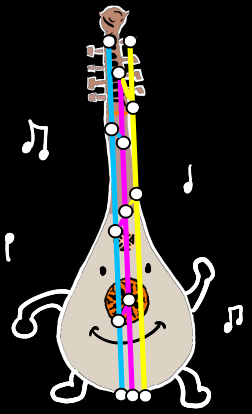
Random generation of Git graphs

Julien COURTIEL (Université de Caen Normandie)
with Martin PEPIN (Université de Caen Normandie)

Instrument beginning by "git" but not a guitar (answer at the end)



GASCom 2024

QUESTION: What do you use to share files with your coauthors?

THE TIER LIST

QUESTION: What do you use to share files with your coauthors?

god-like

good

trash

spawn of the devil

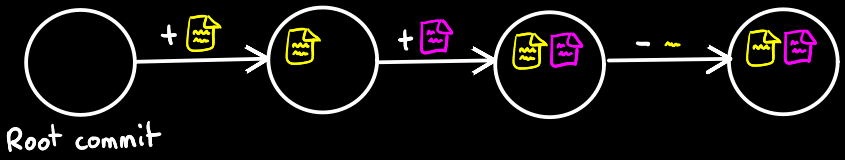
THE TIER LIST

QUESTION: What do you use to share files with your coauthors?

god-like	 
good	 
trash	
spawn of the devil	 

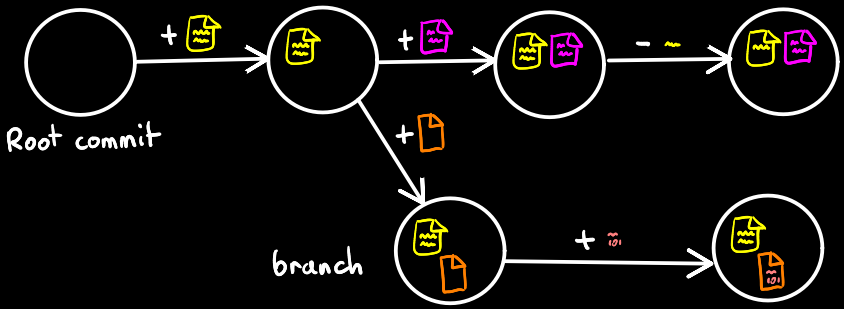
GIT FEATURES

git is a Version Control System (VCS): it stores all the project states over time.



GIT FEATURES

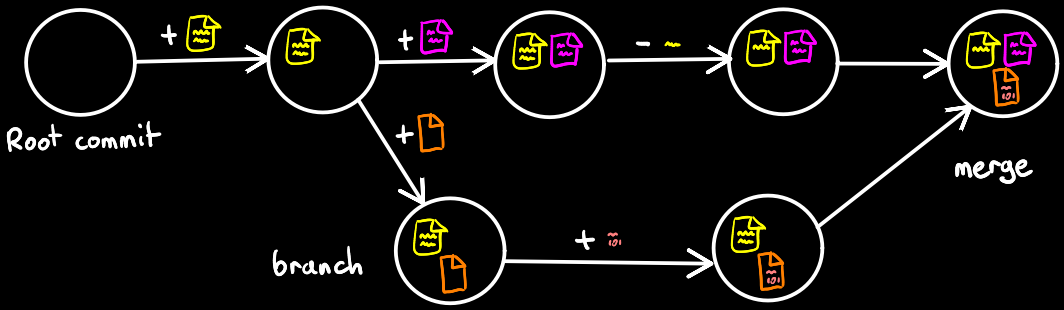
git is a Version Control System (VCS): it stores all the project states over time.



Git lets you create parallel development branches

GIT FEATURES

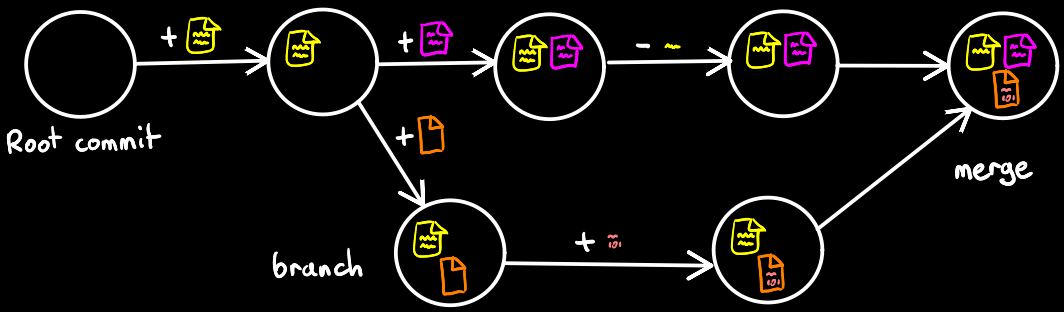
git is a Version Control System (VCS):
it stores all the project states over time.



Git lets you create parallel development branches
that can be integrated later.

GIT FEATURES

git is a Version Control System (VCS):
it stores all the project states over time.



Git lets you create parallel development branches
that can be integrated later.

This forms a Directed Acyclic Graph (DAG),
where the vertices are the project states, also named commits.

OUR MOTIVATION

Objective : Design random generators for graphs of commits

Why?

OUR MOTIVATION

Objective : Design random generators for graphs of commits

Why?

1. VCS unit testing

2. Benchmarks for VCS algorithms

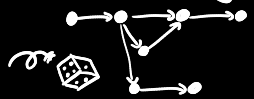
OUR MOTIVATION

Objective : Design random generators for graphs of commits

Why?

1. VCS unit testing

use random graphs



to check program correctness

```
void test_remove_last_commit() {

 srand(time(NULL));

 for (int i = 0; i < NUM_TESTS; i++) {

  GitGraph graph1 = create_random_git_graph();
  size_t n1 = graph1.num_commits;

  GitGraph graph2 = remove_last_commit(graph1);
  size_t n2 = graph2.num_commits;

  assert(n1 == n2 + 1);

 }
}
```

2. Benchmarks for VCS algorithms

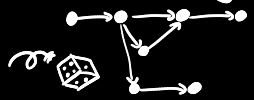
OUR MOTIVATION

Objective : Design random generators for graphs of commits

Why?

1. VCS unit testing

use random graphs



to check program correctness

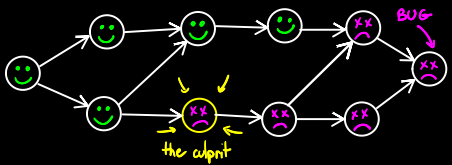
```c
void test_remove_last_commit() {

 srand(time(NULL));

 for (int i = 0; i < NUM_TESTS; i++) {

  GitGraph graph1 = create_random_git_graph();
  size_t n1 = graph1.num_commits;

  GitGraph graph2 = remove_last_commit(graph1);
  size_t n2 = graph2.num_commits;

  assert(n1 == n2 + 1);

 }
}
```

2. Benchmarks for VCS algorithms

e.g.

Regression Search Problem

Find the commit that introduces a bug

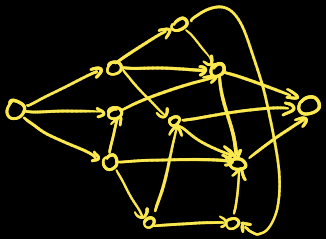


Analysis of algorithms such as git bisect

[C. Dorbec Lecoq 2023]

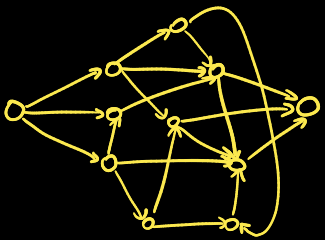
WHICH GRAPHS TO GENERATE?

In git, every DAG without restriction can be generated...

