Current PSP 52.15 → 60

Nov23_PSP_11May
Piyush Kumar
kameswarreddy Yeddula
Sawan
Gobika K
Yash Malviya
Vijay V A
Sai Sharath
Harshil Dabhoya
Suraj Devraye
sakthivel R
Rajeev
Manjunatha I
Kevin Theodore E

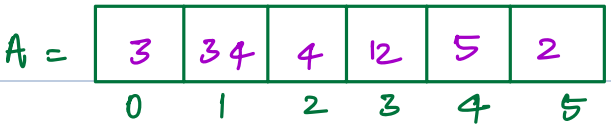
Nov23_PSP_11May
Manoj Vijayakumar
Suchita Sinha
Hemal Makwana
MD JASHIMUDDIN
Mohammad Mateen
manikandan m
Vigneshwaran K
kumkum
Pradeep Kumar Chandra
Shaurya Srivastava
Prashant Kumar Soni
Pranadarth S
Mohammed Arshad

Important Events

1. Full syllabus contest
2. Mock Interview
3. Time & Backlog management

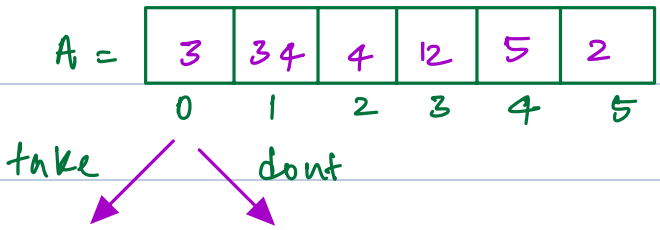
Target Sum

You are given a set of non-negative integers & a target sum. Determine whether there exist a subset whose sum is equal to target sum



sum = 9 ans = True

Brute Force Approach



sum = 9

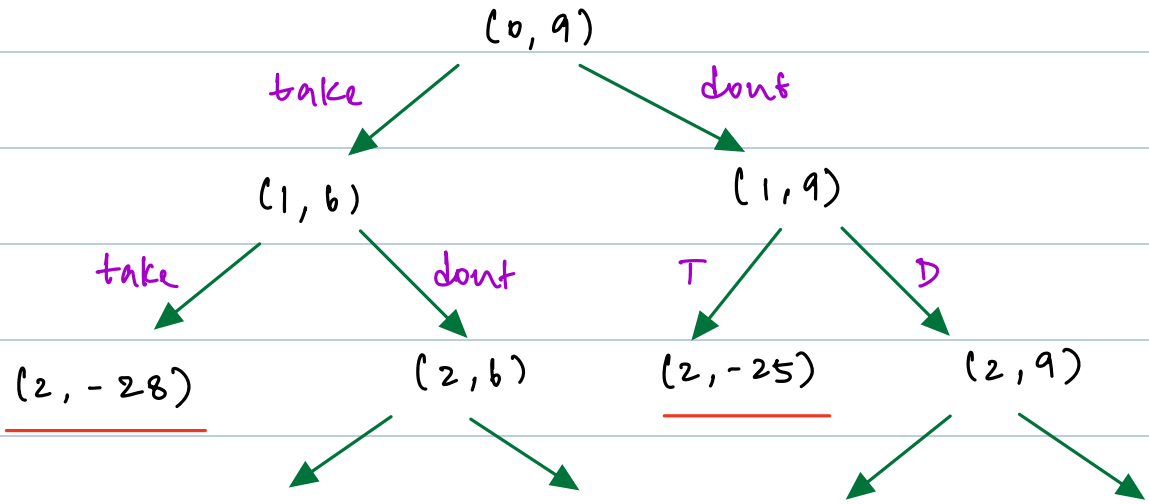
```
boolean targetSum ( Index, total ) {
    If (total == 0) return true;
    If ( Index >= N) return false;
    If (total < 0) return false;
    take = targetSum (Index+1, total - A[Index])
    dont = targetSum (Index+1, total);
    return take || dont;
}
```

T.C = $O(2^n)$

Dry Run

A = | 3 | 34 | 4 | 12 | 5 | 2 |
0 1 2 3 4 5

sum = 9



Memoize based on DP (Pndex, total)

size total

DP [N] [M] = -1

pre req

```
boolean targetSum ( Pndex, total ) {
    Pf (total == 0) return true;
    Pf ( Pndex >= N) return false;
    Pf (total < 0) return false;
    Pf (DP[Pndex] [total] != -1)
        return DP[Pndex] [total]        // Recurr
    take = targetSum (Pndex+1, total - A[Pndex])
    dont = targetSum (Pndex+1, total);
    DP[Pndex] [total] = take || dont;   // store
    return take || dont;
}
```

Flipkart 's Suggestion Problem

Flipkart wants to make shopping easier for their customers. They plan to ask customers what they need and how much money they want to spend. Then, based on this information, Flipkart will suggest the best products for them to buy. This way, customers can quickly find what they want within their budget and maximizing the customer satisfaction at the same time.

