Planes, Hyperplanes, and Beyond: Understanding Higher-Dimensional Spaces

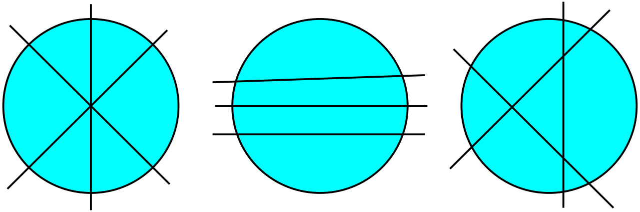
Jeremy L. Martin
Department of Mathematics
University of Kansas

The Frank S. Brenneman Lectures
Tabor College
March 28, 2017

Part 1: How Many Pieces of Cake?

The Cake-Cutting Problem

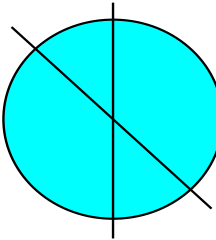
What is the greatest number of pieces that a cake can be cut into with a given number of cuts?



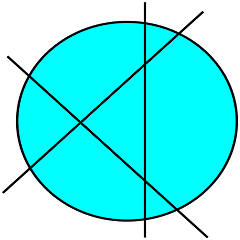
- The cuts must be straight lines and must go all the way through the cake.
- The sizes and shapes of the pieces don't matter.
- For the moment, we'll focus on 2-dimensional cakes (think of them as pancakes).

Solutions with 2, 3 or 4 Cuts

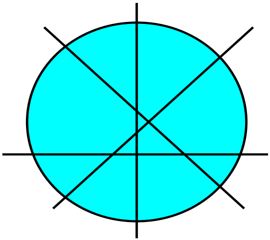
Let's write $P_2(N)$ for the maximum number of pieces obtainable using N cuts. (The 2 stands for dimension.)



2 cuts:
$P_2(2) = 4$



3 cuts:
$P_2(3) = 7$



4 cuts:
$P_2(4) = 11$

Solutions with N Cuts

Cuts N	Pieces $P_2(N)$
1	2
2	4
3	7
4	11
5	16
6	22

Cuts N	Pieces $P_2(N)$
7	29
8	37
9	46
10	56
...	...
100	5051

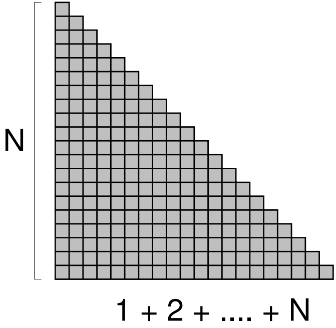
Do you see the pattern?

The Pattern

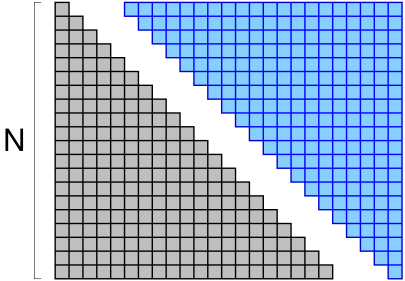
N	$P_2(N)$				
0	1				
1	2	$=$	$1+1$		
2	4	$=$	$2+2$	$=$	$1+1+2$
3	7	$=$	$4+3$	$=$	$1+1+2+3$
4	11	$=$	$7+4$	$=$	$1+1+2+3+4$
5	16	$=$	$11+5$	$=$	$1+1+2+3+4+5$
...	...		...		...

- How do we prove that the pattern works for every N?
- What does $1+2+\cdots+N$ equal anyway?

The Staircase Theorem

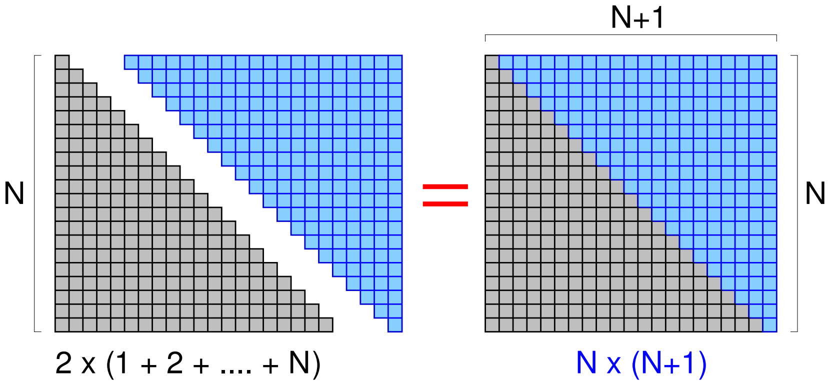


The Staircase Theorem

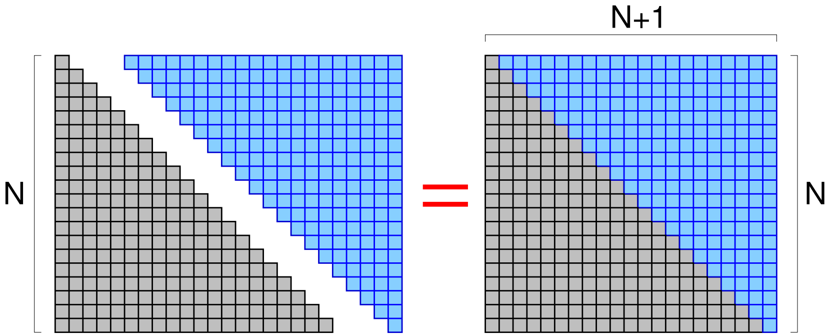


2 x (1 + 2 + ... + N)

The Staircase Theorem



The Staircase Theorem



2 x (1 + 2 + ... + N)

N x (N+1)

$$\boxed{1 + 2 + \dots + N = N(N+1)/2}$$

The Pattern

N	$P_2(N)$				
0	1				
1	2	$=$	$1+1$		
2	4	$=$	$2+2$	$=$	$1+1+2$
3	7	$=$	$4+3$	$=$	$1+1+2+3$
4	11	$=$	$7+4$	$=$	$1+1+2+3+4$
5	16	$=$	$11+5$	$=$	$1+1+2+3+4+5$
...	...		...		...

By the Staircase Theorem, we can conjecture that

$$P_2(N) \;=\; 1+(1+2+\cdots+N) \;=\; 1+\frac{N(N+1)}{2}.$$

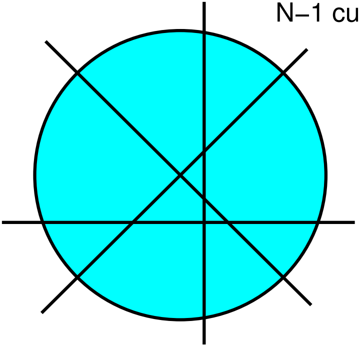
Maximizing the Number of Pieces

How can we ensure obtaining as many pieces as possible?

Maximizing the Number of Pieces

How can we ensure obtaining as many pieces as possible?

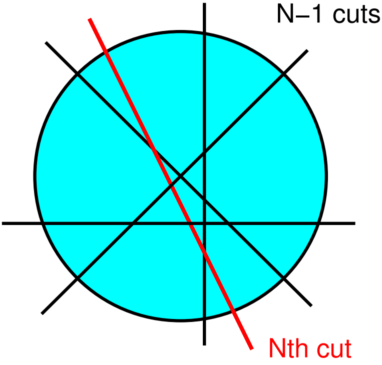
- First cut the pancake into $P_2(N-1)$ pieces using $N-1$ cuts.



Maximizing the Number of Pieces

How can we ensure obtaining as many pieces as possible?

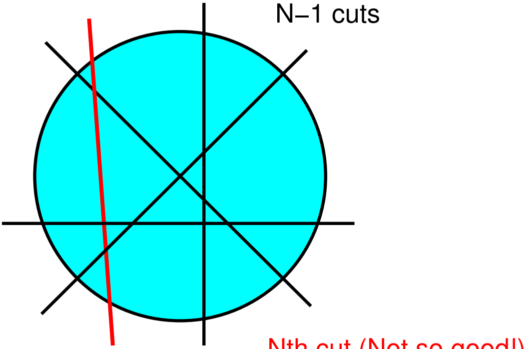
- First cut the pancake into $P_2(N-1)$ pieces using $N-1$ cuts.
- Now make the Nth cut, hitting as many pieces as possible.



Maximizing the Number of Pieces

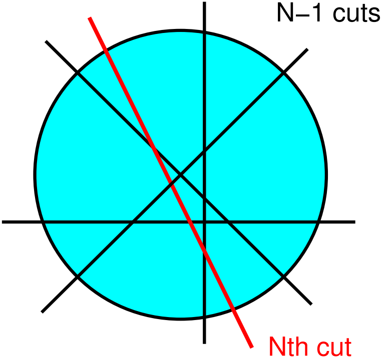
How can we ensure obtaining as many pieces as possible?

- First cut the pancake into $P_2(N-1)$ pieces using $N-1$ cuts.
- Now make the Nth cut, hitting as many pieces as possible.



