Answering Questions about parking functions

These are the counts for parking functions giving the resulting counts for a permutation

```
In [126]: print_by_conjugacy_class()
Partition = 1,1,1,1
24

Partition = 2,1,1
12 8 6 2
4 2

Partition = 3,1
8 4 3 2
8 6 3 2

Partition = 2,2
3 1 4

Partition: 4
6 6
4 3
2 2
```

```
In [127]: print_by_cosets_of_cyclic_group()
coset representative: 1,2,3,4.
e, (1234), (13)(24), (1432)
24 6 4 6

coset representative: 1,2,4,3
(34), (124), (1423), (132)
6 2 2 8

coset representative: 1,3,2,4
(23), (134), (1243), (142)
8 3 4 2

coset representative: 1,3,4,2
(234), (1324), (143), (12)
6 2 3 12

coset representative: 1,4,2,3
(243), (14), (123), (1342)
4 2 8 3

coset representative: 1,4,3,2
(24), (14)(23), (13), (12)(34)
2 1 4 3
```

One interesting observation is that the counts for the permutations that result from linear probing style circular parking lots (see 6/7) for the whole module are coming from the maximum count of parking functions for each coset of the diagonal subgroup. You can tell which element will have the maximum count since that element will send n to n (I'm not 100% sure that this is always true, but I think it is). This makes sense because then for the parking function, there are n options for where the last car can park. I think that studying circular parking as a proceedure is very related to studying parking functions which have $\pi_n = n$.

From here, figuring out the number of parking functions is an application of Stanley's exercise (pg 95, part d) [1].

One way of understanding the number of preference lists which yield a particular permutation for circular parking is to leverage the diagonal subgroup. Note that given a coset of the diagonal subgroup, every element of that coset will have the same distribution of displacements for each of the cars. That means that the function of interest will be constant on cosets of $S = \langle s \rangle$ where s is the shift operator in S_n. Note that exactly one element of every coset of σS will have the last car parked in the last spot (aka fix n). The other cars must be a parking function for $n-1$ cars in $n-1$ spots, with a resulting permutation $\sigma' \in S_{n-1}$. Then the total number of circular preference lists which give the permutation σ is n times the number of parking functions for $n-1$ cars which give σ'.

Double Cosets and Actions in Terms of Wreath Products

For m cars and n spots, the group that is convenient to describe all of the actions that we might care about is $C_n \wr S_m$. First, let us make precise what it means for an element of this group to act on a preference list $\pi \in P_{n,m}$ (n parking spots, m cars). Let us denote the preference of the ith car as $\pi(i)$

Given an element $(c, \sigma) \in C_n \wr S_m$ where $c \in C_n^m$ and $\sigma \in S_m$ and an element $\pi \in P_{n,m}$, the action of (c, σ) first permutes the order of the cars in the line, then adjusts the preferences of the cars. This can be written as follows:

$$(c, \sigma) \cdot \pi(x) = (c, e) \cdot \pi(\sigma^{-1}x) = c_{\sigma^{-1}x} \cdot \pi(\sigma^{-1}x)$$

Then, by restricting to particular subgroups of the wreath product, you can recover the actions of S_n, C_n, and C_n^m, as described on the whiteboard on 6/7.

Note that in this case, a preference list can be thought of as a left coset of S_m in $C_n \wr S_m$. One way that you can explicitly write down this correspondence is that the preference list π is sent to the coset of S_m containing (π, e).