Given budget of user and cost and happiness value for N items of the desired product.

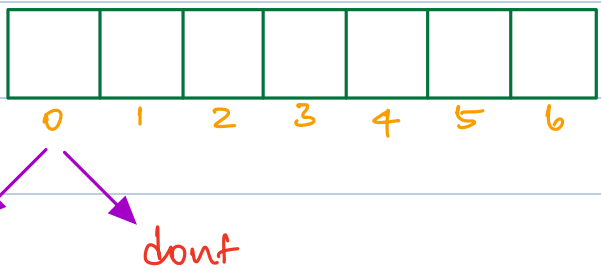
Compute max happiness value.

Namkeen Type	Price	Happiness
Type 1	110	39
Type 2	180	87
Type 3	50	13
Type 4	120	44
Type 5	100	24

Budget = 300

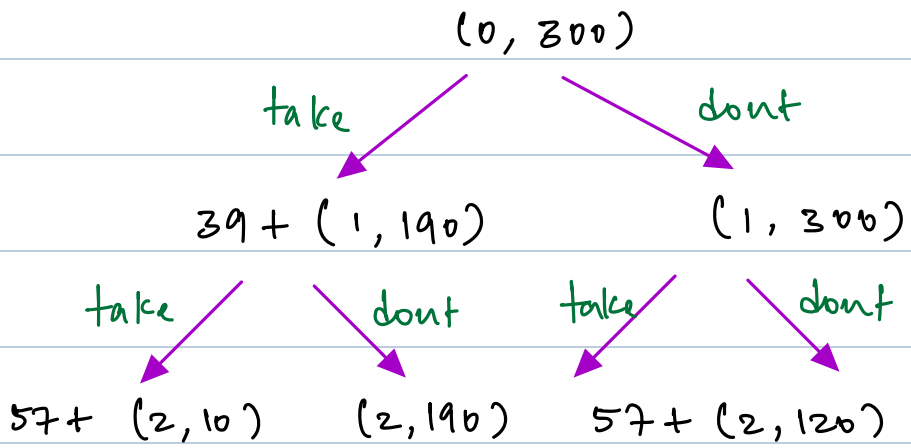
Output = 101

Is it 0/1 Knapsack or unbounded knapsack.



pseudo code

	P	H
0	110	39
1	180	57
2	50	13
3	120	44
4	100	24



pseudo code

```
int maxHappiness (Index, budget) {
    if (Index >= N) return INT_MIN;
    dont = maxHappiness (Index+1, budget)
    take = INT_MIN;
    if (budget >= price[Index])
        take = H[Index] + maxHappiness (
                Index+1, budget - p[Index])
    }
    return max (take, dont);
}
```

Memoize

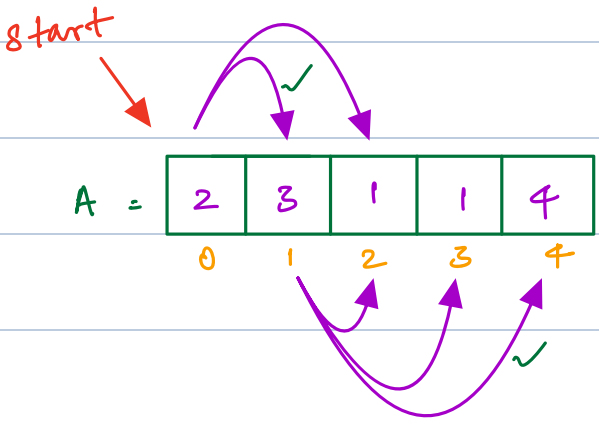
Minimum Jumps to reach end ★★★ Amazon

You are given A[N] you are initially positioned at nums[0]

Each A[i] represents the max length of forward jump from index i

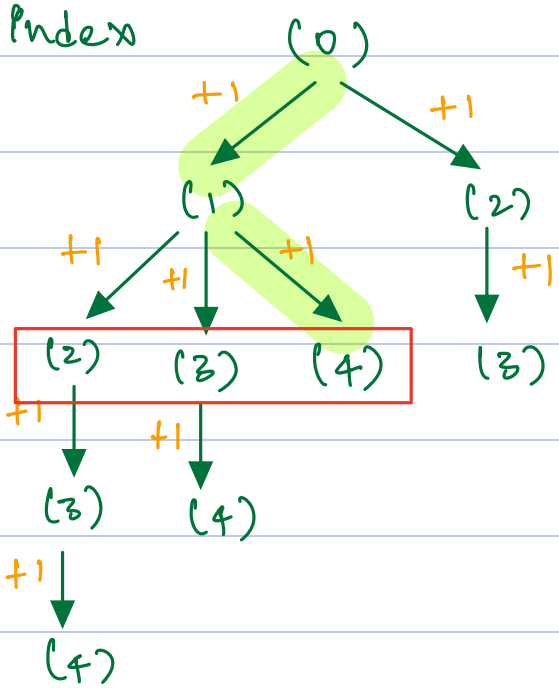
Return min no. of jumps to reach nums[n-1]

