368. Largest Divisible Subset

Medium 484 23 Favorite Share

Given a set of **distinct** positive integers, find the largest subset such that every pair (S_i, S_j) of elements in this subset satisfies:

S_i % S_j = 0 or S_j % S_i = 0.

If there are multiple solutions, return any subset is fine.

Example 1:

```
Input: [1,2,3]
Output: [1,2] (of course, [1,3] will also be ok)
```

Example 2:

```
Input: [1,2,4,8]
Output: [1,2,4,8]
```

Accepted 43,745 | Submissions 126,794

```java
package Algorithm;

import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;

/*
dp数组表示从0~i包括第i个元素最大的divisible subset size
pre数组用来标记 状态转移过程中的方向，用于回溯最大值时的解集。
dp[i]=max{dp[i],dp[j]+1}
 */
public class L368 {
    public List<Integer> largestDivisibleSubset(int[] nums) {
        LinkedList<Integer> res = new LinkedList<Integer>();
        if(nums == null || nums.length == 0)
            return res;
        Arrays.sort(nums);
        int [] dp = new int [nums.length];
        int [] pre = new int [nums.length];
        int maxIdx = -1, max = -1;

        for(int i = 0; i < nums.length; i ++) {
            dp[i] = 1;
            pre[i] = -1;
        }

        for(int i = 1; i < nums.length; i ++) {
            for(int j = 0; j < i; j ++) {
                if(nums[i] % nums[j] == 0 && dp[i] < dp[j] + 1) {
                    dp[i] = dp[j] + 1;
                    pre[i] = j;
                }
            }
        }
        for(int i = 0; i < nums.length; i ++) {
            if(dp[i] > max) {
                max = dp[i];
                maxIdx = i;
            }
        }
        for(int i = maxIdx; i >= 0; ) {
            res.addFirst(nums[i]);
            i = pre[i];
        }
        return res;
    }

}
```