Therefore, the representation theory of preference lists under the action of group elements from $G \leq C_n \wr S_m$ can be approached as

$$\mathrm{Res}_G^{C_n \wr S_m} \mathrm{Ind}_{S_m}^{C_n \wr S_m} 1$$

Double cosets in the wreath product are a very nice tool for describing some of the properties we have encountered on different functions when they are pulled back to the wreath product context. For example:

- Functions on preference lists where order of the cars doesn't matter are constant on double cosets of the form $S_m x S_m$
- Functions on preference lists which are constant when the diagonal subgroup acts are constant on double cosets of the form DxS_m
- Functions on preference lists which are both order invariant and which are constant under the action of the diagonal subgroup are constant on double cosets of the form $S_m D x S_m$

This also gives another way of rephrasing what $N_d(\lambda)$ counts. $N_d(\lambda)$ counts the double cosets $S_m x S_m$ where x has type λ and defect d

Some Useful/Frequently Used Maps

Connecting parking functions to circular preference lists (ie linear probing)

Let PK_n be the space of parking functions with n spots and n cars. Let PC_{n+1} be the space of circular preference lists with n spots and n cars. Let D be the diagonal subgroup in PC_{n+1}. Let the map $\varphi : PK_n \to PC_{n+1}/D$ be the bijection between these two spaces. If φ is a homomorphism, how does $C_n \wr S_n$ act on PC_{n+1}/D? Can this be identified with a subgroup of $C_{n+1} \wr S_{n+1}$ acting on PC_{n+1}/D?

Connecting circular preference lists to permutations

Let PC_n be the space of circular preference lists with n spots and n cars. Let the map $f : PC_n \to S_n$ be the map from parking functions to their corresponding filling permutations describing which car parked in the ith spot (so the preference list 231 would be sent to the permutation written in one line notation 312). Let $p : PC_n \to S_n$ be the map from parking functions to their corresponding parking permutations describing where the ith car in line parked. What is the largest subgroup of $C_n \wr S_m$ for which this is a module homomorphism?

Decomposing Circular Parking functions

Given a function on a preference list, that function can be thought of as a function on C_m^n. Given a function on, C_m^n this can be decomposed into the character basis for C_m^n using the Fourier transform. Note that the characters for C_m^n can be indexed by elements of the group since the group is commutative. One particular way to achieve this is the following:

$$\chi_{a_1 a_2 \cdots a_n}(g) = \omega^{a_1 g_1 + a_2 g_2 + \cdots a_n g_n}$$

where $\omega = e^{2\pi i/m}$ and each $a_i \in [0, m)$.

This is basically an extension of the way in which Valyuzhenich indexed characters for $\mathbb{Z}_2^n$ from last semester [2].

Note that several functions of interest (lucky, circular displacement, etc) are constant on cosets of the diagonal subgroup, $D = \langle (1, \cdots, 1) \rangle$. This plays nicely with the representation theory. Remember from class, when we discussed the uncertainty principle, the bound $G \leq |supp f||supp \hat{f}|$ was achieved when f was an indicator function on a coset. My (not fully thought through thought) is that in general for abeilan groups, you could probably show that if H is a subgroup and 1_H has support $\{\chi_g | g \in H'\}$, then H' is a subgroup of G. Since G is abelian in this case, that immediately means that H' is a normal in G. So in general for abelian groups that if f is constant on cosets of a group H, then $|supp \hat{f}| = |G|/|H|$

Given the notation for indexing the characters, we can explicitly find these characters by considering what characters are constant on cosets of the diagonal subgroup. Note that if

$$a_1 + a_2 + \cdots + a_n = 0 (\text{mod } m)$$

then $\chi_{a_1 a_2 \cdots a_n}(g) = \chi_{a_1 a_2 \cdots a_n}(dg)$ for any $d \in D$.

There are m^{n-1} choices which satisfy this equation since you can choose anything you would like for the first $n-1$ elements. At that point, your choice is fixed for the last element. (for more counting shenanigans which were attempting to structure this count in terms of partitions, which was deeply silly but also interesting, see handwritten notes).

Therefore the set of characters which are constant on cosets of D forms an m^{n-1} dimensional space. Note that functions which are constant on cosets of D also forms an m^{n-1} dimensional space, so that's all the characters we need to sweep out the whole space.

