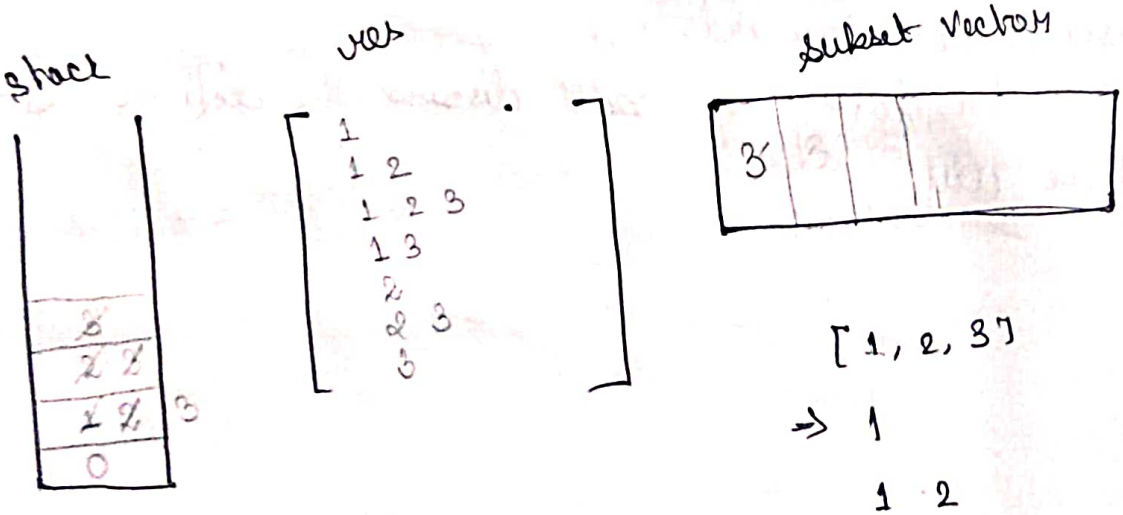
(read as 5 left shift 1)

5 << 1 → means in the binary representation of 5 which is (0 0 0 1 0 1) Here we'll shift all the representation towards left.

In doing so, we'll discard the leftmost bit and we will

Finding all the subset of an array (Backtracking)

```
void getsubset (vector<int> &v, vector<vector<int>> &res,
                vector<int> &subset, int index)
{
    for(int i = index; i < v.size(); i++)
    {
        subset.push_back(v[i]);
        res.push_back(subset);
        getSubsets(v, res, subset, i+1);
        subset.pop_back();
    }
}
```



[1, 2, 3]

→ 1

1 2

1 2 3

1 3

2

2 3

3

Problem statement

→ you will be given an array. you have to find out all the possible subsets of the given array.

Eg Array → [1, 2, 3]

possible subsets are: 1, 12, 123, 13, 2, 23, 3

Things we gonna use -

1) 1 vector to store the given array

2) A 2-D vector for storing all possible subsets

3) A index variable for keeping track of all index

4) A stack cuz we are using recursion.

5) A subset vector.

GIT AND GITHUB

In git bash

↳ git init

↓

It initialise empty git repository in a directory

work flow

1 → Make changes to a file

2 → Prepare to share the changes

3 → Share the changes

* To select a file, use command

↳ git add file_name

↓

name of the file

if you wanted to add a file to the project

shift operators

<< left shift
>> Right

if we have 5
binary representation - 000101
Then we'll shift all the bits to left then discard the 1st value & add if Zero to the end.

Eg 5<<1

binary representation
000101 ⇒ 001010 = 10 = (5×2)
← left shift by one
↳ this is our final ans

Or left shift 1 means multiplication by 2

Eg a<<1 ⇒ a*2

a<<3 = a*2*2*2

Eg a<<2 ⇒ $a * 2^2$

∴ $a << b = a * 2^b$ formula

Thus we can say that- left shift is multiplication by 2.

