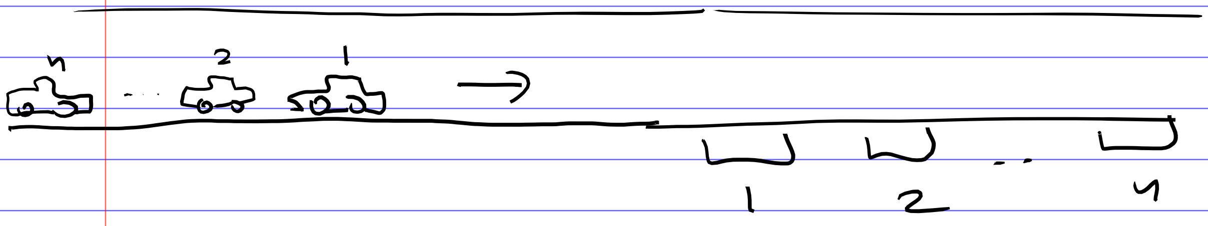
18.212 Lecture 26 04/21/2021

Parking functions

- n cars, n parking spots on a one-way road.



Preference function $f : [n] \to [n]$

$f : i \mapsto f_i$ $\qquad f = (f_1, \ldots, f_n)$

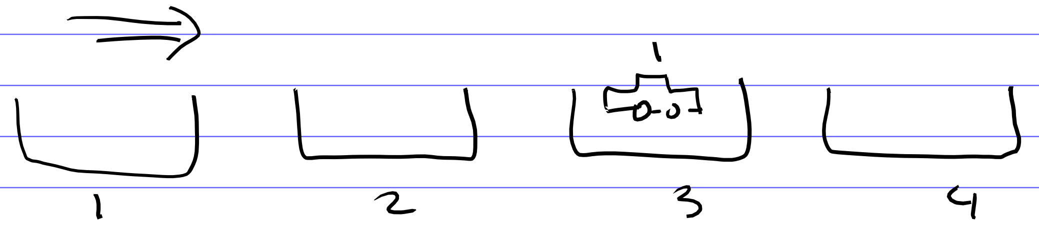
The driver of the i^{th} car prefers to park in the f_i^{th} parking spot.

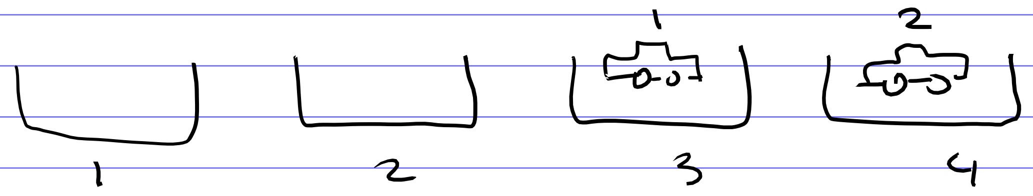
- i^{th} car drives to f_i^{th} spot and parks there if the spot is empty.
  Otherwise, it keeps driving untill it finds an empty spot and parks there.

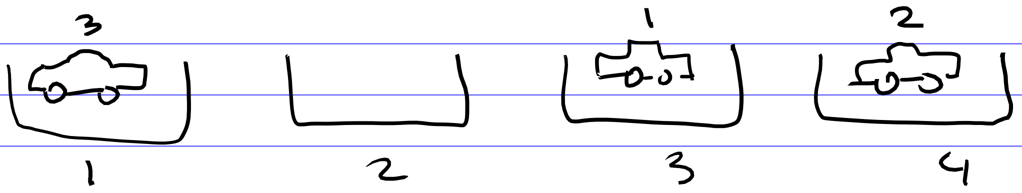
Definition $f = (f_1, \ldots, f_n)$ is called a parking function if all n cars will park.