Note from 6/21: In the Decomposing Circular parking functions section, m and n are swapped from their usual meanings :(

6/8

			defect	lucky	irred circ lucky	χ_{000}	χ_{100}	χ_{200}	χ_{010}	χ_{020}	χ_{001}	χ_{002}	χ_{012}	χ_{021}	Circular disp	max circdisp	circ 1off	circ 2off	permutation
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	3	2	1	1	e
1	1	2	0	1	1	1	1	1	w	w	w	w	1	1	2	1	2	0	e
1	1	3	0	2	2	1	1	1	w^2	w^2	w^2	w^2	1	1	1	1	1	0	e
1	2	1	0	2	2	1	1	w	1	w^2	w	w^2	w^2	1	2	2	0	1	e
1	2	2	0	2	2	1	1	w	w	w^2	w^2	1	1	w	1	1	1	0	e
1	2	3	0	3	3	1	1	w	w^2	1	1	w	w	w^2	0	0	0	0	e
1	3	1	0	2	2	1	1	w^2	1	w	w^2	w	w^2	w	1	1	1	0	(23)
1	3	2	0	3	3	1	1	w^2	w	1	w	w^2	w	w^2	0	0	0	0	(23)
1	3	3	1	2	2	1	1	w^2	w^2	w	w^2	w	w	w	2	2	0	1	(23)
2	1	1	0	2	2	1	w	1	1	w	w^2	w^2	w	1	2	2	0	1	(12)
2	1	2	0	2	2	1	w	1	w	w^2	1	w	w	w^2	1	1	1	0	(12)
2	1	3	0	3	3	1	w	1	w^2	1	w	w^2	w	1	0	0	0	0	(12)
2	2	1	0	2	2	1	w	w	1	w	w^2	w	1	w^2	1	1	1	0	(132)
2	2	2	1	1	1	1	w	w	w	w^2	1	w^2	1	1	3	2	1	1	(132)
2	2	3	1	1	1	1	w	w	w^2	1	w^2	1	1	1	2	1	2	0	(132)
2	3	1	0	3	3	1	w	w^2	1	w^2	w	w^2	w^2	w^2	0	0	0	0	(132)
2	3	2	1	2	2	1	w	w^2	w	w^2	1	w^2	w^2	w^2	2	2	0	1	(132)
2	3	3	1	2	2	1	w	w^2	w^2	w^2	w^2	w^2	w^2	w^2	1	1	1	0	(132)
3	1	1	0	2	2	1	w^2	1	1	w^2	w	w^2	w^2	w	1	1	1	0	(123)
3	1	2	0	3	3	1	w^2	1	w	w^2	1	w^2	w^2	1	0	0	0	0	(123)
3	1	3	1	2	2	1	w^2	1	w^2	w^2	w^2	w^2	w^2	w^2	2	2	0	1	(123)
3	2	1	0	3	3	1	w^2	w	1	w	w	w	w	w	0	0	0	0	(13)
3	2	2	1	2	2	1	w^2	w	w	w	w^2	w	w	w^2	2	2	0	1	(13)
3	2	3	1	2	2	1	w^2	w	w^2	w	1	w	w	1	1	1	1	0	(13)
3	3	1	1	2	1	1	w^2	w^2	1	1	w	1	w^2	w	2	1	2	0	(123)
3	3	2	1	2	2	1	w^2	w^2	w	1	w^2	w	w	w^2	1	1	1	0	(123)
3	3	3	2	1	1	1	w^2	w^2	w^2	1	w	w^2	w^2	w	3	2	1	1	(123)

note: lucky + 1off + 2off = $3\chi_{000}$

defect = $\frac{12}{27}\chi_{000} + \frac{1+1w+2w^2}{27}\chi_{100}$ ...

circular lucky = $2\chi_{000} + 0\chi_{100} + 0\chi_{200} + 0\chi_{010} + 0\chi_{020} + 0\chi_{001} + 0\chi_{002}$
$+ 0\chi_{110} + 0\chi_{101} + 0\chi_{011}$

From Prof O

1, 6, 8, 12, 14, 16, 20, 22, 27

$\chi_{000}, \chi_{012}, \chi_{021}, \chi_{102}, \chi_{111}, \chi_{120}, \chi_{201}, \chi_{210}, \chi_{222}$ (χ_{120}: -9; χ_{210}: -9)

→ are these characters always those that add up to 0 mod 3?