Right shift

$a >> b = a / 2^b$

a right shift b is $a/2^b$. In right shift- we'll discard the bit from right and add 1 zero at the left.

a = 10

000101 ⇒ 000101 ⇒ 5

∴ $a >> b = a / 2^b$

Unique Number (2N+1)

Q. Given a list of numbers where every numbers occurs twice except one, find.

Eg arr → { 1, 2, 2, 3, 3 } O/P → 1

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin>> n;
    int no;
    int ans=0;
    for (int i=0; i<n; i++)
    {
        cin >> no;
        ans = ans^ no; // Bitwise XOR
                       // 1^5^5 = 6
    }
    cout << ans << endl;
    return 0;
}
```

Bit Manipulation Grd of Set Bits

common Bit Operations

* Given an no. find wether its odd / even.

Ans Binary config of 5 → 0000101
↑
last bit = 1 (odd)

because,

in the in the binary representation of a no. the

$\ldots 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 1$ so only 2^0 is odd else all even

even | odd

Thus, to find if the no. is odd or even just do bitwise and (&) with 1.

↳ if result is 1 then odd

↳ else even.

N = 5 ⇒ (N & 1) == 1 (odd)

Eg 5 → 0000101
1 → 0000001
0000001 → odd

Pseudo code

```
bool isodd (int n)
{ return (n & 1); }

main ()
{ cout << isodd (n) << "\n"; }
```

Getting ith Bit of a number from Right

Eg N = 5, i = 2

5 => 0000101

So we can create a mask like if we wanted to know the 2nd bit from right then we'll left shift 1 by 2 places and do and (&)

Eg 5 = 0000101 i = 2

1<<2 => 0000100 (&)

0000100

→ If the Ans > 0 then Set bit (1)

→ else = = 0

↳ the bit is zero

Pseudo code

```
int getbit ( int n, int i)
{   int mask = (1 << i);
    int bit = (n & mask) > 0 ? 1 : 0;
    return bit;
}
```

```
int main ()
{ int n = 5; int i;
  cin >> i;
  cout << getbit(n, i) << "\n";
  return 0;
}
```

// How to set a particular bit at a given position.

What we basically do will be creating a mask and left shift 1 by ith bit then perform or (|) and save the ans.

N = 5 , i = 3

5 = 0000101
(k<<3) = 0001000
0001101 → Ans

if that particular bit is zero then it will become 1 then, else if it is 1 then it will remain 1.

pseudo code

```
int setBit (int n , int i)
{
    int mask = (1<< i);
    int ans = n | mask;
    return ans;
}
main()
{
    n = setbit ( n, i); }
```

BIT MANUPULATION ⇒ clear and update Bits

clear ith Bit

↳ S1 make a mask variable and assign (1<< i)
↳ S2 Negate the mask variable
↳ S3 perform and operation (&) with (N & (Mask))

Eg N = 5 0000101 i = 2
mask & 11111011
~(1<<i) 0000001 ⇒ Ans

Pseudo code (Passing n by reference)

```
void clearBit (int &n, int i)
{ int mask = ~(1<<i);
  n = n & mask;
}
int main()
{ int n=5, i=2;
  clearBit (n, i);
}
```

// update a particular bit (0/1)

↳ Here we wanted to set a value v to a particular position i

S1 First we'll clear the position given

Eg # N = 5, i = 2, v = 1

5 = 0000101 ⇒ 0000001

clear it

S2 Then make a mask variable with v (v<<i) and left shift it by i.

S3 Then do Or function with (N | mask).

Pseudo code

```
void updatebit ( int fn ,int i , int v)
{   int  mask = ~ (1 << i); // for clearing
    int  clear_num = n & mask;
    n = clear_num | (v << i); // for updating
}
    int main()
{   int n = 5, v = 0, i = 2;
    updatebit (n, 2, 0);
}
```

Bit Manipulation - Clear Range of Bits

Q clear last i bits.

Eg N = 1010 1101 1101 , i = 4

result ⇒ 1010 1100 0000 ⇒ Ans

↳ make a mask with all 1 but ith position 0.
↳ then perform and operation

pseudo code

