Sieve

https://discuss.codechef.com/t/please-proofread-the-atricle-for-tutorial-section-on-sieve-methods/7955
https://brilliant.org/wiki/modular-arithmetic/
All in one

- product of divisors of a number can be written as N D/2, where N is number and D is number of divisors of N.
- A number cannot have more than **log N** prime factors. So even if there is a large prime number its prime factors can not exceed 7.
- Only Perfect number has an odd no of divisors.
- Let $\mathrm{F(n)}$, $F(n)$ denote the number of factors of N. Then $\mathrm{F(N)} <= 1344$ $F(N) \leq 1344$ for $\mathrm{N} <= 10\hat{}9$ $\mathrm{F(N) = O(N\hat{}\{1/3\})}$ in the worst case.
- power(A,power(B,C), M) = power(A,power(B,C,M-1),M))
- The number of ways to put n identical objects into k labeled boxes is

  (n+k−1 n).
- Derangement formula : Dp[i] = (i-1)*(dp[i-1]+dp[i-2])
  https://www.geeksforgeeks.org/count-derangements-permutation-such-that-no-element-appears-in-its-original-position/

1. Sieve to count no of prime factors of each no from 1 to n

```
for(int i=2;i<MAX;i++){
        if(isPrime[i]){
            primefact[i] = 1;
            for(int j=2*i;j<MAX;j+=i){
                isPrime[j] = false;
                primefact[j]++;
            }
        }
    }
```

2. Represent Each no in p1^a1 * p2^a2 * p3^a3 form

```
int prime_factor[N];
vector<pair<int, int> > prime_factors[N];

for(int i = 2; i < N; i++) {
    if(prime_factor[i] != 0)  continue;
    prime_factor[i] = i; //IMPORTANT
    for(int j = 2*i; j < N; j += i) {
      prime_factor[j] = i;
    }
  }


  for(int i = 2; i < N; i++) {
    int x = i;
    map<int, int> M; //HW - use vector instead of map here
s    while(x != 1) {
      M[prime_factor[x]]++;
      x /= prime_factor[x];
    }
    for(auto v : M) {
      prime_factors[i].push_back(v);
    }
  }
```

3. https://cp-algorithms.com/algebra/binary-exp.html***
    a.

```
long long binpow(long long a, long
long b) {
    if (b == 0)
        return 1;
    long long res = binpow(a, b / 2);
    if (b % 2)
        return res * res * a;
    else
        return res * res;
}
```

```
long long binpow(long long a, long
long b) {
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a;
        a = a * a;
        b >>= 1;
    }
    return res;
}
```


b. Matrix multiplication using binpow
    i. https://www.spoj.com/problems/MPOW/

4. Multiplicative Inverse
    - N^{-1} % mod = $N^{\text{mod - 2}}$ mod
      https://cp-algorithms.com/algebra/module-inverse.html#:~:text=Practice%20Problems-,Definition,x%E2%89%A11modm.&text=It%20can%20be%20proven%20that,a%2Cm)%3D1)

    - If gcd(n, m) == 1 then only multiplicative inverse exists.

```
int mulinverse(ll m, int  n){
    int t1 = 0, t2 = 1, t, q, r = 1, q1 = m, q2 =
n;

    while(true){
        q = q1 / q2;
        r = q1 - (q*q2);
        t = t1 - (q*t2);
        if(r == 0) break;
        q1 = q2;
        q2 = r;
        t1 = t2;
        t2 = t;

    }
    if(t2 < 0)
        t2 += m;
    return t2;
}
```

        i. Find ncr using mult inverse
        ii. https://cses.fi/problemset/task/1079/ (n C r)
            1. Using ncr = n! * (r!)^(-1) * (n-r)!^(-1)
            2. n!/(n-r)!*r! == n! * power((n-r)! , mod-2) * power(r!,mod-2)
               Reference :

5. Matrix Exponent
    a. Fibonaci : https://cses.fi/problemset/task/1722
    b. https://cses.fi/problemset/task/1096/

Backtracking

1. N queen Problem
    a. https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/
    b. interviewbit.com/problems/nqueens/

