Lecture :- Interview problems

Agenda

- Target sum
- Minimum no of jumps to reach end.
- N digit numbers

Target sum subsets

Given A[n] and an integer B. find whether there exist a subset in an array whose sum = B. If such subset exist, return 1 else return 0.

0	1	2	3	4	5
3	34	4	12	5	2

B	ans
9	true {4, 5} {3, 4, 2}
30	false.

Brute force

Generate all subsets — $O(2^n)$

Check the sum. — $O(n)$

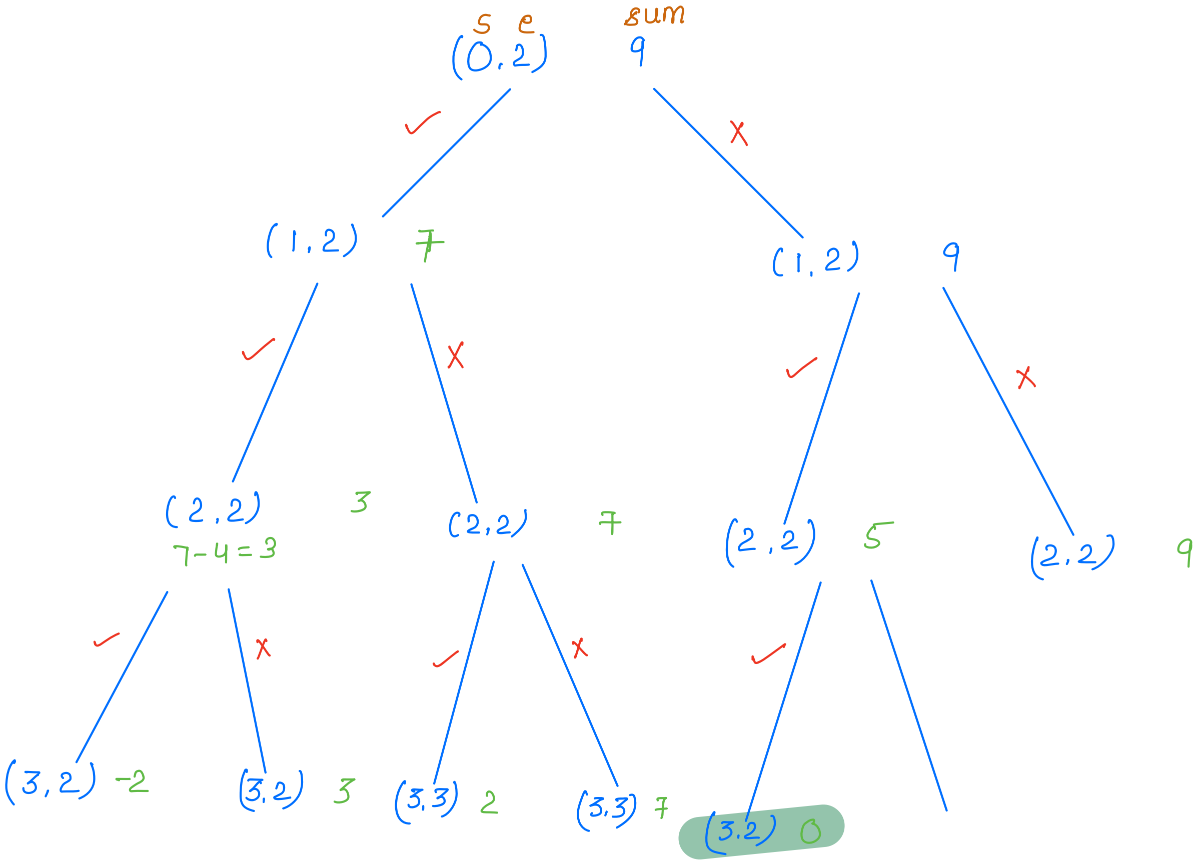
TC: $O(n * 2^n) \simeq O(2^n)$

SC: $O(2^n)$

Approach

0	1	2
2	4	5

target = 9



Recursive code

```
boolean targetSum( int[] A, int idx, int target) {
    if ( target == 0) {
        return true;
    }
    if (target < 0 ) {
        return false;
    }
    if ( idx >= A.length) {
        return false;
    }
    inc = targetsum ( A, idx+1, target - A[i]);
    exc = targetsum( A, idx +1, target);
    return inc || exc;
}
```

0	1	2	3	4	5
3	34	4	12	5	2

target ⇒ 5

dp[n+1] [target +1]

dp[i][j] ⇒ if it is possible to achieve target = j from first i elements.

elements		0	1	2	3	4	5
	0	t	f	f	f	f	f
3	1	t	f	f	t	f	f
34	2	t	f	f	t	f	f
4	3	t	f	f	t	t	f
12	4	t	f	f	t	t	f
5	5	t	f	f	t	t	t
2	6	t	f	t	t	t	t

(5, 3)

pick 5 → (4, -2)

dont pick 5 → (4, 3)

(3, 4)

pick → (2, 0)

dont pick → (2, 4)

dp[3][4] = dp[2][0] ||
dp[2][4]

(6, 5) t

Pick 6th → (5, 3) t

Dont pick → (5, 5) t

(i, j)