```
int clearLastBit (int n, int i)
{  int mask = (-1<<i);
   return n & mask; }

int main()
{ int n = 15, i = 2; // from pos i+1 to 0.
  cout << clearLastBits(n,i) << "\n"; }
```

Q clear in Range of bits from i to j.

N = 0 0 1 1 1 1 1 , i = 1, j = 3

final answer ⇒ 0 0 1 0 0 0 0 1

Sol ↳ create a mask variable a like 1 1 1 1 0 0 0 0 0 (left) ①
↳ create a mask variable b like 0 0 0 0 0 0 1 1 (right)
↳ Then perform the OR operation a | b = 1 1 1 1 0 0 1 1

mask = a | b

Eg a = (~0) << (j+1)

↳ 1 1 1 1 0 0 0 0 0 (j = 5)

By to get a no. like 0 0 0 1 1 1 or some zero followed by some one then $2^n - 1$ eg = n = 4

$b = 2^n - 1 = 0\,0\,0\,1\,1\,1\,1$ ⇒ $b = 2^i - 1$

$\boxed{b = (1<<i)-1}$ ⇒ $(1<<i)-1$

```
int clearRangeItoJ (int n, int i, int j)
{
    int a = (~0) << (j+1);
    int b = (1<<i) - 1;
    int mask = a | b
    int ans = n & mask;
}   return ans;
```

```
int main()
{   int n=31, i=1, j=3;
    cout<< clearRangeItoJ(n, i, j);
    return 0;
}
```

Algo

BIT MANIPULATION

REPLACE BITS IN N BY M

Q You are given a 32-bit numbers, N & M and 2 bit position i & j. Write a method to set all bits between i and j in N equal to M (Eg M becomes a substring of N located at i & starting at j).

Eg I/P N = 10000000000

M = 10101, i = 2, j = 6

```
          j     i
          6 5 4 3 2 1 0
O/P ⇒ 1000 1 0 1 0 1 0 0
          \___M___/
```

New no.

Steps

1) clear all the bits in N in the range i to j by using clear bit funcs. (N')

2) Left shift m by i places so that zeros will be added in the end (M')

3) Perform OR operation with N' | M'.

Pseudo code

```
int replaceBits (int n, int m, int i, int j)
{ int n1 = clearRangeItoJ(n, i, j); // discussed before
  int ans = n1 | (m<<i);
  return ans;
}
```

Decimal to Binary using Bitwise

Convert the decimal no. to its binary form.

Eg N = 13 O/P = 1101

by default all the no.s are stored in Binary form in the main memory.

↳ S1 first we'll extract the bits from right

thus, 1101 ⟹ O/P 1011 (print in reverse order)

S2 convert this into base 10 int.

S3 Keep a power, 10 next 100, 1000, ...

S4 Keep a var ans and ans += power * i;

Eg 1101

ans = 1*1 + 0*10 + 1*100 + 1*1000

⇒ 1101

pseudo code

```
int decimaltobin (int n)
{
    int ans = 0;
    int p = 1;
    while (n > 0)
    {
        int last bit = (n & 1);
        ans += p * last_bit;
        p = p * 10;
        n = n >> 1;
    }
    return ans;
}
```

Counting Set Bits And Optimation

Given a number N, find the no. of set Bits in the binary representation.

Eg N = 13 . Binary = 1101

O/P = 3.

(Set bits = 1) (unset bits = 0)

Ans There are 3 way to ans this problem

1) As we know that the no. is stored in binary form in the memory so we will run a while loop if the last bit is 1 then increment sum & right-shift the no.

pseudo code

```
int countbits (int n)
{
    int a = 0;
    while (n > 0)
    {
        a += (n & 1);
        n = n >> 1;
    }
    return a;
}
```

Time O(log N)

2nd Method

