→ ✓ Unique BST's.

→ ✓ Inlays to Decode

→ ✓ Flip Array

→ Interleaving Strings.

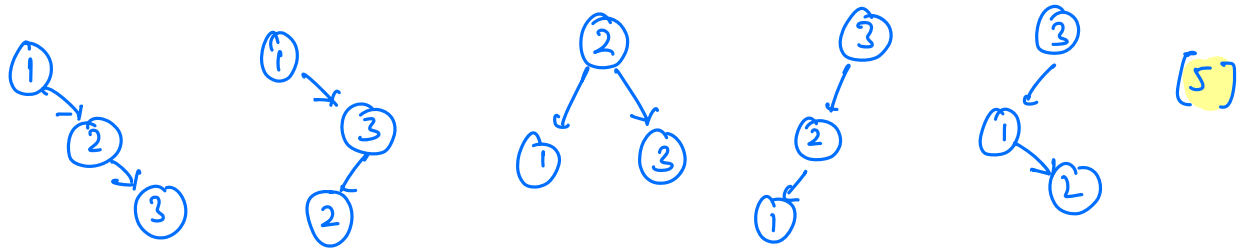
①

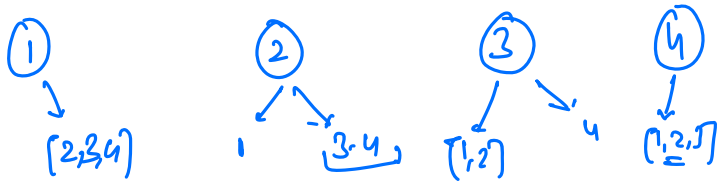
Given an integer A, how many structurally unique BST's (binary search trees) exist that can store values 1...A?

→ Catalan Number

A = 1. ① [1]

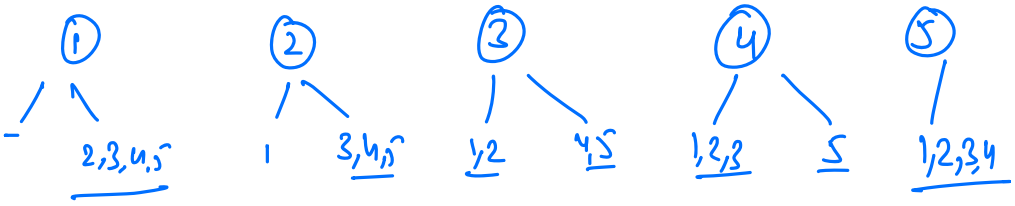
A = 2  [2]

A = 3  [5]

A = 4 

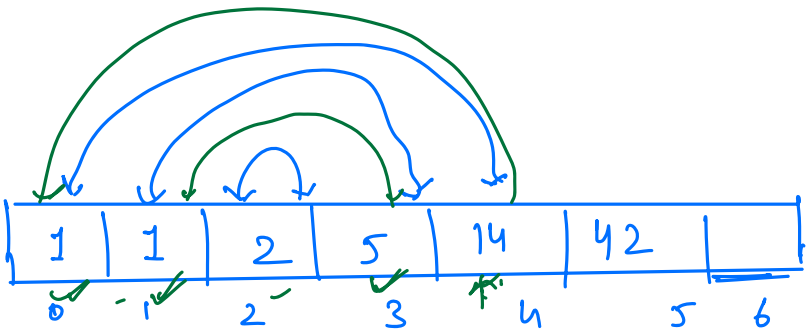
$(1*5) + (1*2) + (2*1) + (5*1) = [14]$

A = 5.



$(1*14) + (1*5) + (2*2) + (5*1) + (14*1) = 42$

A = 6.