✓ → [i-1] [j - arr[i-1]]

x → [i-1][j]

DP code

```
boolean targetSum(int[] A, int target) {
    n = A.length;
    dp[n+1][target+1];
    // first column
    for( i=0; i<=n; i++) {
        dp[i][0] = true;
    }

    for(i=1; i<=n; i++) {
        for(j=1; j<=target; j++) {
            if( j - arr[i-1] >=0 ) {
            inc = dp[i-1][j - arr[i-1]]
            }
            exc = dp[i-1][j];

            dp[i][j] = inc || exc;
        }
    }
    return dp[n][target];
}
```

TC: $O(n^2)$

SC: $O(n^2) \longrightarrow O(n)$ could be done

Qu Given A[n]. You are initially present at A[0]. Each element represents max. no of jump from index i.
Return minimum no of jumps to reach A[n-1].

0	1	2	3	4
2	3	1	1	4

ans = 2

2	3	0	1	4

ans = 2

2	0	0	1	4

x

x

return -1

dp[n]

dp[i] $\Rightarrow$ Min no of jumps required to reach last idx from ith idx.

Approach

A =

0	1	2	3	4	5	6	7	8	9
3	3	0	2	1	2	4	2	0	0

jumps

$\cancel{\infty}$ 4	$\cancel{\infty}$ 4	∞	$\cancel{\infty}$ 3	$\cancel{\infty}$ 3	$\cancel{\infty}$ 2	$\cancel{\infty}$ 1	$\cancel{\infty}$ 1	∞	$\cancel{\infty}$ 0

dp[3]
1 → dp[4] (3)
2 → dp[5] (2)

ans = 2+1=3

dp[7] = min(dp[8], dp[9]) + 1
0+1=1

dp[7]
1 → dp[8] ∞
2 → dp[9] 0

dp[0] → 4
1 → dp[1] 4
2 → dp[2] ∞
3 → dp[3] 3

Pseudocode

```
int minJumps(int[] A, int n) {
    dp[n];
    Arrays.fill(dp, ∞);
    dp[n-1] = 0;
    for(i = n-2; i >= 0; i--) {
        min = ∞;
        jumps = A[i];
        for(j = 1; j <= jumps; j++) {
            if(i+j < n) {
                min = min(min, dp[i+j]);
        }   }
    if(min != ∞) {
        dp[i] = min + 1;
    }   }
    return dp[0];
}
```

TC: $O(n^2)$
SC: $O(n)$

Break: 8:32 - 8:42

Qu find out the no. of n digit +ve numbers whose sum of digits is equal to B.

Leading zeroes are not allowed.

Ex $n = 2$ and $B = 4$

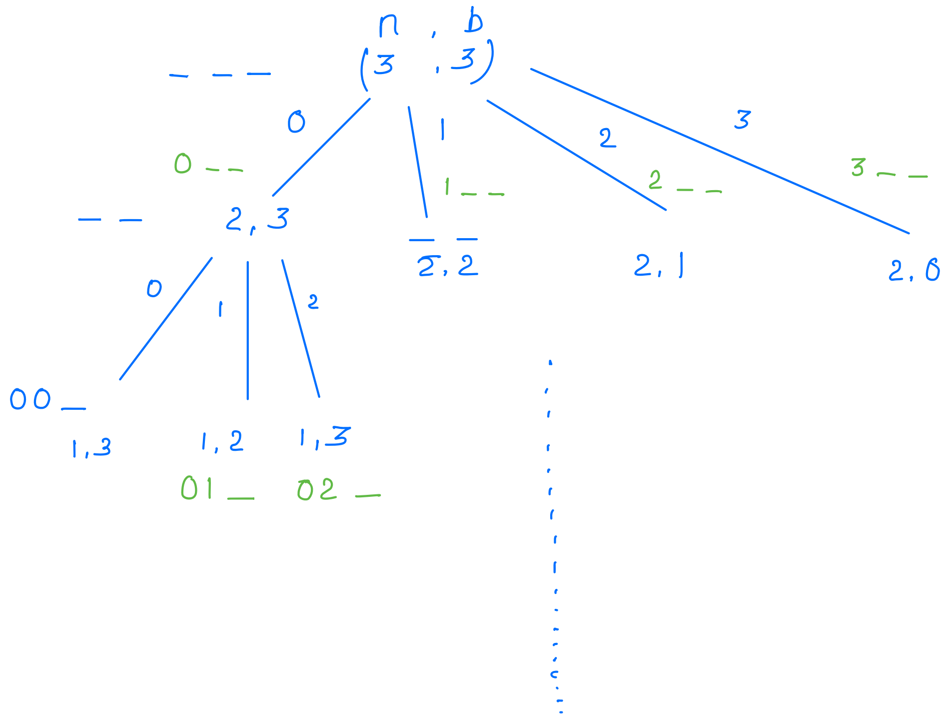
13, 22, 31, 40, ~~04~~

$n = 3$ and $b = 3$

111, 120, 210, 102, 300, 201

Approach

n = 3 and b = 3



Overlapping subproblems

dp [n+1] [sum+1]

Thankyou :)

Pseudocode

```
int numberwithsum(int n, int sum) {
```