Product of lists with Backtracking

#itertools #product #backtracking

You have `n` number of lists. You have to produce a list of combinations such that, every combination contains only one element from each list.

*You can also do the following with `itertools.product(lists)` . Anyway, let's flex some Algorithmic muscles.

```
def product(lists, i, li, prod, res):
    if len(prod)==len(lists):
        res.append(prod[:])
        return
    for j in range(i,len(lists[li])):
        prod.append(lists[li][j])
        product(lists,i, li+1, prod, res)
        prod.pop()
    return res

if __name__== '__main__':
    lists=[
        [0,1,2,3],
        [0,1,2,3,4],
        [0,1,2,3,4,5]
        ]
    result = product(lists,i=0, li=0, prod=[], res=[])
    print("Output:\n",result)
```

Output:

```
[
[0, 0, 0], [0, 0, 1], [0, 0, 2], [0, 0, 3], [0, 0, 4], [0, 0, 5], [0, 1, 0],
[0, 1, 1], [0, 1, 2], [0, 1, 3], [0, 1, 4], [0, 1, 5], [0, 2, 0], [0, 2, 1],
[0, 2, 2], [0, 2, 3], [0, 2, 4], [0, 2, 5], [0, 3, 0], [0, 3, 1], [0, 3, 2],
[0, 3, 3], [0, 3, 4], [0, 3, 5], [0, 4, 0], [0, 4, 1], [0, 4, 2], [0, 4, 3],
[0, 4, 4], [0, 4, 5], [1, 0, 0], [1, 0, 1], [1, 0, 2], [1, 0, 3], [1, 0, 4],
[1, 0, 5], [1, 1, 0], [1, 1, 1], [1, 1, 2], [1, 1, 3], [1, 1, 4], [1, 1, 5],
[1, 2, 0], [1, 2, 1], [1, 2, 2], [1, 2, 3], [1, 2, 4], [1, 2, 5], [1, 3, 0],
[1, 3, 1], [1, 3, 2], [1, 3, 3], [1, 3, 4], [1, 3, 5], [1, 4, 0], [1, 4, 1],
[1, 4, 2], [1, 4, 3], [1, 4, 4], [1, 4, 5], [2, 0, 0], [2, 0, 1], [2, 0, 2],
[2, 0, 3], [2, 0, 4], [2, 0, 5], [2, 1, 0], [2, 1, 1], [2, 1, 2], [2, 1, 3],
[2, 1, 4], [2, 1, 5], [2, 2, 0], [2, 2, 1], [2, 2, 2], [2, 2, 3], [2, 2, 4],
[2, 2, 5], [2, 3, 0], [2, 3, 1], [2, 3, 2], [2, 3, 3], [2, 3, 4], [2, 3, 5],
[2, 4, 0], [2, 4, 1], [2, 4, 2], [2, 4, 3], [2, 4, 4], [2, 4, 5], [3, 0, 0],
[3, 0, 1], [3, 0, 2], [3, 0, 3], [3, 0, 4], [3, 0, 5], [3, 1, 0], [3, 1, 1],
[3, 1, 2], [3, 1, 3], [3, 1, 4], [3, 1, 5], [3, 2, 0], [3, 2, 1], [3, 2, 2],
[3, 2, 3], [3, 2, 4], [3, 2, 5], [3, 3, 0], [3, 3, 1], [3, 3, 2], [3, 3, 3],
[3, 3, 4], [3, 3, 5], [3, 4, 0], [3, 4, 1], [3, 4, 2], [3, 4, 3], [3, 4, 4],
[3, 4, 5]
]
```

HackerRank Problem: Maximize it

Hard

#heapq #hackerrank #backtracking

You are given a function $f(X) = X^2$. You are also given K lists. The i^{th} list consists of N_i elements.

You have to pick one element from each list so that the value from the equation below is maximized:

$S = (f(X_1) + f(X_2) + \ldots + f(X_k))\%M$

X_i denotes the element picked from the i^{th} list . Find the maximized value S_{max} obtained.

$\%$ denotes the modulo operator.

Note that you need to take exactly one element from each list, not necessarily the largest element. You add the squares of the chosen elements and perform the modulo operation. The maximum value that you can obtain, will be the answer to the problem.

Input Format

The first line contains 2 space separated integers K and M.
The next K lines each contains an integer N_i, denoting the number of elements in the i^{th} list, followed by N_i space separated integers denoting the elements in the list.

Constraints

$1 \le K \le 7$
$1 \le M \le 1000$
$1 \le N_i \le 7$
$1 \le Magnitude\ of\ elements\ in\ list \le 10^9$

Output Format

Output a single integer denoting the value S_{max}.

Sample Input

```
3 1000
2 5 4
3 7 8 9
5 5 7 8 9 10
```

Sample Output

```
206
```

Explanation

Picking 5 from the 1^{st} list, 9 from the 2^{nd} list and 10 from the 3^{rd} list gives the maximum S value equal to $(5^2 + 9^2 + 10^2)\%$ $1000 = 206$.

Solution

```python
import heapq
def maximize_it(arrays, M, i = 0, li =0, prod =[], maxHeap=[]):
    if len(prod)==len(arrays):
        modulo = -1* (sum(x**2 for x in prod)%M)
        heapq.heappush(maxHeap, modulo)
        return
    for j in range(i,len(arrays[li])):
        prod.append(arrays[li][j])
        maximize_it(arrays, M, i, li+1, prod, maxHeap)
        prod.pop()
    return maxHeap[0]*(-1)

if __name__== '__main__':
    arrays=[
        [2, 5, 4],
        [3, 7, 8, 9],
        [5, 5, 7, 8, 9, 10]
        ]
    M = 1000
    result = maximize_it(arrays, M)
    print(result)
```

Output

```
206
```

Intuitive Solution using itertools

#itertools #product

```python
import itertools
def maximize_it(arrays,M):
    max_modulo = 0
    for product in itertools.product(*arrays):
        modulo = sum(x**2 for x in product)%M
        if modulo > max_modulo:
            max_modulo = modulo
    return max_modulo
if __name__== '__main__':
        arrays=[ [2, 5, 4], [3, 7, 8, 9], [5, 5, 7, 8, 9, 10] ]
        M = 1000
        result = maximize_it(arrays, M)
        print(result)
```

Output

```
206
```