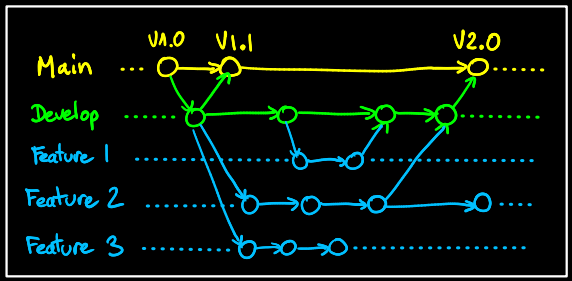


WHICH GRAPHS TO GENERATE?

In git, every DAG without restriction can be generated...

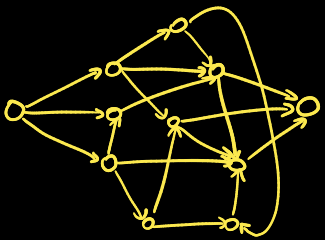


... but many projects follow a workflow

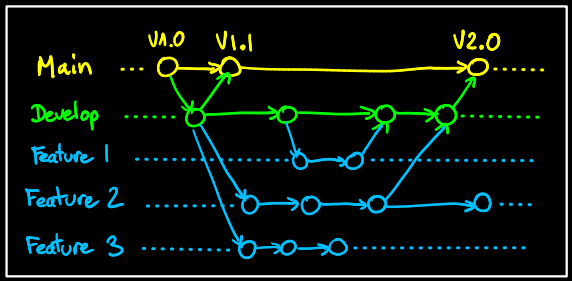


WHICH GRAPHS TO GENERATE?

In git, every DAG without restriction can be generated...



... but many projects follow a workflow



In this work, we consider a simple workflow but widely used in industry: the feature branch workflow

GIT GRAPH

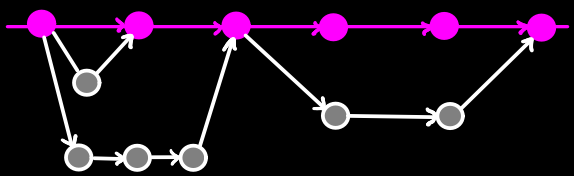
DEFINITION

(feature branch) Git graph = DAG with

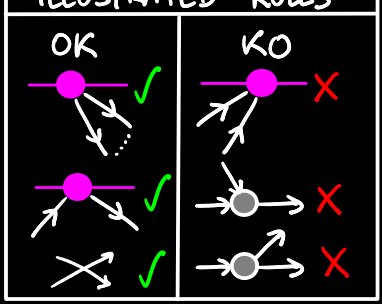
- a main branch (path of magenta vertices)
- 0, 1 or several feature branches, paths of ≥ 1 white vertices starting and ending on magenta vertices
- indegree ≤ 2 for all vertices

previously defined in [Lecoq 2024]

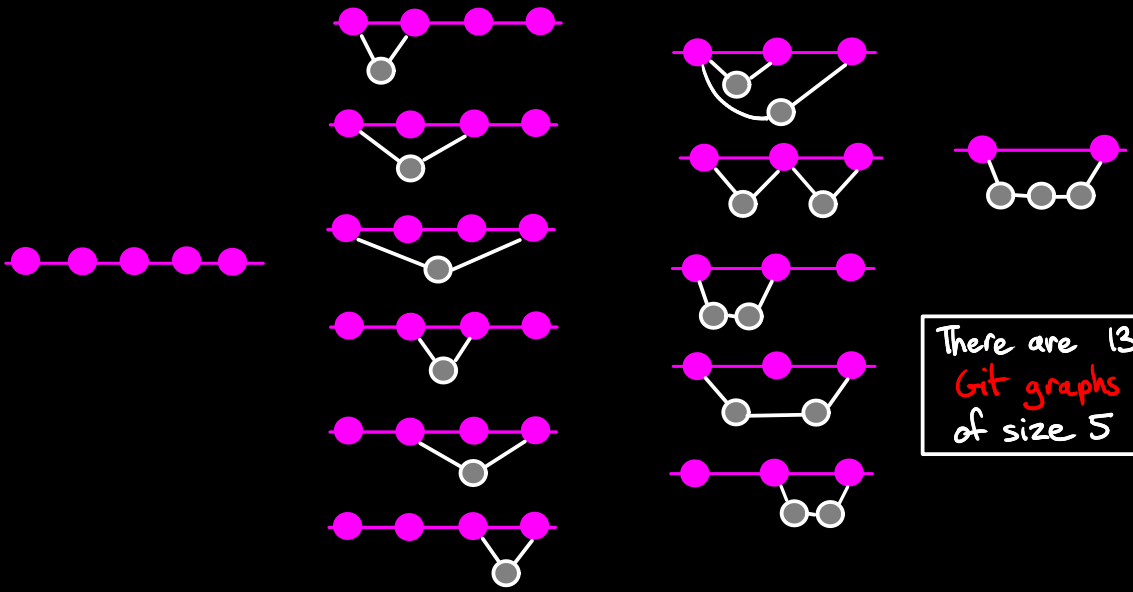
e.g.



