COMPUTATION OF THE CHARACTER TABLE OF THE SYMMETRIC GROUP

YOUXING Z

Abstract. We introduce an algorithm to compute the character table of the symmetric group S_n and implement it in Golang.

1. Introduction

Character table is a square matrix, whose rows correspond to irreducible representations, and columns correspond to conjugacy classes of group elements. Symmetric group is a very important group in the study of algebra. And the character table of symmetric group is very useful in some subjects such as physical chemistry, quantum chemistry, etc. However, we cannot quickly compute large tables using existing tools such as Sagemath, GAP. Therefore we implement a specific algorithm to calculate the table for symmetric group.

2. Conjugacy Classes of Symmetric Group

Definition 2.1 (Conjugate)**.** Let G be a group, $g, h \in G$, if there is some $x \in G$ such that $xgx^{-1} = h$, we say that g and h are **conjugate** in G.

Definition 2.2 (Conjugacy Classes)**.** We can easily verify that conjugate relation is an equivalence relation, then we say **conjugacy classes** are the equivalence classes under this relation of a group G. i.e. the conjugacy classes of $g \in G$ is $[g] = \{xgx^{-1} | x \in G\}$.

Let S_n be a symmetric group with order $n!$, and permutation $\alpha \in S_n$. We know that we can write α as a product of disjoint cycles, denote by $\alpha = c_1c_2c_3 \cdots c_k$, where c_i $(i = 1, 2, \cdots, k)$ is a cycle. Then we have definition:

Definition 2.3 (Cycle Type)**.** Let $l_i = |c_i|$ be the length of each cycle c_i, $i = 1, 2, \cdots, k$, and $l_i \geq l_{i+1}$, we say that $(l_1, l_2, \cdots, l_k)$ is a **cycle type** of the permutation $\alpha = c_1c_2c_3 \cdots c_k$.

Example 2.4. Let $\sigma \in S_8$ and $\sigma = (123)(45)(67)$, then the cycle type is $(3, 2, 2, 1)$

Theorem 2.5. The permutations g and h are conjugate in symmetric group S_n if and only if they have same cycle type.

Proof. We can easily verify that conjugate of an m-cycle is an m-cycle, and the conjugate of product is a product of conjugates. Since g, h can be written as a product of cycles, if g and h are conjugate, then they have same cycle type.

Conversely, suppose g, h have same cycle type $(l_1, l_2, \cdots, l_k)$, let $g = g_1 g_2 \cdots g_k$ and $h = h_1 h_2 \cdots h_k$, where g_i and h_i are l_i-cycles, $i = 1, 2, \cdots, k$. For each i, we write $g_i = (g_{i1}, g_{i2}, \cdots, g_{il_i})$ and $h_i = (h_{i1}, h_{i2}, \cdots, h_{il_i})$, then define permutation τ by $\tau(g_{ij}) = h_{ij}$, $j = 1, 2, \cdots, l_i$, hence $\tau g_{ij} \tau^{-1} = h_{ij}$. Then we have

$$\tau g \tau^{-1} = \tau g_1 g_2 \cdots g_k \tau^{-1} = \tau g_1 \tau^{-1} \tau g_2 \tau^{-1} \cdots \tau g_k \tau^{-1} = h_1 h_2 \cdots h_k = h$$

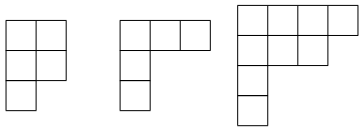
Hence the conjugacy classes of symmetric group S_n are determined by cycle type.

3. Partition

Definition 3.1 (Partition)**.** A **partition** is a sequence $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_l)$ of non-negative integers in decreasing order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_l > 0$, l is the **length** of λ, $|\lambda| = \sum_{i=1}^{l} \lambda_i$ is the **size** of λ.

Corollary 3.2. Cycle type is a partition.

Definition 3.3 (Young Diagram)**.** For a given partition λ, the **Young diagram** is a finite collection of boxes, which contains λ_i boxes in row i.



$(2, 2, 1)$ $(3, 1, 1)$ $(4, 3, 1, 1)$

Definition 3.4 (Young Tableau)**.** A **Young tableau** is obtained by filling in the boxes of the Young diagram with symbols taken from some ordered alphabet. We say it is **standard** if the entries in each row and each column are increasing.