```
while (n>0)
{ n = n + (n-1);
  a++;
}
```

3rd func In build funct^n

cout << __builtin_popcount(n) << "\n"; (→ 2 underscores)

↳ this directly counts no. of set bits

Unique Number 2

Q Given an array containing n number. All the no.s are present twice except for 2 no.s which are present only once. Find the unique no. in linear time without using any extra space.

Eg 3 1 2 1 O/P ⇒ 2, 3

smaller no. will come first

Eg 1, 1, 2, 3, 5, 2, 5, 7 O/P ⇒ 3, 7

★ Linear time and const space.

Concept S1 First we will do XOR of all the elements present

S2 Then we'll get the resultant XOR value of both the unique no.s.

S3 For sure the resultant cannot be zero coz they are unique

S4 There must be atleast 1 set bit present in the resultant XOR

S5 Then to find the pos of set bit.

S6 Then to find a search for all the element who is having set bit at that particular position.

Then do XOR of all the nos which are having set bit at that position. Thus we gets our 1st unique no.

Then to find B, just do XOR of resultant with this we'll find our 2nd uniquen no.

Eg 5 1 2 1 2 3 5

XOR = 3 ^ 7 = 4

4 → 0100

↳ pos of 1st bit = 2

Right shift the num till its 1st bit for finding the first set bit

no. with 1 at 2 pos.

⇒ 5, 5, 7

5 ^ 5 ^ 7 = 7 **a = 7**

4 ^ 7 = 3 ∴ **b = 3**

Pseudo code

```
int main()
{
    int n;
    int a[100005];
    cin>>n;
    int no;
    int res = 0;

    for (int i=0; i<n; i++)
    {   cin>>a[i];
        res = res ^ a[i];
```

```
int t = res;
int pos = 0;
while((t & 1) != 1)
{ pos++;
  t = t >> 1;
}
int mask = (1 << pos);
int x = 0;
int y = 0;
for(int i=0; i<n; i++)
{ if(a[i] & mask) > 0)
  { x = x ^ a[i];
  }
}
y = res ^ x;
cout << min(x, y) << " " << max(x, y) << "\n";
return 0;
}
```

Unique No. 3 (Imp)

Q In an array all the no. are occuring 3 times except for 1 number which is unique no. (Occuring 1) find the unique no.

Eg 1 1 1 2 2 2 3 ; ans = 3

Eg 3, 3, 2, 1, 1, 1, 3

Sol We will right every no. in the binary form

S2 Then when will add the all the bits of the particular position. One thing we'll notice will that that the sum of the bits at any pos will either be 3n or 3n+1.

S3 Once we'll add all the bits at all particular position then ~~add~~ we will do % with each sum digit.

S4 Thus, we'll get our required ans.

Eg Sol

3 → 0 1 1
3 → 0 1 1
2 → 0 1 0
1 → 0 0 1
1 → 0 0 1
1 → 0 0 1
3 → 0 1 1

0 4 6

%3 %3 %3 %3

0 1 0 ⇒ 2 Ans

if unique no. has bit at a given pos ⇒ 3N+1 else 3N.

pseudo code

```
int main()
{   // we are making an array of final bits :. Space => O(1)
    int cnt[64] = {0};
    int n, no;
    cin >> n;
    for (int i=0; i<n; i++)
    {   cin >> no;
        // update the cnt array by extracting bits
        int j = 0;
        while (no > 0)
        {   int last_bit = (no & 1);
            cnt[j] += last_bit;
            j++;
            no = no >> 1;
        }
    }
// traversing over the array & form the answer by converting 0s and 1s into a number.
    int p = 1;
    int ans = 0;
    for (int i=0; i<64; i++)
    {   cnt[i] %= 3;
        ans += (cnt[i] * p);
        p = p << 1; }
    cout << ans << endl; }
    return 0;
```

2^6	2^5	2^4	2^3	2^2	2^1	2^0	
128	64	32	16	8	4	2	1	
	0	1	0	1	1	0	0	44
	0	0	0	0	1	0	0	4
	0	0	0	0	1	1	1	7
1	0	0	1	1	0	1		77
0	1	1	0	0	0	1		49
1	0	0	1	0	1	0		74

Fast exponentiation using Bitwise Bitmasking

↳ Optimising power funct^n (a^n) using Bitmasking

Eg $a = 3$, $b = 5$, $3^5 = 243$

↳ $3 \times 3 \times 3 \times 3 \times 3$ ⇒ Multiplying 3, 5 no. of times.

a^n ⇒ It take $O(n)$ times to compute the power

a^5 step 1: we will write the form of 5 in binary. then.

Eg $a^5 = a^{\overset{2^2\,2^1\,2^0}{101}} = a^{4+0+1} = a^4 a^0 a^1$ (101 ← logn bits)

So for any no. n the max no. of set bits are logn

Eg $a = 3$, $n = 5$, ans = 1

a^{101} = if the bit = 1 then multiply by a (current val)

else a ib bit = 0 skip

$a = 3^1 = 3 = 1 \times 3$

next $a = 3^2 = 9 = 3 \times 3$

next $a = 3^4 = 81 = 9 \times 3$

$2^6 = 2^{110}$ ans = 1, a = 2, n = 6 = 110

take an ans variable.

if 1 then multiply by a else skip

ans = $1 \times 2^2 \times 2^4 = 4 \times 16 = 64$

a = 1, a = 2^0, a = 2^4.

pseudo code