2. General Approach for solving backtracking

```
Pick a starting point.
while(Problem is not solved)
    For each path from the starting point.
        check if selected path is safe, if yes select it
        and make recursive call to rest of the problem
        before which undo the current move.
    End For
If none of the move works out, return false, NO SOLUTON.
```

https://leetcode.com/problems/coin-change

https://leetcode.com/problems/subsets/discuss/27281/A-general-approach-to-backtracking-questions-in-Java-(Subsets-Permutations-Combination-Sum-Palindrome-Partitioning)

Combination sum variations blog
https://leetcode.com/problems/combination-sum-iv/discuss/85120/C%2B%2B-template-for-ALL-Combination-Problem-Set

1. https://leetcode.com/problems/coin-change

```
dp[0] = 0;
    for(int i=0;i<n;i++)
    {
      for(int j=coins[i];j<amount+1;j++)
        dp[j] = min(dp[j], dp[j - coins[i]] + 1);

    return dp[amount] == 999999 ? -1 : dp[amount];
```

2. https://leetcode.com/problems/coin-change-2/

```
 dp[0] = 1;
        for(int i=0;i<n;i++)
            for(int j=coins[i];j<=amount;j++)
                dp[j] = dp[j-coins[i]] + dp[j];
return dp[amount];
```

3. https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/practice-problems/algorithm/knapsack-with-large-weights-33a2433a/submissions/
   When weight range is very high we can use index and value for dp

   Dp[i][j] → min weight required to get value j upto index j

   Both approach in code
   For O(n) space when we override in same array try to reverse inner loop so will not override value tha is used in future

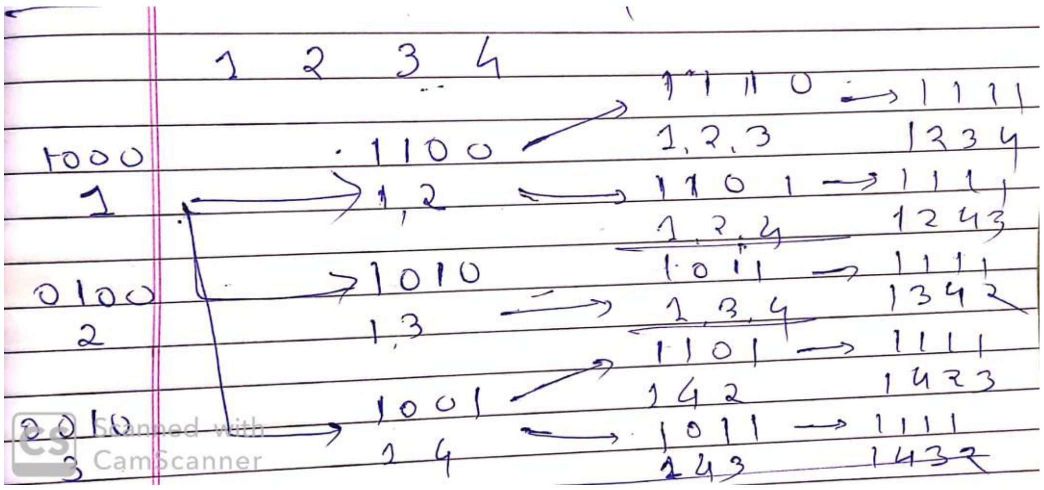
4. https://leetcode.com/problems/combination-sum-iii/
5. https://leetcode.com/problems/combination-sum-iv
6. https://leetcode.com/problems/number-of-dice-rolls-with-target-sum/
7. https://leetcode.com/problems/last-stone-weight-ii/
8. https://leetcode.com/problems/combinations/

Subsets

1 2 3 4

```
1 - 1 2 - 1 2 3 - 1 2 3 4          2 - 2 3 -2 3 4
            1 2 4                      - 2 4
  - 1 3 - 1 3 4                    3 - 3 4
  - 1 4
```

Permutation



3. Solve Sudoku

    https://leetcode.com/problems/sudoku-solver/
    Start from 0,0 move till 8,8
    If i,j is . try all 0 to 9 and if invalid backtrack

