Author

[Ayamin](https://codefights.com/profile/Ayamin)

https://codefights.com/img/coins_new.png2000

For the given set S its *powerset* is the set of all possible subsets of S.

Given an array of unique integers nums, your task is to return the *powerset* of its elements.

Implement an algorithm that does it in a depth-first search fashion. That is, for every integer in the set, we can either choose to take or not take it. At first, we choose NOT to take it, then we choose to take it.

**Example**

* For nums = [1, 2], the output should be  
  Powerset(nums) = [[], [2], [1], [1, 2]].  
  Here's how the answer is obtained:

don't take element 1

----don't take element 2

--------add []

----take element 2

--------add [2]

take element 1

----don't take element 2

--------add [1]

----take element 2

--------add [1, 2]

* For nums = [1, 2, 3], the output should be  
  Powerset(nums) = [[], [3], [2], [2, 3], [1], [1, 3], [1, 2], [1, 2, 3]].
* **[input] array.integer nums**

Array of unique positive integers, 1 ≤ nums.length ≤ 10.

* **[output] array.array.integer**

The *powerset* of nums.

<https://codefights.com/challenge/WgTYsRejc9qeToQdC/main?utm_source=challengeOfTheDay&utm_medium=email&utm_campaign=email_notification>

static List<List<int>> Powerset(int[] nums)

{

List<List<int>> ps = new List<List<int>>();

ps.Add(new List<int>()); // add the empty set

// for every item in the original list

foreach (int item in nums)

{

List<List<int>> newPs = new List<List<int>>();

foreach (List<int> subset in ps)

{

// copy all of the current powerset's subsets

newPs.Add(subset);

// plus the subsets appended with the current item

List<int> newSubset = new List<int>(subset);

newSubset.Add(item);

newPs.Add(newSubset);

}

// powerset is now powerset of list.subList(0, list.indexOf(item)+1)

ps = newPs;

}

return ps;

}