Q→ Find the count of N digit numbers whose digit sum = S. ✓

I/P → N & S leading 0's are not allowed → 007 ✗ 7 ✓
041 ✗ 41

S > 0 ✓

Eg → N = 2, S = 4 {13, 22, 31, 40} Ans = 4 ✓

N = 2 → [10, 11, 12, ... 98, 99] $(10^2 - 1) - (10^1) + 1$

$= 99 - 10 + 1 = 99 - 9 = 90$

N → $(10^N - 1) - (10^{N-1}) + 1$

$= 10^N - 10^{N-1} = 10^{N-1}(10-1) = 9 * 10^{N-1}$ ✓

$9 * 10 * 10 * 10 * 10 \ldots \Rightarrow 9 * 10^{N-1}$ ←

9 (1–9) (0–9)

2 parameters to consider → N & S ✓

N = 5 ✓ — — — — — → 0–9

S = 30 N = 4 ←

x numbers of 4 digits with sum = 22

25 + 5 = 30

21 — 30 → (9 – 0) = 30

+8

⇒ count[N][S] → count of N digit numbers with digit sum = S. ✓

count[i][j] → 1 if (i == j == 9); 0 otherwise ✓

i > 1

$count[i][j] = \sum_{k=0}^{9} count[i-1][j-k]$ ✓

(i-1) → (i)

— — — — (i-1) — (i)

j-9, j-8, j-7 ... j

[0–9] → digits that can be placed at i^{th} location

3 2 6 8 I → [0, 1, 2, 3, 4 ... 9] → k

(j-k) + k = j ✓

(i-1) digits

if (S > 9 * N) return 0;

$\forall i,j$ `count[i][j] = 0;` ✓

```
for (i = 1; i <= N; i++) {                    ✓
    prv = cur;   ∀j cur[j] = 0;               ✓
    for (j = 1; j <= S; j++) {                ✓
        if (i == 1) count[i][j] = (j <= 9 ? 1 : 0);
        else {
            for (k = 0; k <= 9; k++) {        // digit
                if (j > k)
                    count[i][j] += prv[j-k];  ✓
            }
        }
    }
}
return count[N][S];
```

TC = $O(N*S*10)$
$= O(N*9*N*10)$
$= O(90*N^2)$
$\approx O(N^2)$

SC = $O(N*S)$
$= O(N*9N)$
$= O(9N^2) \approx O(N^2)$

3 → 5 ; N = 2, S = 2, j = 3, k = 5, j − k < 0

i = 2, j = 4

k = 0 to 3 — 0

count[1][4−3] = count[1][1]

sum is j−k of (i−1) digits then add k as i^{th} digit to get sum = j.

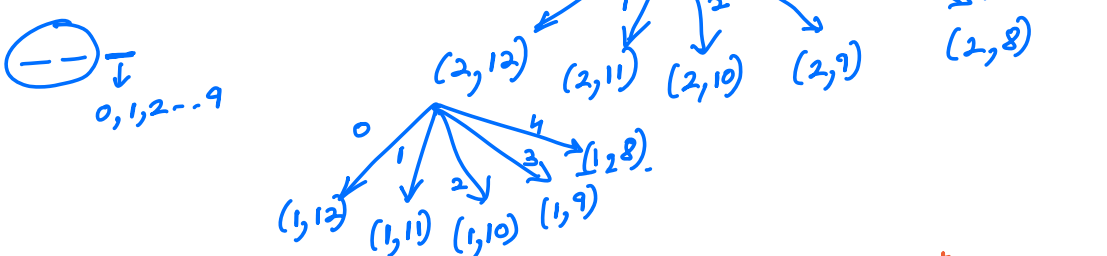
N = 2, S = 4

count	1	2	3	4
1	1	1	1	1
2	1	2	3	(4) ✓

S = 2 { 2 0 ; 1 1 }

S = 3 { 3 0 ; 2 1 ; 1 2 }

S = 4 { 4 0 ; 3 1 ; 2 2 ; 1 3 }



0, 1, 2 .. 9

max S = 9*N

9 9 9 9 . . 9

9+9+9+9 − − +9) N times

= 9*N

to get ans for i digits we only need answers till (i−1) digits

∴ only need to store 2 rows at a time

current i ; previous (i−1)

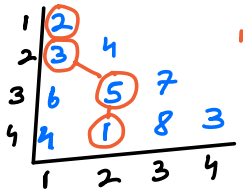
SC = $O(N^2) \approx O(N)$ ✓

Q → Find min path sum in given △.

