Competitive programming

Contents

[Basic Programming 4](#_Toc86487300)

[Bit Manipulation 4](#_Toc86487301)

[How to check if a given number is a power of 2 4](#_Toc86487302)

[Count the number of ones in the binary representation of the given number 4](#_Toc86487303)

[Check if the ith bit is set in the binary from of the given number 4](#_Toc86487304)

[How to generate all the possible subsets of a set 5](#_Toc86487305)

[Find the largest power of 2(most significant bit in binary form), which is less than or equal to the given number N. 5](#_Toc86487306)

[Tricks with bits 6](#_Toc86487307)

[Data structures 7](#_Toc86487308)

[Trees 7](#_Toc86487309)

[Heaps/Priority Queues 7](#_Toc86487310)

[Advanced Data Structures 12](#_Toc86487311)

[Trie 12](#_Toc86487312)

[Segment Tree 12](#_Toc86487313)

[Fenwick (Binary Indexed) Tree 15](#_Toc86487314)

[Suffix Trees 16](#_Toc86487315)

[Suffix Arrays 16](#_Toc86487316)

[Disjoint Data Structure 16](#_Toc86487317)

[Algorithms 17](#_Toc86487318)

[Searching 17](#_Toc86487319)

[Ternary Search 17](#_Toc86487320)

[Sorting 17](#_Toc86487321)

[Radix Sort 17](#_Toc86487322)

[Bucket Sort 18](#_Toc86487323)

[Graphs 19](#_Toc86487324)

[Breadth First Search 19](#_Toc86487325)

[Depth First Search 19](#_Toc86487326)

[Minimum Spanning Tree 20](#_Toc86487327)

[Shortest Path Algorithms 24](#_Toc86487328)

[Math 29](#_Toc86487329)

[Number Theory 29](#_Toc86487330)

[Modular arithmetic 29](#_Toc86487331)

[Modular exponentiation 29](#_Toc86487332)

[Greatest Common Divisor (GCD) 29](#_Toc86487333)

[Modular Multiplicative Inverse 30](#_Toc86487334)

[Sieve of Eratosthenes 31](#_Toc86487335)

[Modification of Sieve of Eratosthenes for fast factorization 31](#_Toc86487336)

[Sieve of Eratosthenes on the segment 32](#_Toc86487337)

[Fermat Primality Testing 33](#_Toc86487338)

[Miller-Rabin Primality Testing 33](#_Toc86487339)

[Totient Function 34](#_Toc86487340)

[Combinatorics 35](#_Toc86487341)

[Basic of Combinatorics 35](#_Toc86487342)

[Inclusion-Exclusion 36](#_Toc86487343)

[Geometry 37](#_Toc86487344)

[Computational Geometry I 37](#_Toc86487345)

# Basic Programming

## Bit Manipulation

### How to check if a given number is a power of 2

Properties for numbers which are powers of 2, is that they have one and only one bit set in their binary representation. If the number is neither zero nor a power of two, it will have 1 in more than one place. So if x is a power of 2 then x & (x-1) will be 0.

Text, letter

Description automatically generated

### Count the number of ones in the binary representation of the given number

With bitwise operations, we can use an algorithm whose running time depends on the number of ones present in the binary form of the given number. This algorithm is much better, as it will reach to logN, only in its worst case.

Text

Description automatically generated

### Check if the ith bit is set in the binary from of the given number

Using Left shift operator, we can write 2i as 1 << i . Therefore:

Text, letter

Description automatically generated

### How to generate all the possible subsets of a set

A big advantage of bit manipulation is that it can help to iterate over all the subsets of an N-element set. As we all know there are 2N possible subsets of any given set with N elements. What if we represent each element in a subset with a bit. A bit can be either 0 or 1, thus we can use this to denote whether the corresponding element belongs to this given subset or not. So, each bit pattern will represent a subset.