ILLUSTRATED RULES

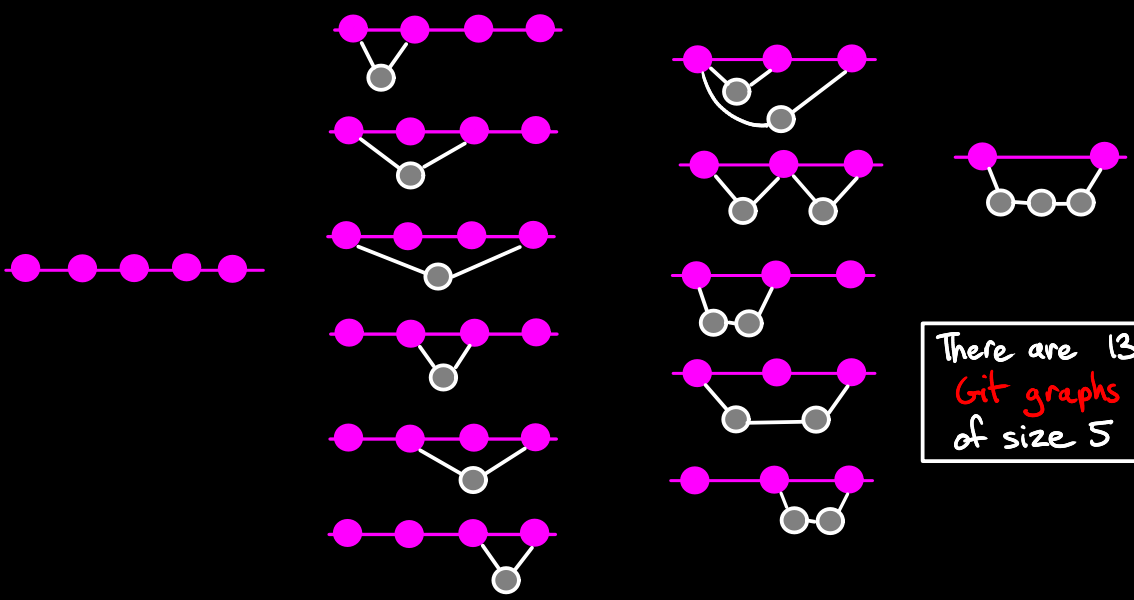


ENUMERATING & SAMPLING



There are 13 Git graphs of size 5

ENUMERATING & SAMPLING

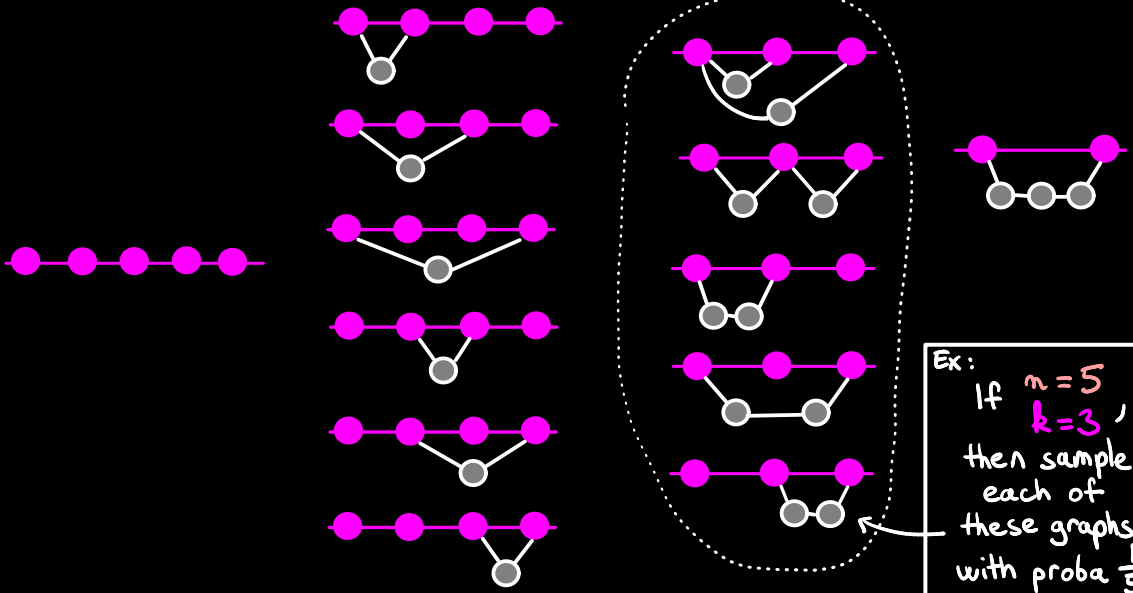


There are 13 Git graphs of size 5

GOAL

Sample a Git graph uniformly at random
given a size n and a number k of magenta vertices

ENUMERATING & SAMPLING



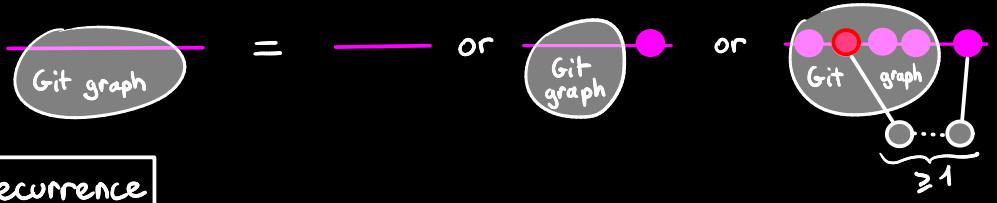
Ex: If $n=5$, $k=3$, then sample each of these graphs with proba $\frac{1}{5}$

GOAL

Sample a Git graph uniformly at random
given a size n and a number k of magenta vertices

Decomposition

RECURSIVE DECOMPOSITION



Recurrence

$$g_{n,k} = g_{n-1,k-1} + \sum_{l \geq 1} (k-1)\, g_{n-1-l,k-1} \qquad \text{for } n \geq 1$$

where $g_{n,k}$:= number of Git graphs with n vertices, k of them being magenta

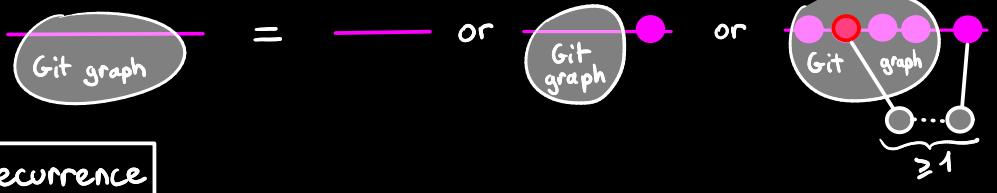
Differential Equation for the Generating Function

$$G(z,u) = 1 + zu\,G(z,u) + \frac{z^2u^2}{1-z}\frac{\partial G}{\partial u}(z,u)$$

where $G(z,u) = \sum_{n \geq 0} \sum_{k \geq 0} g_{n,k}\, z^n u^k$

RECURSIVE DECOMPOSITION

Decomposition



Recurrence

$$g_{n,k} = g_{n-1,k-1} + \sum_{l \geq 1} (k-1)\, g_{n-1-l,k-1} \qquad \text{for } n \geq 1$$

where $g_{n,k}$:= number of Git graphs with n vertices, k of them being magenta

Differential Equation for the Generating Function

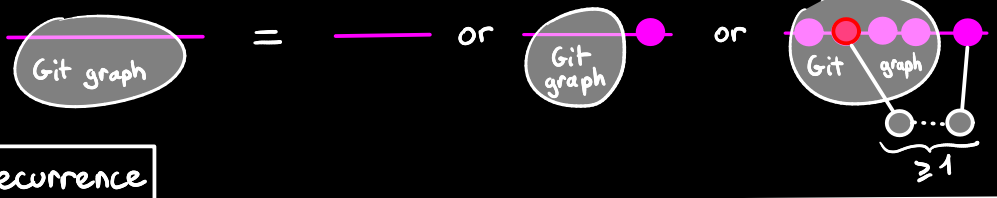
$$G(z,u) = 1 + zu\,G(z,u) + \frac{z^2u^2}{1-z}\frac{\partial G}{\partial u}(z,u)$$

where $G(z,u) = \sum_{n \geq 0} \sum_{k \geq 0} g_{n,k}\, z^n u^k$

→ possible to write a recursive generator from this but it is inefficient

RECURSIVE DECOMPOSITION

Decomposition



Recurrence

$$g_{n,k} = g_{n-1,k-1} + \sum_{\ell \geq 1} (k-1)\, g_{n-1-\ell,k-1} \qquad \text{for } n \geq 1$$

where $g_{n,k}$:= number of Git graphs with n vertices, k of them being magenta

Differential Equation for the Generating Function

$$G(z,u) = 1 + zu\,G(z,u) + \frac{z^2u^2}{1-z}\frac{\partial G}{\partial u}(z,u)$$

where $G(z,u) = \sum_{n\geq 0}\sum_{k\geq 0} g_{n,k}\, z^n u^k$

→ possible to write a recursive generator from this but it is inefficient

→ Bolzmann sampling impossible (?) since $G(z,u)$ not analytic

TRANSFORMING THE EQUATION

Recurrence	$g_{m,k} = g_{m-1,k-1} + \sum_{l \geq 1} (k-1)\, g_{m-1-l,k-1}$
Differential Equation	$G(z,u) = 1 + zu\,G(z,u) + \frac{z^2u^2}{1-z}\frac{\partial G}{\partial u}(z,u)$

Usual trick:

Ordinary Generating Function $\underbrace{\sum_{m,k \geq 0} g_{m,k} z^m u^k}_{G(z,u),\ \text{not analytic}\ \times}$

TRANSFORMING THE EQUATION

Recurrence	$g_{n,k} = g_{n-1,k-1} + \sum_{l \geq 1} (k-1)\, g_{n-1-l,k-1}$
Differential Equation	$G(z,u) = 1 + zu\,G(z,u) + \frac{z^2u^2}{1-z}\frac{\partial G}{\partial u}(z,u)$

Usual trick:

Ordinary Generating Function: $\underbrace{\sum_{n,k \geq 0} g_{n,k} z^n u^k}_{G(z,u),\ \text{not analytic}\ \times}$