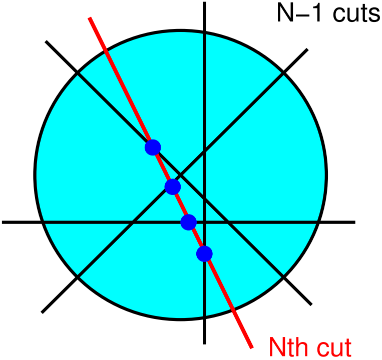
Maximizing the Number of Pieces

Key observation: The number of pieces subdivided by the Nth cut equals one more than the number of previous cuts it meets.



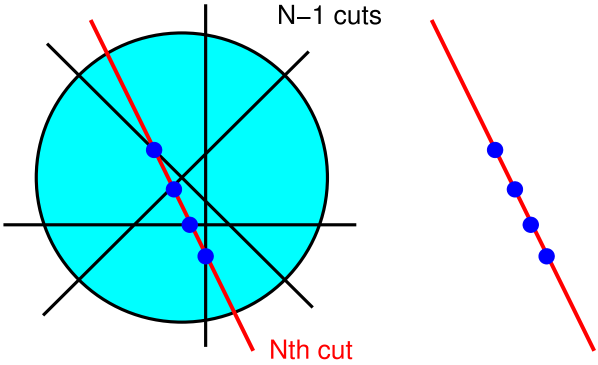
Maximizing the Number of Pieces

Key observation: The number of pieces subdivided by the Nth cut equals one more than the number of previous cuts it meets.



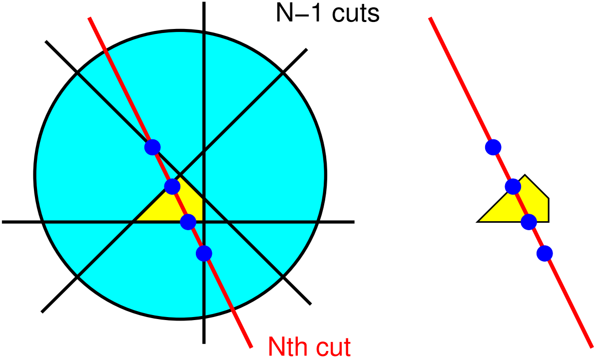
Maximizing the Number of Pieces

Key observation: The number of pieces subdivided by the *N*th cut equals one more than the number of previous cuts it meets.



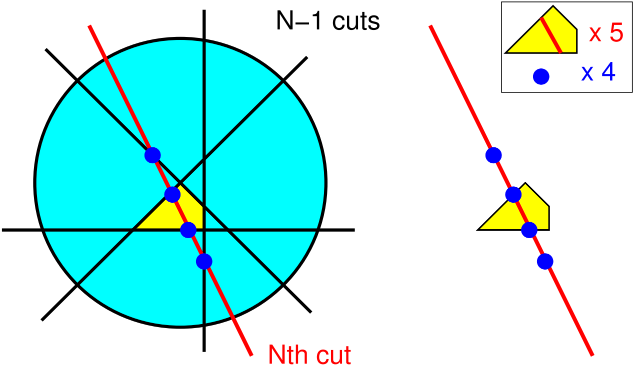
Maximizing the Number of Pieces

Key observation: The number of pieces subdivided by the Nth cut equals one more than the number of previous cuts it meets.



Maximizing the Number of Pieces

Key observation: The number of pieces subdivided by the Nth cut equals one more than the number of previous cuts it meets.



Maximizing the Number of Pieces

If we make sure that

- every pair of cuts meets in some point, and
- no more than two cuts meet at any point,

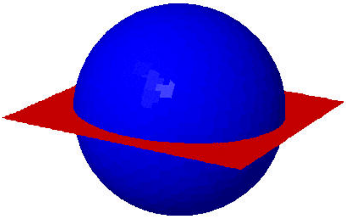
then the N^{th} cut will meet each of the previous $N-1$ cuts, and therefore will make N new pieces.

Since the original pancake had one piece, we have proved that

$$\boxed{\mathbf{P_2(N) \;=\; 1+(1+2+\cdots+N) \;=\; 1+\frac{N(N+1)}{2}.}}$$

From 2D to 3D

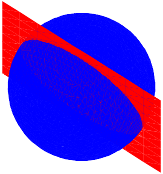
What about 3-dimensional cakes?



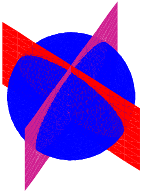
A cut in 3-dimensional space means a plane, not a line.

From 2D to 3D

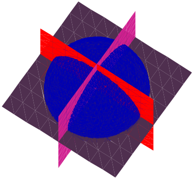
Let's write $P_3(N)$ for the maximum number of pieces obtainable from a 3-dimensional cake with N cuts.



$P_3(1) = 2$



$P_3(2) = 4$

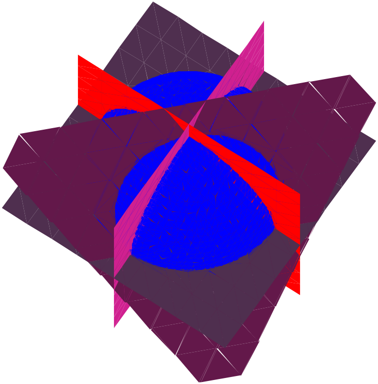


$P_3(3) = 8$

Compare 2D: $P(1) = 2$, $P(2) = 4$, $P(3) = 7$.

$P_3(4) = 15$

With four planes, we can make 15 pieces (though only 14 are visible from the outside).



From 2D to 3D

N	0	1	2	3	4	5	6	7	8
$P_2(N)$	1	2	4	7	11	16	22	29	37
$P_3(N)$	1	2	4	8	15	26	42	64	93

Do you see the pattern?

From 2D to 3D

N	0	1	2	3	4	5	6	7	8
$P_2(N)$	1	2	4	7	**11**	16	22	29	37
$P_3(N)$	1	2	4	8	**15**	**26**	42	64	93

The pattern is

$$\boxed{P_3(N) = P_3(N-1) + P_2(N-1).}$$

From 2D to 3D

N	0	1	2	3	4	5	6	7	8
$P_2(N)$	1	2	4	7	**11**	16	22	29	37
$P_3(N)$	1	2	4	8	**15**	**26**	42	64	93

The pattern is

$$\boxed{P_3(N) = P_3(N-1) + P_2(N-1).}$$

(In fact $P_3(N) = \frac{N^3+5N+6}{6}$ — but the pattern is more important than this formula!)

Pancakes, Cakes and Hypercakes

How about four-dimensional pancakes?

Pancakes, Cakes and Hypercakes

How about four-dimensional pancakes?

(Never mind whether they actually exist!)

Pancakes, Cakes and Hypercakes

How about four-dimensional pancakes?

(Never mind whether they actually exist!)

In general, if you have a d-dimensional cake and you can make N cuts, how many pieces can you make? (Call this number $P_d(N)$.)

- We already know the answers for $d = 2$ and $d = 3$.
- For $d = 1$: N cuts give $N + 1$ pieces.
- For any d: 0 cuts give 1 piece, 1 cut gives 2 pieces.

Pancakes, Cakes and Beyond

- Each number is the sum of the numbers immediately "west" ($\leftarrow$) and "northwest" ($\nwarrow$).
- Formula: $P_d N) = P_d(N-1) + P_{d-1}(N-1)$.

	N								
	0	1	2	3	4	5	6	7	8
$P_1(N)$	1	2	3	4	5	6	7	8	9
$P_2(N)$	1	2	4	7	11	16	22	29	37
$P_3(N)$	1	2	4	8	15	26	**42**	64	93
$P_4(N)$	1	2	4	8	16	31	**57**	**99**	163
$P_5(N)$	1	2	4	8	16	32	63	120	219
...	...								

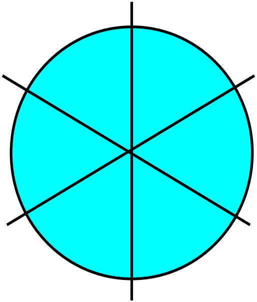
Pancakes, Cakes and Beyond

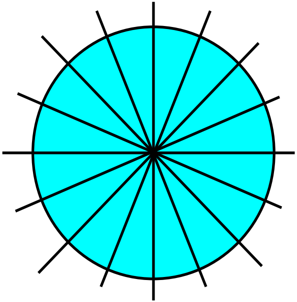
Theme: Understanding patterns in dimensions we can see enables us to understand dimensions we can't see.

Part 2: Symmetric Cake-Cutting

Symmetric Cake-Cutting

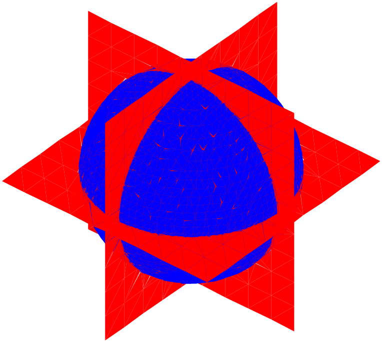
What are the possible ways to cut a perfectly round cake so that all pieces are congruent (i.e., geometrically the same)?