Text, letter

Description automatically generated

### Find the largest power of 2(most significant bit in binary form), which is less than or equal to the given number N.

Approach 1

A picture containing table

Description automatically generated

Approach 2

Text

Description automatically generated

### Tricks with bits

1. x ^ ( x & (x-1)) : Returns the rightmost 1 in binary representation of x.
2. x & (-x) : Returns the rightmost 1 in binary representation of x
3. x | (1 << n) : Returns the number x with the nth bit set.

# Data structures

## Trees

### Heaps/Priority Queues

Building max heap

Text

Description automatically generatedText

Description automatically generated

Building min heap

Text

Description automatically generated

Text, letter

Description automatically generated

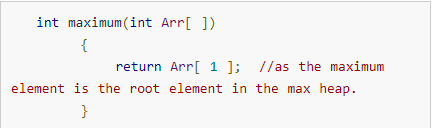
Implementing hep\_sort

Text, letter

Description automatically generated

Priority Queues

Get maximum:



Extract Maximum:

In this operation, the maximum element will be returned and the last element of the heap will be placed at index 1 and max\_heapify will be performed on node 1 as placing last element on index 1 will violate the property of max-heap.

Graphical user interface, text

Description automatically generated

Increase Value:

In case increasing value of any node, it may violate the property of max-heap, so we may have to swap the parent’s value with the node’s value until we get a larger value on parent node.

Text

Description automatically generated

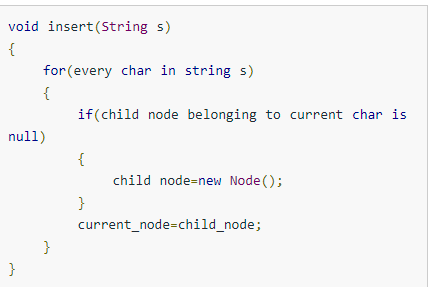
Insert Value:

Text, letter

Description automatically generated

## Advanced Data Structures

### Trie



Graphical user interface, text, application

Description automatically generated

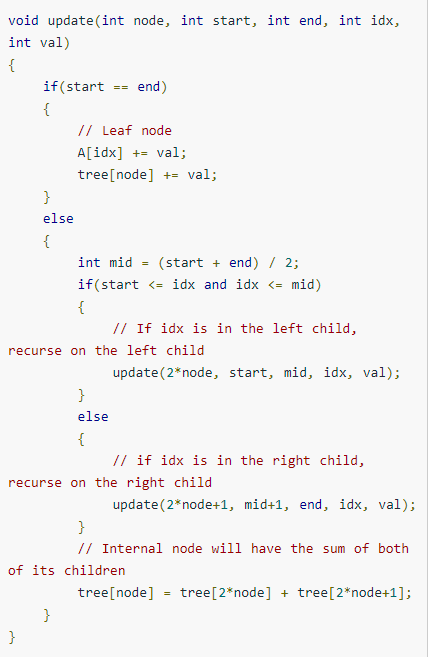
### Segment Tree

Buid O(N)

Text

Description automatically generated

Update O(lonN)



Query O(logN)

Text

Description automatically generated

### Fenwick (Binary Indexed) Tree

Update O(logN)

Text

Description automatically generated with medium confidence

Query O(logN)

Text, letter

Description automatically generated

### Suffix Trees

### Suffix Arrays

## Disjoint Data Structure

# Algorithms

## Searching

### Ternary Search

This is used in unimodal functions to determine the maximum or minimum value of that function. Unimodal functions are functions that, have a single highest value.

Graphical user interface, text, application

Description automatically generated

## Sorting

### Radix Sort

Graphical user interface, text, application, email

Description automatically generated

### Bucket Sort

After dividing elements into buckets we perform insertion sort on each bucket.

Graphical user interface, text, application

Description automatically generated

## Graphs

### Breadth First Search

Text

Description automatically generated

### Depth First Search

Text

Description automatically generated

### Minimum Spanning Tree

#### Kruskal’s Algorithm

Disjoint sets

Text, letter

Description automatically generated

Kruskal algorithm

Kruskal’s algorithm builds the spanning tree by adding edges one by one into a growing spanning tree.