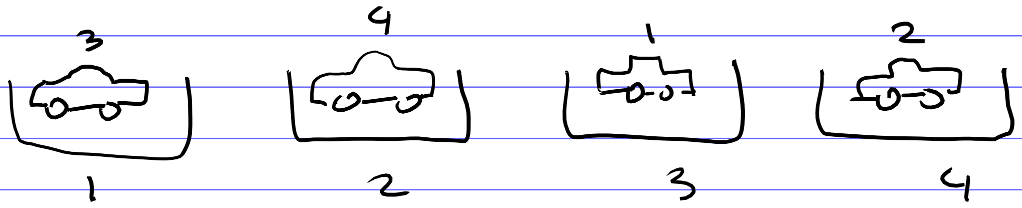
Example $n = 4$

$(f_1, \ldots, f_4) = (3, 3, 1, 1)$



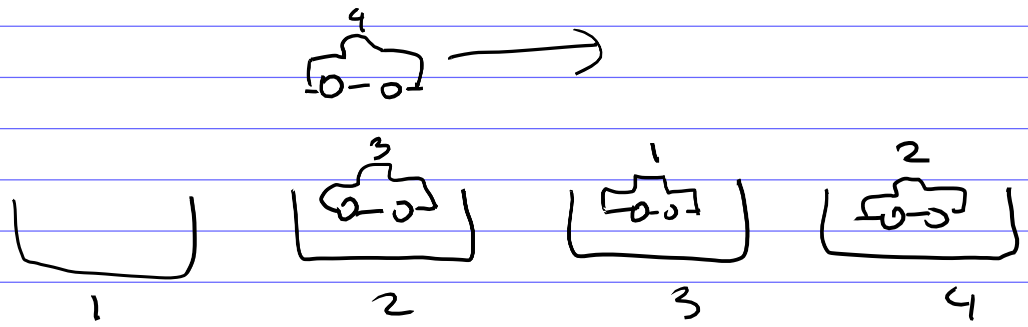






So $(3, 3, 1, 1)$ is a parking function.

But $(3, 3, 2, 2)$ is <u>not</u> a parking function, because 4^{th} car cannot park:



Lemma $f = (f_1, \dots, f_n)$ $f_i \in [n]\ \forall i$

TFAE:

(A) f is a parking function

(B) The sequence $f_1, \dots, f_n$ contains

at most 1 entry n;

at most 2 entries $\geq n-1$;

at most 3 entries $\geq n-2$;

...

at most k entries $\geq n-k+1$

for $k = 1, 2, \dots, n$.

$$\#\{f_i \geq n-k+1\} \leq k \text{ for } k \in [n].$$

(C) There exists a permutation $w_1, \dots, w_n$ of $1, 2, \dots, n$ s.t. $f_i \leq w_i$ $\forall$ $i = 1, 2, \dots, n$

Exercise. Prove that these 3 conditions are equivalent.

Note: It is clear that (A) $\Rightarrow$ (B)

Example $n=3$.

All parking functions.

6 permutations:

123, 132, 213

231, 312, 321

and everything obtained from these permutations by decreasing some entries:

122, 212, 221

113, 131, 311

112, 121, 211

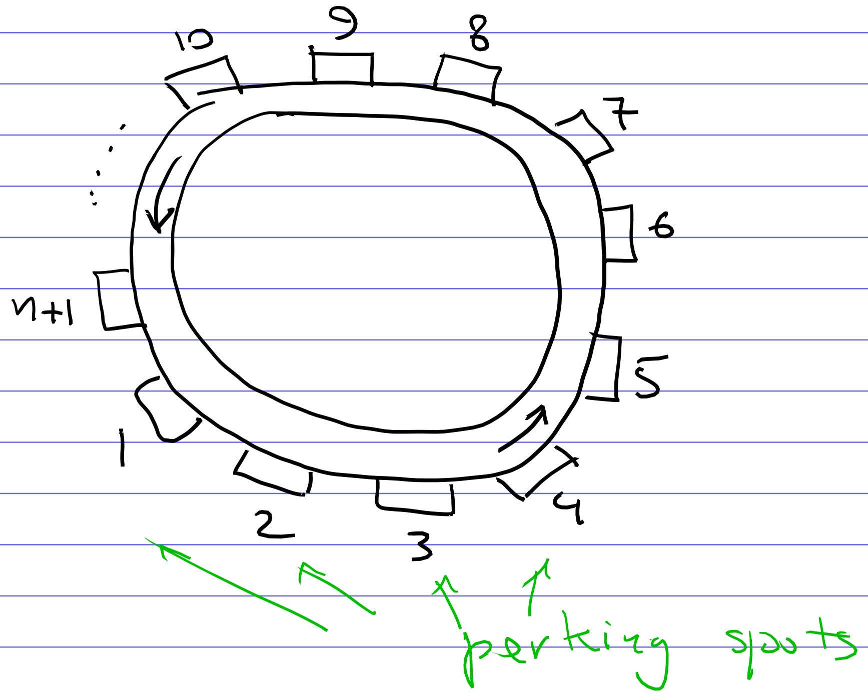
111

In total, $6+3+3+3+1=16$ parking functions for $n=3$.

