Division - Writeup

This exercise is marked as the challenge for VG101(SU2020) - Lab 4.

Problem Statement:

Divide number n into k numbers $a_1 \cdots a_k$, such that

$\sum_{i=1}^{k} a_i = n$.

Find how many different methods are there. Two division a, b are considered same if they can be rearranged to be the same.

For $n = 7, k = 3$, the only four methods are

$\{1, 1, 5\}; \{1, 2, 4\}; \{1, 3, 3\}; \{2, 2, 3\}$.

Challenge: find the case for $n = 1000$ and $k = 100$. If the answer is too large, please give the answer modulo by 998244353.

Writeups collected from VG101-SU2020:

MatrixpecKer | MATLAB | Naïve Solution

Simulate the process and test the divisions one by one:

```
% mydiv([],7,3)

function mydiv(s,n,k)
    if k==1
        s=[s n];
        disp(s);
        return;
    end

    if length(s)==0
        for i=1:floor(n/k)
            mydiv([s i],n-i,k-1);
        end
    else
        for i=s(length(s)):floor(n/k)
```

```
            mydiv([s i],n-i,k-1);
        end
    end
end
```

Note that this process includes many repeated calculations. For $n > 100$, a better algorithm is needed.

Scarlet | N/A | Optimized Solution

Denote the set of distinct divisions for the problem as $p(n,k)$.

Consider any division from the case $p(n,k)$

1. The division includes "1": just remove (any of) this 1, this new division must belong to the case $p(n-1,k-1)$.
2. The division doesn't include "1": substract 1 from all the elements in the division, then we will get a new division exactly from the case $p(n-k,k)$.

For example, for $n=7$ and $k=3$, we have.

1. $\{1,1,5\};\{1,2,4\};\{1,3,3\}$: Here we remove the 1 in each division and get $\{1,5\};\{2,4\};\{3,3\}$. They are just the cases in $p(6,2)$.
2. $\{2,2,3\}$: Subtract 1 from each element and get $\{1,1,2\}$. It is just the case in $p(4,3)$;

It's obvious that the divisions get from the previous step are distinct. So we can count the number in the set by $|p(n,k)| = |p(n-1,k-1)| + |p(n-k,k)|$. It can be implemented by recursion with $p(n,k)$ saved for each time.

See **Binhao Qin's** recursion formula (*state transition equation*) for a clear reference.

The math fact tells us that we can perform the modulo after each addition without affecting the final result. See **Grid's** MATLAB code for implementing details.

Binhao Qin | Python | 0.069s

Basically, all of us are using the set of recursion conditions that

$$f(n,k) = \begin{cases} 0, \text{if } n < k \\ 1, \text{if } n == k \text{ or } k == 1 \\ f(n-k,k) + f(n-1,k-1) \text{ otherwise} \end{cases}$$

(Python code implementation in Appendix.)

SaltyFish(Grid) | MATLAB | 0.101s

> The recursion formula is found at Wikipedia.

A caching mechanism is employed to avoid repetitive calculation.

In MATLAB, the answer is too large to remain precise, so we can only have the answer modulo 998244353:

```
global cache
cache = zeros(1000, 100);
disp(p(1000, 100))

function ret = p(n, k)
```

```
    global cache
    if k == 1 || k == n || k == n-1
        ret = 1;
        return
    end
    if k > n
        ret = 0;
        return
    end
    if ~cache(n, k)
        cache(n, k) = mod(p(n-1, k-1) + p(n-k, k),998244353);
    end
    ret = cache(n, k);
end
```

In MATLAB, the time cost is increased to 0.462s (0.392s on the second run).

Limlimg | MATLAB | 0.044602s

```
% non-recursive. More calculation done than necessary .

% Let a1..ak be a sorted sequence with the sum of n. a1 as the smallest
% number, falls in the range 1:floor(n/k).

% a2..ak are all greater than or equal to a1. Define b1..b(k-1), where
% bi = a(i+1) - a1 + 1.

% Then b1..b(k-1) is a positive sorted sequence with the sum of
% n - a1 - (k-1)*(a1-1)

% The number of possible b1..b(k-1) is exactly the number of possible
% a2..ak. Sum these numbers for all a1 to get the result.

function result = Division(n, k)
    D = zeros(n, k);
    % D is a matrix where D(n, k) = Division(n, k).
    % To calculate D(n, k), only elements in smaller rows and columns are
    % used. So fill out this matrix by row or column.
    for i = 1:n
        D(i, 1) = 1;
        for j = 2:k
            if j == i
                D(i, j) = 1;
                break;
                % break because the remaining elements are unused
            else
                for a1 = 1:floor(i/j)
                    D(i, j) = mod(D(i, j) + D(i-a1-(j-1)*(a1-1), j-1),
998244353);
                end
            end
        end
    end
    result = D(n, k);
end
```

A better version following the solution described by Scarlet (0.002768s):

```
function result = Division(n, k)
    D = zeros(n, k);
    D(1, 1) = 1;
    for i = 2:n
        D(i, 1) = 1;
        for j = 2:k
            if j == i
                D(i, j) = 1;
                break;  % leaving zeros
            else
                D(i, j) = mod(D(i-j, j) + D(i-1, j-1), 998244353);
            end
        end
    end
    result = D(n, k);
end
```

Yifan Shen

SaltierFish | MATLAB | 0.005236s

Some explanations on the boundary conditions

Dividing any number into only one number can have only one method (itself), so $p(n,1)=1$ for all n.

Since every number used in the division must be ≥ 1, there are no ways to divide a number into more parts than itself, so if $n<k$, $p(n,k)=0$.

Dividing some number n into n parts would also have only one method $(1,1,\ldots,1)$ so $p(n,n)=1$.

These three conditions ensure that when doing $p(n,k)=p(n-1,k-1)+p(n-k,k)$ the indices don't go out of range.

A straight forward, easy to read code (This code uses iteration from p(1,1) to p(n,k) instead of recursion)

```
function solution = division(n,k)
    p = zeros(n,k);
    for j = 1:k
        for i = 1:n
            if j == 1 || i == j
                p(i,j) = 1;
            elseif i < j
                p(i,j) = 0;
            else
                p(i,j) = mod(p(i-1,j-1)+p(i-j,j),998244353);
            end
        end
    end
    solution = p(n,k);
end
```

Haichuan Wang | Matlab | 0.199574s

The same method as above. An implementation with recursion and dp table in Matlab. The use of variables is inspired by Yuxuan Xia.

```matlab
function [dp,count] = divCounter3(dp,n,k)
    if k == 1 || n == k || n-1 == k
        count = 1;
    elseif n < k
        count = 0;
    else
        if dp(n-1,k-1) ~= 0
            countTemp1 = dp(n-1,k-1);
        else
            [dp,countTemp1] = divCounter3(dp,n-1,k-1);
        end
        if dp(n-k,k) ~= 0
            countTemp2 = dp(n-k,k);
        else
            [dp,countTemp2] = divCounter3(dp,n-k,k);
        end
        count = mod(countTemp1 + countTemp2,998244353);
        dp(n,k) = count;
    end
end
```

Mysterious Challenger | python | N/A

LRU-cache used, see Appendix.

Appendix

For python implementation, see here!