$(1*14) + (1*5) + (2*2) + (5*1) + (14*1)$

code →

```
dp[A+1];

dp[0] = 1, dp[1] = 1;

for( i = 2; i <= A; i++) {
    for( j = 0; j < i ; j++) {
        dp[i] += dp[j] * dp[i-j-1];
    }
}

return dp[A];
```

T.C → $O(A^2)$
S.C → $O(A)$

② Ways to Decode

'A' → 1

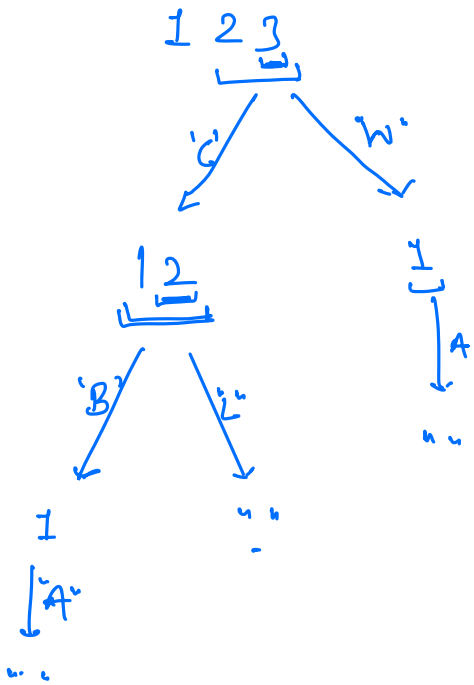
'B' → 2

.

.

'Z' → 26

Given an encoded message denoted by string A containing digits, determine the total number of ways to decode it modulo 109 + 7.



23 → W

24 → X

25 → Y

26 → Z

A B C

1 2 3

L C

12 3

A W

1 23

1203

↓ C

120

↓ T

1

↓ A

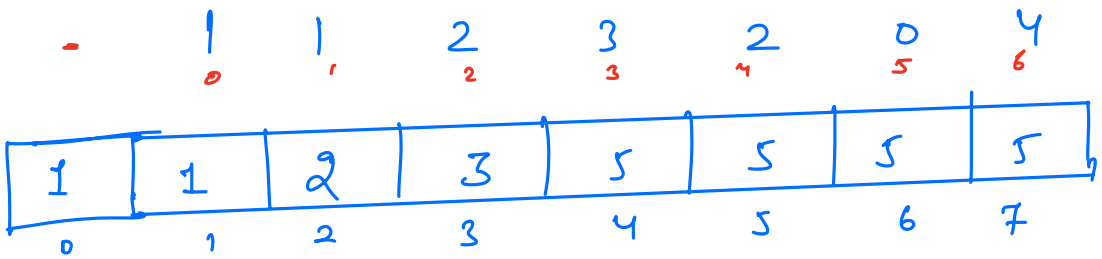
1302

↓ B

130

x x

Eg →



"A" "AA" AAB AABC AABCB AABCT

"K" KB KBC KBCB KBCT

AM AMC AMCB AMCT

AAW AAWB AAWT

KW KWB KWT

#code. →

```
int dp[N+1];
dp[0] = 1;
for( i = 1; i <= N; i++){
    char cc = str[i-1];
    if( cc != 0){
        dp[i] += dp[i-1];
    }
    if( i > 1){
        char pc = str[i-2];
        int num = pc*10 + cc;
        if( pc != 0 && num <= 26){
            dp[i] += dp[i-2];
        }
    }
}
return dp[N];
```

2 6

T.C → O(N)
S.C → O(N)