Theorem There are exactly $(n+1)^{n-1}$ parking functions of size n.

Proof We will modify the setup as follows:

- n cars
- $n+1$ parking spots on a circular road (with counterclockwise direction)
- preference function $f: [n] \to [n+1]$.



The cars park according to the same procedure.

Now all n cars will always park for any preference function $f: [n] \to [n+1]$.

Moreover, one parking spot will be left empty after all n cars park.

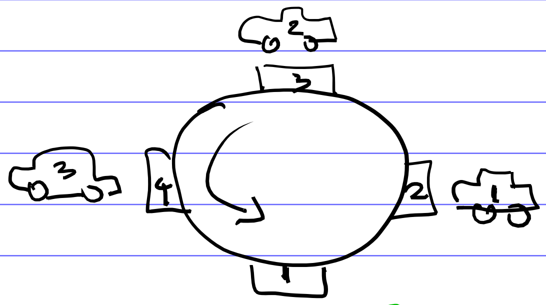
This construction has circuler symmetry:

If $\tilde{f}_i = f_i + 1 \mod n+1$ for all $i \in [n]$,

Then the resulting perking arrengement for $\tilde{f}$ is obtein from the perking arrengement for f by shifting it 1 step couter clockwise.

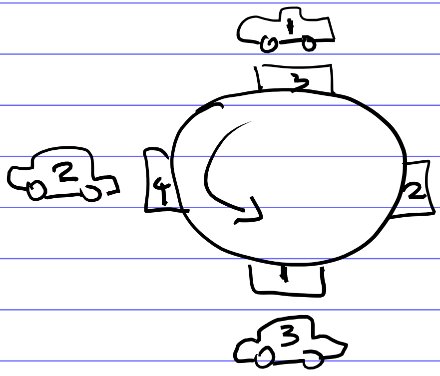
Exemple $n = 3$

$f = (2, 3, 3)$



1^{st} spot remains empty

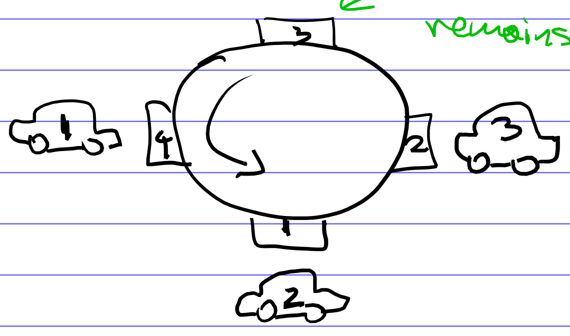
$\tilde{f} = (3,4,4)$



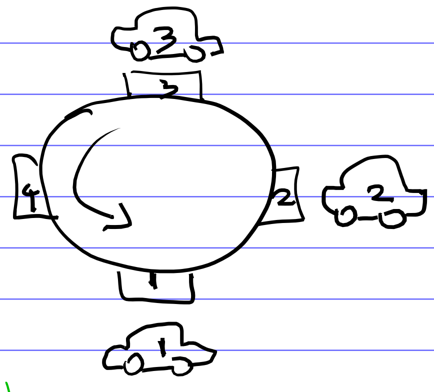
2^{nd} spot remains empty

$\tilde{\tilde{f}} = (4,1,1)$

3^{rd} spot remains empty



$\tilde{\tilde{\tilde{f}}} = (1,2,2)$



4^{th} spot remains empty

<u>Observation</u>. A preference function $f : [n] \to [n+1]$ is a parking function iff in the resulting parking arrangement of the cars the $(n+1)^{st}$ spot remains empty.

This means that no car drives past $(n+1)^{st}$ parking spot. In other words, this is equivalent to parking on a non-circular road.

Let F_i be the set of preference functions for which i^{th} spot remains empty.

F_{n+1} = the set of parking functions.

