# Problem 638: Shopping Offers

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 52.10%
**Paid Only:** No
**Tags:** Array, Dynamic Programming, Backtracking, Bit Manipulation, Memoization, Bitmask

## Problem Description

In LeetCode Store, there are `n` items to sell. Each item has a price. However, there are some special offers, and a special offer consists of one or more different kinds of items with a sale price.

You are given an integer array `price` where `price[i]` is the price of the `ith` item, and an integer array `needs` where `needs[i]` is the number of pieces of the `ith` item you want to buy.

You are also given an array `special` where `special[i]` is of size `n + 1` where `special[i][j]` is the number of pieces of the `jth` item in the `ith` offer and `special[i][n]` (i.e., the last integer in the array) is the price of the `ith` offer.

Return _the lowest price you have to pay for exactly certain items as given, where you could make optimal use of the special offers_. You are not allowed to buy more items than you want, even if that would lower the overall price. You could use any of the special offers as many times as you want.

**Example 1:**

**Input:** price = [2,5], special = [[3,0,5],[1,2,10]], needs = [3,2] **Output:** 14 **Explanation:** There are two kinds of items, A and B. Their prices are $2 and $5 respectively. In special offer 1, you can pay $5 for 3A and 0B In special offer 2, you can pay $10 for 1A and 2B. You need to buy 3A and 2B, so you may pay $10 for 1A and 2B (special offer #2), and $4 for 2A.

**Example 2:**

**Input:** price = [2,3,4], special = [[1,1,0,4],[2,2,1,9]], needs = [1,2,1] **Output:** 11 **Explanation:** The price of A is $2, and $3 for B, $4 for C. You may pay $4 for 1A and 1B, and $9 for 2A ,2B and 1C. You need to buy 1A ,2B and 1C, so you may pay $4 for 1A and 1B (special offer #1), and $3 for 1B, $4 for 1C. You cannot add more items, though only $9 for 2A ,2B and 1C.

**Constraints:**

* `n == price.length == needs.length` * `1 <= n <= 6` * `0 <= price[i], needs[i] <= 10` * `1 <= special.length <= 100` * `special[i].length == n + 1` * `0 <= special[i][j] <= 50` * The input is generated that at least one of `special[i][j]` is non-zero for `0 <= j <= n - 1`.

## Code Snippets

**C++:**

```
class Solution {
public:
int shoppingOffers(vector<int>& price, vector<vector<int>>& special,
vector<int>& needs) {

}
};
```

**Java:**

```
class Solution {
public int shoppingOffers(List<Integer> price, List<List<Integer>> special,
List<Integer> needs) {

}
}
```

**Python3:**

```
class Solution:
def shoppingOffers(self, price: List[int], special: List[List[int]], needs:
List[int]) -> int:
```