Representation Theory through Parking Functions

Alexander Wilson
York University

1. Parking Functions

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.

If every car parks, the tuple of preferences is a Parking function

(e.g. (2, 1, 2))

Write PF_n for the set of parking functions of n.

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.

If every car parks, the tuple of preferences is a <u>Parking function</u>

(e.g. (2, 1, 2))

Write PF_n for the set of parking functions of n.

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.

If every car parks, the tuple of preferences is a <u>Parking function</u>

(e.g. $(2, 1, 2)$)

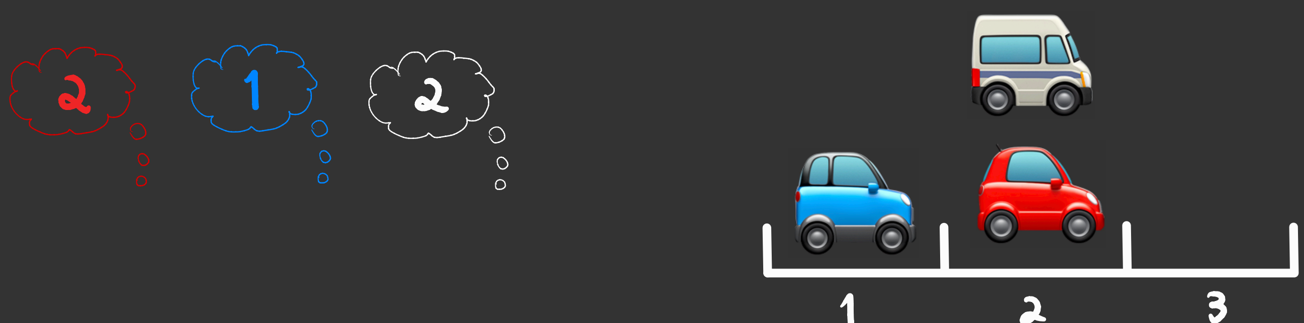
Write PF_n for the set of parking functions of n.

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.



If every car parks, the tuple of preferences is a Parking function

(e.g. (2, 1, 2))

Write PF_n for the set of parking functions of n.

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.

If every car parks, the tuple of preferences is a Parking function

(e.g. (2, 1, 2))

Write PF_n for the set of parking functions of n.

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.

$(2,3,2)$ is <u>not</u> a parking function

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.

(2,3,2) is <u>not</u> a parking function

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.

(2,3,2) is <u>not</u> a parking function

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.

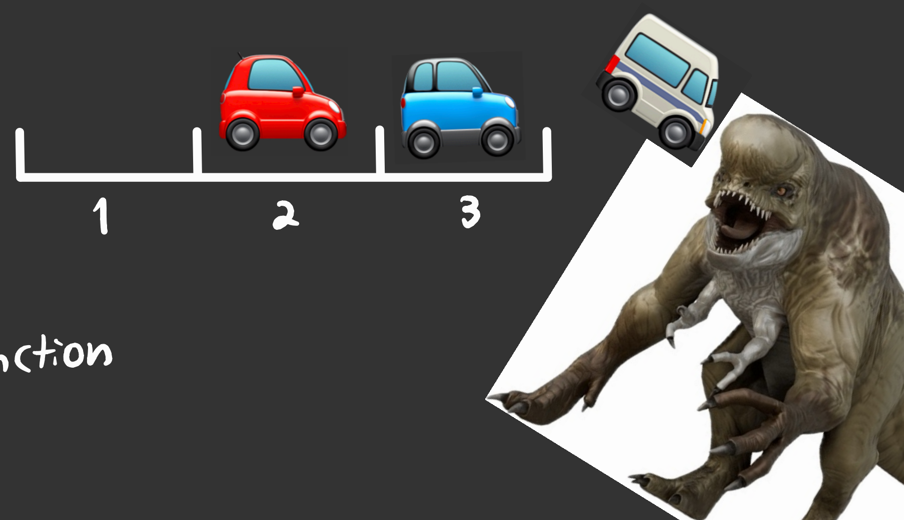
(2,3,2) is <u>not</u> a parking function