Steps:

* Sort the graph edges with respect their weights.
* Start adding edges to the MST from the edge with the smallest weight until the edge of the largest weight.
* Only add edges which doesn’t form a cycle, edges which connect only disconnected components.

Graphical user interface, text, application, email

Description automatically generated

Text

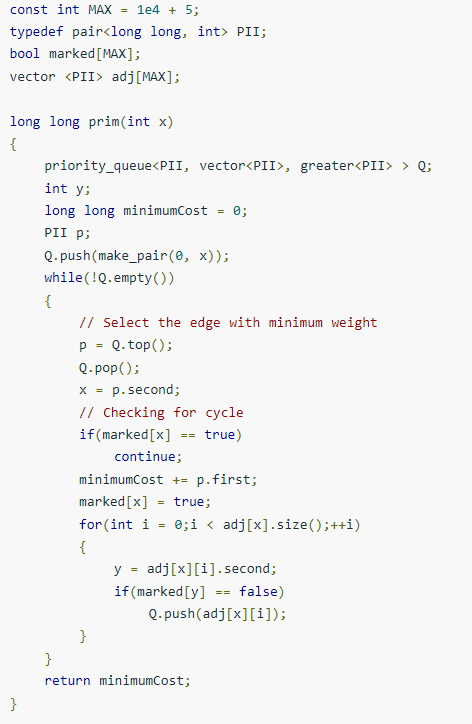
Description automatically generated

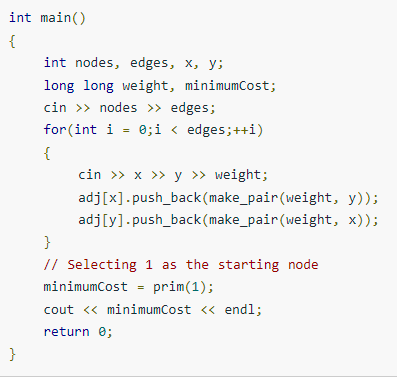
#### Prim’s Algorithm

In Prim’s Algorithm we grow the spanning tree from a starting position. Unlike an edge in Kruskal’s, we add vertex to the growing spanning tree in Prim’s.

Steps:

* Maintain two disjoint sets of vertices. One containing vertices that are in the growing spanning tree and other that are not in the growing spanning tree.
* Select the cheapest vertex that is connected to the growing spanning tree and is not in the growing spanning tree and add it into the growing spanning tree. This can be done using Priority Queues. Insert the vertices, that are connected to growing spanning tree, into the Priority Queue.
* Check for cycles. To do that, mark the nodes which have been already selected and insert only those nodes in the Priority Queue that are not marked.





### Shortest Path Algorithms

#### Bellman Ford’s Algorithms

Bellman Ford’s Algorithm is used to find the shortest paths from the source vertex to all other vetices in a weighted graph.

A very important application of Bellman Ford algorithm is to check if there is a negative cycle in the graph.

Time complexity of Bellman Ford algorithm is relatively high



Steps:

* The outer loop traverses from 0: n-1
* Loop over all edges, check if the next node distance > current node distance + edge weight, in this case update the next node distance to “current node distance + edge weight”.

Text

Description automatically generated

#### Dijkstra’s Algorithm

Dijkstra’s Algorithm is used to find the shortest path from the source vertex to all other vertices in the graph.

Steps:

* Set all vertices distances = infinity except for the source vertex, set the source distance = 0
* Push the source vertex in a min-priority queue in the form (distance, vertex) as the comparison in the min-priority queue will be according to vertices distances.
* Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source)
* Update the distances of the connected vertices to the popped vertex in case of “current vertex distance + edge weight <next vertex distance”, then push the vertex with the new distance to the priority queue.
* If the popped vertex is visited before, just continue without using it.
* Apply the same algorithm again until the priority queue is empty.

Time complexity of Dijkstra’s Algorithm is O(v^2) but with min-priority queue it drops down to O(V + E log V).