36 -9 -9 -9 -9 -9 -9

$\chi_{000}, \chi_{012}, \chi_{021}, \chi_{102}, \chi_{111}, \chi_{120}, \chi_{201}, \chi_{210}, \chi_{222}$

For 3 elements

0 0	$\lambda = 000$	X 1 order	] 1, 1
1 3	$\lambda = 210$	X 6 orders	] 1 2
	$\lambda = 111$	X 1 order	
2 6	$\lambda = 222$	X 1 order	] 1 1

For 4 elements

0 0	χ_{0000}		] ⇒ 1 1
1 4	$\lambda = 3100$	X 12 orders	] ⇒ 3 1 4
	$\lambda = 2200$	X 6 orders	
	$\lambda = 2110$	X 12 orders	
	$\lambda = 1111$	X 1 order	
2 8	$\lambda = 3320$	X 12 orders	] ⇒ 3 1 4
	$\lambda = 3311$	X 6 orders	
	$\lambda = 3221$	X 12 orders	
	$\lambda = 2222$	X 1 order	
3 12	$\lambda = 3333$	X 1 order	] ⇒ 1 1

The symmetry always holds since the partition options for adding up to $a \cdot n$ are the same as those for $(n-1-a) \cdot n$ since there is a correspondance from

adds to $a \cdot n$: $\lambda = \lambda_1 \dots \lambda_n$ with
adds to $(n-1-a)n$: $\lambda = (n-1-\lambda_1) \dots (n-1-\lambda_n)$

For $n=5$ elements

0 0	$\lambda = 00000$	X 1	] ⇒ 1 1
1 5	$\lambda = 41000$	X 20	] ⇒ 12 1 6
	$\lambda = 32000$	X 20	
	$\lambda = 31100$	X 30	
	$\lambda = 22100$	X 30	
	$\lambda = 21110$	X 20	
	$\lambda = 11111$	X 1	
2 10	$\lambda = 44200$	X 30	] ⇒ 38 1 12
	$\lambda = 44110$	X 30	
	$\lambda = 43300$	X 30	
	$\lambda = 43210$	X 120	
	$\lambda = 43111$	X 20	
	$\lambda = 42220$	X 20	
	$\lambda = 42211$	X 30	
	$\lambda = 33310$	X 20	
	$\lambda = 33220$	X 30	
	$\lambda = 33211$	X 30	
	$\lambda = 32221$	X 20	
	$\lambda = 22222$	X 1	

So Far:

	# irred. to decompose in C_m^n constant on $D \times S_m$	# distinct values of coef. constant on $S_n D \times S_m$
	1	1
	1 1	2
OEIS A108267 →	1 7 1	9
	1 31 31 1	64
	1 121 381 121 1	625

A128562

1	1
1 1	2
1 2 1	4
1 4 4 1	10
1 6 12 6 1	26
1, 10, 29	80
1, 14, 61, 94	246
1, 21, 120, 263	810
1, 29, 222, 645, 910	2704

OEIS A003239 related to rooted trees

Partition calculator: int: k·n, size: 2·n, only ints: 1→n+1
dcode.fr/partition-generator

Better way to count all χ_g with the total sum of $g \equiv 0 \pmod n$
↳ pick any of the first $n-1$ digits
↳ Fix the last one to make the count correct

total #: $(n)^{n-1}$

Connecting n-dimensional parking functions to $n+1$ dimensional circular parking functions. & wreath products. - PK_n - PC_{n+1}

If you have $f \in \mathbb{C}PK_n$
corresponds to $f \in \mathbb{C}PC_{n+1}$, constant on Dx and order doesn't matter? $S_n Dx$
corresponds to $f \in \mathbb{C}(C_{n+1} \text{ wr } S_{n+1})$ constant on $D \times S_{n+1}$ $S_n D \times S_{n+1}$

Code to compute these coefficients

I wrote code because I realized how silly it was that I didn't have a nice way to do that earier, and that needed to be amended. Also, I had more functions that I wanted to decompose.