```
#include <iostream>
using namespace std;
int power_optimisation(int a, int n)
{ int ans = 1;
  while (n>0)
{ int last_bit = (n&1);
  if (last_bit)
  { ans = ans * a; }
    a = a * a;
    n = n>>1;
}
  return ans;
  return 0;
}
  int main()
{ int a, n;
  cin>>a>>n;
  cout<< power_optimisation(a,n) << "\n";
  return 0;
}
```

Everytime we get a set bit we will multiply the ans by a

if the bit is 1 then multiply else skip.

Every step square up a

Discarding the last bit of n.

Generate Subset Using Bitmasking

1. Finding subsequence / subset of a Given string

Eg Input :- "abc"

Output :- "", a, ab, abc, ac, b, bc, c

Explanation

Here every element of the string has 2 option either to include or to exclude.

a b c

↓ ↓ ↓

$2 \times 2 \times 2 = 8$

This total makes,

$2 \times 2 \times 2 = 8$ possible subsets

* If we include all values a, b, c then we get a whole string.

* If we don't include element then we get a null subset.

possibilities

"", a, b, c, ab, bc, ac, abc

ac is not a subsequence but it is a subset (Subsequence is a contigious)

So to generate all possible subsets we will write the binary configuration of 0 - 7

	cba	
0	000	""
1	001	a
2	010	b
3	011	ba
4	100	c
5	101	ca
6	110	cb
7	111	cba

we will break the prob in 2 parts

1. which iterates over the list of element from 0-n
2. Filter. That extract the alphabet from the string having set bit.

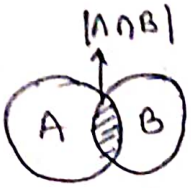
code

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
void filter (int n, char a[])
{  int j=0;
   while (n>0)
   {  int l_b = (n&1); → checking for the last bit
      if (lb == 1) ⟶ if the bit is 1 then print the char
        { cout<<a[j]; }
        j++; ⟶ incrementing j every time
        n = n>>1; → for discarding last bit.
   }
}
  void printSubsets(char a[])
{    int n = strlen(a); // calculating the size of string
     for (int i=0; i<(1<<n); i++) // iterating over
      { filter (i, a); } // range & pass to filter.
      return;
}
     int main()
{ char a[100];
   cin>>a;
   printSubsets(a);
   return 0;
}
```

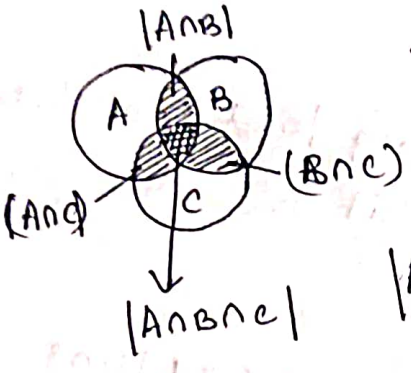
O/P a, b, ab, c, ac, bc, abc, " ".