Because of the circular symmetry

$$|F_1| = |F_2| = \dots = |F_{n+1}|.$$

So $\#\{\text{parking functions}\} = |F_{n+1}|$

$$= \frac{1}{n+1} \# \left\{ \begin{array}{c} \text{all preference functions} \\ f : [n] \to [n+1] \end{array} \right\}$$

$$= \frac{1}{n+1} \cdot (n+1)^n = (n+1)^{n-1}.$$

$\square$

Exercise Find a bijection between parking functions of size n and spanning trees of K_{n+1}.

parking functions $\overset{?}{\longleftrightarrow}$ labelled trees

Dyck paths again

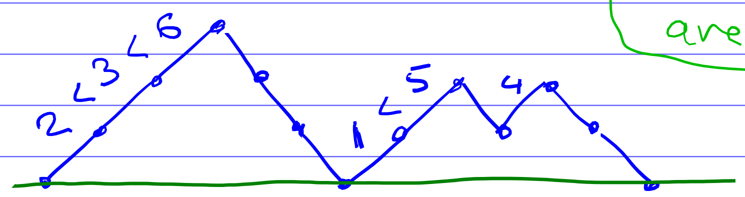
Consider a Dyck path P of length $2n$ (i.e. P has n "up" steps and n "down" steps.)

Let us label all "up" steps in P by $1, 2, \dots, n$ (without repetitions) such that the labels in any consecutive sequence of "up" steps increase.

We'll call them *labelled Dyck paths*.

only the "up" steps are labelled

Example
$n = 6$



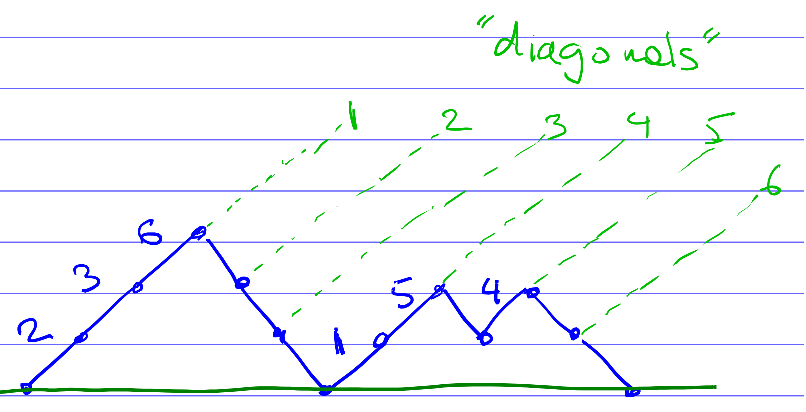
a labelled Dyck path

Theorem The number of labelled Dyck paths of length $2n$ equals $(n+1)^{n-1}$.

Proof. Let's construct a bijection:

$$\left\{\begin{matrix}\text{labelled}\\ \text{Dyck}\\ \text{paths}\end{matrix}\right\} \xleftrightarrow{1:1} \left\{\begin{matrix}\text{parking}\\ \text{functions}\end{matrix}\right\}$$

Example



Let's mark the diagonals on which the "up" steps are located by $1, 2, \dots, n$, as shown above.

A labelled Dyck path P corresponds to the parking function $f = (f_1, \dots, f_n)$ s.t.

$f_i = j$ iff the "up" step labelled j is on the i^{th} diagonal.

Example (the above labelled Dyck path) $\longleftrightarrow f = (f_1, \dots, f_6)$

- 1^{st} diagonal contains the "up" steps labelled $2, 3, 6$.
  So $f_2 = f_3 = f_6 = 1$
- 4^{th} diagonal contains the "up" steps labelled $1, 5$
  So $f_1 = f_5 = 4$.
- 5^{th} diagonal contains label 4.
  So $f_4 = 5$.

We get

$$\begin{pmatrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\ 4 & 1 & 1 & 5 & 4 & 1 \end{pmatrix}$$

$f = (4, 1, 1, 5, 4, 1)$.

Here indices i in f_i correspond to the labels in the Dyck path; and the values of f_i correspond to diagonals in the Dyck path.

It is easy to see that the condition that f is a parking function correspond to condition that P is a Dyck path

Indeed, a lattice path P from $(0,0)$ to $(2n, 0)$ is a Dyck path iff P has

- no "up" steps on $(n+1)^{st}$ diag.
- at most 1 "up" step on n^{th} diag.
- at most 2 "up" steps on $(n-1)^{st}$ and n^{th} diag.
- at most 3 "up" steps on $(n-2)^{nd}$, $(n-1)^{st}$, and n^{th} diag.

etc.