Parking Functions

a) Each car has a spot it prefers, and #cars = #spots

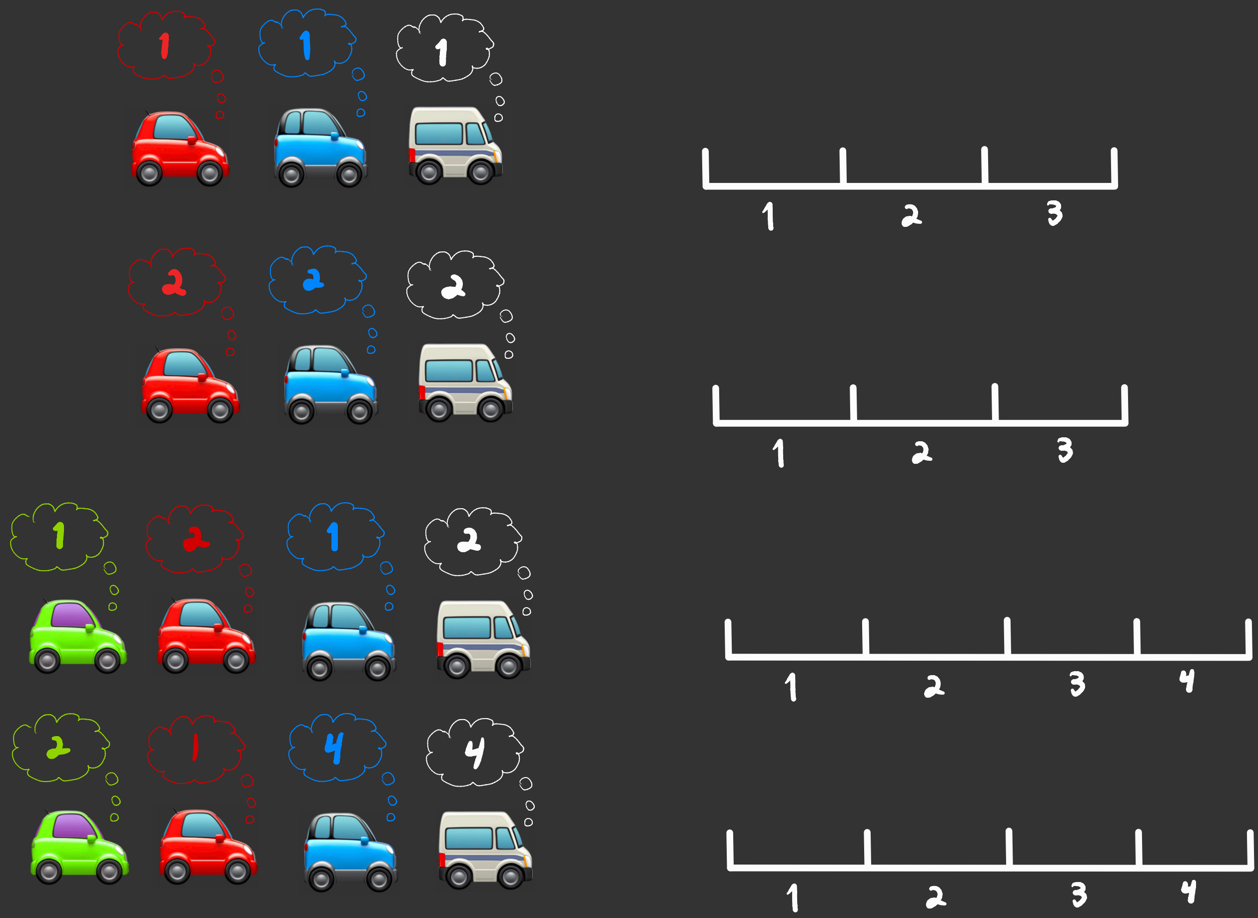
b) A car first drives to that preferred spot, taking it if it's open

c) If it can't park there, it keeps driving until the first available spot.



(2,3,2) is <u>not</u> a parking function

Which are Parking functions?



Failing to Park

- A preference tuple fails to park when too many cars prefer spots too far along. That is,

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in PF_n$$

exactly when

$$\underbrace{\#\{i \mid \alpha_i > k\}}_{\text{\# cars wanting to park past spot } k} \leq \underbrace{n-k}_{\text{\# spots available past spot } k} \qquad \text{for all } k = 1, 2, \ldots, n-1$$

☆ This has nothing to do with how the tuple is ordered!

Sorting Parking Functions

The observation above means rearranging the preference tuple won't change whether it is a parking function.

Ex) PF_3

111					
112	121	211			
113	131	311			
122	212	221			
123	132	213	231	312	321

PF_n breaks up into pieces based on which tuples rearrange to each other

Comparing Pieces

Ex) PF_7

⋮

1111234	1112134	...
1112233	1112323	...