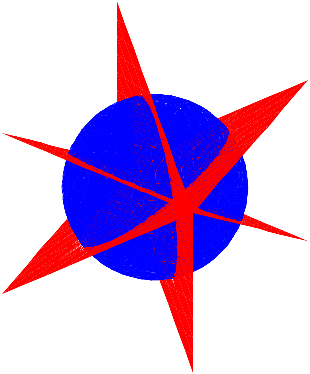




Symmetric Cake-Cutting

What are the possible ways to cut a perfectly round cake so that all pieces are congruent (i.e., geometrically the same)?





Refresher: N-Dimensional Algebra

Lines in 2-dimensional space have equations like

$$x = y \qquad x = 0 \qquad x + 2y = 4.$$

Refresher: N-Dimensional Algebra

Lines in 2-dimensional space have equations like

$$x = y \qquad x = 0 \qquad x + 2y = 4.$$

Planes in 3-dimensional space have equations like

$$x = y \qquad x = z \qquad x = 0 \qquad x + 3y + 2z = 1.$$

Refresher: N-Dimensional Algebra

Lines in 2-dimensional space have equations like

$$x = y \qquad x = 0 \qquad x + 2y = 4.$$

Planes in 3-dimensional space have equations like

$$x = y \qquad x = z \qquad x = 0 \qquad x + 3y + 2z = 1.$$

Hyperplanes in 4-dimensional space have equations like

$$x + y = z \qquad w = 0 \qquad 3w - 2x + 7y + 2z = 2012.$$

Refresher: N-Dimensional Algebra

Lines in 2-dimensional space have equations like

$$x = y \qquad x = 0 \qquad x + 2y = 4.$$

Planes in 3-dimensional space have equations like

$$x = y \qquad x = z \qquad x = 0 \qquad x + 3y + 2z = 1.$$

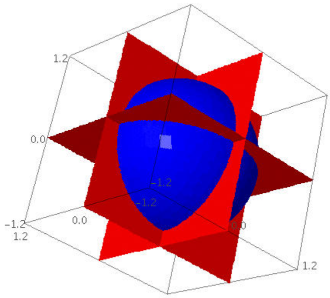
Hyperplanes in 4-dimensional space have equations like

$$x + y = z \qquad w = 0 \qquad 3w - 2x + 7y + 2z = 2012.$$

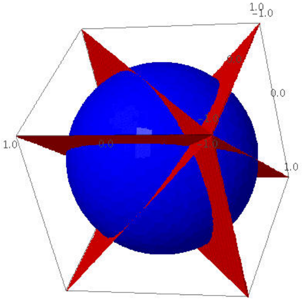
The two sides of a hyperplane are described by inequalities.
For example, the plane $x = z$ cuts 3D-space into two pieces:

$$x < z \quad \text{and} \quad z < x.$$

Symmetric Cake-Cutting



Three planes:
$x = 0, \ y = 0, \ z = 0$



Three planes:
$x = y, \ x = z, \ y = z$

Symmetric Cake-Cutting in Higher Dimensions

We can cut up a 3-dimensional sphere into congruent pieces using the planes defined by the equations

$$\boxed{x = 0,\ y = 0,\ z = 0} \qquad \text{or} \qquad \boxed{x = y,\ x = z,\ y = z}$$

to produce 8 or 6 regions respectively.

Symmetric Cake-Cutting in Higher Dimensions

We can cut up a 3-dimensional sphere into congruent pieces using the planes defined by the equations

$$\boxed{x = 0,\ y = 0,\ z = 0} \qquad \text{or} \qquad \boxed{x = y,\ x = z,\ y = z}$$

to produce 8 or 6 regions respectively.

Question: Suppose we cut up a 4-dimensional sphere into pieces using the hyperplanes

$$\boxed{\begin{array}{cc} w = 0 & x = 0 \\ y = 0 & z = 0 \end{array}} \qquad \text{or} \qquad \boxed{\begin{array}{ccc} w = x & w = y & w = z \\ x = y & x = z & y = z \end{array}}$$

How many regions will result?

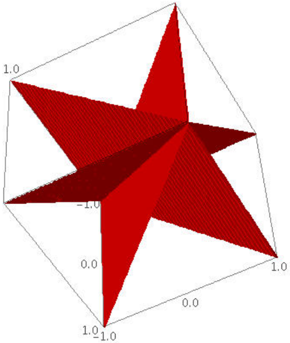
Symmetric Cake-Cutting in Higher Dimensions

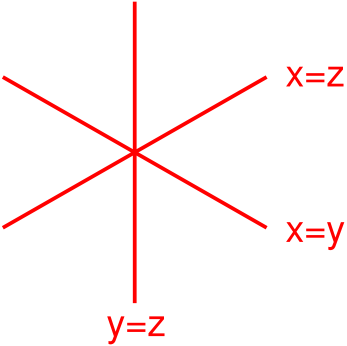
Some tools for visualizing 4-dimensional space:

- Work by analogy: understanding low-dimensional space can help us understand higher dimensions
- Project into lower dimension to make visualization easier
- Reexpress high-dimensional problems mathematically

The Braid Arrangement

The arrangement of planes $x = y$, $x = z$, $y = z$ is called the *3-dimensional braid arrangement* (**Braid3** for short).





Projecting from 3D to 2D makes the diagrams simpler, and preserves the geometry (and number!) of the regions.

Regions Between The Planes of **Braid3**

Each region of **Braid3** lies on one side of each of the planes $x = y$, $x = z$, $y = z$. Therefore,

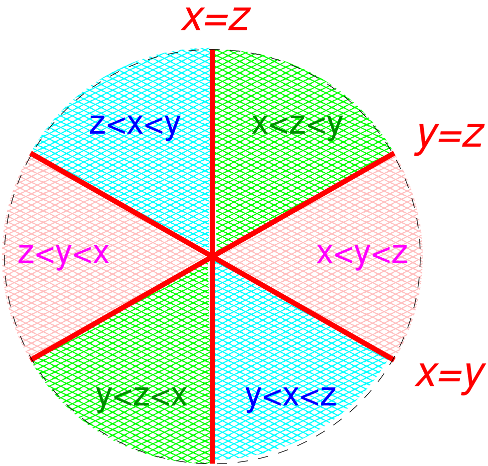
- either $x < y$ or $y < x$;
- either $x < z$ or $z < x$;
- either $y < z$ or $z < y$.

So every region can be specified by the **order** of x, y, z.

There are six possibilities:

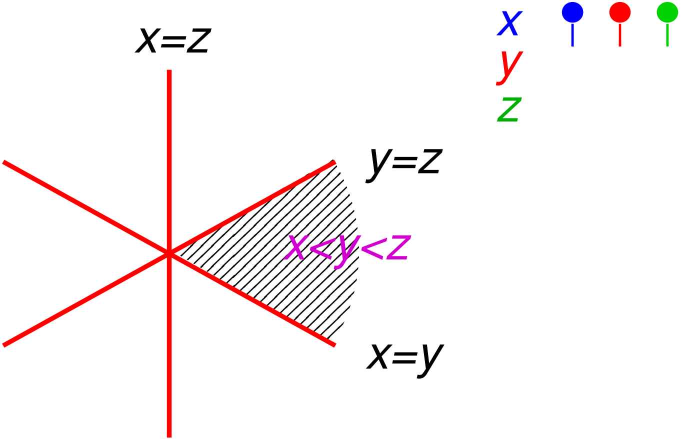
$$\begin{array}{ccc} x < y < z & y < x < z & z < x < y \\ x < z < y & y < z < x & z < y < x \end{array}$$

Regions of Braid3



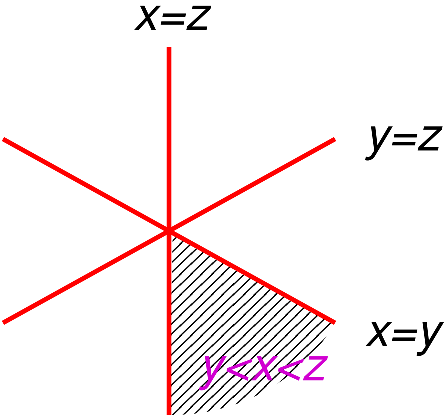
Why “Braid”?

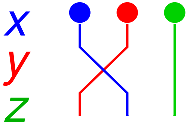
Crossing a border corresponds to reversing one inequality.



Why “Braid”?

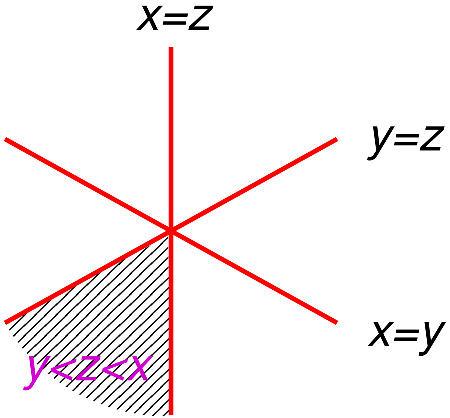
Crossing a border corresponds to reversing one inequality.

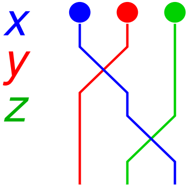




Why "Braid"?