NOTE: You can always reach end



ans = 2

Dry Run



pseudo code

```
DP[index] = INT_MAX,

int minJumps(index){
    if(index >= N-1) return 0;
    jumps = INT_MAX;
    if(DP[index] != INT_MAX) return DP[idx]
    for(step=0; step<=A[index]; step++){
        n_index = index + step;
        jumps = min(jumps, 1+minJumps(
                                n_index));
    }
    DP[index] = jumps;          // store
    return jumps;
}
```

$$T.C = \frac{\text{No. of unique DP states}}{N} * \frac{\text{T.C per state}}{\max(A)}$$

$$T.C = (N * \max(A))$$

Break : 10:21 pm

N digit Numbers ★★★

Find out no. of A digit positive numbers whose digits on being added equal to a given no. B

Note: valid no. starts from 1-9. leading 0's not allowed

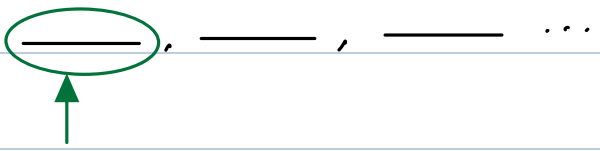
A = 2 B = 4 → {13, 31, 22, 40} → 4 ways

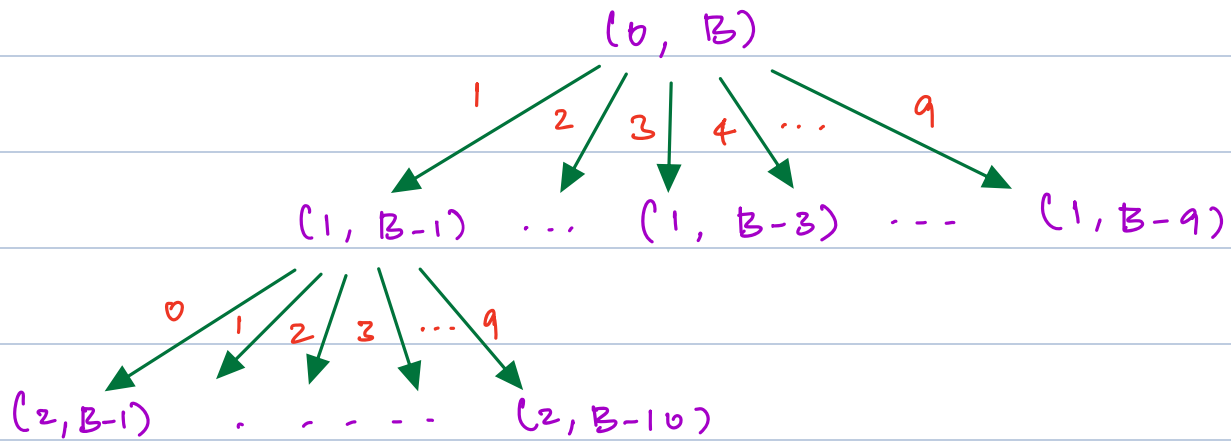
A = 1 B = 3 → {3} → 1 way

Brute force

Run from 10^{A-1} to 10^{A} and check all digits with sum = B

Observation

A digits 



pseudo code

```
int nDigits ( pos, target, start) {
    if (target < 0) return 0; // Impossible
    if (pos == A) {
        if (target == 0) return 1;
        else return 0;
    }

    if (DP[pos][target] != -1)
            return DP[pos][target];

    ways = 0
    for ( i = start; i < 10; i++) {
        ways += nDigits ( pos+1, target - i, 0)
        way %= MOD;
    }

    DP[pos][target] = ways;
    return ways;
}
```

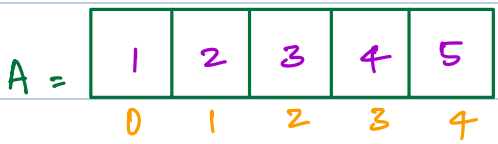
A — B — 0/1 → Initial

T.C = O(AB) S.C = O(A*B)

