You have n boxes labeled from 0 to n - 1. You are given four arrays: status, candies, keys, and containedBoxes where:

* status[i] is 1 if the ith box is open and 0 if the ith box is closed,
* candies[i] is the number of candies in the ith box,
* keys[i] is a list of the labels of the boxes you can open after opening the ith box.
* containedBoxes[i] is a list of the boxes you found inside the ith box.

You are given an integer array initialBoxes that contains the labels of the boxes you initially have. You can take all the candies in **any open box** and you can use the keys in it to open new boxes and you also can use the boxes you find in it.

Return *the maximum number of candies you can get following the rules above*.

**Example 1:**

**Input:** status = [1,0,1,0], candies = [7,5,4,100], keys = [[],[],[1],[]], containedBoxes = [[1,2],[3],[],[]], initialBoxes = [0]

**Output:** 16

**Explanation:** You will be initially given box 0. You will find 7 candies in it and boxes 1 and 2.

Box 1 is closed and you do not have a key for it so you will open box 2. You will find 4 candies and a key to box 1 in box 2.

In box 1, you will find 5 candies and box 3 but you will not find a key to box 3 so box 3 will remain closed.

Total number of candies collected = 7 + 4 + 5 = 16 candy.

**Example 2:**

**Input:** status = [1,0,0,0,0,0], candies = [1,1,1,1,1,1], keys = [[1,2,3,4,5],[],[],[],[],[]], containedBoxes = [[1,2,3,4,5],[],[],[],[],[]], initialBoxes = [0]

**Output:** 6

**Explanation:** You have initially box 0. Opening it you can find boxes 1,2,3,4 and 5 and their keys.

The total number of candies will be 6.

**Constraints:**

* n == status.length == candies.length == keys.length == containedBoxes.length
* 1 <= n <= 1000
* status[i] is either 0 or 1.
* 1 <= candies[i] <= 1000
* 0 <= keys[i].length <= n
* 0 <= keys[i][j] < n
* All values of keys[i] are **unique**.
* 0 <= containedBoxes[i].length <= n
* 0 <= containedBoxes[i][j] < n
* All values of containedBoxes[i] are unique.
* Each box is contained in one box at most.
* 0 <= initialBoxes.length <= n
* 0 <= initialBoxes[i] < n