Knapsack

	All combination	Unique Combination

Infinite	`vector<int> dp(amount+1, 0);`
	`dp[0] = 1;`
	`  for(int i=0;i<=amount;i++)`
	`      for(int j=0;j<n;j++)`
	`            if(i - a[j] >= 0)`
	`              dp[i] = dp[i] + dp[i-a[j]];`
		`vector<int> dp(amount+1, 0);`
		`dp[0] = 1;`
		`  for (int j=0;j<n;j++)`
		`      for(int i=coins[j];i<=amount;i++)`
		`           dp[i] = dp[i] + dp[i-coins[j]];`

finite		`vector<int> dp(amount+1, 0);`
		`dp[0] = 1;`
		`for(int i=0;i<n;i++){`
		`       for(int j=amount;j>=coint[i];j--)`
		`              dp[j] = dp[j] + dp[j-a[i]];`

Bit Manipulation

1. Count no of 1s in number
   Let's use n = 00101100 as an example. This binary representation has three 1s.
   If n = 00101100, then n - 1 = 00101011, so n & (n - 1) = 00101100 & 00101011 = 00101000. Count = 1.
   If n = 00101000, then n - 1 = 00100111, so n & (n - 1) = 00101000 & 00100111 = 00100000. Count = 2.

If n = 00100000, then n - 1 = 00011111, so n & (n - 1) = 00100000 & 00011111 = 00000000. Count = 3.

```
int count = 0;
    while (n) {
        n &= (n - 1);
        count++;
    }
    return count;
```

2. Given array where all occurs twice except one element which occur once find it
Having noticed that if X has 1 in that position, we will have 3x+1 number of 1s in that position. If X has 0 in that position, we will have 3x+1 number of 0 in that position.

3. Min xor pair from given array
Sort an array and take min xor of consecutive elements

4. Hamming distance sum of all pair in given array
Take each bit one by one
    Count zeros and onces
    Ans += 2*ones*zeros

Two Pointers and Sliding windows

1. Two Pointers and sliding window
    a. https://leetcode.com/problems/subarray-sum-equals-k/discuss/301242/General-summary-of-what-kind-of-problem-can-cannot-solved-by-Two-Pointers

If a wider scope of the sliding window is valid, the narrower scope of the window is also valid.
If a narrower scope of the sliding window is invalid, the wider scope of that narrower scope is invalid must hold.

    b. Remove duplicates in place in

```
  int i = 0, j = 0;
   while(j < a.size()){
       while(j < a.size() && a[j] ==
a[j+1]) j++;
       a[i++] = a[j];
       j++;
   }
```

Two duplicates allowed

```
 int i = 0, j = 0, n = a.size();
 while(j < n){
     int count = 1;
    while(j < n && a[j] == a[j+1]){
      j++;  count++;
    }
    a[i++] = a[j];
    if(count > 1) a[i++] = a[j];
    j++;
}
```

```
                                        return i;
```

c. Maximum consecutive 1s in sequence on 0/1 where k flips allows
   i. https://www.interviewbit.com/problems/max-continuous-series-of-1s/
   ii. Take negate of input now find maximum window with sum = k
d. Container with Most water
   i. https://www.interviewbit.com/problems/container-with-most-water/
      [4 (left) , 9, 12, 7 , 8, 14 (right)] --------> this array is indexed from 1 to 6 and we looks at left and right and compare the value
      So eliminate all pairs of index [1,2], [1,3],[1,4],[1,5] <- here the index[] means possible windows of solutions (4,9), (4,9,12), (4,9,12,7)...
      Why? area of window [1,6] = (6 - 1) * MIN(4, 14) = 20
      all the windows that have [1,2...5] even if all the numbers on the right are changed to infinity must be less than 20.
      EX:
      (5-1) * min(4, infinity) < 20 for all right indices less than 6.
      So this means its ok to exclude all windows starting with index 1
e. Find duplicate in array of size n + 1 where range is 1 to n
   i. https://leetcode.com/problems/find-the-duplicate-number/solution/
   ii. Using fast and slow pointer the num which is duplicate create cycle
f. Longest substring without repeating char
      https://www.interviewbit.com/problems/longest-substring-without-repeat/

