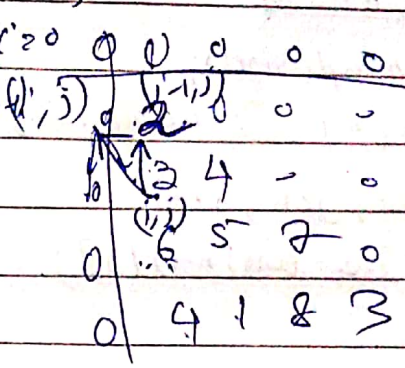
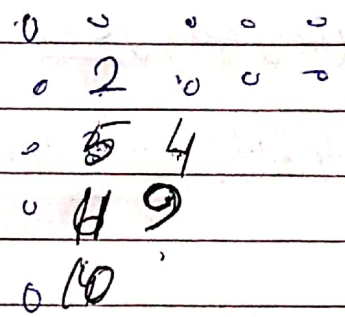
Leetcode (120. Trimgle)

→

```
[
   [2],
   [3,4],
   [6,5,7],
   [4,1,8,3]
]
```





6 5 7

4 1 8 3

9 10

7 6 10

4 1 8 3

2

3 4

6 5 7

4 1 8 3

↓

2

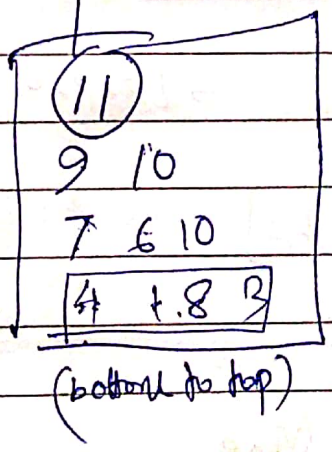
5 4

10 8 11

14 11 18 14

dp

1) check for both bottom-up + top-down approach

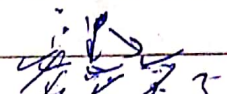


__/__/____

Go from the bottom to the top

```
int n = triangle.size();
int dp[n]; // 1D dp array
for(int j=0; j<n; j++)
    dp[j] = arr[n-1][j];

for (int i=n-2; i>=0; i--){
    for (int j=0; j<=i; j++){
        dp[j] = min(dp[j], dp[j+1]) + triangle[i][j];
    }
}
return dp[0];
```



---

137. Single Number II (every element appears twice instead of once)

→ [2, 2, 3, 2] → [0, 1, 0, 1, 0, 1, 99]

[x, x, 2, 3, x, x, 1]

(2 ^ 3 ^ 1)

(1) Naive Soln → use a HashMap

(2)

5, 5, 5, 8

1 0 1
0 1
0 1
1 0 0 0
0 0 0

10
100
10 | 100
10 + 100
=
10
11
11
1 00 0 0 0 1
0 0 0, 0 0 0

→ Just go on adding the (ith bits) of every number in the array.
If (sum%3 == 0) ⇒ they all occur thrice or 0 is the bit of the single occurrence

Else
if result != 0, set the ith bit.

result |= X;

→ sizeof operator ⇒ gives o/p in bytes
sizeof(int) → 4 (bytes)
→ 32 bits

→

1 00 0 1 0 1

__/__/____

129. Sum Root to Leaf Numbers

→ String ans = "";
vector<string> req;
dfs(root, ans, req);

```
for (auto x : req)
    sum += (stoi(x));
```

{

String to int → stoi()
int to string → to_string()

451. Sort Characters by Frequency

> for (auto x : mp)
ans.pb(x.first, x.second)

{e 4 ← compact
{t 2

vs

for (auto x : s)
ans.pb(mp[x], x)

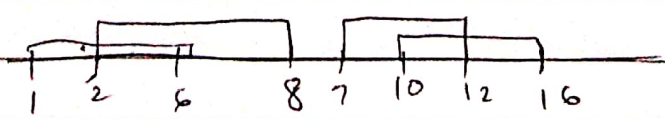
{e 2 ← 1 character by letter
{e 2
{t 2

> reverse sort

sort(vec.begin(), vec.end());
reverse(vec.begin(), vec.end());

452. Min No of Arrows to Burst Balloons

[10,16] [2,8] [1,6] [7,12]



__/__/__

453. Min Moves to Equal Array Elements

[1, 2, 3] → ③

+1 +1

-1 -1, -1

③ 2

increase rest by 1

///

decrease current by 1

[1, 2, 3, 4, 5] ...

① + ② + ③ + ④ → ⑩

(*) Incrementing remn elements by +1 ≡ Decrementing current element by -1.

(*)

∴ If we do (-1) on all elements to equalise array, so we will reach min.

∴ ans += abs(arr[i] - min(arr));

1+2+3 = ⑥

0 + (3-1)(3-2)(3-3)

[1, 2, 3, 4]

(*). 454- Four Sum II

→ map vs unordered-map

map[A(i)+B(j)]

↓ search O(log n).