Inclusion - Exclusion Principle ...

$$|A \cup B| = |A| + |B|$$



$$|A \cup B| = |A| + |B| - |A \cap B|$$



$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|$$

$$|A \cup B \cup C \cup D| = |A| + |B| + |C| + |D| - |A \cap B| - |B \cap C| - |C \cap D| \dots + |A \cap B \cap C| + |B \cap C \cap D| + \dots - |A \cap B \cap C \cap D|$$

General Trend

If we are combining odd num of terms then sign = -ve else +ve

odd ⇒ +ve
even ⇒ -ve

⚠ So we will generate all subsets.

if the subset has odd no. of element we'll add the elements else substract the elements

Lets solve a prob

Q How many no.s are there which are < 1000 and are divisible by 2, 3, 5.

Sol first we'll see for how many no. which are divisible by 2, 3, 5

$$|2| = \frac{999}{2} = 499$$

$|3| = \frac{999}{3} = 333$

$|5| = \frac{999}{5} = 199$

$\therefore |2 \cup 3 \cup 5| = |2| + |3| + |5|$

~~$\Rightarrow 499 + 333 + 199$~~

$|2 \cup 3 \cup 5| = |2| + |3| + |5| + |2 \cap 3 \cap 5| - |2 \cap 3| - |2 \cap 5| - |3 \cap 5|$

for finding the no.

$|2 \cap 3| = \frac{999}{2 \times 3} = \frac{999}{6} = 166$

$|2 \cap 5| = \frac{999}{10} = 99$

$|3 \cap 5| = \frac{999}{3 \times 5} = \frac{999}{15} = 66$

$|2 \cap 3 \cap 5| = \frac{999}{2 \times 3 \times 5} = \frac{999}{30} = 33$

$\therefore |2 \cup 3 \cup 5| = 499 + 333 + 199 - 166 - 99 - 66 + 33$

$\Rightarrow 733$

so what basically we gonna do is we'll make a list of numbers which are prime and using bitmasking we will find out all the subsequence of the list using ("finding Subsequence technique") if the sig of subsequence is odd than we will add into the result else we'll subtract from the result.

Not-So-EASY · MATH

Q You are given a number n, you have to find out the number of number between 1 and n which are divisible by any of the prime no. less than 20.

code

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
#define ll long long
int main()
{
    ll t;
    cin >> t;
    ll primes[] = {2,3,5,7,11,13,17,19};
    while(t--)
    {
        ll n;
        cin >> n;
        ll subsets = (1<<8) - 1;
        ll ans = 0;
for(ll i = 1; i <= subsets; i++)
    {
        ll denom = 1ll;
        ll setBits = __builtin_popcount(i);
        for(ll j=0; j<=7; j++)
        {
            if(i & (1<<j))
            { denom = denom * primes[j]; }
        }
        if(setBits & 1)
        { ans += n/denom; }
```

we are taking long long because when we are taking product it can become large

count set bits so that it is odd then + else -

if the bit is set then we will mul it with the denominator.

```
else
{ ans *= n/denom; }
cout << ans << endl;
}
return 0;
}
```

Summary

First we figured out how many subset are present. Eg for 8 no.s possible subsets = $2^8 - 1$ because we are not talking about null subset

S2 Then we'll start iterate over all subsets starting from 1 to $2^8 - 1$ and for every no. we have a corresponding no. in binary which is going to have atmost 8 set bits.

S3 Then we calculated no. of set bits if it is odd then we will multiply the product and add it to our denominator and final and

S4 else we subtracted from final ans.

S5 print the result.

Lucky Numbers

A lucky no. is a no containing '4' & '7' only. Eg 4, 7, 47, 444, ...
If we sort all lucky no. what is the k bound index of n.

Idea To find smaller nos
If my number is a 4 digit no. then we call find out all possible combition for 3 digit no.

Eg $n = 7447$ if 3 digit no $= 2^3$

$\overline{2} \cdot \overline{2} \cdot \overline{2}$

each place can be filled using 2 nos only
Total 2^3.

for $\underbrace{\_\_\_\_\_}_{x \text{ digits}}$

$2^1 + 2^2 + 2^3 \cdots + 2^{x-1}$

2^x digit no. $= \overline{2} \cdot \overline{2}$

$\Rightarrow 2^2 = 4$

1 digit $= \overline{2}^1$

$\Rightarrow \frac{2(2^{x-1}-1)}{2-1}$ (GP series Part 1)

$\Rightarrow \boxed{2^x - 2}$ x-1 term of no.

Now for x digits,

4447 → 1

3 2 1 0
4 7 4 4
1 · 2 · 2

$2^2 = 4$

4 7 4 7
↓ → 1
2 × 2 = 5

for every 7 we have to see for its contribution

for every 7 $\Rightarrow 2^{\text{Pos of 7}}$ will be added to ans

code

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
int main()
{
  char a[20];
  cin>>a;
  int l = strlen(a);
```

```
ll ans = 0;
ans = (1<<l) - 2;  → for finding (l-1) digits contribution
// iterate over from unit place and add contribution
for (int i = l-1; p = 0; i >= 0; i--, pos++)
{
    if (a[i] == '1')
        ans += (1<<p); // if 1 then 2^pos.
}
    cout << ans + 1 << "\n";
}
return 0;
}
```