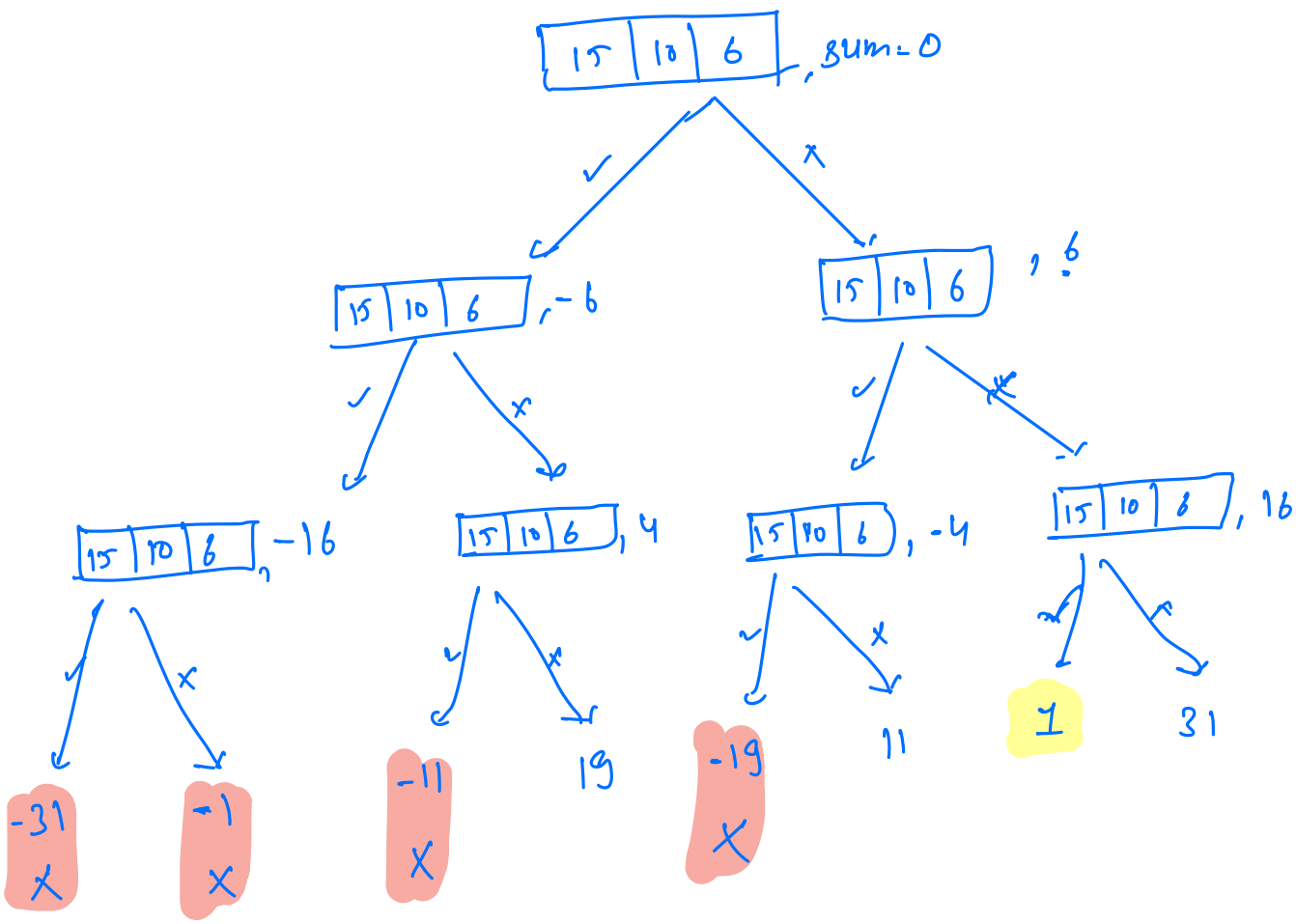
Flip Array

Given an array A of positive elements, you have to flip the sign of some of its elements such that the resultant sum of the elements of array should be minimum non-negative(as close to zero as possible).

Return the minimum number of elements whose sign needs to be flipped such that the resultant sum is minimum non-negative.

A → [15, 10, 6] ans = 1
    0 1 2

A → [14, 10, 4] ans = 1.