↓ O(1) or O(N)

cnt += mp[-(C(i)+D(j))] → if exists

(1,2) Sum = -6

for each (k, l), all the equal-sum (i,j) go.

Sum = 6

(i,j)

1 2

2 3

5 6

6

27

__/__/____

455. Assign Cookies

[1,2,3] → [1,1]

→ Lower bound → 1st element >=

upper bound → 1st element strictly greater

→ what if lower_bound is not in the array?

vect → {1,2,3,4,5,6} sz = 6

auto it = lower_bound(v.begin(), v.end(), 150);

cout << it - v.begin() << endl;

→ // (6) ← if absent, returns sz of the vector

{

2

3 → 1

1 → 3

g: g1 g2 g3

s: s1 s2

2 pointer approach

[1, 2, 3] [1, 1]

if (arr[i] <= arr[j]) {

i++; j++;

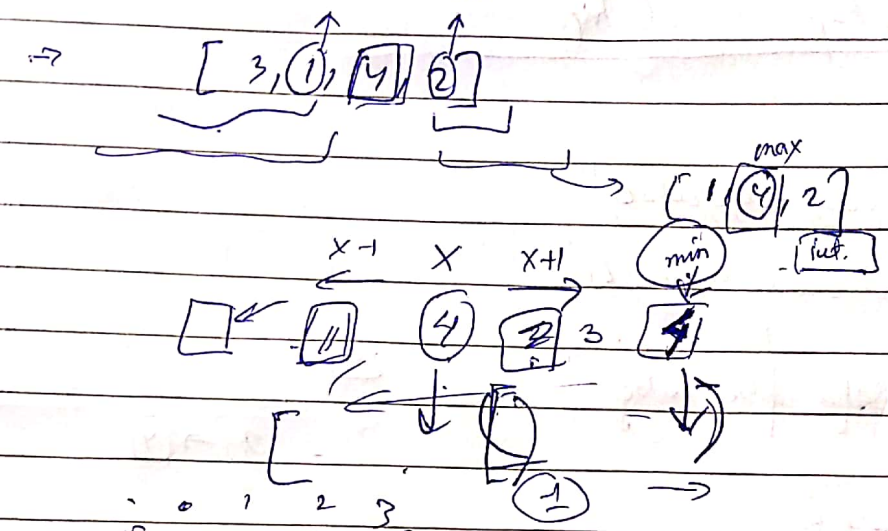
} else {

j++;

}

__/__/__

456. 132 Pattern

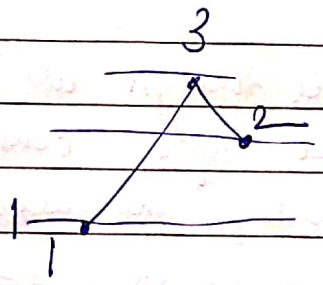


→ [3, 1, 4, 2]

(3,0) (1,1) (4,2) (2,3)

(1,1) (2,3) (3,0) (4,2)

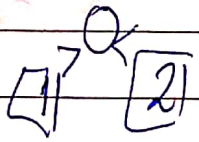
(1,1) (3,0) (2,3)



[-1, 3, 2, 0]

small big

(-1, 3)



459. Repeated Substring Pattern

> "abcabcabcabc"

→ "abcabcabcabc"

"abc abc abc abc" 4

size → [1 ↔ n/2]

n/2 → 6

abc

(n%i == 0) ✓

4. i length substring present.

3 n/3 → 3

→ iterate through all possible lengths of the substring. (1 - n/2)

→ For each length, construct the to be string by repeatedly appending the substr.

→ if to-be-string == original str,
True.

→ False.

→ Can be solved with KMP substring search.

1228. Missing Number in A.P

Q → Sum of A.P = $\frac{n}{2}[a_1 + a_{last}]$

(a_1 → first term, a_{last} → last term)

$$a_n = a_1 + (n-1)d$$

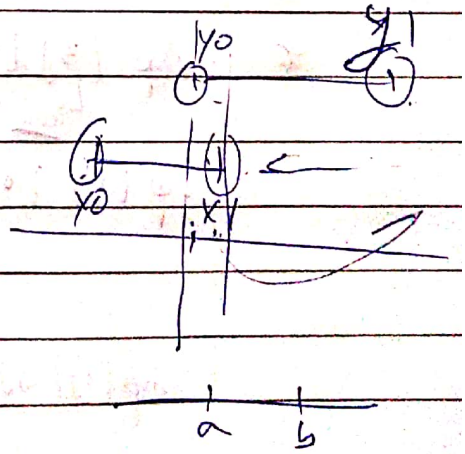
→ missing no = (total sum of AP − sum of arr).