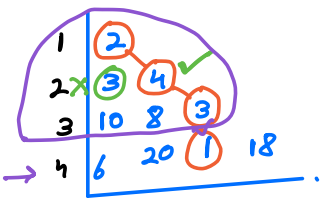
(1,1) → last row any location ✓

(i,j) → (i+1, j) ↓ ✓

→ (i+1, j+1) ↘ ✓



minSum = 2+3+5+1 = 11 ✓



minSum = 2+4+3+1 = 10

⇒ greedy i.e select min(A[i+1][j], A[i+1][j+1]) ✗

If min sum of (i-1) row is known ⇒ min sum of i^{th} row can be calculated easily ✓

optimal substructure ✓

overlapping subproblems ✓

(1,1) → (i,j) → last row ✓ ✓

sum[i][j] → min sum to reach last row from (i,j) cell. ✓

sum[N][j] = A[N][j] // last row ✓

sum[i][j] = A[i][j] + min(sum[i+1][j], sum[i+1][j+1]) ✓

Eg →

	1	2	3	4
1	2			
2	3	4		
3	10	8	3	
4	6	20	1	18

N*N

⇒

	1	2	3	4
1	10 → Ans			
2	12	8		
3	16	9	4	
4	6	20	1	18

minSum[i][j] ✓

TC = O(N*N)

SC = O(N^2) ⇒ O(2N) ✓

Q → Knight

	1	2	3	4
1	10	-5	3	8
2	10	-50	-3	-12
3	-12	1	12	-1
4	6	-2	-8	6
5	3	1	-5	-10

1 room → princess

Moves

(i,j) → (i, j+1)

((i+1), j)

N=5

M=4

A[i][j] == 0 ⇒ empty room

< 0 ⇒ guard in room will reduce the health by |A[i][j]|

> 0 ⇒ magic herbs in room to increase health by A[i][j]

If at any point health <= 0 ⇒ knight is dead ✓

Find min initial health to reach the princess alive.

$(1,1) \rightarrow (i,j) \rightarrow (N,M)$ → v. important

	1	2	3
1	2	-1	0
2	-3	-6	-50
3	-8	-20	100

initial health = 26

Ans = health[1][1]

health[i][j] → min health to reach princess (N,M) when we enter cell (i,j)

h[N][M] → if(A[i][j] >= 0) h[i][j] = 1

else h[i][j] = 1 − A[i][j] (guard)

A[N][M] = −5

1 − (−5) = 6

	1	2	3
1	28 →	27 →	27
2	25	21	-23 (dead)
3	17	1 →	101

<= 0 if initial health < 26

$h[i][j] = \max\left(\min\left(h[i+1][j], h[i][j+1]\right) - A[i][j], 1\right)$

(i+1, j) (i, j+1)

1 − 6 = −5 < 0

Knight	1	2	3	4
1	10	-5	3	8
2	10	-50	-3	-12
3	-12	1	12	-1
4	6	-2	-8	6
5	17	1	-5	-10 (princess)

Moves

N = 5, M = 4

N * M

$(i,j) \rightarrow (i,j+1)$

$((i+1),j)$

Ans

	1	2	3	4
1	1	6 →	1	10
2	3	51	4	18
3	13 →	1 →	1 →	6
4	1	15 →	13 →	5
5	1 →	15 →	16 →	11

$h[N][j] = \max(h[N][j+1] - A[i][j], 1)$

11 − (−5) = 16

15 − 17 = −2 < 0 ⇒ 1

TC = O(N * M)

SC = O(N * M) → O(2 * M) = O(M)

only store 2 rows

HW. can SC = O(1) (use i/p matrix) ?

∀(i,j)

if(h[i+1][j] < h[i][j+1])

go down;

else

go right;

(1,1) → (N,M)