Borel transform →

Exponential Generating Function: $\underbrace{\sum_{n,k \geq 0} \frac{g_{n,k}}{n!} z^n u^k}_{\text{analytic}}$

TRANSFORMING THE EQUATION

Recurrence	$g_{n,k} = g_{n-1,k-1} + \sum_{l \geq 1} (k-1)\, g_{n-1-l,k-1}$
Differential Equation	$G(z,u) = 1 + zu\,G(z,u) + \frac{z^2u^2}{1-z}\frac{\partial G}{\partial u}(z,u)$

Usual trick:

Ordinary Generating Function $\sum_{n,k \geq 0} g_{n,k} z^n u^k$ $= G(z,u)$, not analytic ✗

Borel transform

Exponential Generating Function $\sum_{n,k \geq 0} \frac{g_{n,k}}{n!} z^n u^k$ analytic, but no pretty equation ✗

TRANSFORMING THE EQUATION

Recurrence	$g_{n,k} = g_{n-1,k-1} + \sum_{l \geq 1} (k-1)\, g_{n-1-l,k-1}$
Differential Equation	$G(z,u) = 1 + zu\,G(z,u) + \frac{z^2u^2}{1-z}\frac{\partial G}{\partial u}(z,u)$

Usual trick:

Ordinary Generating Function: $\underbrace{\sum_{n,k \geq 0} g_{n,k} z^n u^k}_{G(z,u),\ \text{not analytic}}$ ✗

Borel transform

Exponential Generating Function: $\underbrace{\sum_{n,k \geq 0} \frac{g_{n,k}}{n!} z^n u^k}_{\text{analytic, but no pretty equation}}$ ✗

Borel transform on u

$$\tilde{G}(z,u) = \sum_{n,k \geq 0} \frac{g_{n,k}}{k!} z^n u^k$$

TRANSFORMING THE EQUATION

Recurrence	$g_{n,k} = g_{n-1,k-1} + \sum_{l \geq 1} (k-1)\, g_{n-1-l,k-1}$
Differential Equation	$G(z,u) = 1 + zu\,G(z,u) + \frac{z^2u^2}{1-z}\frac{\partial G}{\partial u}(z,u)$

Usual trick:

Ordinary Generating Function: $\underbrace{\sum_{n,k\geq 0} g_{n,k} z^n u^k}_{G(z,u),\ \text{not analytic}}$ ✗

Borel transform

Exponential Generating Function: $\underbrace{\sum_{n,k\geq 0} \frac{g_{n,k}}{n!} z^n u^k}_{\text{analytic, but no pretty equation}}$ ✗

Borel transform on u

$$\tilde{G}(z,u) = \sum_{n,k\geq 0} \frac{g_{n,k}}{k!} z^n u^k$$

analytic ✓

and

Differential Equation for $\tilde{G}$
$\frac{\partial \tilde{G}}{\partial u} = z\tilde{G} + \frac{z^2 u}{1-z}\frac{\partial \tilde{G}}{\partial u}$ ✓

TRANSFORMING THE EQUATION

$\tilde{G}(z, u)$ = Exponential (in u) Generating Function of Git graphs

z counts vertices u counts magenta vertices

$$\tilde{G}(z,u) = \sum_{m,k \geq 0} g_{m,k} \frac{z^m u^k}{k!}$$

analytic ✓

and

Differential Equation for $\tilde{G}$
$\frac{\partial \tilde{G}}{\partial u} = z\tilde{G} + \frac{z^2 u}{1-z} \frac{\partial \tilde{G}}{\partial u}$ ✓

TRANSFORMING THE EQUATION

$\tilde{G}(z, u)$ = Exponential (in u) Generating Function of Git graphs

z counts vertices u counts magenta vertices

$$\tilde{G}(z,u) = \sum_{m,k \geq 0} g_{m,k} \frac{z^m u^k}{k!}$$

analytic ✓

and

Differential Equation for $\tilde{G}$

$$\frac{\partial \tilde{G}}{\partial u} = z\tilde{G} + \frac{z^2 u}{1-z}\frac{\partial \tilde{G}}{\partial u}$$

✓

this can be solved!

TRANSFORMING THE EQUATION

$\tilde{G}(z_g, u)$ = Exponential (in u) Generating Function of Git graphs

z_g counts vertices u counts magenta vertices

$$\tilde{G}(z_g, u) = \sum_{n,k \geq 0} g_{n,k} \frac{z_g^n u^k}{k!}$$

analytic ✓

and

Differential Equation for $\tilde{G}$

$$\frac{\partial \tilde{G}}{\partial u} = z_g \tilde{G} + \frac{z_g^2 u}{1 - z_g} \frac{\partial \tilde{G}}{\partial u}$$

✓

this can be solved!

Theorem

$$\tilde{G}(z_g, u) = \left(1 - \frac{z_g^2 u}{1 - z_g}\right)^{-\frac{1 - z_g}{z_g}}$$

TRANSFORMING THE EQUATION

$\tilde{G}(z, u)$ = Exponential (in u) Generating Function of Git graphs

z counts vertices u counts magenta vertices

$$\tilde{G}(z,u) = \sum_{n,k \geq 0} g_{n,k} \frac{z^n u^k}{k!}$$

analytic ✓

and

Differential Equation for $\tilde{G}$

$$\frac{\partial \tilde{G}}{\partial u} = z\,\tilde{G} + \frac{z^2 u}{1-z}\frac{\partial \tilde{G}}{\partial u}$$

✓

this can be solved!

Theorem

$$\tilde{G}(z,u) = \left(1 - \frac{z^2 u}{1-z}\right)^{-\frac{1-z}{z}}$$

How can it be exploited?

RANDOM GENERATION?

RANDOM GENERATION?

"Boltzmann model" (exponential in u, ordinary in z)

Fix $z > 0^*$ and $u > 0^*$.

We wish to sample a Git graph $\mathcal{G}$ with a weight proportional to $z^{\#\text{vertices in } \mathcal{G}} \dfrac{u^{\#\text{magenta vertices in } \mathcal{G}}}{(\#\text{magenta vertices in } \mathcal{G})!}$

(Size is not fixed)

Examples

$\mathbb{P}(\text{—}) \propto 1$ | $\mathbb{P}(\text{—●—}) \propto z\,u$ | $\mathbb{P}(\text{●●●●}) \propto z^5 \frac{u^4}{24}$ | $\mathbb{P}(\text{●●●●}) \propto z^5 \frac{u^3}{6}$

*: in the disk of convergence of $\tilde{G}$

RANDOM GENERATION?

"Boltzmann model" (exponential in u, ordinary in z_γ)

Fix $z_\gamma > 0^*$ and $u > 0^*$.

We wish to sample a Git graph $\mathcal{G}$ with a weight

~~proportional~~ equal to $\dfrac{z_\gamma^{\#\text{vertices in } \mathcal{G}}}{\widetilde{G}(z_\gamma, u)} \dfrac{u^{\#\text{magenta vertices in } \mathcal{G}}}{(\#\text{magenta vertices in } \mathcal{G})!}$

(Size is not fixed)

where $\widetilde{G}(z_\gamma, u) = \sum_{n,k \geq 0} g_{n,k} \dfrac{z_\gamma^n u^k}{k!} = \left(1 - \dfrac{z_\gamma^2 u}{1 - z_\gamma}\right)^{-\frac{1-z_\gamma}{z_\gamma}}$.