g. K window maximum
   i. https://leetcode.com/submissions/detail/408974744/
h. Longest Repeating Character Replacement
   i. https://leetcode.com/problems/longest-repeating-character-replacement/
   ii. Current window size = j - i + 1

      maxCount = max freq in curr window

      Diff char = cur window size - maxCount

      If Diff char < k expand window

      Else shift window

      -- sometimes window may contain invalid maxCount but its okay bcaz we want longest one so when window expands it should be valid
i. Longest subarray with absolute difference less than or equal to limit

    i. https://leetcode.com/problems/longest-continuous-subarray-with-absolute-diff-less-than-or-equal-to-limit/
    ii. Approach 1 O(nlogn)
    Use multiset to maintain max and min in current window
    Expand window if max - min < limit
    Shrink until condition not satisfied
    iii. Approach 2 O(N) using queue
    Use monotonically increasing min queue and max queue to get maximum and minimum in given sliding window in O(1) time
    Expand window if max - min < limit
    Shrink until condition not satisfied

j. No of subarray having exactly k distinct integers
    i. https://leetcode.com/problems/subarrays-with-k-different-integers/
    ii. Ans = atmost(k) - atmost(k-1)

k. Min no of flips
    i. https://leetcode.com/problems/minimum-number-of-k-consecutive-bit-flips/
    ii. Use queue to insert if a[i] = target insert i + k - 1 into queue and result++
    iii. Target = 0 q.size() is even
    1 else

l. Sliding window median
    i. https://leetcode.com/problems/sliding-window-median/submissions/
    ii. Use multiset slice window and push median into ans

m. K concatenation Maximum
    i. https://leetcode.com/problems/k-concatenation-maximum-sum/

        ii. https://leetcode.com/problems/k-concatenation-maximum-sum/discuss/383302/C%2B%2B-clean-code-beat-97-with-detailed-explanation

    n. Count subarray each having each having value strictly greater than K
        i. https://practice.geeksforgeeks.org/problems/count-of-subarrays/0

    o. Shortest subarray having sum at least k
        i. https://leetcode.com/problems/shortest-subarray-with-sum-at-least-k/discuss/143726/C%2B%2BJavaPython-O(N)-Using-Deque

Binary Search

1. Find the smallest number x such that f(x) <= Target, where F(x) is inc or dec
   Look at constraint if A[i] < 10pow9 and some target value is given and f(x) is inc/dec
    a. https://leetcode.com/problems/find-the-smallest-divisor-given-a-threshold/

```
int low = 1, high = 1e6, ans = -1;
        while(low <= high){
            int mid = low + (high-low)/2;
            if(check_mid_satisfy_cond(nums, mid, threshold)){ //cond f(x) <= Target
                ans = mid;
                high = mid - 1;
            }
            else{
                low = mid + 1;
            }
        }
        return ans;
```

b. Template :

```
 int n = nums.size();
int low = 0, high = n;
 while(low < high){
int mid = low + (high - low)/2;
if(check(mid, nums))
  high = mid;
else
  low = mid + 1;
return low;
```

https://leetcode.com/problems/minimum-number-of-days-to-make-m-bouquets/discuss/769703/Python-Clear-explanation-Powerful-Ultimate-Binary-Search-Template.-Solved-many-problems.

i. https://leetcode.com/problems/split-array-largest-sum/
ii. https://www.interviewbit.com/problems/painters-partition-problem/
iii. https://leetcode.com/problems/kth-smallest-number-in-multiplication-table/submissions/
iv. https://leetcode.com/problems/find-k-th-smallest-pair-distance/
v. https://leetcode.com/problems/ugly-number-iii/

2. Lower Bound

```
l = 0, r = N - 1
while(l <= r) {
            int mid = (l+r)/2;
            if (nums[mid] == target) {
                answer = mid; // keep the current answer as you might not be
able to comeback if this was the answer.
                r = mid - 1; // lower bound - first index of element
                l = mid + 1; // last index of element
            }
            else if (nums[mid] > target) {
                r = mid - 1;
            } else {
                l = mid + 1;
            }
        }
return ans
```