⋮

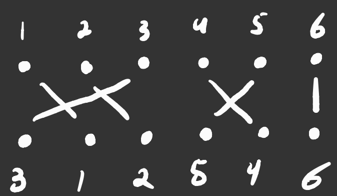
These two pieces both have 210 elements, but do they have the same structure?

This question will motivate the rest of what we do today.

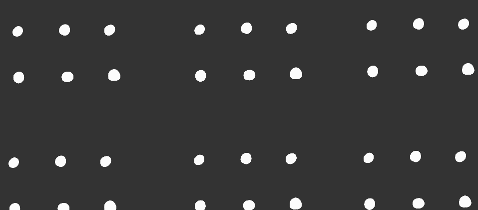
Symmetric group

- S_n consists of all permutations (rearrangements) of the numbers $1, 2, \ldots, n$.

- We'll draw these permutations like this:

$\sigma =$ 

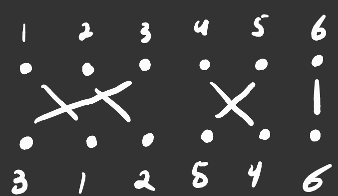
Ex) The six elements of S_3 are:



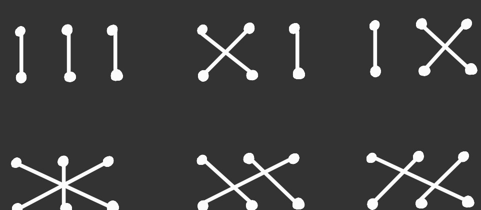
Symmetric group

- S_n consists of all permutations (rearrangements) of the numbers $1, 2, \ldots, n$.

- We'll draw these permutations like this:

$\sigma =$ 

Ex) The six elements of S_3 are:

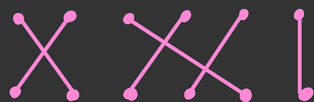


Action on Parking Functions

- Given a parking function $\alpha \in PF_n$ and a permutation $\sigma \in S_n$, we have σ "act on" α as follows:

$$\alpha = (1,\ 2,\ 1,\ 3,\ 3,\ 1)$$

$\sigma =$



$$\sigma . \alpha = (2,\ 1,\ 3,\ 3,\ 1,\ 1)$$

- The <u>orbit</u> of α under the S_n-action is

$$S_n . \alpha = \{\sigma . \alpha \mid \sigma \in S_n\}$$

Orbits of Parking Functions

Ex) PF_3

111	111	111	111	111	111	111
112	112	121	112	121	211	211
113	113	131	113	131	311	311
122	122	122	212	221	212	221
123	123	132	213	231	312	321

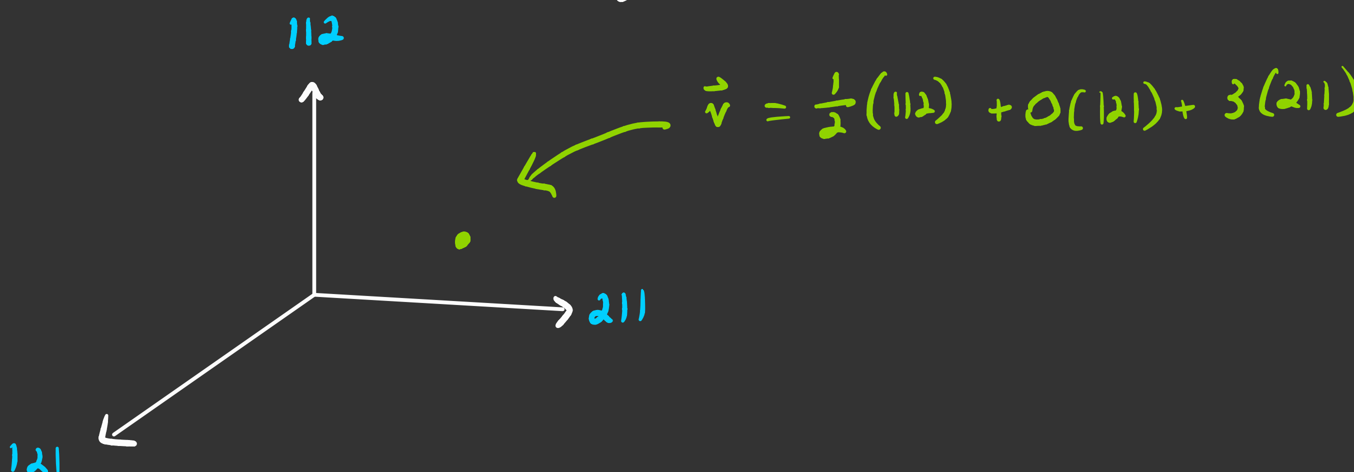
The way we split up PF_3 earlier is into S_3-orbits!