Examples

$\mathbb{P}(\text{—}) = \dfrac{1}{\widetilde{G}(z_\gamma, u)}$ $\quad$ $\mathbb{P}(\text{—●—}) = \dfrac{z_\gamma u}{\widetilde{G}(z_\gamma, u)}$ $\quad$ $\mathbb{P}(\text{●—●—●}) = \dfrac{z_\gamma^5}{\widetilde{G}(z_\gamma, u)} \dfrac{u^4}{24}$ $\quad$ $\mathbb{P}(\text{●—●—●}) = \dfrac{z_\gamma^5}{\widetilde{G}(z_\gamma, u)} \dfrac{u^3}{6}$

*: in the disk of convergence of $\widetilde{G}$

CHOOSING z AND u

Proposition

Let γ be a random Git Graph sampled with respect to the previous Boltzmann model, conditioned to have size n

$$\mathbb{E}(\#\text{magenta vertices}(\gamma)) \sim \frac{1-\rho_u}{2-\rho_u} n$$

$$\mathbb{V}(\#\text{magenta vertices}(\gamma)) \sim \frac{\rho_u(1-\rho_u)}{(2-\rho_u)^3} n$$

where $\rho_u = \dfrac{\sqrt{1+4u}-1}{2u}$

Proof : Transfer Theorem from $\tilde{G}(z,u) = \left(1 - \frac{z^2 u}{1-z}\right)^{-\frac{1-z}{z}}$

CHOOSING z_γ AND u

Proposition

Let γ be a random Git Graph sampled with respect to the previous Boltzmann model, conditioned to have size n

$$\mathbb{E}(\#\text{magenta vertices}(\gamma)) \sim \frac{1-\rho_u}{2-\rho_u} n$$

$$\mathbb{V}(\#\text{magenta vertices}(\gamma)) \sim \frac{\rho_u(1-\rho_u)}{(2-\rho_u)^3} n$$

where $\rho_u = \dfrac{\sqrt{1+4u}-1}{2u}$

Proof : Transfer Theorem from $\tilde{G}(z_\gamma, u) = \left(1 - \dfrac{z_\gamma^2 u}{1-z_\gamma}\right)^{-\frac{1-z_\gamma}{z_\gamma}}$

Consequence : Given any $\alpha \in (0, \frac{1}{2})$, and $n \geq 0$, we can tune u to target αn magenta vertices and then z_γ to target size n.

Different Perspective

Is there a combinatorial explanation for the formula

$$\widetilde{G}(z_\gamma, u) = \left(1 - \frac{z_\gamma^2 u}{1 - z_\gamma}\right)^{-\frac{1 - z_\gamma}{z_\gamma}} ?$$

Different Perspective

Is there a combinatorial explanation for the formula

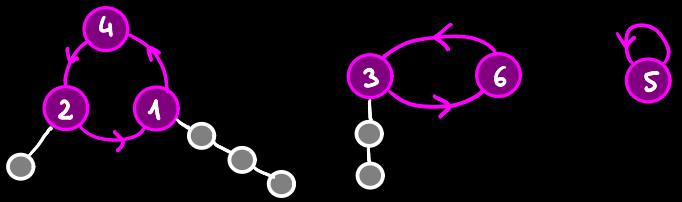
$$\widetilde{G}(z_\delta, u) = \left(1 - \frac{z_\delta^2 u}{1 - z_\delta}\right)^{-\frac{1-z_\delta}{z_\delta}} = \exp\left(\frac{1}{\frac{z_\delta}{1-z_\delta}} \ln\left(\frac{1}{1 - u z_\delta \frac{z_\delta}{1-z_\delta}}\right)\right)?$$

DIFFERENT PERSPECTIVE

Definition

cyclarium = set of cycles of magenta vertices labeled from 1 to k where a chain of white unlabeled vertices is attached to each magenta vertex, except to the ones having the largest label in their cycles.

e.g.:



Is there a combinatorial explanation for the formula

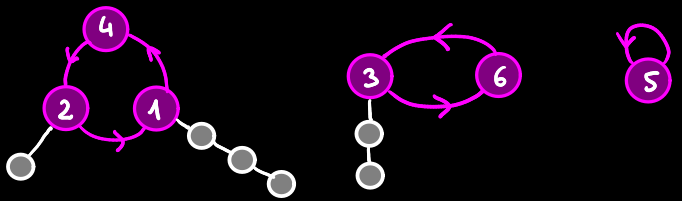
$$\widetilde{G}(z_0, u) = \left(1 - \frac{z_0^2 u}{1 - z_0}\right)^{-\frac{1-z_0}{z_0}} = \exp\left(\frac{1}{\frac{z_0}{1-z_0}} \ln\left(\frac{1}{1 - u z_0 \frac{z_0}{1-z_0}}\right)\right) ?$$

DIFFERENT PERSPECTIVE

Definition

cyclarium = set of cycles of magenta vertices labeled from 1 to k where a chain of white unlabeled vertices is attached to each magenta vertex, except to the ones having the largest label in their cycles.

e.g.:



Is there a combinatorial explanation for the formula

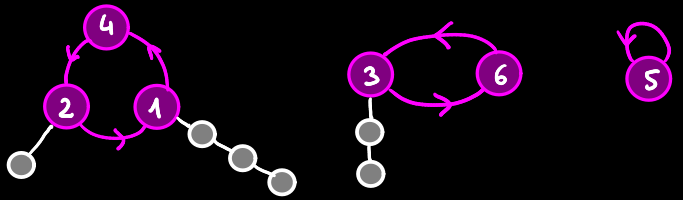
$$\widetilde{G}(z_0, u) = \left(1 - \frac{z_0^2 u}{1 - z_0}\right)^{-\frac{1-z_0}{z_0}} = \exp\left(\frac{1}{\frac{z_0}{1-z_0}} \ln\left(\frac{1}{1 - u z_0 \frac{z_0}{1-z_0}}\right)\right) ?$$

It's the generating function of cyclariums!

BIJECTION

Proposition

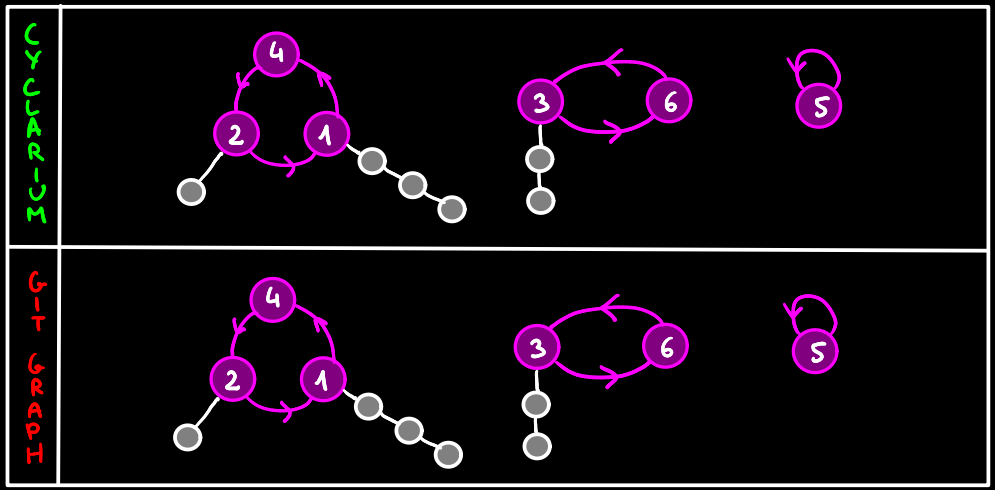
There is a bijection from cyclariums to Git graphs.



BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs.