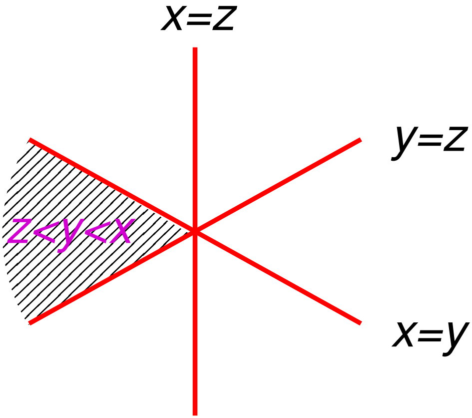
Crossing a border corresponds to reversing one inequality.

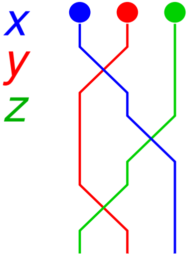




Why "Braid"?

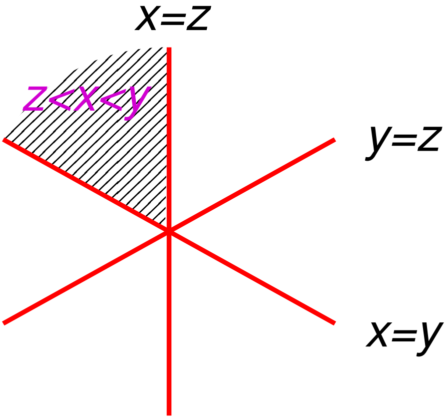
Crossing a border corresponds to reversing one inequality.

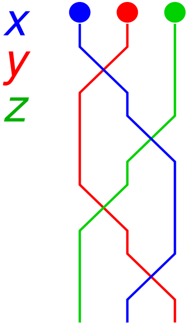




Why "Braid"?

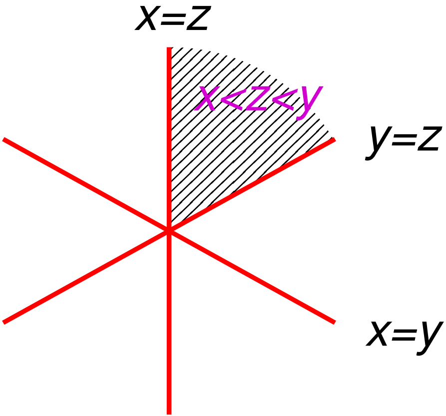
Crossing a border corresponds to reversing one inequality.

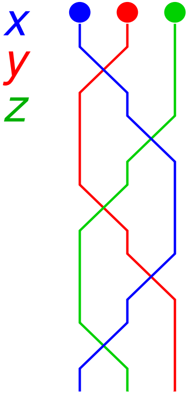




Why "Braid"?

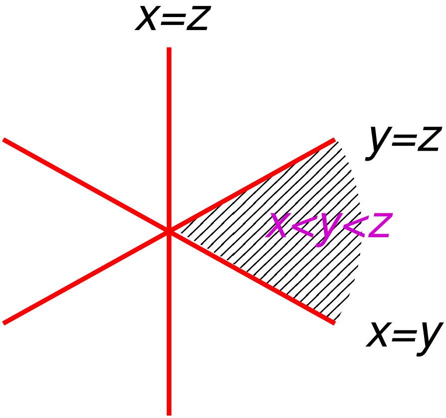
Crossing a border corresponds to reversing one inequality.

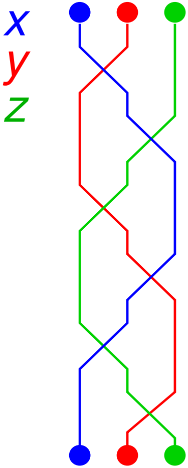




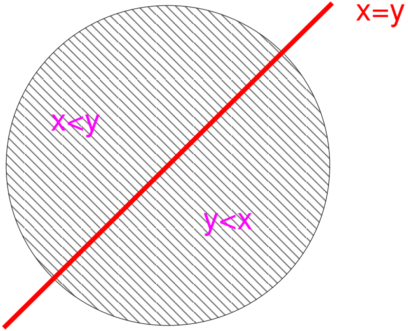
Why "Braid"?

Crossing a border corresponds to reversing one inequality.

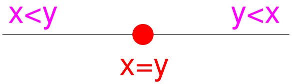




The 2-Dimensional Braid Arrangement



Braid2



Projection into 1D

Note that there are 2 regions.

The 4-Dimensional Braid Arrangement

The arrangement **Braid4** consists of the hyperplanes defined by the equations

$$w = x, \quad w = y, \quad w = z, \quad x = y, \quad x = z, \quad y = z$$

in four-dimensional space.

Key observation: We can project **Braid2** from 2D to 1D, and **Braid3** from 3D to 2D, so. . .

by analogy, we should be able to project **Braid4** from 4D to 3D.

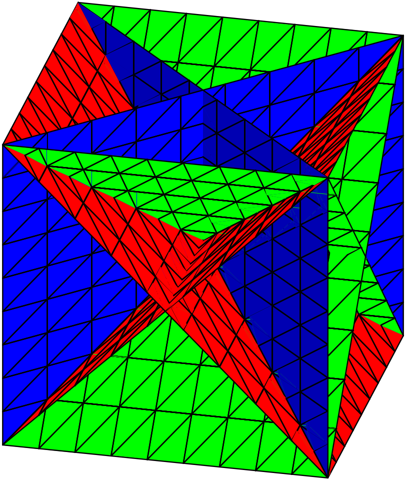
A Technical Interlude

- The six equations

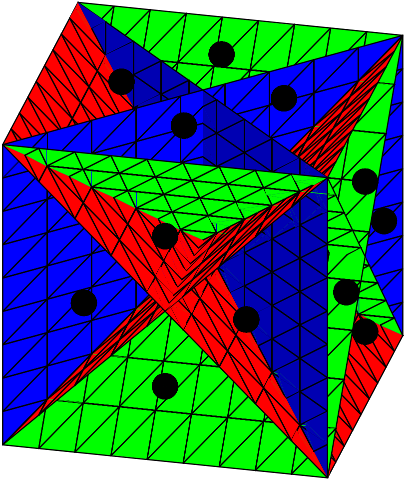
$$w = x, \quad w = y, \quad w = z, \quad x = y, \quad x = z, \quad y = z$$

  are all satisfied if $w = x = y = z$.
- That is, the six hyperplanes of **Braid4** intersect in a line L.
- As in the previous cases, we can "squash" (or project) 4D along L to reduce to 3D.
- The hyperplane perpendicular to L is defined by $w + x + y + z = 0$.
- To make the pictures that follow, I gave my computer the equations for **Braid4** and added the equation $w + x + y + z = 0$, i.e., $w = -x - y - z$.

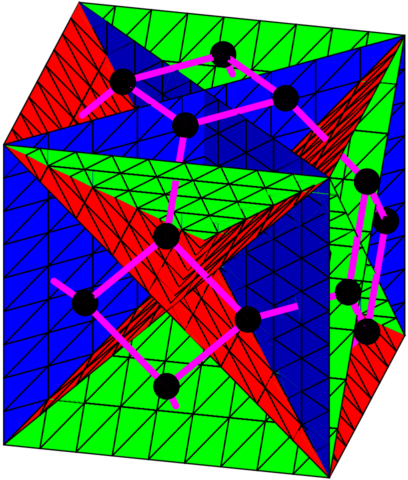
Here's what **Braid4** looks like!



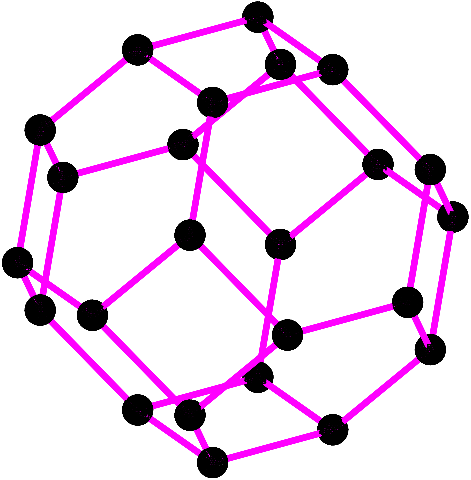
Suppose we put a dot in each region and connect adjacent dots. . .



Suppose we put a dot in each region and connect adjacent dots. . .



. . . and then remove the hyperplanes, leaving only the dots.



Regions of Braid4

The regions of **Braid4** correspond to the orderings of the four coordinates w, x, y, z:

$wxyz$	$wxzy$	$wyxz$	$wyzx$	$wzxy$	$wzyx$
$xwyz$	$xwzy$	$xywz$	$xyzw$	$xzwy$	$xzyw$
$ywxz$	$ywzx$	$yxwz$	$yxzw$	$yzwx$	$yzxw$
$zwxy$	$zwyx$	$zxwy$	$zxyw$	$zywx$	$zyxw$

- There are 4 possibilities for the first letter;
- 3 possibilities for the second, once the first is determined;
- 2 possibilities for the third, once the first two are determined;
- only 1 possibility for the last letter.

Total: $4 \times 3 \times 2 \times 1 =$ 24 orderings $=$ 24 regions.