These conditions mean that

$\#\{f_i = n\} \leq 1$

$\#\{f_i \in \{n-1, n\}\} \leq 2$

$\#\{f_i \in \{n-2, n-1, n\}\} \leq 3$

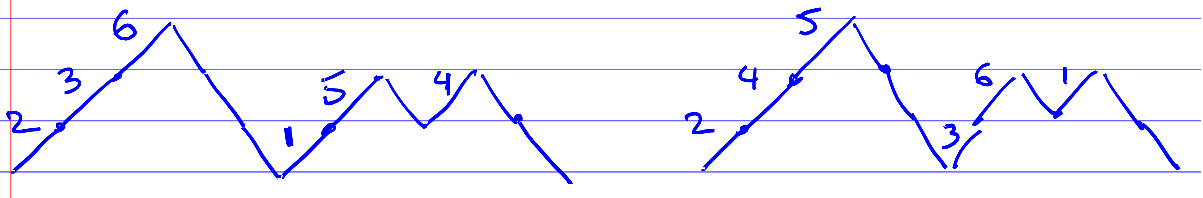
etc.

This is exactly the condition (B) that defines parking functions.

$\square$

Remark. Clearly, if we permute entries in a parking function we get a parking function. For example

$$\left(\overset{1}{4},\overset{2}{1},\overset{3}{1},\overset{4}{5},\overset{5}{4},\overset{6}{1}\right) \rightsquigarrow \left(\overset{1}{5},\overset{2}{1},\overset{3}{4},\overset{4}{1},\overset{5}{1},\overset{6}{4}\right)$$



Permutations of $(f_1, \dots, f_n)$ correspond to relabellings of labelled Dyck paths.

Thus the equivalence classes of parking function under permutation of entries correspond to usual (unlabelled) Dyck paths.

Propositions # weakly increasing parking functions of size n equals the n^{th} Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$.

Example $n=3$.

$C_3 = 5$ weakly increasing parking functions:

123, 122, 113, 112, 111.

Generalized parking functions.

Fix $n \geq 1$ and $\alpha = (\alpha_1 \leq \dots \leq \alpha_n)$ (where α_i's are positive integers)

Definition $f = (f_1, \dots, f_n)$ is an α-parking function iff

- $f_1, \dots, f_n \in \{1, 2, \dots, \alpha_n\}$
- $\forall\ k = 1, 2, \dots, n$

$$\#\{f_i \leq \alpha_k\} \geq k.$$

Equivalently, the set of α-parking functions consists of all $(f_1, \dots, f_n)$ obtained from $(\alpha_1, \dots, \alpha_n)$ by

- permutation the entries,

and/or

- decreasing some entries.

For example, the usual parking functions are exactly the α-parking functions for $\alpha = (1, 2, 3, \dots, n)$

Example $n=2$ $\alpha=(1,4)$

α-parking functions:

14, 41, 13, 31, 12, 21, 11

7 α-parking functions for $\alpha=(1,4)$.

Theorem. Fix $n, k, \ell \geq 1$.

Let $\alpha =$

$= (\ell, \ell+k, \ell+2k, \ell+3k, \ldots, \ell+(n-1)k)$

Then $\#$ α-parking functions equals $\ell \cdot (\ell + kn)^{n-1}$.

Example (above example)

$n=2$, $\ell=1$, $k=3$

$\#\{\alpha\text{-parking funct.}\}$

$= (1+2\cdot 3)^{2-1} = 7.$

Chip-firing game

(a.k.a. Abelian sandpile model)

This is a simple mathematical model for complicated natural processes such as avalanches.

- [Per Bak, Chao Tang, Kurt Wisenfeld, 1987]

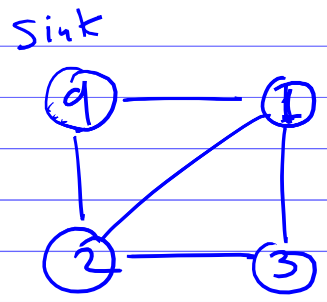
- [Anders Björer, Laslo Lovász, Peter Shor, 1991]

$G = (V, E)$ a connected finite graph with one selected vertex $q \in V$, called the sink.

We'll assume $V = \{0, 1, \dots, n\}$ and $q = 0$.