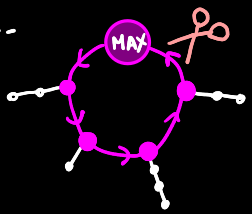
BIJECTION

Proposition

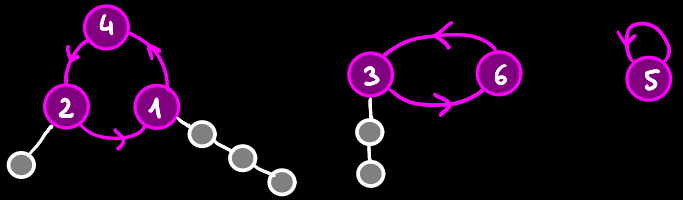
There is a bijection from cyclariums to Git graphs:

STEPS

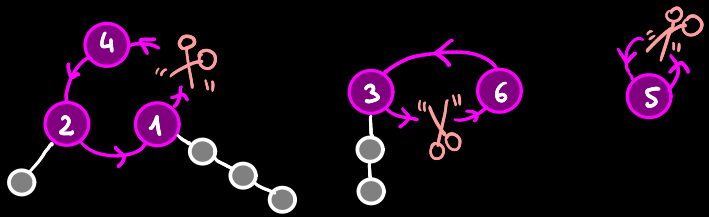
1-



CYCLARIUM



GIT GRAPH



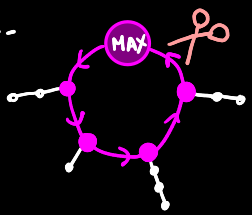
BIJECTION

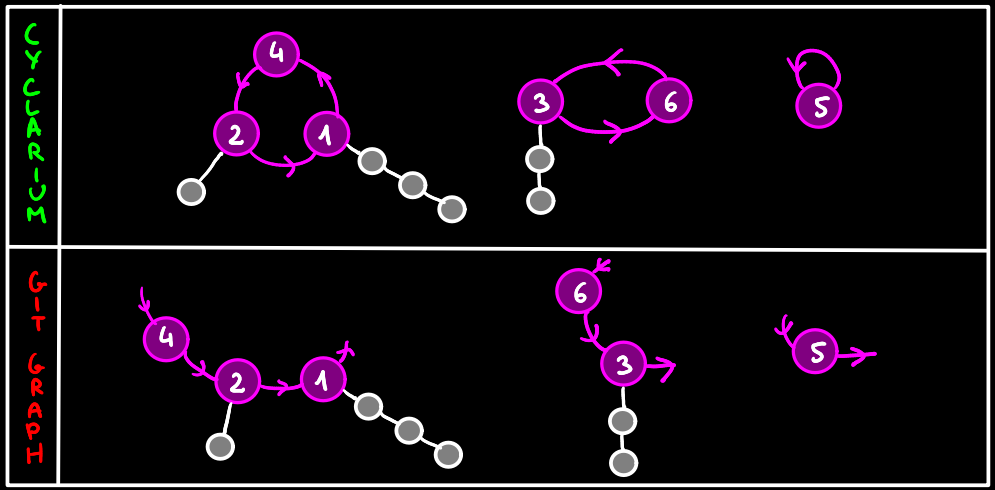
Proposition

There is a bijection from cyclariums to Git graphs:

STEPS

1-





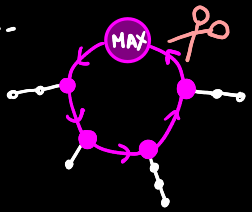
BIJECTION

Proposition

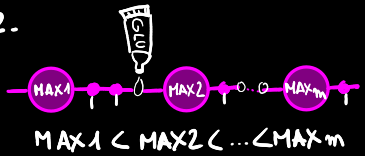
There is a bijection from cyclariums to Git graphs:

STEPS

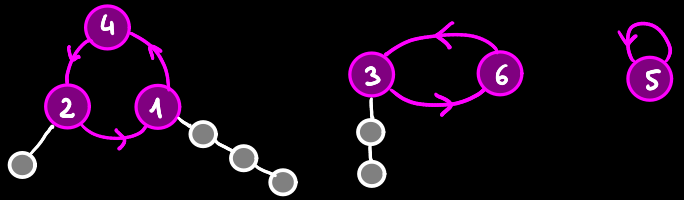
1.



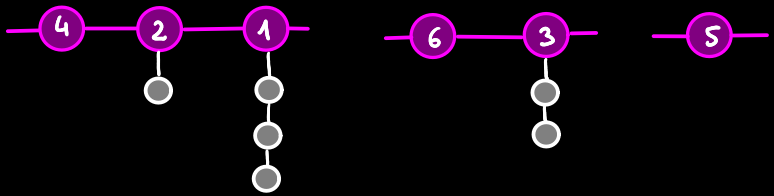
2.



CYCLARIUM



GIT GRAPH



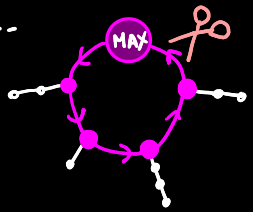
BIJECTION

Proposition

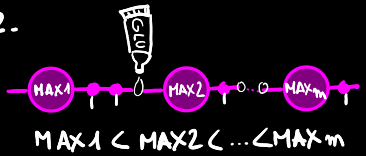
There is a bijection from cyclariums to Git graphs:

STEPS

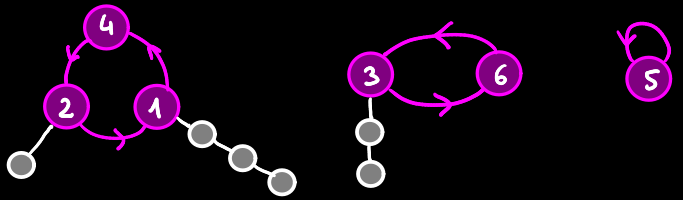
1.



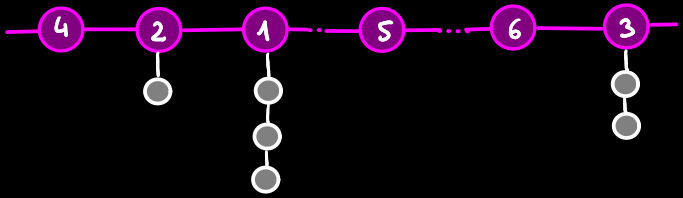
2.



CYCLARIUM



GIT GRAPH



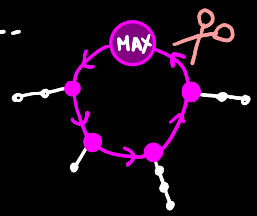
BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs:

STEPS

1.

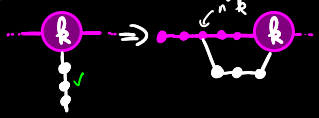


2.

MAX1 < MAX2 < ... < MAXm

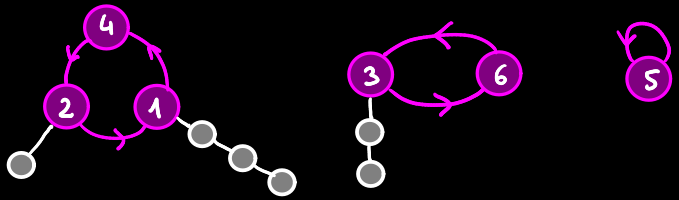
3. Right to left:

3a.

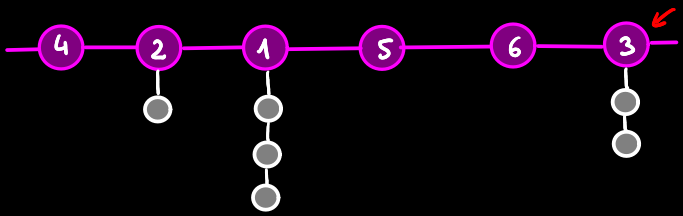


3b. Label removing + shift

CYCLARIUM



GIT GRAPH



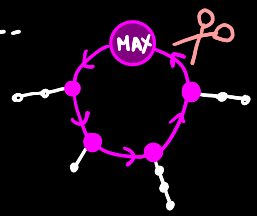
BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs:

STEPS

1.