Young tableua:

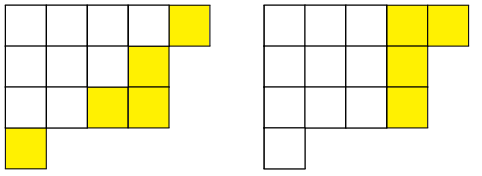
2	1	7	4	5
6	3	8	9	
12	11	14	13	
10				

Standard Young tableau (STY):

1	2	3	4	5
6	7	8	9	
10	11	12	13	
14				

Definition 3.5 (Border Strip (BS))**.** Let λ, μ be partitions with $\lambda \supseteq \mu$, then the set-theoretic difference $\lambda \backslash \mu$ is called a **skew diagram**. A skew diagram is **border strip** if it is connected (two boxes have a common edge) and does not contain a 2×2-block of boxes. The **height** $ht(\lambda)$ of a border strip is the number of rows it touches minus one. We define $BS(\lambda, l)$ as the set of all border strips with length $l = |\lambda \backslash \mu|$ in partition λ.

Example 3.6.



$(5,4,3,1)\backslash(4,3,2)$ $(5,4,3,1)\backslash(3,3,3,1)$

The shaded region in the figure above are skew diagrams, and $\rho = (5,4,3,1)\backslash(3,3,3,1)$ is a border strip with height $ht(\rho) = 3 - 1 = 2$.

Definition 3.7 (Border Strip Tableau (BST))**.** The **border strip tableau** $T(\lambda, \rho)$ of a partition λ and a weight partition ρ is a filling of the Young diagram λ such that exactly ρ_i boxes are labeled, and rows and columns are non-decreasing. Moreover, for each i the boxes labeled i must form a border strip. The **height** $ht(T)$ is the sum of the heights of the border strips in tableua $T(\lambda, \rho)$, i.e. $ht(T) = \sum_{\xi \in T(\lambda,\rho)} ht(\xi)$. We define $BST(\lambda, \rho)$ as the set of all possible border strip tableaux determined by partition λ and ρ. i.e. $BST(\lambda, \rho) = \{T_1(\lambda, \rho), T_2(\lambda, \rho), \cdots\}$.

Example 3.8. Consider size 8, a partition $\lambda = (5,2,1)$ and a weight partition $\rho = (3,3,1,1)$. Then we have 6 border strip tableaux, $BST((5,2,1),(3,3,1,1))$:

1	1	1	3	4
2	2			
2				

T_1

1	1	2	2	2
1	3			
4				

T_2

1	1	2	2	2
1	4			
3				

T_3

1	2	2	3	4
1	2			
1				

T_4

1	2	2	2	4
1	3			
1				

T_5

1	2	2	2	3
1	4			
1				

T_6

The height $ht(T_1) = \sum_{\xi \in T_1} ht(\xi) = 0 + 1 + 0 + 0 = 1$.

4. Irreducible Representation of Symmetric Group

Definition 4.1 (Representation)**.** A **representation** of a finite group G on a finite-dimensional complex vector space V is a homomorphism $\rho : G \to GL(V)$ of G to the group of automorphisms of V. We often call V itself a representation of G.

Definition 4.2 (Irreducible)**.** A representation V is called **irreducible** if there is no proper nonzero invariant subspace W of V.

Definition 4.3 (P_λ, Q_λ)**.** Given a standard Young tableau of Young diagram of partition λ with size n, then we define two subgroups of S_n:

$$P = P_\lambda = \{\sigma \in S_n | \ \sigma \text{ preserves each row of } \lambda\}$$
$$Q = Q_\lambda = \{\sigma \in S_n | \ \sigma \text{ preserves each column of } \lambda\}$$

Example 4.4. For S_7, $\lambda = (3, 2, 2)$ we have standard Young tableua:

1	2	3
4	5	
6	7	

Then $P = \{(123)(45)(67), (132)(45)(67), \cdots\}$, $Q = \{(146)(257), (164)(257), \cdots\}$.