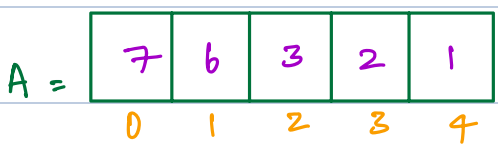
Maximum Profit from Stock Price

Given an array A where i^{th} element represents the price of stock on day i, the objective is to find the max profit

We are allowed to complete as many transactions as desired but engaging in multiple transactions simultaneous is not allowed

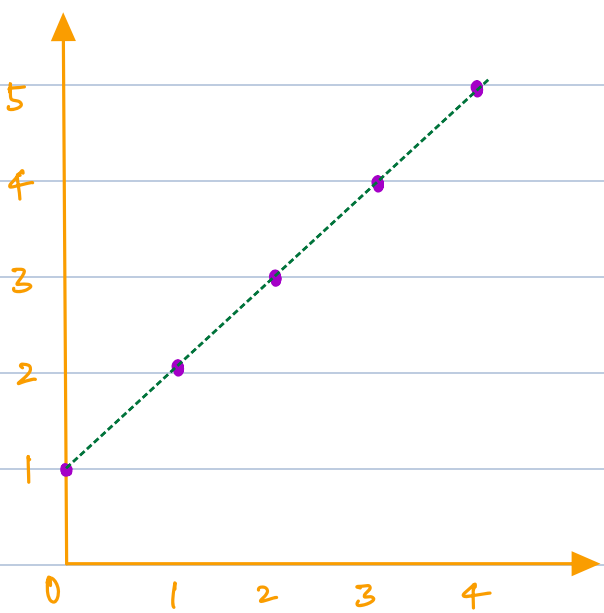


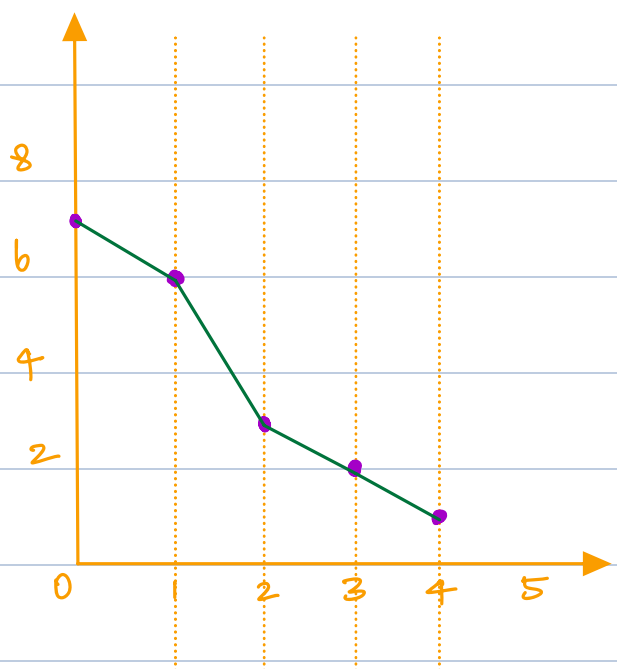
profit = 4



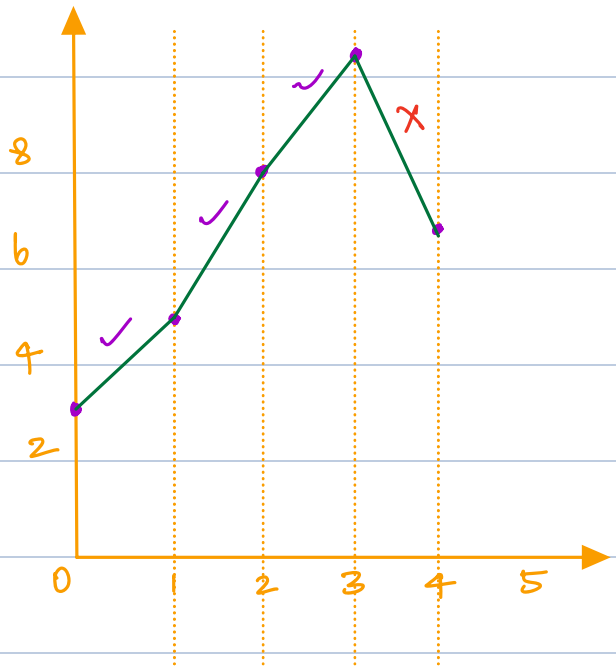
profit = 0

Visualize





#pseudo code




```
total = 0;
for (i=0; i<n-1; i++) {
    fv = A[i+1];
    pv = A[i]
    if (fv > pv) {
        total += (fv - pv);
    }
}
```

T.C = O(n) S.C = O(1)