A Further Decomposition

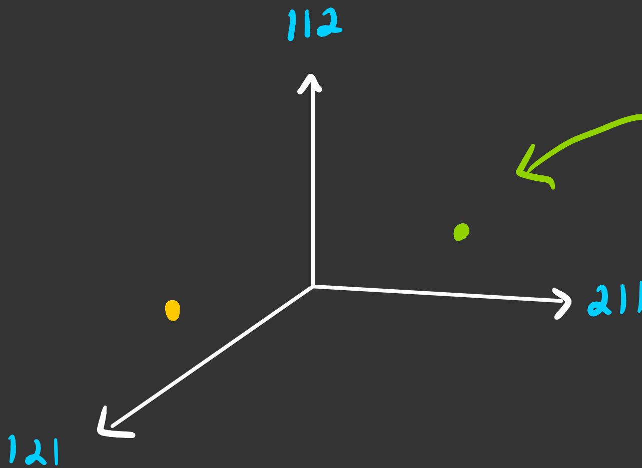
Let's Zoom in on one of these orbits:

$$S_3 . 112 = \{112, 121, 211\}$$

We're going to imagine a 3-dimensional space with a coordinate for each parking function in this orbit.



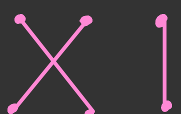
A Further Decomposition



$$\vec{v} = \frac{1}{2}(112) + 0(121) + 3(211)$$

$$\updownarrow$$

$$\left(\frac{1}{2}, 0, 3\right)$$

$\sigma =$ 

$\sigma . 112 = 112$

$\sigma . 121 = 211$

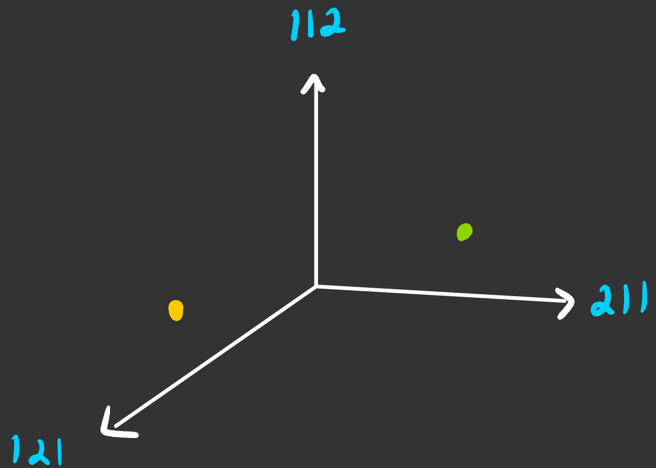
$\sigma . 211 = 121$

$$\sigma . \vec{v} = \frac{1}{2}(112) + 0(211) + 3(121)$$

$$\updownarrow$$

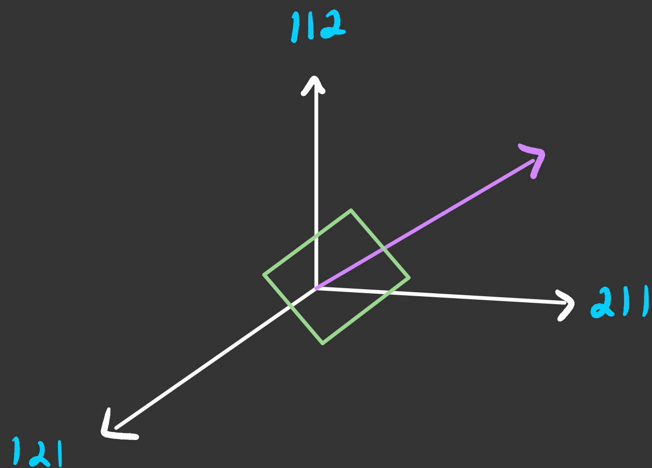
$$\left(\frac{1}{2}, 3, 0\right)$$

A Further Decomposition



This space is an example of a representation of S_3.