Regions of Braid4

- We have just seen that **Braid4** has 24 regions.
- The regions correspond to permutations of w, x, y, z.
- Each region has exactly 3 neighboring regions.
- If two regions are neighbors, then the corresponding permutations differ by a single flip:

x **z** **w** y $\longleftrightarrow$ x **w** **z** y

Beyond the Fourth Dimension

The *n*-dimensional braid arrangement consists of the hyperplanes defined by the equations

$$\begin{aligned}
&x_1 = x_2,\\
&x_1 = x_3, \quad x_2 = x_3,\\
&\dots\\
&x_1 = x_n, \quad x_2 = x_n, \quad \dots, \quad x_{n-1} = x_n
\end{aligned}$$

- There are $n(n-1)/2$ hyperplanes (by the staircase formula!)
- The regions correspond to the possible orderings of the coordinates $x_1, \dots, x_n$.
- The number of regions is $n \times (n-1) \times (n-2) \cdots \times 3 \times 2 \times 1$ (also known as *n* factorial and written $n!$).
- Each region has $n-1$ neighboring regions.

Part 3: Cars, Trees and Keeping Score

Parking Cars

- A group of cars enter a parking lot, one by one.

Parking Cars

- A group of cars enter a parking lot, one by one.
- \# of parking spaces = # of cars (say n).

Parking Cars

- A group of cars enter a parking lot, one by one.
- # of parking spaces = # of cars (say n).

- The parking spaces are arranged along a one-way road.

Parking Cars

- A group of cars enter a parking lot, one by one.
- # of parking spaces = # of cars (say n).

- The parking spaces are arranged along a one-way road.
- Each car has a preferred parking space that it drives to first. If that spot is not available, it continues to the first empty space.

Parking Cars

- A group of cars enter a parking lot, one by one.
- # of parking spaces = # of cars (say n).

- The parking spaces are arranged along a one-way road.
- Each car has a preferred parking space that it drives to first. If that spot is not available, it continues to the first empty space.
- If there is no empty space, too bad!

Parking Cars

- A group of cars enter a parking lot, one by one.
- \# of parking spaces = # of cars (say n).

- The parking spaces are arranged along a one-way road.
- Each car has a preferred parking space that it drives to first. If that spot is not available, it continues to the first empty space.
- If there is no empty space, too bad!

- A parking function is a list of preferences such that all cars are able to park successfully.

Parking Cars

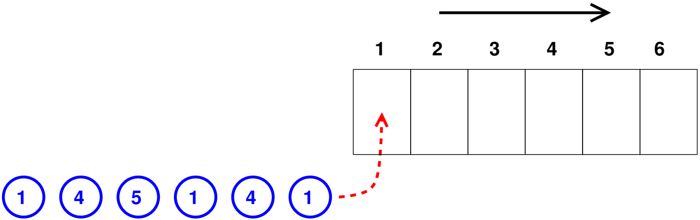
- A group of cars enter a parking lot, one by one.
- # of parking spaces = # of cars (say n).

- The parking spaces are arranged along a one-way road.
- Each car has a preferred parking space that it drives to first. If that spot is not available, it continues to the first empty space.
- If there is no empty space, too bad!

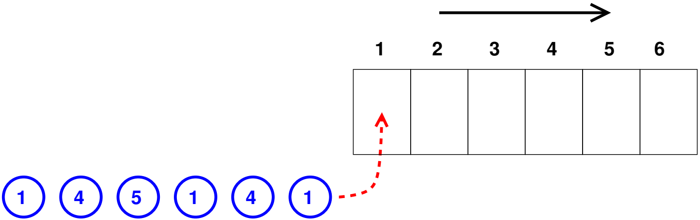
- A parking function is a list of preferences such that all cars are able to park successfully.

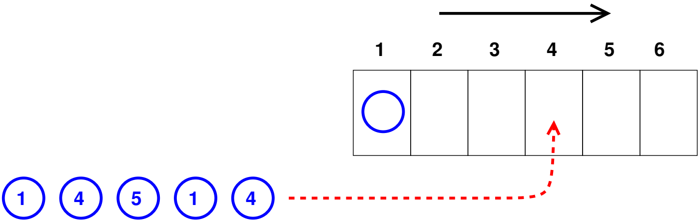
- Applications: database indexing, hash tables)

Parking Functions

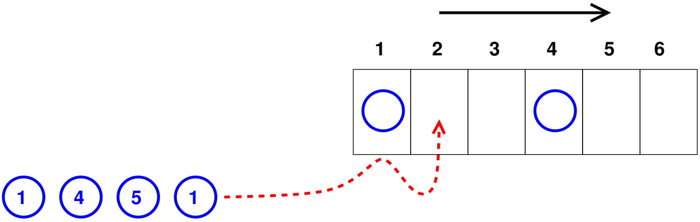


Parking Functions

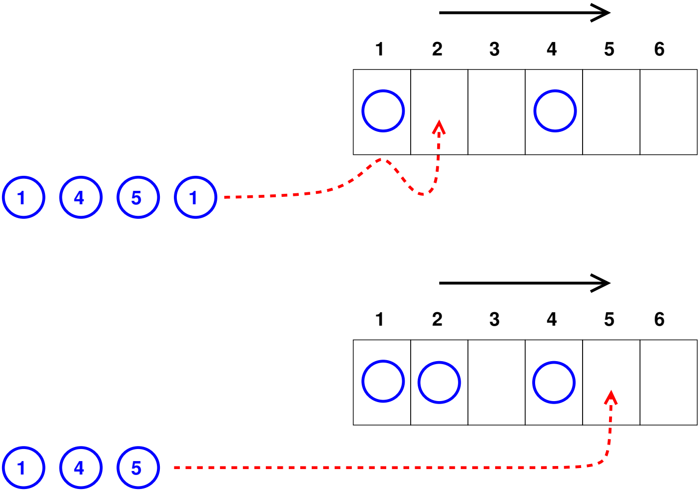




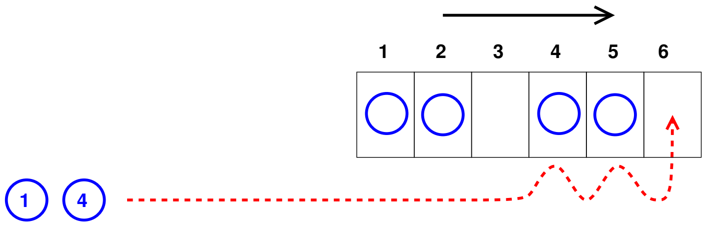
Parking Functions



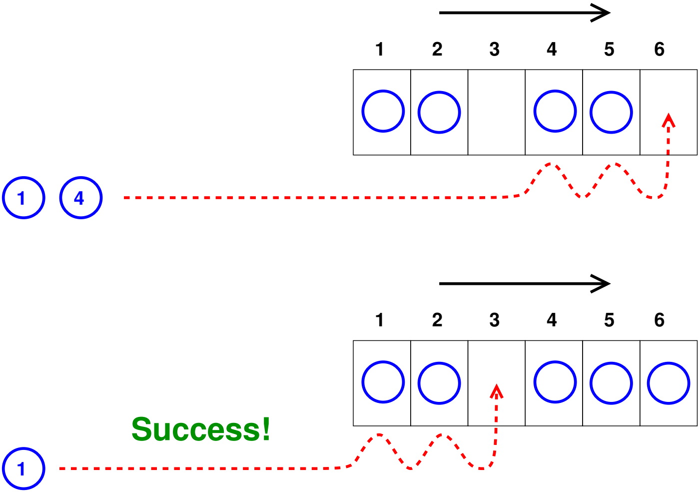
Parking Functions



Parking Functions



Parking Functions



Parking Functions

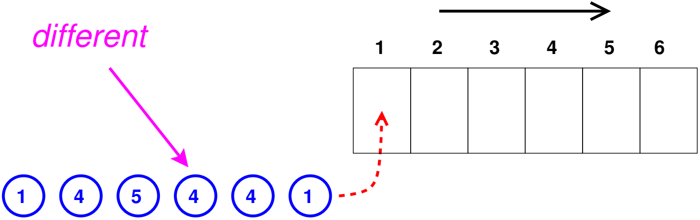
Therefore

1 4 5 1 4 1

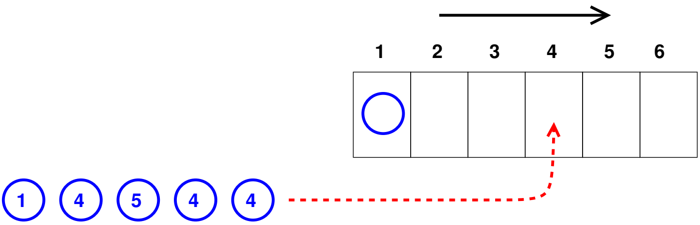
is a parking function. What about

1 4 5 4 4 1 ?