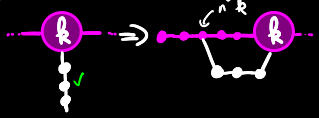


2.

MAX1 < MAX2 < ... < MAXm

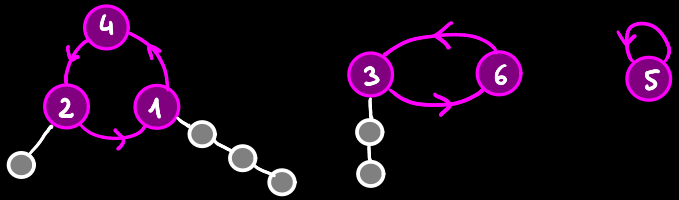
3. Right to left:

3a.

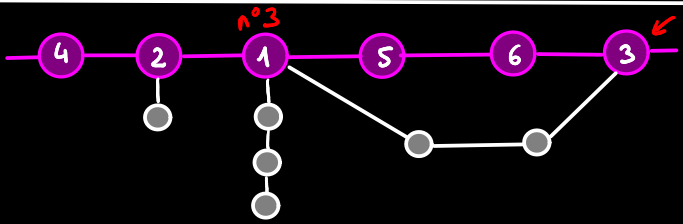


3b. Label removing + shift

CYCLARIUM



GIT GRAPH



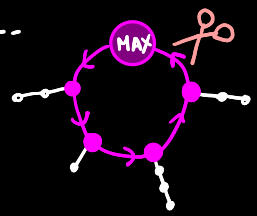
BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs:

STEPS

1.

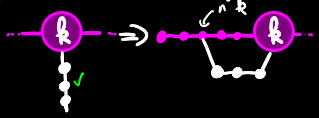


2.

MAX1 < MAX2 < ... < MAXm

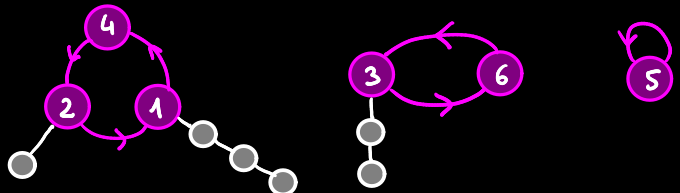
3. Right to left:

3a.

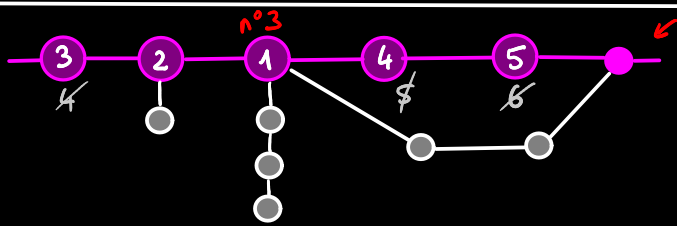


3b. Label removing + shift

CYCLARIUM



GIT GRAPH



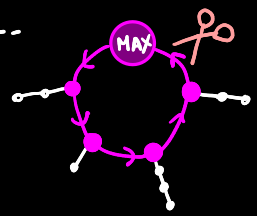
BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs:

STEPS

1.

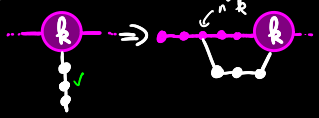


2.

MAX1 < MAX2 < ... < MAXm

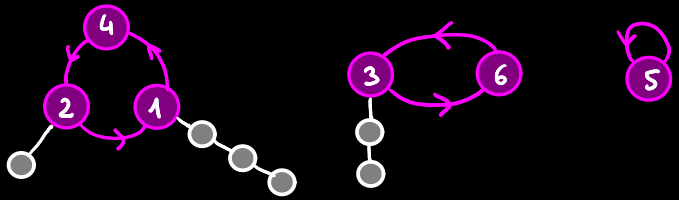
3. Right to left:

3a.

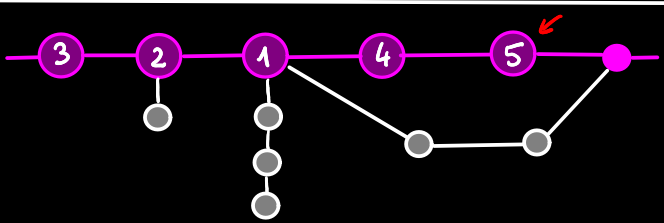


3b. Label removing + shift

CYCLARIUM



GIT GRAPH

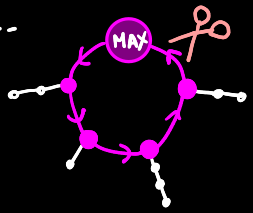


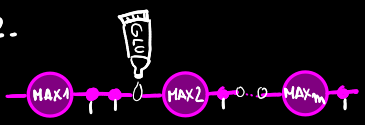
BIJECTION

Proposition

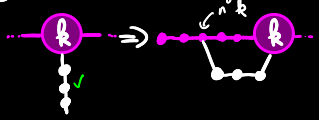
There is a bijection from cyclariums to Git graphs:

STEPS

1. 

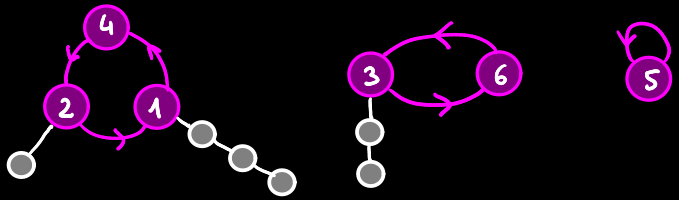
2. 

3. Right to left:

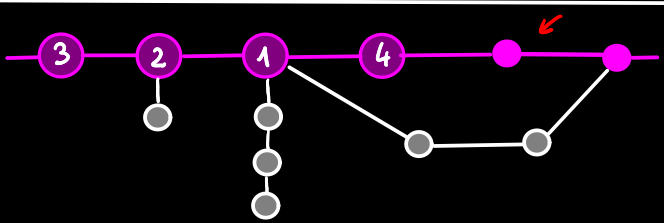
3a. 

3b. Label removing + shift

CYCLARIUM



GIT GRAPH

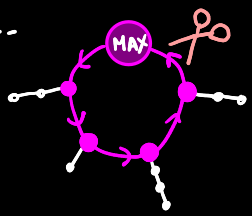


BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs:

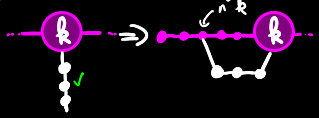
STEPS

1. 

2. 

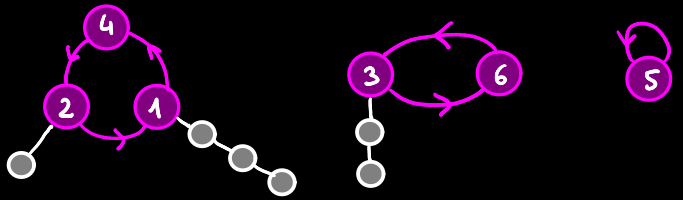
MAX1 < MAX2 < ... < MAXm

3. Right to left:

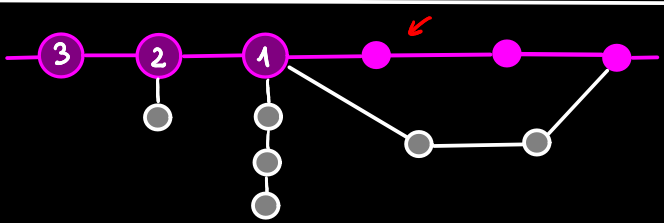
3a. 