A Further Decomposition



$$V = \{a(112) + a(211) + a(121) \mid a \in \mathbb{R}\}$$

$$W = \{a(112) + b(211) + c(121) \mid a+b+c=0\}$$

These subspaces are <u>S_n-invariant</u>:

For any $\vec{v} \in V$, $\sigma . \vec{v} \in V$.

we say that this representation <u>decomposes</u> as

$V \oplus W$

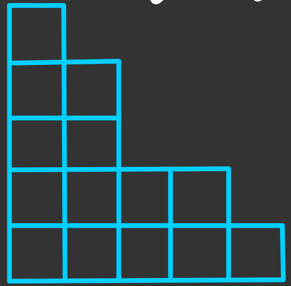
2. Representations of S_n

Tableaux

A **Partition** of a positive integer n is a weakly-decreasing sequence λ adding to n

$$(5,4,2,2,1) \vdash 14$$

The **diagram** of λ is a left-aligned array of boxes with row-lengths given by λ.

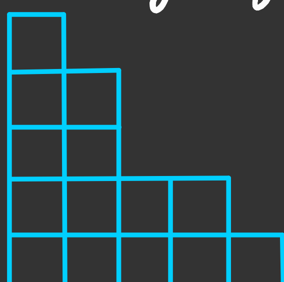


Tableaux

A **Partition** of a positive integer n is a weakly-decreasing sequence λ adding to n

$$(5,4,2,2,1) \vdash 14$$

The **diagram** of λ is a left-aligned array of boxes with row-lengths given by λ.



A **(standard Young) tableau** of shape λ is a filling of its diagram with the numbers $1, \ldots, n$ so that

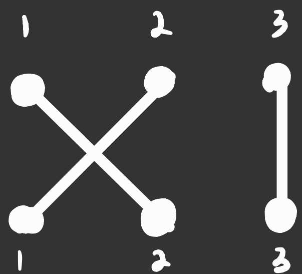
i) Rows increase left-to-right

ii) Columns increase bottom-to-top.

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

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

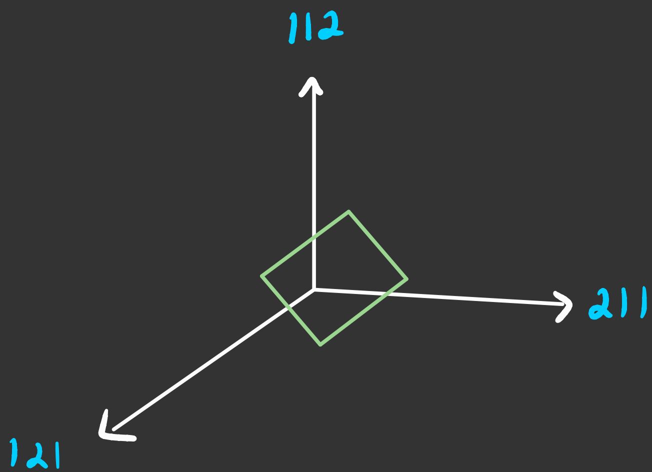
Acting on Tableaux



$$\begin{ytableau} 3 \\ 1 & 2 \end{ytableau} = \begin{ytableau} 3 \\ 2 & 1 \end{ytableau}$$

$$= \begin{ytableau} 3 \\ 1 & 2 \end{ytableau} - \begin{ytableau} 2 \\ 1 & 3 \end{ytableau}$$

"straightening algorithm"

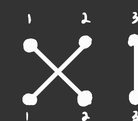


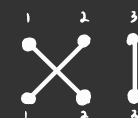
$$W = \{a(112) + b(211) + c(121) \mid a+b+c=0\}$$

is the same as acting on shape $\begin{array}{|c|c|}\cline{1-1} & \multicolumn{1}{c}{} \\ \hline & \\ \hline\end{array}$ tableaux

$$\begin{array}{|c|c|}\cline{1-1} 2 & \multicolumn{1}{c}{} \\ \hline 1 & 3 \\ \hline\end{array} \longleftrightarrow (211)-(121) \in W$$

$$\begin{array}{|c|c|}\cline{1-1} 3 & \multicolumn{1}{c}{} \\ \hline 1 & 2 \\ \hline\end{array} \longleftrightarrow (211)-(112) \in W$$