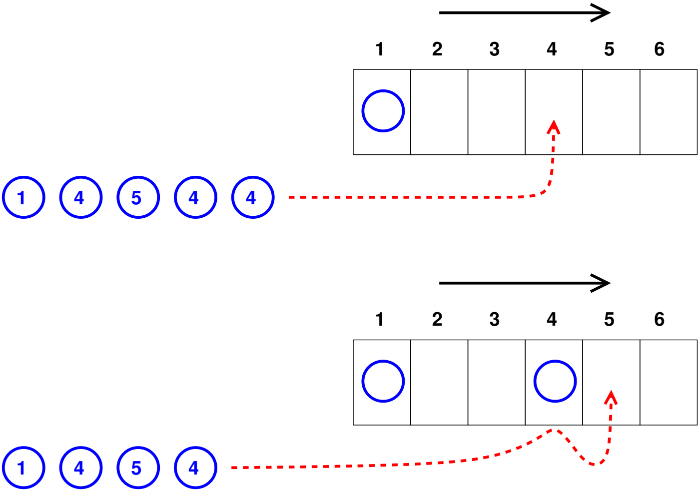
Parking Functions



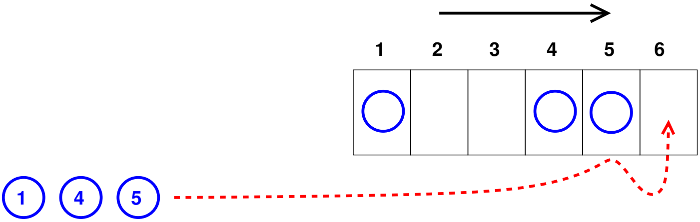
Parking Functions



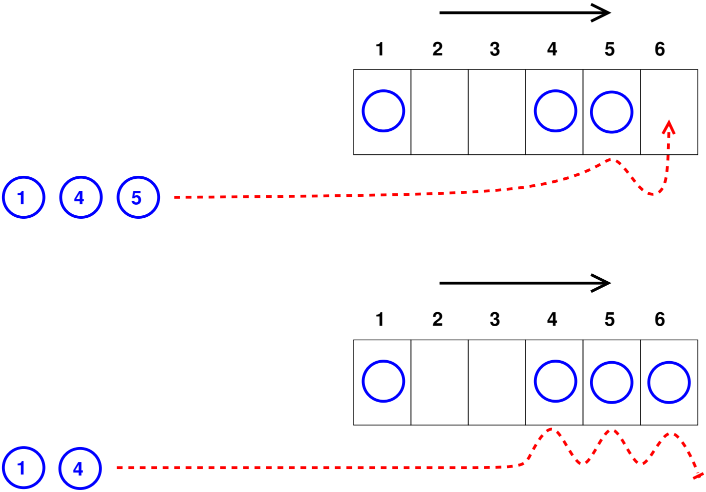
Parking Functions



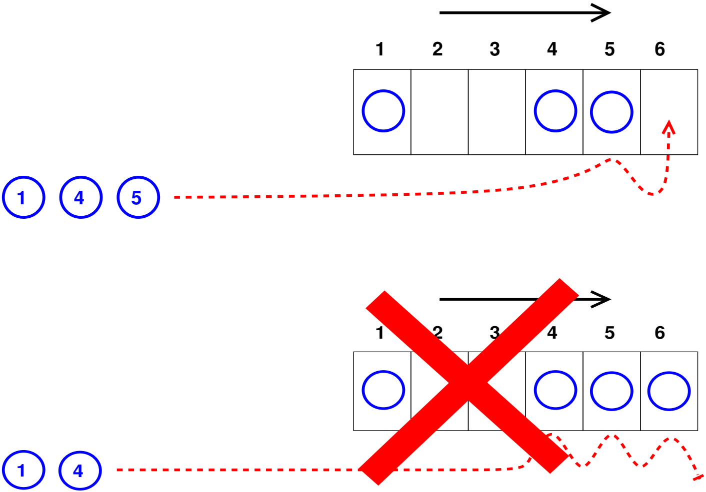
Parking Functions



Parking Functions



Parking Functions



Parking Two Cars

There are $4 = 2^2$ possible lists of preferred spots.
3 of them successfully park both cars.

1	1	OK	2	1	OK
1	2	OK	2	2	Not OK

Parking Three Cars

There are $27 = 3^3$ possible lists of preferred spots.
16 of them successfully park all three cars.

Parking functions (the ones that work):

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

Non-parking functions (the ones that don't work):

133	222	223	233	333
313		232	323	
331		322	332	

Parking n Cars

Observation #1: Whether or not all the cars can park depends on what their preferred spaces are, but not on the order in which they enter the parking lot.

For example, if there are 6 cars and the preference list includes two 5's and one 6, not all cars will be able to park.

Also, every parking function must include at least one 1. (What are some other conditions that must be satisfied?)

Parking n Cars

Observation #2: 3 cars $\implies$ 16 parking functions.

Parking n Cars

Observation #2: 3 cars $\implies$ 16 parking functions.

Number of cars (n)	Number of parking functions
1	1
2	3
3	16

Parking n Cars

Observation #2: 3 cars $\Longrightarrow$ 16 parking functions.

Number of cars (n)	Number of parking functions
1	1
2	3
3	16
4	125

Parking n Cars

Observation #2: 3 cars $\Longrightarrow$ 16 parking functions.

Number of cars (n)	Number of parking functions
1	1
2	3
3	16
4	125
5	1296

Do you see the pattern?

Parking n Cars

Observation #2: 3 cars $\implies$ 16 parking functions.

Number of cars (n)	Number of parking functions	
1	1	$= 2^0$
2	3	$= 3^1$
3	16	$= 4^2$
4	125	$= 5^3$
5	1296	$= 6^4$

Parking n Cars

Observation #2: 3 cars $\implies$ 16 parking functions.

Number of cars (n)	Number of parking functions	
1	1	$= 2^0$
2	3	$= 3^1$
3	16	$= 4^2$
4	125	$= 5^3$
5	1296	$= 6^4$

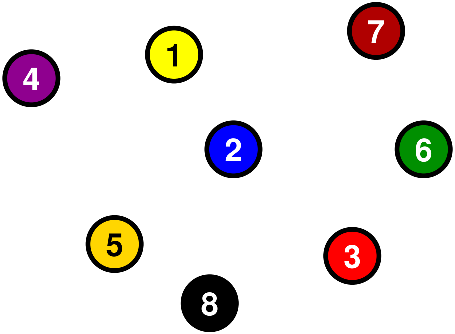
Conjecture: $\mathbf{n}$ **cars** $\implies$ $\mathbf{(n+1)^{n-1}}$ **parking functions.**

Connecting Points

Problem: Connect n points with as few links as possible.

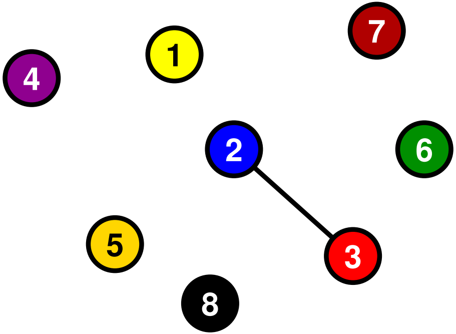
Connecting Points

Problem: Connect n points with as few links as possible.



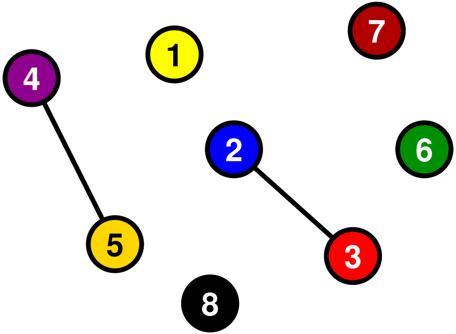
Connecting Points

Problem: Connect n points with as few links as possible.



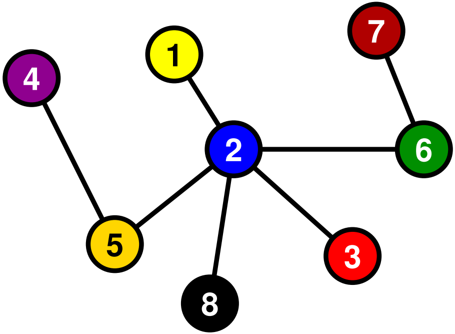
Connecting Points

Problem: Connect n points with as few links as possible.



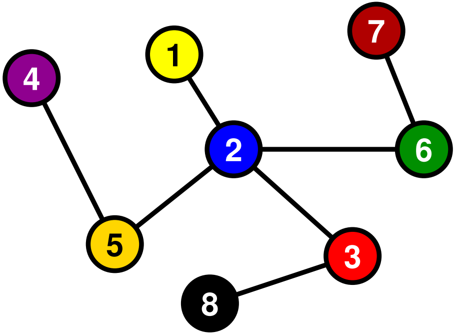
Connecting Points

Problem: Connect n points with as few links as possible.



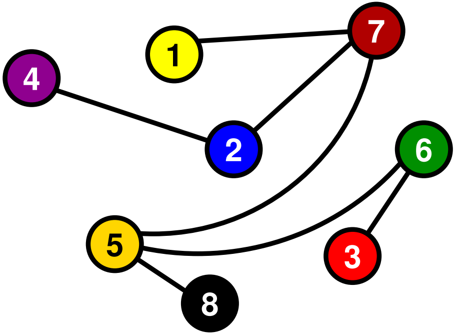
Connecting Points

Problem: Connect n points with as few links as possible.