Implementation

Table

Description automatically generated with medium confidence

#### Floyd Warshall’s Algorithm

Floyd Warshall’s Algorithm is used to find the shortest paths between all pairs of vertices in a graph, where each edge in the graph has a weight which is positive or negative. The biggest advantage of using this algorithm is that all the shortest distances between any 2 vertices could be calculated in O(V^3), where V Is the number of vertices in a graph.

Steps:

* Initialize the shortest paths between any 2 vertices with infinity.
* Find all pairs shortest paths that use 0 intermediate vertices, then find the shortest paths that use 1 intermediate vertex and so on.. until using all N vertices as intermediate nodes.
* Minimize the shortest paths between any 2 pairs in the previous operation
* For any 2 vertices(i,j), one should actually minimize the distances between this pair using the first K nodes, so the shortest path will be: **min(dist[i][k] + dist[k][j], dist[i][j]).**

**dist[i][k]** represents the shortest path that only uses the first **K** vertices, **dist[k][j]** represents the shortest path between the pair k,j. As the shortest path will be a concatenation of the shortest path from I to k, then from k to j.

Implementation

Text, letter

Description automatically generated

Articulation Points and Bridges

Biconnected Components

Strongly Connected Components

Topological Sort

Hamiltonian Path

Maximum flow

Minimum Cost Maximum Flow

Min-cut

# Math

## Criterios de divisibilidad

2: El último número es par

3: La suma de los dígitos es múltiplo de 3

4: Los 2 últimos dígitos son múltiplos de 4

5: El ultimo digito es 5 o 0

6: Criterios del 2 y 3

8: Los últimos 3 dígitos son múltiplos de 8

9: Suma de dígitos múltiplo de 9

## Number Theory

### Modular arithmetic

Properties:

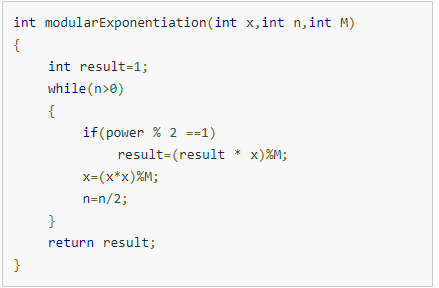
Text, letter

Description automatically generated

### Modular exponentiation

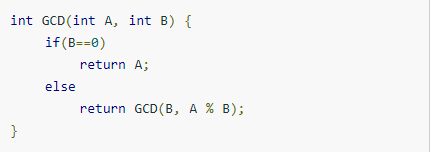
T=O(logN)

M=O(1)

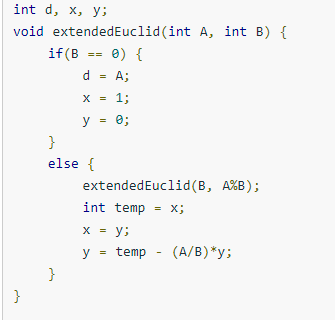


### Greatest Common Divisor (GCD)

Euclid’s algorithm O(log(max(A, B)))

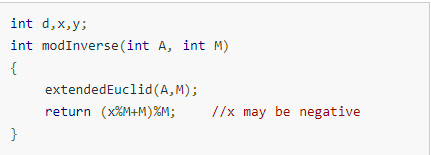


Extended Euclidean Algorithm O(log(max(A, B)))



### Modular Multiplicative Inverse

A and M are coprime. GCD(A,M)=1



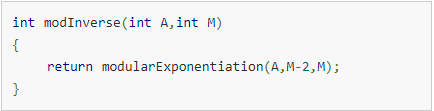
Use only when M is prime

This approach uses Fermat’s Little Theorem



By multiplying with A-1 both sides, the equation can be rewritten as follows:





### Sieve of Eratosthenes

Text

Description automatically generated

### Modification of Sieve of Eratosthenes for fast factorization

Factorization in sqrt(N)