3. Upper bound

```
l = 0, r = N - 1
while(l <= r) {
            int mid = (l+r)/2;
            if (nums[mid] == target) {
                r = mid - 1; // lower bound - first index of element
                l = mid + 1; // last index of element
            }
            else if (nums[mid] > target) {
                ans = mid
                r = mid - 1;
            } else {
                l = mid + 1;
            }
     }
     return ans;
```

4. Lower bound and Upper bound gfg **

```cpp
int lower_bound(vector<int> arr, int
target)
{
    int n = arr.size();
    int low = 0, high = n - 1;
    while(low < high){
        int mid = low + (high - low)/2;
        if(arr[mid] < target)
           low = mid + 1;
        else
           high = mid;
     }
     return low;
}
```

```cpp
int upper_bound(vector<int> arr,int
target)
{
    int n = arr.size();
    int low = 0, high = n - 1;
    while (low < high) {
        int mid = low + (high - low)/2;
        if (arr[mid] <= target)
            low = mid + 1;
        else
            high = mid;
    }
  return low;
}
```

1. https://leetcode.com/problems/search-insert-position/
2. https://leetcode.com/problems/find-peak-element/
3. https://leetcode.com/problems/find-the-duplicate-number/
4. https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/
5. https://leetcode.com/problems/find-minimum-in-rotated-sorted-array-ii/

Trie

1. https://www.interviewbit.com/problems/shortest-unique-prefix/
    a. Make the trie and store freq with each node
    b. Now for each string shortest prefix to identify it uniquely is till node with freq = 1
2. https://www.interviewbit.com/problems/xor-between-two-arrays/
    a. Store first array in trie
    b. Now for each element of B search for nearest complement of number say x
    Update ans = max(B[i] xor x, ans)

3. https://www.hackerearth.com/problem/algorithm/subarrays-xor/description/
4. Go to Everyday beautiful pdf

DIGIT DP

https://www.spoj.com/problems/CPCRC1C/

```
#include <bits/stdc++.h>
using namespace std;

int digitsumRecursion(int pos, int size, int cursum){
    if(pos > size) return cursum;
    int res = 0;
    for(int i=0;i<=9;i++)
        res += digitsumRecursion(pos+1, size, cursum + i);
    return res;
}

// sum of 1 to size digit numbers
// n = 1 --> 0 to 9
// n = 1 --> 0 to 00
int digitDp(int pos, int size, int cursum, vector<vector<int>> &dp){
    if(pos > size) return cursum;
    if(dp[pos][cursum] != -1) return dp[pos][cursum];
    int res = 0;
    for(int i=0;i<=9;i++)
        res += digitDp(pos+1, size, cursum + i, dp);
    return dp[pos][cursum] = res;
}

int digitDpUptoN(int pos, string num, int cursum, bool islimit, vector<vector<vector<int>>>
&dp){

   // cout << pos <<  " " << cursum << endl;
    if(pos > num.size()) return cursum;
    if(dp[pos][cursum][islimit] != -1) return dp[pos][cursum][islimit];
    int res = 0, limit = 9;
    if(islimit) limit = (num[pos-1] - '0');
    for(int i=0;i<=limit;i++)
        if(i == limit && islimit)
            res += digitDpUptoN(pos+1, num, cursum + i, true, dp);
        else
            res += digitDpUptoN(pos+1, num, cursum + i, false, dp);
    return dp[pos][cursum][islimit] = res;
}

int main(void) {
      // your
      int n = 999;
//    vector<vector<int>> dp(20, vector<int> (180, -1));
//    cout << digitsumRecursion(1, 2, 0) << endl;
//    cout << digitDp(1, 3, 0, dp) << endl;
      string s = to_string(n);
    vector<vector<vector<int>>> dp1(20, vector<vector<int>> (180, vector<int> (2, -1)));

      cout << digitDpUptoN(1, s, 0, true, dp1);
}
```
string binnum = bitset<32>(n).to_string(); // convert a number to 32 bit binary representation