Always INT typecast vector.length()

__/__/____

122. Best Time to Buy & Sell Stock

→ [7, 1, 5, 3, 6, 4]

7 . . . . .

[7, 1, 2, 3, 4, 1]

+1 +1 → (2)

— 4 → (+3)

1, 2, 3, 5

+1 +1 +2 → (4)

+1 +1 +4 → (6)

[1, 2, 3, 1, 5]

+2 +4 → +6

→ So locally go on maximising profit without caring about the big pic. They will all added up.

profit += (prices[i] > prices[i-1]) ? prices[i] - prices[i-1] : 0;

125. Valid Palindrome

(A-Z) (a-z) (0-9)

→ alphanumeric → alphabet + numeric

Python

for x in s.lower():

if x.isalnum():

→ isalnum(x)

tolower(x)

toupper(x)

while (i<=j)

if (!isalnum(s[i])) { i++; continue; }

↓ if (!isalnum(s[j])) { j--; continue; } ↓

A if (tolower(s[i]) != tolower(s[j])) { return false; }

i++; j--;

}

→

A (3) A +32

-32

a +32

P P P-32 ✓

0/0

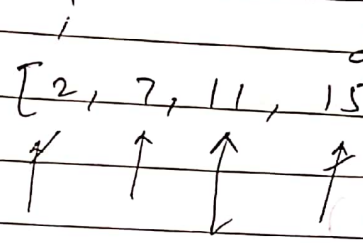
136. Single Number

→ $a \oplus a = 0$

→ $a \oplus 0 = a$

→ So if all items except one item appears twice, the xor of the array gives the single occuring unique element

167. Two Sum II - Input Array Is Sorted




```
s=17
if (s > target)  j--;   // 13
if (s < target)  i++;   // 18
if (s == target) {
    index -> {i+1, j+1}
    break;
}
```

169. majority element

→ Iterate thru hash

```
for(auto x : hash){
    x.first, x.second
}
```

172. Factorial Trailing Zeroes

→ ~~5! = 120~~

~~10! = 1200~~

Trailing 0s in n! = Count of 5s in prime factors of n!

$= \text{floor}(n/5) + \text{floor}(n/25) + \text{floor}(n/125) + \dots$

$(n/i) >= 1$

→ Note

method long long int → input must be unchanged

int trailing((int)n) {

↳ we can change the return datatype as we want to.

}

for (int i=5; (n/i)>=1; i*=5) {

}

(n/i)>=0 ✗ → ∞ loop

② trailing zeroes always long long int.

② Edge Case

169. Maximum Depth of BT nip ([])

— normal dfs + vector.pb + max(vector)

— if (root==NULL) return 0;

↳ bcoz here vector would be empty but ans that has to be returned is 0.

171. Excel Sheet Column Number

A → 1 AA 2 AA

AB — — — —

Z → 26 26^3 26^2 26 20

Z Y

int ans=0;

for (i=0; i<s.length(); i++) {

int val = (s[i] - 65 + 1); A → 1

ans = ans*26 + val;

[1-26] [0-26] [0] [26] ASCII A-Z (65-90) a-z (97-122) 0-9 (48-57)

168. Excel Column Sheet Title

701 → "ZY"

A→1, Z→26 — 1-based indexing, that's why.

```
string ans = " "
while (n-- > 0) {        // n-- : Imp
    int val = n % 26;
    char c = val + 65 - 1;
    ans = ans + c;
    n /= 26;
}
reverse(ans.begin(), ans.end());
return ans;
}
```

Normal decimal to binary time.

---

189. Rotate Array (to the right)

[1, 2, 3, 4, 5, 6, 7] k=3 → k = k % n

[5, 6, 7, 1, 2, 3, 4]

[1, 2, 3] — 2 % 3 → 1

Brute (me)

```
for (i = n-k; i < n; i++)
    tmp.pb(nums[i]);

for (i = 0; i < n-k; i++)
    tmp.pb(nums[i]);

nums = tmp;
```

O(N) extra space.

Imp
→ 3rd element from the last
(Py) arr[-3] ✓
(C++) arr[n-3] ✓

Scanned by CamScanner

__/__/__

Revere Method (LeetCode + Codeforces)

0 1 2 3 4 5 6

Org → 1 2 3 4 5 6 7

rev → (7 6 5 4 3 2 1)

rev(1st K elem) → 5 6 7 4 3 2 1

rev(last (n-k) elem) → [5 6 7 1 2 3 4]

K

n = 7, K = 3 (5 6 7 1 2 3 4)

index

v.begin() ≡ v.begin() + 0

v.end() ≡ v.begin() + (n)

↑ excluding last index

```
int n = nums.size();
k = k % n;

reverse(nums.begin(), nums.end());      // full → [0, K)
reverse(nums.begin(), nums.begin()+k);  // 1st K
reverse(nums.begin()+k, nums.end());    // last (n-k) elements
```

↳ [K, end)