Text, letter

Description automatically generated

We can get the minimum prime that divides the current N in O(1) with an array

Text, letter

Description automatically generated

Text, letter

Description automatically generated

### Sieve of Eratosthenes on the segment

Text

Description automatically generated

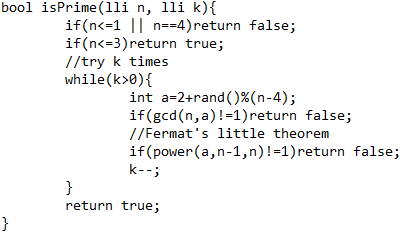
### Number of divisors of a number

Multiply all the exponents of the factors adding one to each of them:

p1+1\*p2+1…..=Number of divisors

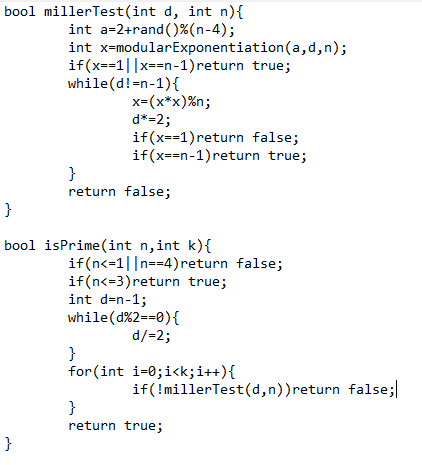
### Fermat Primality Testing

Fermat’s little theorem: Ap-1 =1modp



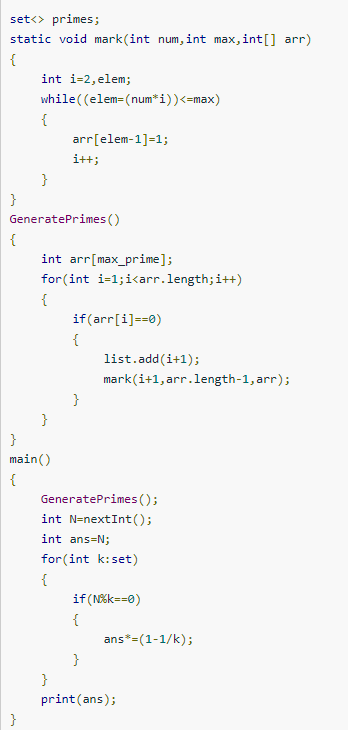
### Miller-Rabin Primality Testing

It is based on a basic principle where if X2 = Y2 modN, but X!=±YmodN, then N is composite



### Totient Function

Euler’s Totient function is a function that is related to getting the number of numbers that are coprime to a certain number X that are less than or equal to it.



Text

Description automatically generated

## Combinatorics

### Basic of Combinatorics

Permutations of choosing R distinct objects out of a collection of N objects can be calculated using the following formula:

Text

Description automatically generated

Combinations of choosing R distinct objects out of a collection of N objects can be calculated using the following formula:

Text

Description automatically generated

**Basic Combinatorics Rules:**

**The Rule of Product:**

The product rule states that if there are X number of ways to choose one element from A and Y number of ways to choose one element from B, then there will be X x Y number of ways to choose two elements, one from A and one from B.

**The Rule of Sum:**

The sum rule states that if there are X number of ways to choose one element from A and Y number of ways to choose one element from B, then there will be X + Y number of ways to choose one element that can belong to either A or B.

Permutation with repetition:

If we have N objects out of which N1 objects are of type 1, N2 objects are of type 2, … Nk objects are of type k, then number of ways of arrangement of these N objects are given by:

A picture containing diagram

Description automatically generated

Combinations with repetition:

If we have N elements our of which we want to choose K elements and it is allowed to choose one element more than once, then number of ways are given by:

Text

Description automatically generated

### Inclusion-Exclusion

Cardinality of the sum of sets

For a given set A of n sets A1, A2, … An, let Sn be the sum of these sets, more formally,

Sn =

The formula:

****

**Implementation**

The value of the formula can be computed by generating all subsets of A, which is the set of sets A1, …, An, and for each such subset, by computing the cardinality of its intersection and adding this value with an appropriate sign to the final result.

**Text

Description automatically generated**

## Geometry

### Computational Geometry I

Closest pair

Graphical user interface, text, application

Description automatically generated

Convex hull

Graphical user interface, text

Description automatically generated