Big Integers

Big integers are used to store large no.

capacity of int = 10^9

capacity of long long int = 10^{18}

One of the famous problem of Big Integer is big factorial

Big factorial

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
int a[1000];
void multiply (int *a, int &n, int no)
{
    int carry = 0;
    for (int i=0; i<n; i++)
    {
        int product = a[i] * no + carry;
        a[i] = product % 10;
        carry = product / 10;
    }
    while (carry)
    {
        a[n] = carry % 10;
        carry = carry / 10;
        n++;
    }
}
```

```
void big-factorial (int number)
{ int *a = new int [10000];
for (int i = 0; i<1000 ; i++)
  { a[i] = 0; }

   a[0] = 1;
   int n=1;
for ( int i = 2; i<= number ; i++)
{ multiply (a, n, i) ; }

   for ( int i = n-1 ; i>=0; i --)
    { cout << a[i]; }
    delete [] a;
    return;
}
    int main ( )
  { int n;
    cin>> n;
    big-factorial (n);
    return 0;
}
```

Maths - 2 combinatorics

Birthday Paradox Problem

What is the minimum no. of people that should be present in a hall so that there's 50% chance of 2 people having the same birthday.

Ans for 100% probability of different birthday

if days = 365 and People = 366

for finding out same birthday, its better to calculate for different birthday.

if same bday = P and diff bd = x

$x = 1 - P$.

if room contain 1 person. probability of diff bd

= 100% = 1

if 2 person for diff bd

$P_1 = \frac{^{365}C_1}{365}$, $P_2 = \frac{^{364}C_1}{365}$

$P_1 \times P_2 = \frac{^{365}C_1}{365} \times \frac{^{364}C_1}{365} = 99.73$

for 3 P

$P_1 = \frac{^{365}C_1}{365} \times P_2 = \frac{^{364}C_1}{365} \times P_3 = \frac{^{363}C_1}{365} = 99.17\%$.

As we increase the people the probability of having different bday decreases and same value increases

Implementation

```
#include <bits/stdc++.h>
using namespace std;
int main()
{ float x = 1.0;
  int people = 0;
  float num = 365;
  float denom = 365;
  float p;
cin >> p;
if ( p == 1.0)
{ cout << "366" << endl;
  return 0;
}
while (x > 1-p)
{ // x is less than the threshold
    x = x * (num)/ denom;
      num--;
      people++;
    cout << "People" << people << "and x" << x << "\n";
}
return 0;
}
```

Maths-III Solving Linear Recurrences

1) Fast Exponentiation : A faster way to compute a^n or a^n

pow(a, n) = $a^n \Rightarrow a(a^{n/2})^2 \Rightarrow$ n is odd

$(a^{n/2})^2 \Rightarrow$ n is even

$\therefore$ n

$\downarrow$

$n/2 \rightarrow n/4 \rightarrow n/8 \rightarrow n/16 \ldots 1$

$5^{11} \rightarrow 5(5^5)^2 \rightarrow 5(5^2)^2 \rightarrow 5(5^1)^2 \rightarrow 5(5^0)^2$

$2^8 \rightarrow (2^4)^2 \rightarrow (2^2)^2 \rightarrow (2^1)^2$

1 2 3

$n = 8$

Steps = $\log_2 N = 3$

Time = $O(\log n)$

Code