First some computations and verifications based on calculations by hand

```
In [65]: max_disp = decompose(max_displacement, 3)
000  :  (30+0j)
012  :  (7.5-2.598076211353316j)
021  :  (7.5+2.598076211353316j)
021  :  (-4.499999999999999+2.598076211353316j)
102  :  (-4.500000000000001-2.598076211353316j)
111  :  (4.5+2.5980762113533173j)
120  :  (-9-2.220446049250313e-16j)
201  :  (-4.500000000000001+2.598076211353316j)
210  :  (-9+2.220446049250313e-16j)
222  :  (4.5-2.5980762113533173j)

In [67]: d1 = decompose(disp1, 3)
000  :  (18+0j)
012  :  5.196152422706631j
021  :  -5.196152422706631j
102  :  5.196152422706631j
111  :  (-4.5+2.598076211353316j)
120  :  (9+0j)
201  :  -5.196152422706631j
210  :  (9+0j)
222  :  (-4.5-2.598076211353316j)

In [68]: d2 = decompose(disp2, 3)
000  :  (9+0j)
012  :  (4.5-2.598076211353316j)
021  :  (4.5+2.598076211353316j)
102  :  (4.5-2.598076211353316j)
111  :  (-1.1102230246251565e-16-5.196152422706632j)
201  :  (4.5+2.598076211353316j)
222  :  (-1.1102230246251565e-16+5.196152422706632j)

In [69]: d0 + d1 + d2
Out[69]:
array([81.+0.j,  0.+0.j,  0.+0.j,  0.+0.j, -0.+0.j,  0.-0.j,  0.+0.j,
       0.+0.j, -0.-0.j,  0.+0.j,  0.+0.j, -0.-0.j, -0.-0.j, -0.+0.j,
      -0.+0.j,  0.-0.j, -0.+0.j, -0.-0.j,  0.-0.j, -0.+0.j,  0.-0.j,
       0.+0.j, -0.+0.j, -0.-0.j, -0.+0.j, -0.-0.j, -0.-0.j])

In [70]: np.isclose(d1 + 2*d2, decompose(total_displacement, 3))
000  :  (36+0j)
012  :  (9+0j)
021  :  (9+0j)
102  :  (9-2.220446049250313e-16j)
111  :  (-4.500000000000001-7.794228634059947j)
120  :  (9-2.220446049250313e-16j)
201  :  (9+2.220446049250313e-16j)
210  :  (9+2.220446049250313e-16j)
222  :  (-4.500000000000001+7.794228634059947j)
Out[70]:
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True])
```