1229. Meeting Scheduler

→ A → [[1,3], [7,9], [12,16]] (n)

B → [[2,3], [4,5], [6,8], [13,14]] (m)

```
while (i<n and j<m){
    x -> A[i];  // [1,3]
    y -> B[j];  // [2,3]
    if ( y[1] < x[0] ){
        j++; continue
    }
    else if ( x[1] < y[0] ){
        i++; continue;
    } else{
    int a = max ( x[0], y[0] );
    int b = min ( y[0], y[1] );
    // [a,b]
    if ( b-a >= desiredgap)
        rec.pb ( {a, a+desiredgap} );
    if (x[1] < y[1]) i++; // (Imp).
    else j++;
}
```



Getting K heads out of N biased coins

Probs of heads → $[P_1, P_2, P_3, \ldots, P_i, \ldots, P_N]$

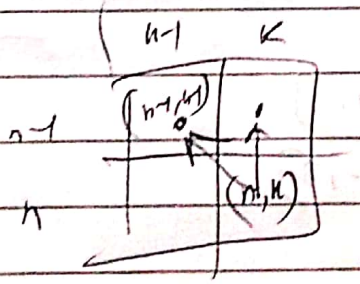
$dp[i][j]$ → prob of getting j heads in i trials

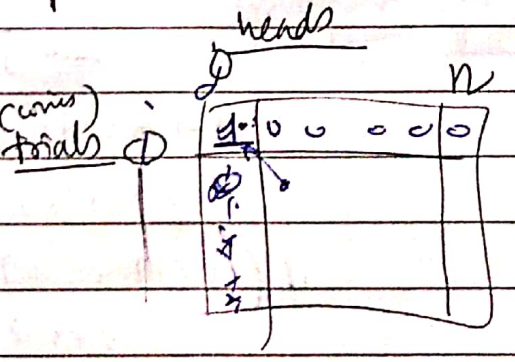
trials heads — K heads — 1 tail

$$dp[n][K] = \left(dp[n-1][K] * (1 - P[n])\right) + \left(dp[n-1][K-1] * P[n]\right)$$

K-1 heads — 1 head

Nth outcome is mutually independent to (N-1)th outcome.



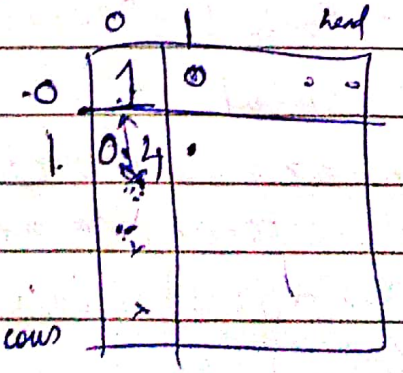


* $dp[3][1]$ → get 1 head out of 3 coins
  - coins
  - heads

$dp[i][0]$ → get 0 head out of i coins

$dp[i-1][0] \times (1 - prob[i-1])$

$dp[0][0]$ →



1 (0, 4)

$dp[1][1]$;

1231. Divide Chocolate

→ sweetness = [1, 2, 3, 4, 5, 6, 7, 8, 9]

K = 5

→ Cut into 5 chunks. [1, 2, 3], [4, 5], [6], [7], [8], [9]

You'll eat the **min sum**.

find the **max sum you can eat.**

int low = 1, high = MAX;

while (l < h) {

mid = (l + h)/2; // optimal **min sum**

slice = 0;

[1, 2, 3] [4, 5] ( ) - - .

K = 6

(1, 2, 3, 4)

Slice < K

low = 1, high = 15

high = mid - 1;

Slice > K

Slice = K

sum →

Scanned by CamScanner

__/__/____

```
int s=0, e=n;
while(s<=e){
int mid = (s+e)/2;
if (isValid(config(books, n, k, mid))
    finalAns = mid;
    e = mid-1;
}else
    s = mid+1;
}
```

if you have left = mid,
use (left + right + 1);

mid = (0 + 1)

```
int stud=1;
int curr=0;
for(i=0; i<n; i++){
    if(curr + books[i] > mid){
        curr = books[i];
        curr++;
    }else
        curr += books(i);
}
if (curr > k) false
else true;
```

(10, 20), 30, 40

[10,5] [2

[10,16], [2,8], [1,6], [7,12]

[1,6], [2,8], [7,12], [10,16]

(6) (12)

Leetcode 452.

```
int point = seg[0].end;
for (int i = 1; i < seg.size(); i++) {
    if (point < seg[i].start || point > seg[i].end) {
        point = seg[i].end;
        ans.pb(point);
    }
}
```

(*) Sort by end pt
(*) Proceed with end-pt comparison

- Also known as min no of points required to cover all segments.

- Note comparator function must be static.

```
sort(vec.begin(), vec.end(), compare);
bool static compare(const vector<int>& a, const vector<int>& b) {
    return a[1] < b[1];
}
```