368. Largest Divisible Subset in an Array

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]
```

Intution:

1. Sort the array <------ so that we check only for the condition A[j]%A[i] == 0, i < j
2. create an Array dp[], where dp[i] means size of the longest divisible subset ending with index element A[i]
3. create an Array prevIndex[], where prevIndex[i] means the index of prev element in subset ending at index i
4. min value of dp[i] = 1 , because a single element is always forming a divisivle Subset
5. For every A[i],

- traverse all array elements A[j]
- if(A[i]%A[j] == 0 && dp[i] < dp[j] + 1)
- dp[i] = dp[j]+1, prevIndex[i] = j

6. Also keep track of the index with largest divisible subset, max_ind = i
7. finally return the array formed by elements at indices {max_ind,prevIndex[max_ind],prevInd[prevIndex[max_ind]],...}

CODE:

```
vector<int> largestDivisibleSubset(vector<int>& A) {
      int n = A.size();
      if(n == 0)   return {};
      sort(A.begin(),A.end());
      vector<int> dp(n,1),prevIndex(n,-1),ans;
      int max_ind = 0;
      for(int i=1;i<n;i++){
          for(int j=0;j<i;j++){
              if(A[i]%A[j] == 0 && dp[i] < dp[j] + 1){
                dp[i] = dp[j]+1;
                prevIndex[i] = j;
              }
           }
           if(dp[max_ind] < dp[i])
               max_ind = i;
       }
       int k = max_ind;
       while(k>=0){
          ans.push_back(A[k]);
```

```
        k = prevIndex[k];
    }
    return ans;
}
```