Definition 4.5 (V_λ)**.** If V is a representation of group algebra $\mathbb{C}[S_n]$ on a vector space V with homomorphism $\mathbb{C}[S_n] \to \text{End}(V)$, and $\{e_\sigma\}$ is a basis of V corresponding to elements $\{\sigma\}$ of S_n, and λ is any partition of n. We then define **Young symmetrizers**:

$$a_\lambda = \sum_{\sigma \in P} e_\sigma \qquad b_\lambda = \sum_{\sigma \in Q} \text{sgn}(\sigma) \cdot e_\sigma \qquad c_\lambda = a_\lambda b_\lambda$$

and V_λ:

$$V_\lambda = \mathbb{C}[S_n] \cdot c_\lambda = \mathbb{C}[S_n] \cdot a_\lambda b_\lambda$$

Example 4.6.

Trivial representation: $V_{(n)} = \mathbb{C}[S_n] \cdot \sum_{\sigma \in S_n} e_\sigma = \mathbb{C} \cdot \sum_{\sigma \in S_n} e_\sigma$

Alternating representation: $V_{(1,\cdots,1)} = \mathbb{C}[S_n] \cdot \sum_{\sigma \in S_n} \text{sgn}(\sigma) e_\sigma = \mathbb{C} \cdot \sum_{\sigma \in S_n} \text{sgn}(\sigma) e_\sigma$

Standard representation: Consider S_3, $\lambda = (2, 1)$, we have:

$$c_\lambda = a_\lambda b_\lambda = (e_1 + e_{(1,2)})(e_1 - e_{(1,3)}) = 1 + e_{(1,2)} - e_{(1,3)} - e_{(1,2,3)}$$

$V_{(2,1)}$ is spanned by c_λ and $(13)c_\lambda$, then $V_{(2,1)}$ is standard representation of S_3.

Theorem 4.7. Each V_λ is an irreducible representation of S_n.

The proof is given in book *Representation Theory* [2], this theorem shows that the irreducible representation of S_n are determined by partition λ.

5. Character Table of Symmetric Group

Definition 5.1 (Character)**.** If V is a representation of G, its **character** χ_V is the complex-valued function on the group defined by

$$\chi_V(g) = \mathrm{Tr}(g|_V)$$

where $\mathrm{Tr}(g|_V)$ is the trace of g on V.

Corollary 5.2. Let $g, h \in G$, if V is a representation of G, then $\chi_V(hgh^{-1}) = \chi_V(g)$, $\chi_V(1) = \mathrm{dim}V$.

Definition 5.3 (Character Table)**.** The irreducible complex characters of a finite group form a **character table**. For a representation $G \to GL(V)$, the character table is: the top row lists the conjugacy classes of G, the left column lists the irreducible representations of V, the rest entries are the character values each corresponding the conjugacy class and irreducible representation.

Example 5.4 (Character table of S_3)**.**

S_3	1	(12)	(123)
trivial	1	1	1
alternating	1	-1	1
standard	2	0	-1

Murnaghan–Nakayama Rule. The **Murnaghan–Nakayama rule** gives an efficient inductive method for computing character values of a symmetric group.

Since partition can determine the conjugacy class and irreducible representation of S_n, we let partition λ specify the conjugacy class, and ρ the irreducible representation, χ^λ_ρ the corresponding character value.

Then we have non-recursive version method:

$$\chi^\lambda_\rho = \sum_{T \in BST(\lambda,\rho)} (-1)^{ht(T)}$$

And recursive version:

$$\chi^\lambda_\rho = \sum_{\xi \in BS(\lambda,\rho_1)} (-1)^{ht(\xi)} \chi^{\lambda\backslash\xi}_{\rho\backslash\rho_1}, \qquad \text{stop condition: } \chi^{()}_{()} = 1$$

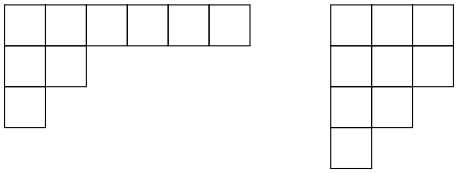
Algorithm 1 Non-Recursive Version

```
function CALCULATE(λ, ρ)
    Sum ← 0
    for Tableau ∈ BST(λ, ρ) do
        if ht(Tableau) is even then          ▷ equivalent to (−1)^{ht(·)} = 1
            Sum ← Sum + 1
        else
            Sum ← Sum − 1
        end if
    end for
    return Sum
end function
```