Some interesting computations of decomposition for 4 spots and cars

```
In [252]: by_value(decompose(get_disp(4,4)[0],4,4, False), 4,4)
23
(-64+0j)            : 3100, 2200, 1300,
(-48-16j)           : 1030, 0130,
(-48+16j)           : 3010, 0310,
(-32+0j)            : 2020, 0220,
(-24+0j)            : 2002, 0202, 0022,
(-20-20j)           : 1003, 0103, 0013,
(-20+20j)           : 3001, 0301, 0031,
(-12-20j)           : 1111,
(-12+20j)           : 3333,
(-8+0j)             : 3212, 2312, 3122, 1322, 2132, 1232,
(-4-4j)             : 2213, 2123, 1223,
(-4+4j)             : 3221, 2321, 2231,
(4-4j)              : 3311, 3131, 1331,
(4+4j)              : 3113, 1313, 1133,
(8-16j)             : 3302, 3032, 0332,
(8+0j)              : 2222,
(8+16j)             : 1102, 1012, 0112,
(12-4j)             : 3203, 2303, 3023, 0323, 2033, 0233,
(12+4j)             : 2101, 1201, 2011, 0211, 1021, 0121,
(16-16j)            : 3230, 2330,
(16+16j)            : 2110, 1210,
(32+0j)             : 1120, 3320,
(640+0j)            : 0000,
```

```
In [253]: by_value(decompose(get_disp(4,4)[1],4,4, False), 4,4)
24
(-48-32j)           : 3320,
(-48+32j)           : 1120,
(-20-20j)           : 3001, 0301, 0031,
(-20+20j)           : 1003, 0103, 0013,
(-16+0j)            : 2110, 1210, 3230, 2330,
(-8-16j)            : 1102, 1012, 0112,
(-8+0j)             : 2222,
(-8+16j)            : 3302, 3032, 0332,
(-4-12j)            : 3203, 2303, 3023, 0323, 2033, 0233,
(-4-4j)             : 3221, 2321, 2231,
(-4+4j)             : 2213, 2123, 1223,
(-4+12j)            : 2101, 1201, 2011, 0211, 1021, 0121,
(4-4j)              : 3113, 1313, 1133,
(4+4j)              : 3311, 3131, 1331,
(8+0j)              : 3212, 2312, 3122, 1322, 2132, 1232,
(16-32j)            : 3010, 0310,
(16+0j)             : 2020, 0220,
(16+32j)            : 1030, 0130,
(20-12j)            : 1111,
(20+12j)            : 3333,
(24+0j)             : 2002, 0202, 0022,
(48+0j)             : 3100, 1300,
(80+0j)             : 2200,
(208+0j)            : 0000,
```

```
In [254]: by_value(decompose(get_disp(4,4)[2],4,4, False), 4,4)
24
(-24+0j)            : 2002, 0202, 0022,
(-16+0j)            : 2200,
(-12-4j)            : 2101, 1201, 2011, 0211, 1021, 0121,
(-12+4j)            : 3203, 2303, 3023, 0323, 2033, 0233,
(-8+0j)             : 3212, 2312, 3122, 1322, 2132, 1232,
(-4-4j)             : 3113, 1313, 1133,
(-4+4j)             : 3311, 3131, 1331,
-16j                : 2110, 1210,
16j                 : 3230, 2330,
(4-4j)              : 3221, 2321, 2231,
(4+4j)              : 2213, 2123, 1223,
(8-16j)             : 3302, 3032, 0332,
(8+0j)              : 2222,
(8+16j)             : 1102, 1012, 0112,
(12-20j)            : 3333,
(12+20j)            : 1111,
(16-32j)            : 1120,
(16+0j)             : 3100, 1300, 2020, 0220,
(16+32j)            : 3320,
(20-20j)            : 3001, 0301, 0031,
(20+20j)            : 1003, 0103, 0013,
(32-16j)            : 1030, 0130,
(32+16j)            : 3010, 0310,
(112+0j)            : 0000,
```

```
In [257]: by_value(decompose(get_disp(4,4)[3],4,4, False), 4,4)
16
(-20-12j)           : 3333,
(-20+12j)           : 1111,
(-8-16j)            : 1102, 1012, 0112,
(-8+0j)             : 2222,
(-8+16j)            : 3302, 3032, 0332,
(-4-4j)             : 3311, 3131, 1331,
(-4+4j)             : 3113, 1313, 1133,
(4-12j)             : 2101, 1201, 2011, 0211, 1021, 0121,
(4-4j)              : 2213, 2123, 1223,
(4+4j)              : 3221, 2321, 2231,
(4+12j)             : 3203, 2303, 3023, 0323, 2033, 0233,
(8+0j)              : 3212, 2312, 3122, 1322, 2132, 1232,
(20-20j)            : 1003, 0103, 0013,
(20+20j)            : 3001, 0301, 0031,
(24+0j)             : 2002, 0202, 0022,
(64+0j)             : 0000,
```

```
In [258]: by_value(decompose(total_disp(4,4),4,4, False), 4,4)
9
(-16-64j)           : 3333,
(-16-32j)           : 2110, 1210, 1120, 2101, 1201, 2011, 0211, 1021, 0121, 1102, 1012, 0112,
(-16+0j)            : 3311, 3131, 1331, 2222, 3113, 1313, 1133,
(-16+32j)           : 3320, 3230, 2330, 3302, 3032, 0332, 3203, 2303, 3023, 0323, 2033, 0233,
(-16+64j)           : 1111,
(16+0j)             : 3221, 2321, 2231, 3212, 2312, 3122, 1322, 2132, 1232, 2213, 2123, 1223,
(48+0j)             : 2200, 2020, 0220, 2002, 0202, 0022,
(80+0j)             : 3100, 1300, 3010, 0310, 1030, 0130, 3001, 0301, 0031, 1003, 0103, 0013,
(624+0j)            : 0000,
```