191. Number of 1 Bits

↳ Number of set bits = Hamming weight

— repeatedly flip least significant bit of the number & update

`n = n & (-n);` → // flips the LSB

(12) (21) __/__/__

190. Reverse Bits

12 → "1100"
21 → "10101"
11 → "1011"
12 → 1100
21 →

i/p → (1011) o/p → (1101) → val

1 + 2 + 4

Traverse the bits from L → R in a 32 bit no.

```
uint32_t ans;
```

32 → "10111---"

```
int ans=0, cnt=1;
for(int i=31; i>=0; i--){
    if(n & (1<<i))
        ans += cnt;
    cnt *= 2;
}
return ans;
```

198. House Robber

i/p: [1,2,3,1] → 1+3=4 — not we can't have 2 cases

where i=0 → +=2
i=1 → +=2
} max

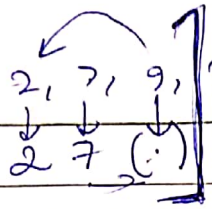
for [2,1,1,2]

but (4) → Any

[2,7,9,3,1]
2 7

[2,1,1,2]

3 3



max(9+2, 7)

rob this, skip next house

don't rob it.

$dp(i) = max(dp(i-2) + nums[i], dp[i-1])$;

if $dp(0) \rightarrow$ } ↑ bfore that, have 2 return statements for $2+1=3$
$dp(1) \rightarrow$ }

2 of these base cases bcoz we dnt sure that atleast 2 elements exist.

Pattern → no adjacent rooms should be selected

202. Happy Number

19 $= 1^2 + 9^2 =$ (82) sum = 1

$8^2 + 2^2$

$x^2 = 4$

$y^2 = 16$

$1^2 + 6^2 = 37$

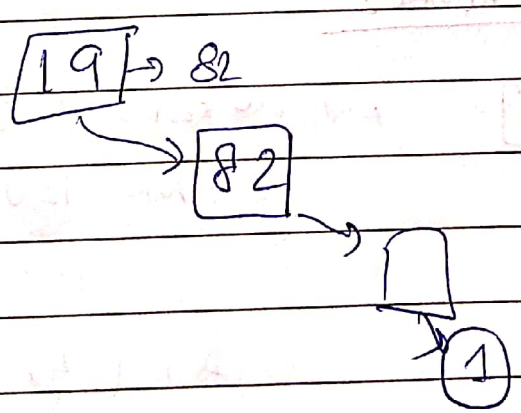
$3^2 + 7^2 =$ (58)

100000

3 digits

$99 \rightarrow 9^2 + 9^2 \rightarrow 81 \times 2 = 162$

Recursive dp solution



```
→ bool isHappy (int n){
    map<int,int> hash;
    return isHappy (n, hash);
}

→ bool isHappy ( int n, map<int,int> &hash) {
    if (n == 1) return true;

    int sum = digSum(n); // 82
    if (hash[sum])
        return false;

    hash[sum]++;
    return isHappy (sum);
}
```

19

258. Add Digits

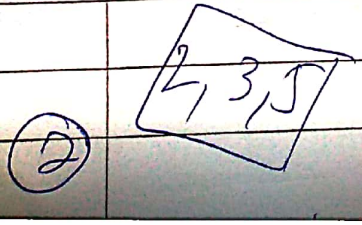
263 Ugly Number

- read problem statement & make return for all base cases

```
for (i=0; i<=n; i++)
```
↳ can cause overflow

```
for (int i=0; i<=sqrt(n); i++)
```

15 / / /



Prime factorisation.cpp

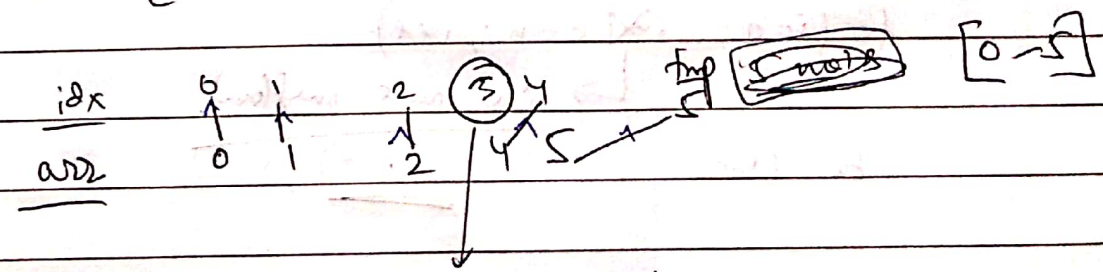
```
// void PF(int n){
    while (n%2==0)
        s.insert(2);
        n/=2;
}

for (int i=3; i<=sqrt(n); i+=2){
    while(n%i==0){
        s.insert(i);
        n/=i;
    }
}
if (n>1)
    s.insert(n);
}
```

ren : 2 3 4

268. Missing Number

(*) [0 - - n-1] 1 number is [0-n] is miss

[0, 1, 2, 4, 5]



tmp = 5;

3 will be left.

tmp = (tmp ^ i ^ arr[i]);

tmp = tmp ^ i ^ nums[i]; // more time

{ tmp = tmp ^ i;
{ tmp = tmp ^ nums[i]; } better time