Algorithm 2 Recursive Version

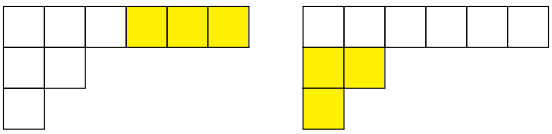
```
function CALCULATE(λ, ρ)
    if λ = () & ρ = () then
        return 1
    end if
    CacheValue ← CACHE(λ, ρ)        ▷ use cache to avoid heavy computation
    if CacheValue ≠ null then
        return CacheValue
    end if
    Length ← ρ_1
    SubWeightPartition ← ρ_{i>1}
    Sum ← 0
    for BorderStrip ∈ BS(λ, Length) do
        RestTableau ← λ\BorderStrip
        if ht(BorderStrip) is even then
            Sum ← Sum+CALCULATE(RestTableau, SubWeightPartition)
        else
            Sum ← Sum−CALCULATE(RestTableau, SubWeightPartition)
        end if
    end for
    CACHE(λ, ρ) ← Sum
    return Sum
end function
```

Example 5.5. We give a recursive version example here for $\lambda = (6, 2, 1)$, $\rho = (3, 3, 2, 1)$.



$\lambda = (6, 2, 1)$ $\rho = (3, 3, 2, 1)$

First, take ρ_1, which is 3, find all the possible border strip with length 3:



Then we have

$$\chi^{(6,2,1)}_{(3,3,2,1)} = (-1)^0\chi^{(3,2,1)}_{(3,2,1)} + (-1)^1\chi^{(5)}_{(3,2,1)}$$

Repeat this process, finally we have

$$\begin{aligned}
\chi^{(3,2,1)}_{(3,2,1)} &= (-1)^1\chi^{(2,1)}_{(1,1,1)} + (-1)^1\chi^{(2,1)}_{(3)} \\
&= (-1)^1[(-1)^1\chi^{(1)}_{(1)}] + (-1)^1[(-1)^0\chi^{(1)}_{(1)}] \\
&= (-1)(-1) + (-1)(1) \\
&= 0 \\
\chi^{(5)}_{(3,2,1)} &= (-1)^0\chi^{(2,1)}_{(3)} \\
&= (-1)^0[(-1)^0\chi^{(1)}_{(1)}] \\
&= (1)(1)(1) \\
&= 1
\end{aligned}$$

Hence, $\chi^{(6,2,1)}_{(3,3,2,1)} = (-1)^0(0) + (-1)^1(1) = -1$.

6. Implementation

Golang is a compiled language with excellent performance. We implemented this algorithm in Golang, and uploaded our codes to Github repository [3].

Performance Improvement of Hash Map.

Since we can not use partition directly as the key of the **map** in Golang, and using **string** as the key is very inefficient even though we can easily turn a partition into a string. We then need to construct an injection or bijection from partition to integer number.

Definition 6.1 (Ordering of Partitions)**.** For the same size partitions λ, ρ, if the number of rows of λ is less than or equal to ρ, we say partition $\lambda > \rho$ if the first different row (denote the index by j) satisfy $\lambda_j > \rho_j$.

Then for partitions $\{\mu^{(i)}\}$ of n, we have a increasing sequence $\{\mu^{(i)}|\mu^{(i+1)} > \mu^{(i)}\}$ by the compare method above, we say i is the **index** of such sequence, where i starts from 0.

Example 6.2 (Case $n = 4$)**.**

$$(1,1,1,1) < (2,1,1) < (2,2) < (3,1) < (4)$$

The index of $(2,1,1)$ is 1.

A Bijection from Partition to Integer. This bijection is provided by Joel Gibson in his blog [1], we will construct it here more specifically.

Let $\{\lambda\}$ be the partitions of n, then we define $P(n,k)$ as the number of partitions of n whose first part is k ($\lambda_1 = k$), and $R(n,k)$ the number of partitions of n with all rows $\leq k$ ($\lambda_i \leq k$).

By the definition, we immediately have

$$\begin{aligned} &P(0,0) = 1 && \\ &P(n,0) = 0 && \text{if } n > 0 \\ &P(n,k) = 0 && \text{if } k > n \\ &P(n,k) = R(n-k,k) && \text{if } k \leq n \\ &R(n,k) = \sum_{i=0}^{k} P(n,i) && \text{if } k \geq 0 \end{aligned}$$

Obviously, $R(n,n)$ is the number of all partitions of n, $R(n,n-1)$ is the number of partitions whose first row is not n, $R(n,n-2)$ is the number of partitions whose first row is not $n, n-1$, $R(n,n-3)$ ....., repeat this process by decreasing of n, we can spilt these partitions into n subsets A_k with size $|A_k| = P(n,k)$. And for each A_k, we remove the first row, it becomes a sub-partitions, and we can repeat the method again and again to find the index of a specific partition of n.