Code →

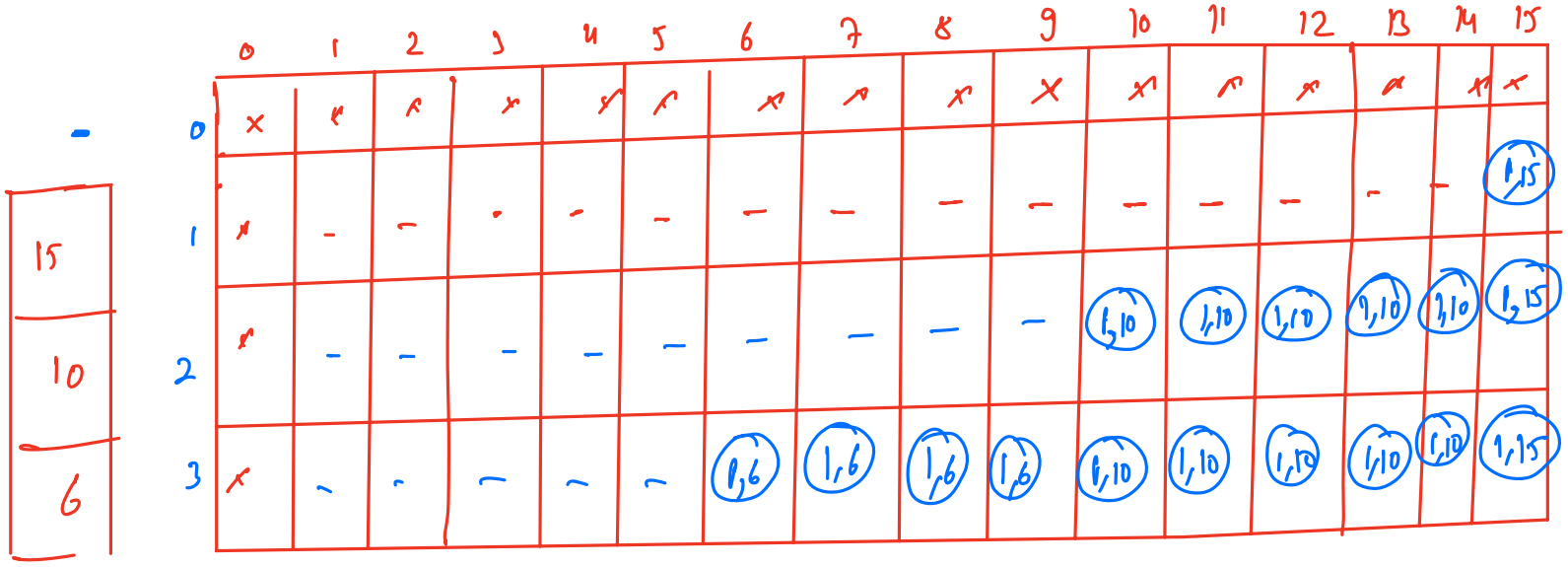
```
int ans = N;          // value of flips
int minSum = Σarr[i];  // sum of element.
                       N-1   0
                       ↑     ↑
void minFlips( arr[], i, sum, flips ){
    if( i < 0 ){
        if( sum < 0 ) { return }
        else {
            if( sum ≤ minSum ){
                minSum = sum;
                ans = flips;
            }
            return;
        }
    }
    minFlips( arr[], i-1, sum + arr[i], flips );
    minFlips( arr[], i-1, sum + (-1 * arr[i]), flips+1);
}
```

idea-2.

```
sum = 0;
for(i=0; i<N; i++){
    sum += arr[i];
}
target = sum/2;
```

Capacity

⇒ Sum of all flipped no's should not exceed sum/2;