```
#include<bits/stdc++.h>
using namespace std;
int pow(int a, int n)
{
    if(n==0)
    { return 1; } // a^0 = 1

    int subprob = pow(a, n/2);

    if(n&1) // if odd
    { return a * subprob * subprob; }

    return subprob * subprob; // if even
}
int main()
{
    int a, n;
    cin>>a>>n;
    cout<<pow(a, n)<<"\n";
    return 0;
}
```

Problems solving involving Linear Recurrences

$$f(i) = f(i-1) + f(i-2) + f(i-4)$$

nth term of the recurrence

A Linear Recurrence is a function in which each term of the sequence is a linear combination of fewer terms

Eg $f(i) = f(i-1) + f(i-2)$ — No. of terms $k = \max no_k = 2$ (Here)

Time $= O(k^3 \log N)$ where k

S1 Find the value of k

$k = 2$

S2 Find the first k terms store it in a vector

$$F1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

S3 Find T (k x k)

↳ T is the transformation matrix of k x k

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} f(i-2) \\ f(i-1) \end{bmatrix} = \begin{bmatrix} f(i-1) \\ f(i) \end{bmatrix}$$

T Fi F_{i+1}

$f(i-2), f(i-1), f(i), f(i+1)$ — T, T, T

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} f(i-1) \\ f(i) \end{bmatrix} = \begin{bmatrix} f(i) \\ f(i+1) \end{bmatrix}$$

S4 $F_n = T F_{n-1} = T(T F_{n-2}) = T \cdot T \cdot T \cdot F_{n-3}$

$\therefore \boxed{F_n = T^{n-1} F_1}$ $\quad T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{n-1}$

Eg $T = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ — 0, 1, 1

$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ — 0, 1, 1, 2

$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ — 0, 1, 1, 2, 3

Ok we can say,

$$F_n = T^{n-1} \cdot F_1$$

Matrix Exponentiation ("How to find transformation Matrix")

Eg $f(i) = 1f(i-1) + 2f(i-2) + 0 \cdot f(i-3) + 4f(i-4)$

formula, $f(i) = \sum_{j=1}^{k} c_j f(i-j)$

First find Fi vector

$$\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}_T \begin{bmatrix} f(i-4) \\ f(i-3) \\ f(i-2) \\ f(i-1) \end{bmatrix}_{Fi} = \begin{bmatrix} f(i-3) \\ f(i-2) \\ f(i-1) \\ f(i) \end{bmatrix}_{Fi+1}$$

Steps

Step 1 make a the first row as zero of T as KXK

Step 2 make the last row as the coefficient of the terms in reverse order.

Step 3 Make the rest matrix as the identity matrix

as
$$\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ c_4 & c_3 & c_2 & c_1 & c_0 \end{matrix}$$

Eg $f(i) = 1f(i-1) + 2f(i-2) + 0f(i-3) + 4f(i-4)$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 0 & 2 & 1 \end{bmatrix}_{4\times 4}$$

Eg $f(i) = f(i-1) + 2f(i-2) + 0f(i-3) + 4f(i-4) + d$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{(T)} \begin{bmatrix} f(i-4) \\ f(i-3) \\ f(i-2) \\ f(i-1) \\ d \end{bmatrix} = \begin{bmatrix} f(i-3) \\ f(i-2) \\ f(i-1) \\ f(i) \\ d \end{bmatrix}$$

1 coefficient of d

Generalised Fibonacci

Eg $c_1 f(i-1) + c_2 f(i-2) + c_3 f(i-3) \cdots c_k f(i-k)$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ c_k & \cdots & c_3 & c_2 & c_1 \end{bmatrix}_{K\times K}$$