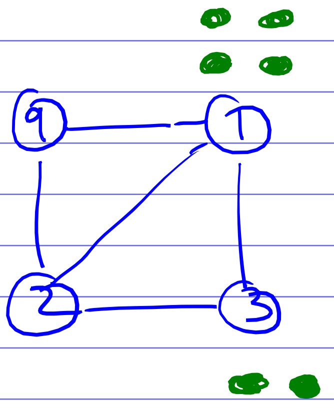
A chip configuration $C = (c_1, \dots, c_n)$ is any non-negative integer vector. We think of c_i as the number of chips at vertex i.

Example $G =$



a chip configuration

$C = (4, 0, 2)$



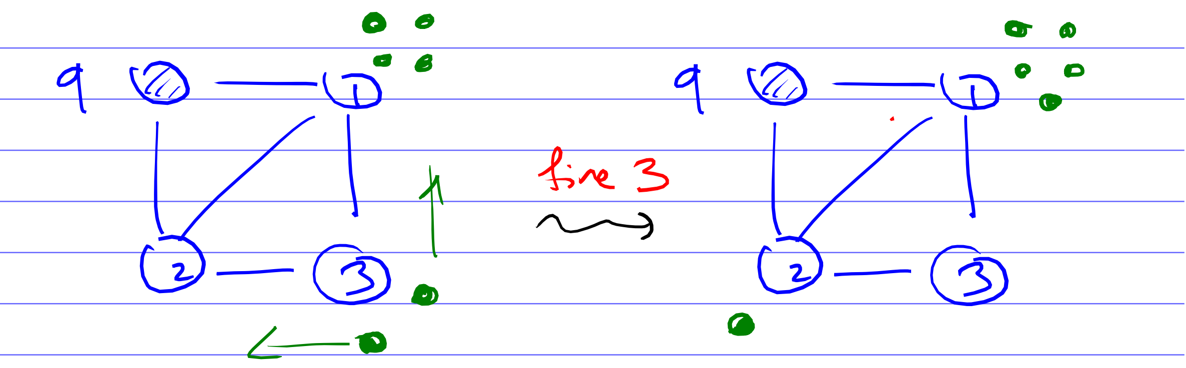
The are no chips at the sink q.

The sink is a "black hole": any chip that goes into it "vanishes"

Let $d_i = \deg_G(i)$ degree of vertex i.

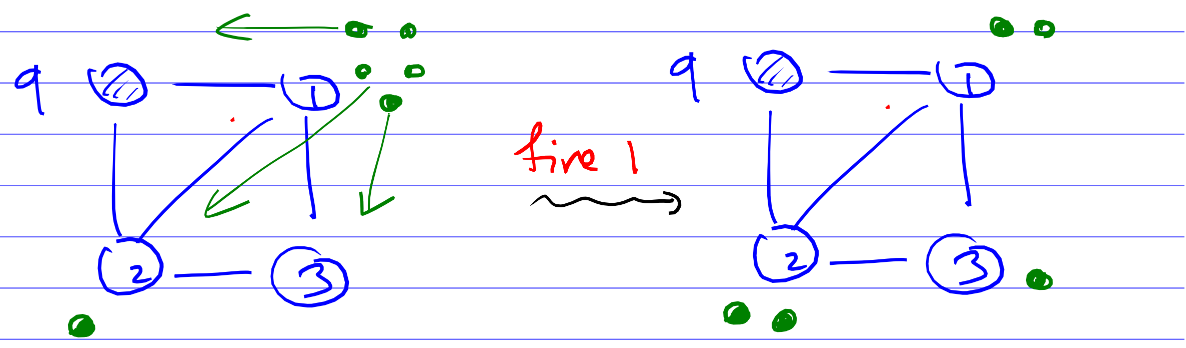
If $c_i \geq d_i$ then i is called a critical site. We can **fire** such vertex i by sending one chip to each neighbour of i.

Example. We can fire vertex 3.



$$C = (4, 0, 2) \rightsquigarrow (5, 1, 0)$$

Then we can fire vertex 1



$$(5, 1, 0) \rightsquigarrow (2, 2, 1)$$

Notice that one chip goes into the sink q and disappears. So the total number of chips decreases.

Now we cannot fire any vertex, because # chips at any vertex < the degree of the vertex.

Def. A chip configuration $(c_1, \ldots, c_n)$ is called **stable** if $c_i < d_i \ \forall i$.

(i.e. we cannot fire any vertex)

Lemma. For any initial chip configuration $C_{init} = (c_1, \dots, c_n)$,

- we will always obtain a stable chip configuration

  $C_{stab} = (c_1', \dots, c_n')$

  after a <u>finite</u> number of firings.
- The resulting stable configuration C_{stab} is <u>unique</u>, i.e., it depends only on C_{init} but not on a choice of firings.