Observations from when $n \neq m$. I think that when $m > n$, you get identical decompositions where the coefficients for most of the characters are just multiplied by a constant, and the coefficient for the trivial representation gets an additional boost

It also seems that there are patterns/relations between n, m and n, n. See the following:

```
In [265]: by_value(decompose(get_disp(4,3)[0],4,3, False), 4,3)
8
(-16+0j)            : 310, 220, 130,
(-12-4j)            : 103, 013,
(-12+4j)            : 301, 031,
(-8+0j)             : 202, 022,
(4-4j)              : 323, 233,
(4+4j)              : 211, 121,
(8+0j)              : 112, 332,
(144+0j)            : 000,

In [266]: by_value(decompose(get_disp(4,4)[0],4,4, False), 4,4)
23
(-64+0j)            : 3100, 2200, 1300,
(-48-16j)           : 1030, 0130,
(-48+16j)           : 3010, 0310,
(-32+0j)            : 2020, 0220,
(-24+0j)            : 2002, 0202, 0022,
(-20-20j)           : 1003, 0103, 0013,
(-20+20j)           : 3001, 0301, 0031,
(-12-20j)           : 1111,
(-12+20j)           : 3333,
(-8+0j)             : 3212, 2312, 3122, 1322, 2132, 1232,
(-4-4j)             : 2213, 2123, 1223,
(-4+4j)             : 3221, 2321, 2231,
(4-4j)              : 3311, 3131, 1331,
(4+4j)              : 3113, 1313, 1133,
(8-16j)             : 3302, 3032, 0332,
(8+0j)              : 2222,
(8+16j)             : 1102, 1012, 0112,
(12-4j)             : 3203, 2303, 3023, 0323, 2033, 0233,
(12+4j)             : 2101, 1201, 2011, 0211, 1021, 0121,
(16-16j)            : 3230, 2330,
(16+16j)            : 2110, 1210,
(32+0j)             : 1120, 3320,
(640+0j)            : 0000,
```

Next Steps (for Monday) - some of these are done

- Answer questions about maps - also think about if/how the correspondence between PK_n and PC_{n+1} works when $n \neq m$ (still to do)
- Write code to figure out distance from uniform distribution? (still to do)
- Write code to be able to decompose circular displacement, max circular displacement, lucky, and displacement k in terms of character basis for $\mathbb{Z}_m^n$ (Done!)
- Write up proof that functions which are constant on Dx can be decomposed using only characters with subscripts which add up to n (Done)
- Could you compute the coefficients in the character decomposition in general? (code dies at 6. Turns out that taking the tensor product of a 6 by 6 matrix with itself 6 times is not a good idea) Or is there a way of computing the character coefficients for total displacement - this one is even lower dimensional? (new item)
- Look at $N_d(\lambda)$ for 2 row partitions (still to do)
- Read! (made progress)

References

[1] Richard P. Stanley and Sergey Fomin. *Enumerative Combinatorics*, volume 2 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1999.

[2] Alexandr Valyuzhenich. Optimal functions with spectral constraints in hypercubes, 2023.