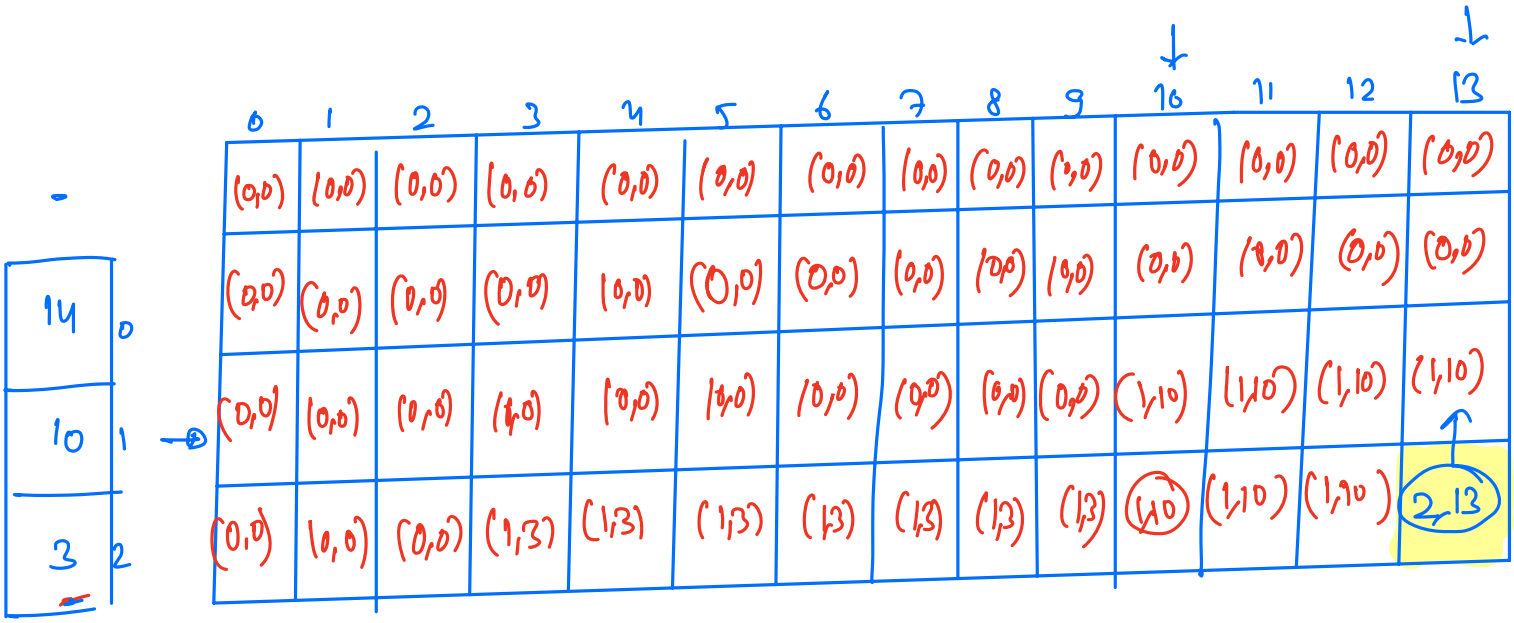


element → sum (wt.)
→ flip. (val)

Cap.

wt → 15 | 10 | 6

val → 1 1 1



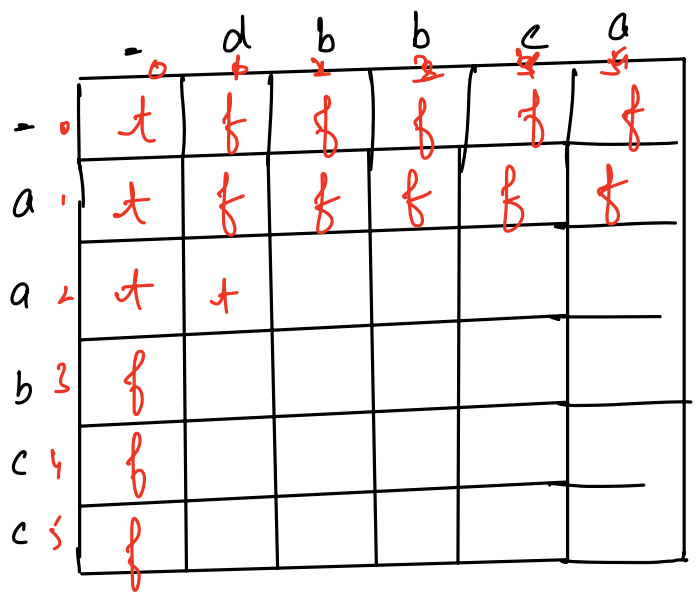
1, 3
1, 10
2, 13.

Interleaving strings

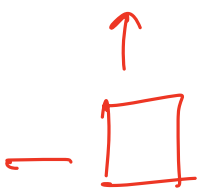
A → a a b c c
0 1 2 3 4

B → d b b c a
0 1 2 3 4

C → a a d b b c b c a c
0 1 2 3 4 5 6 7 8 9



i+j-1.



```
boolean dp[N+1][M+1]

dp[0][0] = true;

for( j = 1; j <= M; j++){
    if( B[j-1] == C[j-1]) { dp[0][j] = dp[0][j-1] }
    else dp[0][j] = false;
}

for( i = 1; i <= N; i++){
    if( A[i-1] == C[i-1] ) { dp[i][0] = dp[i-1][0] }
    else dp[i][0] = false;
}

for( i = 1; i <= N; i++){
    for( j = 1; j <= M; j++) {
        if( (A[i-1] == C[i+j-1] && dp[i-1][j] == true) ||
            (B[j-1] == C[i+j-1] && dp[i][j-1] == true)){
            dp[i][j] = true;
        }
        else { dp[i][j] = false;
    }
}

return dp[N][M];
```