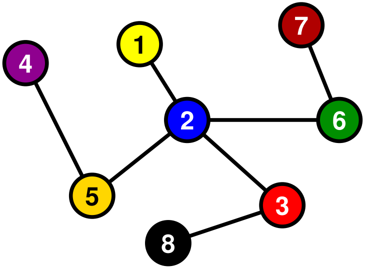
Connecting Points

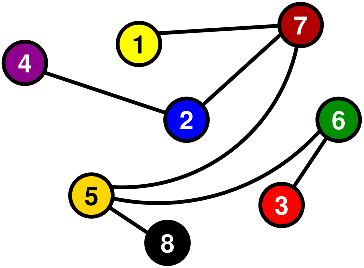
Problem: Connect n points with as few links as possible.



Connecting Points

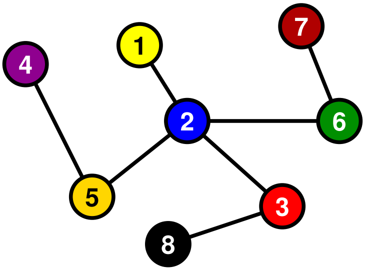
Problem: Connect n points with as few links as possible.

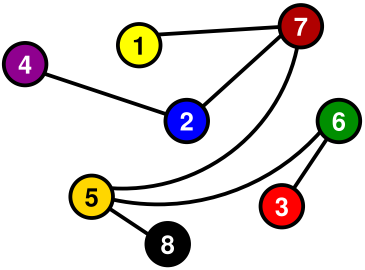




Connecting Points

Problem: Connect n points with as few links as possible.

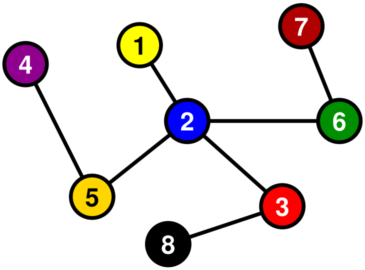


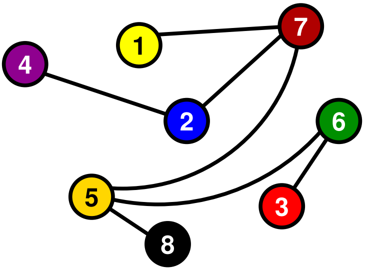


- It doesn't matter where the points are or how you draw the links — just which pairs of points are linked.

Connecting Points

Problem: Connect n points with as few links as possible.





- It doesn't matter where the points are or how you draw the links — just which pairs of points are linked.
- These structures are called trees.

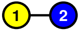
How Many Trees?

1 point:  1 tree

How Many Trees?

1 point: 

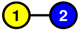
1 tree

2 points: 

1 tree

How Many Trees?

1 point:  1 tree

2 points:  1 tree

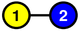
3 points:  3 trees

How Many Trees?

1 point:

1 tree

2 points:

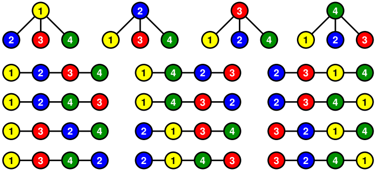


1 tree

3 points:

3 trees

4 points:



16 trees

Trees and Cars

# Cars	# Parking Functions
1	1
2	3
3	16
4	125
5	1296
...	
N	$(N+1)^{N-1}$

# Points	# Trees
1	1
2	1
3	36
4	16
5	125
...	
N	N^{N-2}

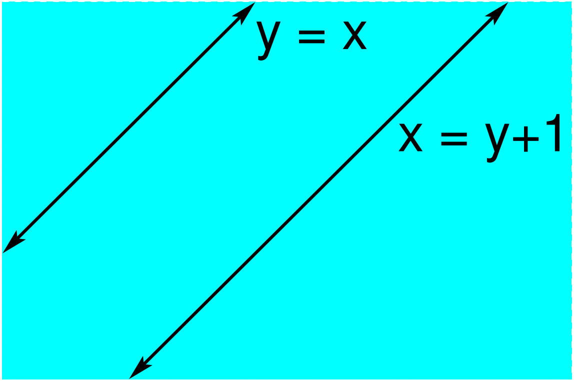
The Shi Arrangement

The *n-dimensional Shi arrangement* consists of the $n(n-1)$ hyperplanes defined by the equations

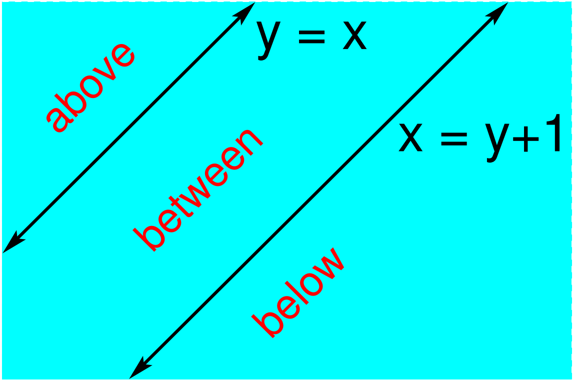
$$\begin{array}{cccc}
x_1 = x_2 & x_1 = x_3 & \dots & x_1 = x_n \\
x_1 = x_2 + 1 & x_1 = x_3 + 1 & \dots & x_1 = x_n + 1 \\
 & x_2 = x_3 & \dots & x_2 = x_n \\
 & x_2 = x_3 + 1 & \dots & x_2 = x_n + 1 \\
 & & & \vdots \\
 & & & x_{n-1} = x_n \\
 & & & x_{n-1} = x_n + 1
\end{array}$$

("Take the braid arrangement, make a copy of it, and push the copy a little bit.")

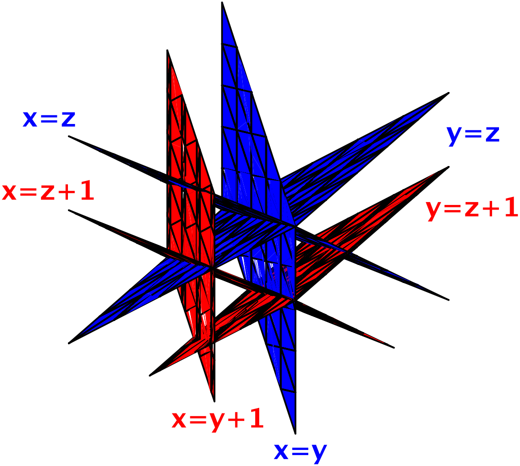
The 2D Shi Arrangement



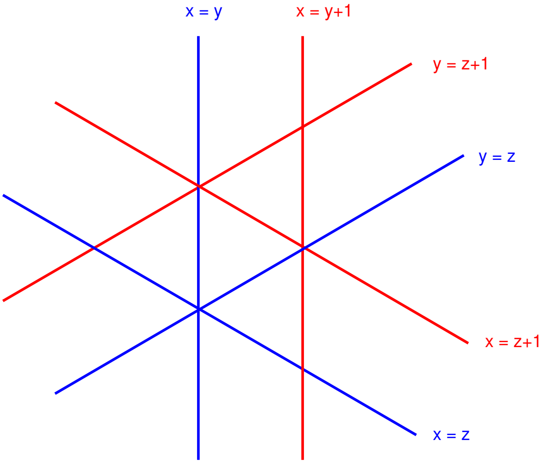
The 2D Shi Arrangement



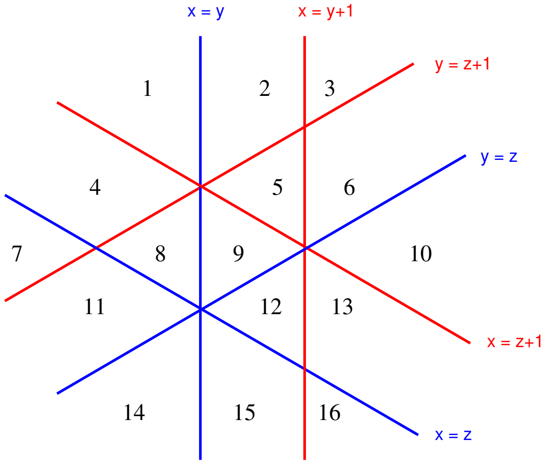
The 3D Shi Arrangement



The 3D Shi Arrangement



The 3D Shi Arrangement



Scoring with a Handicap

- A group of marathon runners are ranked 1 through n.

Scoring with a Handicap

- A group of marathon runners are ranked 1 through n.
- You score one point for each other runner you beat head-to-head.

Scoring with a Handicap

- A group of marathon runners are ranked 1 through n.
- You score one point for each other runner you beat head-to-head.
- But, in order to score a point against a lower-ranked runner, you must beat him/her by at least one minute.