 $\cdot \begin{array}{|c|c|}\cline{1-1} 3 & \multicolumn{1}{c}{} \\ \hline 1 & 2 \\ \hline\end{array}$

$=$  $\cdot \, (211)-(112)$

$$= (121)-(112)$$

$$= (211)-(112)-\big((211)-(121)\big)$$

$$= \begin{array}{|c|c|}\cline{1-1} 3 & \multicolumn{1}{c}{} \\ \hline 1 & 2 \\ \hline\end{array} - \begin{array}{|c|c|}\cline{1-1} 2 & \multicolumn{1}{c}{} \\ \hline 1 & 3 \\ \hline\end{array}$$

Irreducible Representations

- Write S^λ for the representation of S_n obtained from the action of S_n on tableaux of shape λ.

- Every representation of S_n decomposes into pieces that look like S^λ for λ a partition of n.

Ex) PF_3

111						$\cong S^{(3)}$
112	121	211				
113	131	311				$\cong S^{(3)} \oplus S^{(2,1)}$
122	212	221				
123	132	213	231	312	321	$\cong S^{(3)} \oplus S^{(2,1)} \oplus S^{(2,1)} \oplus S^{(1,1,1)}$

Comparing Pieces

Ex) PF_7

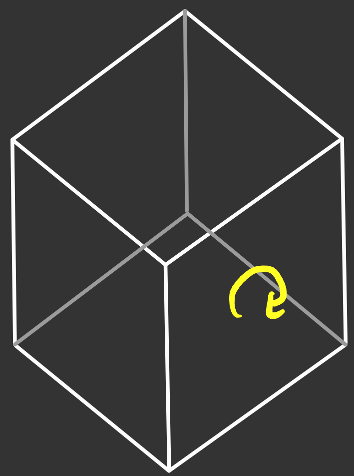
#times S^λ appears in...

λ	1111234 ...	1112233 ...
(3, 2, 2)	0	1
(3, 3, 1)	0	1
(4, 1, 1, 1)	1	0
(4, 2, 1)	2	2
(4, 3)	1	2
(5, 1, 1)	3	1
(5, 2)	3	3
(7)	1	1

3. Representation Theory

- Representation theory is about symmetry

Ex) Geometric objects



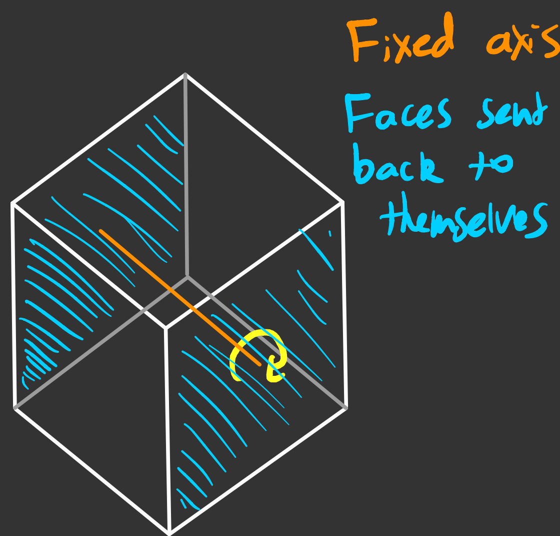
Ex) Combinatorial objects

1 1 2 1 3 3

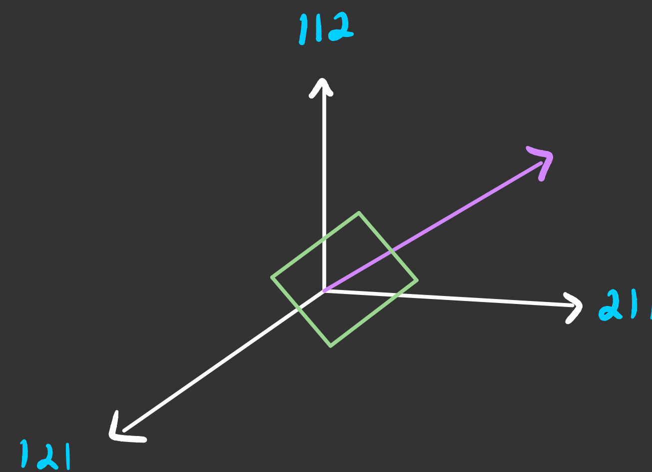
1 1 3 2 1 3

- We like to break symmetries down into smaller pieces

Ex) Geometric objects



Ex) Combinatorial objects



Thank you!