C_{stab} is called the <u>stabilization</u> of the initial conf. C_{init}.

<u>Example</u> (the previous example)

For $C_{init.} = (4, 0, 2)$ the stabilization is

$$C_{stab} = (2, 2, 1).$$

We will arrive to the same stable configuration if we first fire vertex 1 and then fire vertex 3.

Consider the following random model

(0) Start with any initial configuration c_{init}

(1) stabilize the configuration

(2) randomly select a vertex $i \in \{1, 2, \ldots, n\}$ (with uniform distribution) and add 1 chip to vertex i, i.e. increase c_i by 1.

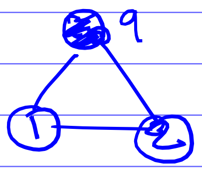
(3) go to step (1)

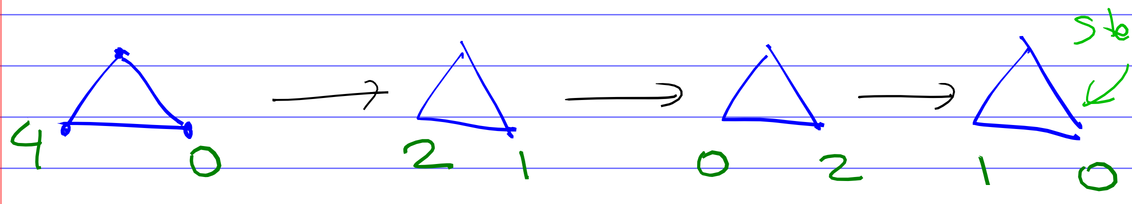
Basically, we keep repeating steps (2) and (1) (randomly drop a chip, stabilize, drop a chip, stabilize, etc.)

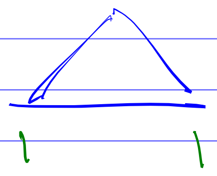
A stable configuration is calle **recurrent** if it keeps occuring in this random process.

Theorem # recurrent chip configurations equals the number of spanning trees of the graph G.

Example G =

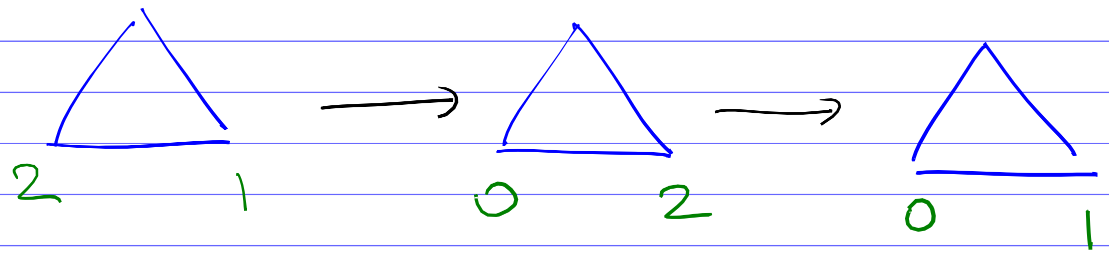


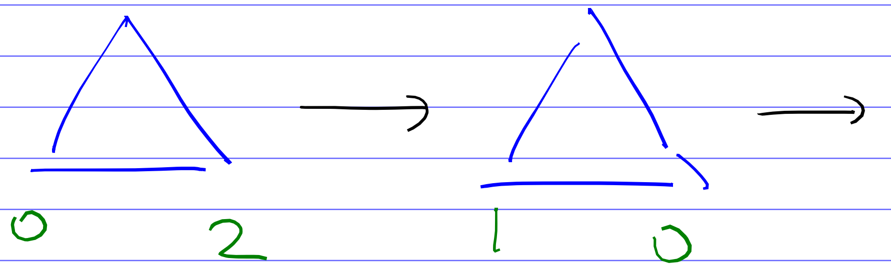




drop a chip at vert. 2 (still a stable conf.)

drop another chip at 1



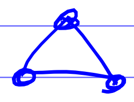


etc.

Recurrent configurations:

(1,1), (1,0), (0,1)

(There are 3 spanning trees of



.)

The configuration (0 0) is stable but not recurrent. It can appear only in initial steps of this random process. But if we run it for a while it can never occur.