Scoring with a Handicap

- A group of marathon runners are ranked 1 through n.
- You score one point for each other runner you beat head-to-head.
- But, in order to score a point against a lower-ranked runner, you must beat him/her by at least one minute.
- **The possible outcomes correspond to regions of the Shi arrangement!**

Slicing n-Dimensional Space

$(n+1)^{n-1} \quad = \quad$ number of regions of the Shi arrangement

Slicing n-Dimensional Space

$$
\begin{aligned}
(n+1)^{n-1} &= \text{number of regions of the Shi arrangement} \\
&= \text{number of handicapped-scoring outcomes}
\end{aligned}
$$

Slicing n-Dimensional Space

$$
\begin{aligned}
(n+1)^{n-1} &= \text{number of regions of the Shi arrangement} \\
&= \text{number of handicapped-scoring outcomes} \\
&= \text{number of trees on } n+1 \text{ points}
\end{aligned}
$$

Slicing n-Dimensional Space

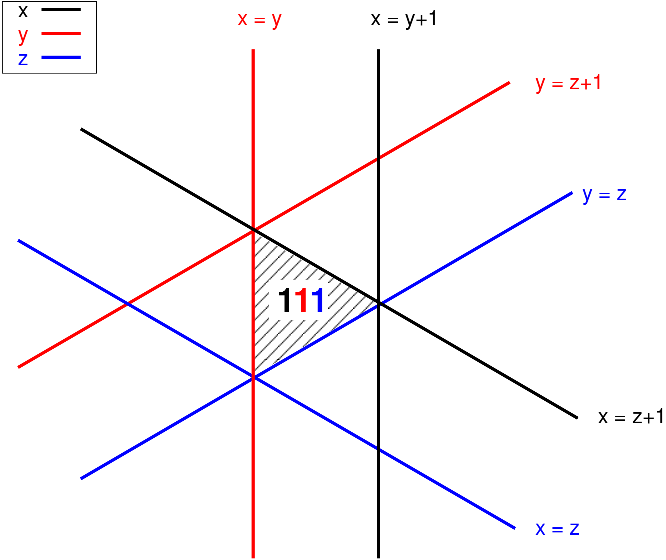
$$
\begin{aligned}
(n+1)^{n-1} &= \text{number of regions of the Shi arrangement} \\
&= \text{number of handicapped-scoring outcomes} \\
&= \text{number of trees on } n+1 \text{ points} \\
&= \text{number of ways to park } n \text{ cars}
\end{aligned}
$$

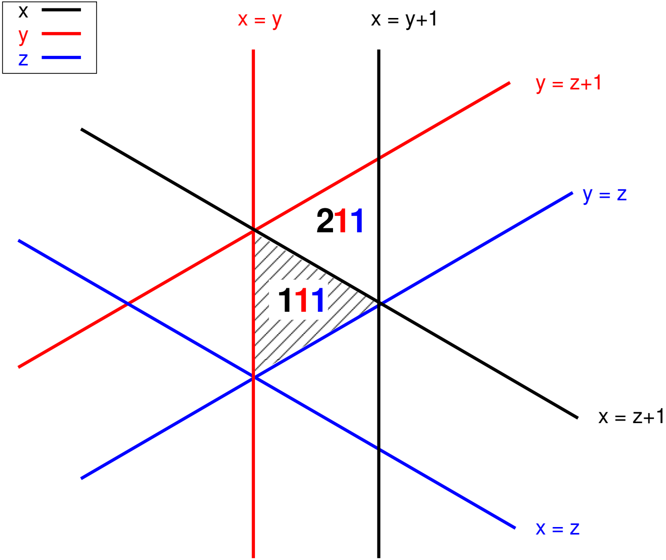
Slicing n-Dimensional Space

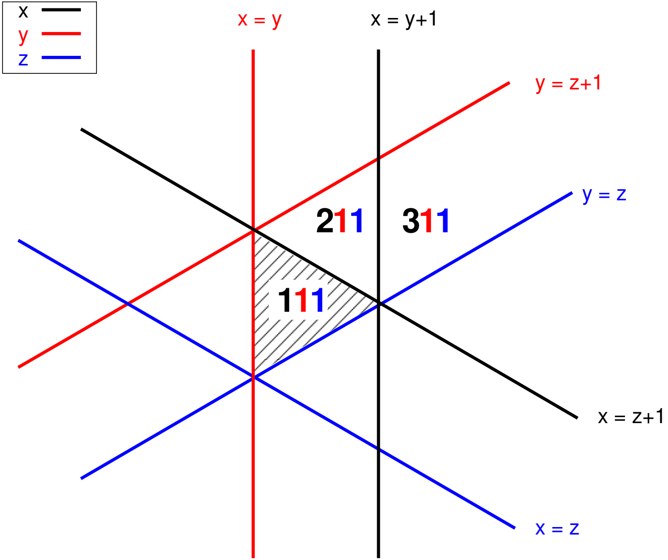
$$
\begin{aligned}
(n+1)^{n-1} &= \text{number of regions of the Shi arrangement} \\
&= \text{number of handicapped-scoring outcomes} \\
&= \text{number of trees on } n+1 \text{ points} \\
&= \text{number of ways to park } n \text{ cars}
\end{aligned}
$$

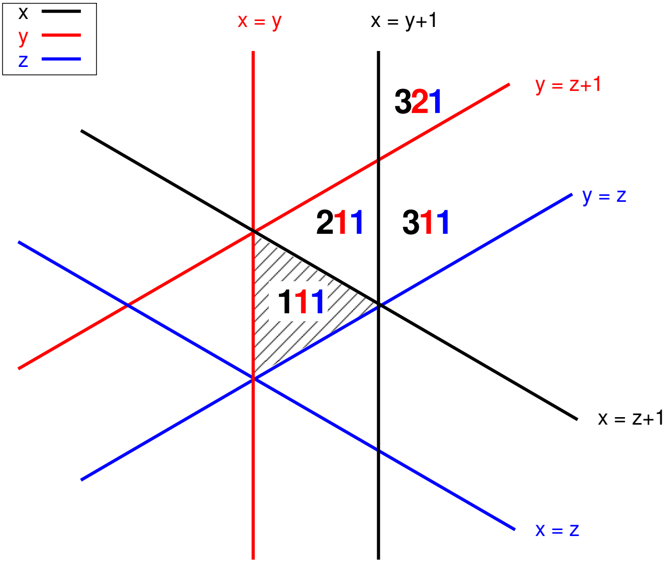
Why are all these numbers the same?

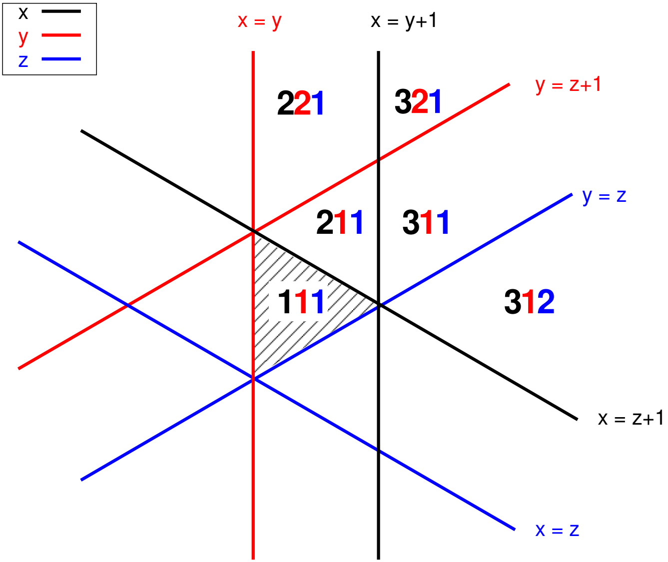
The next figure shows the correspondence between Shi-arrangement regions and parking functions for $n = 3$.

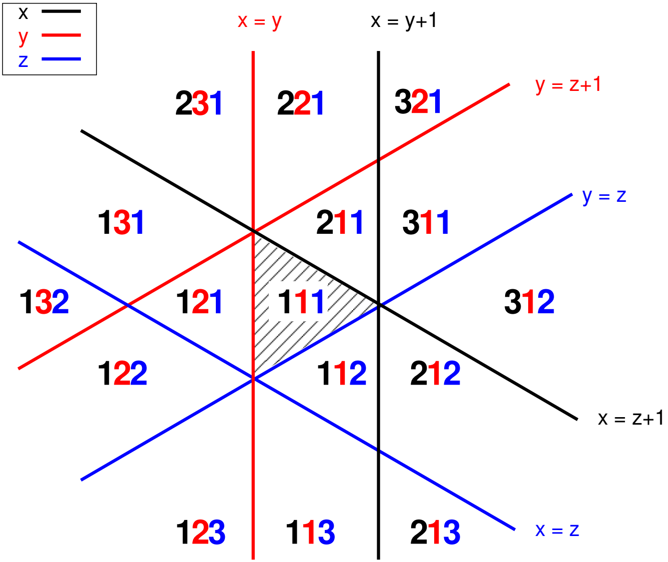

x
y
z
x = y
x = y+1
y = z+1
y = z
x = z+1
x = z
111


x
y
z
x = y
x = y+1
y = z+1
y = z
211
111
x = z+1
x = z


x
y
z
x = y
x = y+1
y = z+1
y = z
x = z+1
x = z
211
311
111


x
y
z
x = y
x = y+1
y = z+1
y = z
x = z+1
x = z
321
211
311
111


x
y
z
x = y
x = y+1
y = z+1
y = z
x = z+1
x = z
221
321
211
311
111
312


x
y
z
x = y
x = y+1
y = z+1
y = z
x = z+1
x = z
231
221
321
131
211
311
132
121
111
312
122
112
212
123
113
213