Then we can define the **index function** I_n : Partition $\to \mathbb{N}$ of n:

$$I_n(\lambda) = \sum_{\substack{i=0,1,\cdots,k \\ m=n}} R(m-\lambda_i, \lambda_{i+1}-1)$$

where k is the number of rows of a given partition λ, and define $\lambda_0 = 0$.

Example 6.3 ($n = 6$)**.** Consider a case $n = 6$, we have ordering partitions of 6:

$$(1,1,1,1,1,1) \quad (2,1,1,1,1) \quad (2,2,1,1) \quad (2,2,2)$$

$$(3,1,1,1) \quad (3,2,1) \quad (3,3) \quad (4,1,1) \quad (4,2) \quad (5,1) \quad (6)$$

$P(6,2)$ is the number of partitions with first row is equal to 2:

$$|\{(2,1,1,1,1),(2,2,1,1),(2,2,2)\}| = 3$$

$R(6,2)$ is the number of partitions with all rows ≤ 2:

$$|\{(1,1,1,1,1,1),(2,1,1,1,1),(2,2,1,1),(2,2,2)\}| = 4$$

And $R(6,3) - R(6,2) = |(3,1,1,1),(3,2,1),(3,3)| = 3$, which show $R(n,k) - R(n,k-1)$ is the number of partitions with first row is k since

$$R(n,k) - R(n,k-1) = \sum_{i=0}^{k} P(n,i) - \sum_{i=0}^{k-1} P(n,i) = P(n,k)$$

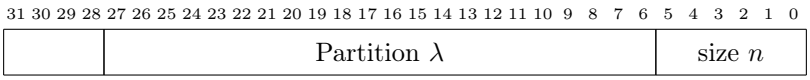
For the index function $I_6((2,2,2)) = R(6,1) + R(4,1) + R(2,1) = 1 + 1 + 1 = 3$.

Convert All Partitions of Different Size into Unique Integer. Since we already have a injection from partition of n to integer, we then need to combine the information of size n and such integer.

We only consider cases $n < 64 = 2^6$, which has up to $p(64) = R(64, 64) = 1741630 < 2^{22}$ partitions. That means we only need $6 + 22 = 28$ bits to uniquely represent a partition.

For $A = \{a | a \in \mathbb{N}, 0 \leq a \leq 2^{28}\}$ and partition λ with $|\lambda| < 64$, we then define a function $f : \lambda \mapsto a$ as $f(\lambda) = I_{|\lambda|}(\lambda) \times 64 + |\lambda|$.

The state stored in bytes of computer memory is



Because computers are very efficient at raising powers of 2, we can convert between partitions and integers very quickly.

Our Test Results. We test our algorithms on personal computer, and the results list below:

n	number of partitions	time spend (non-recursive version, ms)	time spend (non-recursive version, multiple threads, ms)	time spend (recursive version, ms)
5	7	0	0	0
10	42	1	0	1
15	176	17	4	21
20	627	151	46	270
25	1958	1524	434	2886
30	5604	12254	3294	24189
35	14883	92343	25044	216933

(Test Device: MacBook Pro 13, 2.4 GHz 4-Core Intel Core i5, 8 GB 2133 MHz LPDDR3)

Because of the property of recursive algorithm, it uses a cache to avoid a lot of repeated calculations. The bottleneck of the problem lies in the reading and writing of memory, we can not use multiple threads to speed up the computing process.

For the non-recursive version, we use multi-threaded and single-threaded to test, Obviously the multi-threaded will be much faster. Because there are 4 cores, it is almost 4 times faster.

References

[1] Joel Gibson. *Enumerating Partitions*. https://www.jgibson.id.au/articles/characters/#enumerating-partitions. 2021.

[2] Joe Harris William Fulton. *Representation Theory. A First Course*. Springer Science, 2004.

[3] Youxing Z. *Calculate Character Table of S_n*. `https://github.com/youxingz/symmetric_group_character`. Source Code. Version 1.0. 2022.