3b. Label removing + shift

CYCLARIUM



GIT GRAPH



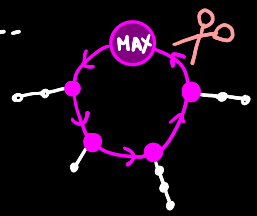
BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs:

STEPS

1.

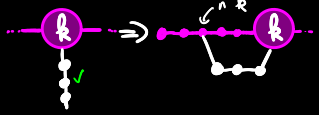


2.

MAX1 < MAX2 < ... < MAXm

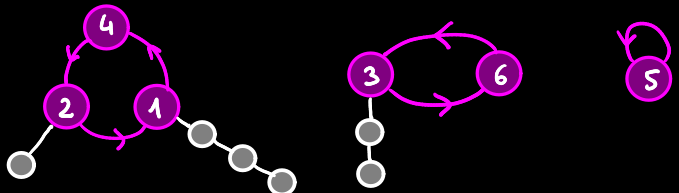
3. Right to left:

3a.

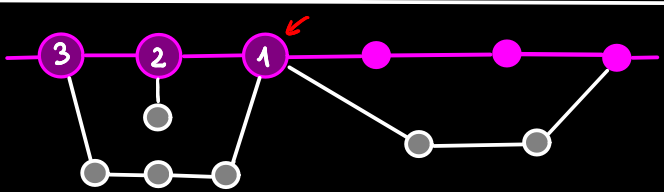


3b. Label removing + shift

CYCLARIUM



GIT GRAPH



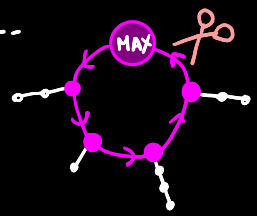
BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs:

STEPS

1.

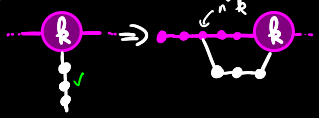


2.

MAX1 < MAX2 < ... < MAXm

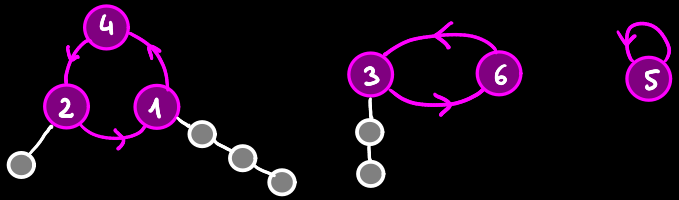
3. Right to left:

3a.

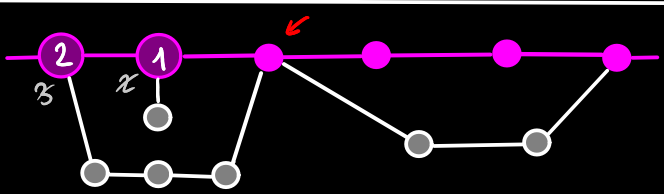


3b. Label removing + shift

CYCLARIUM



GIT GRAPH



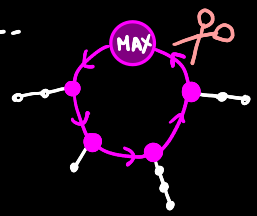
BIJECTION

Proposition

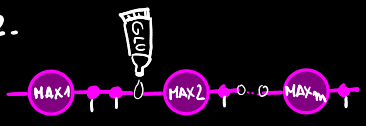
There is a bijection from cyclariums to Git graphs:

STEPS

1.

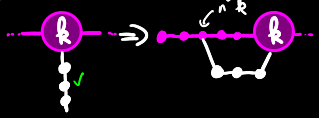


2.



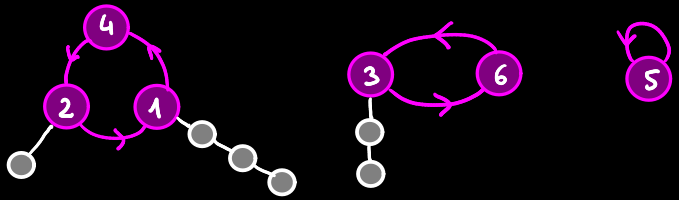
3. Right to left:

3a.

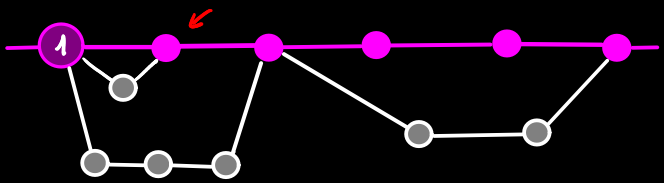


3b. Label removing + shift

CYCLARIUM



GIT GRAPH



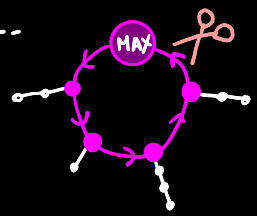
BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs:

STEPS

1.

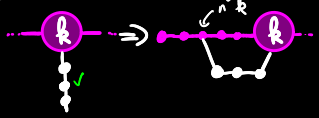


2.

MAX1 < MAX2 < ... < MAXm

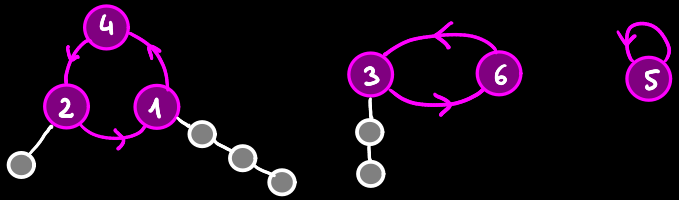
3. Right to left:

3a.

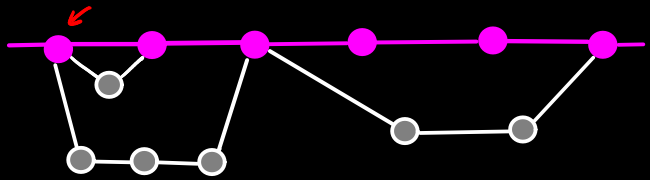


3b. Label removing + shift

CYCLARIUM



GIT GRAPH

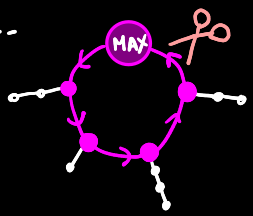


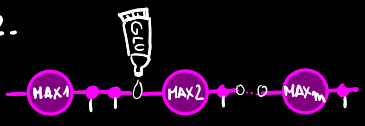
BIJECTION

Proposition

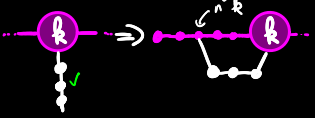
There is a bijection from cyclariums to Git graphs:

STEPS

1. 

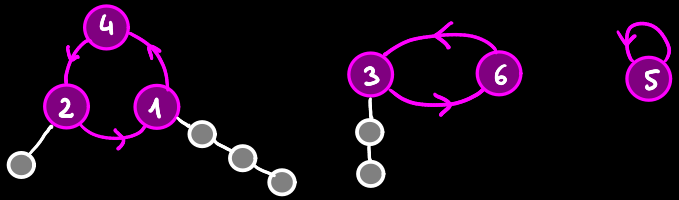
2. 

3. Right to left:

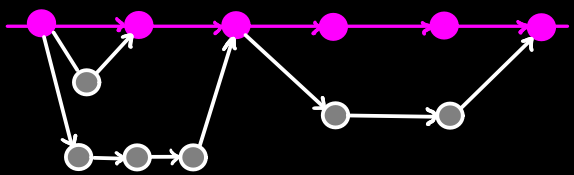
3a. 

3b. Label removing + shift

CYCLARIUM



GIT GRAPH



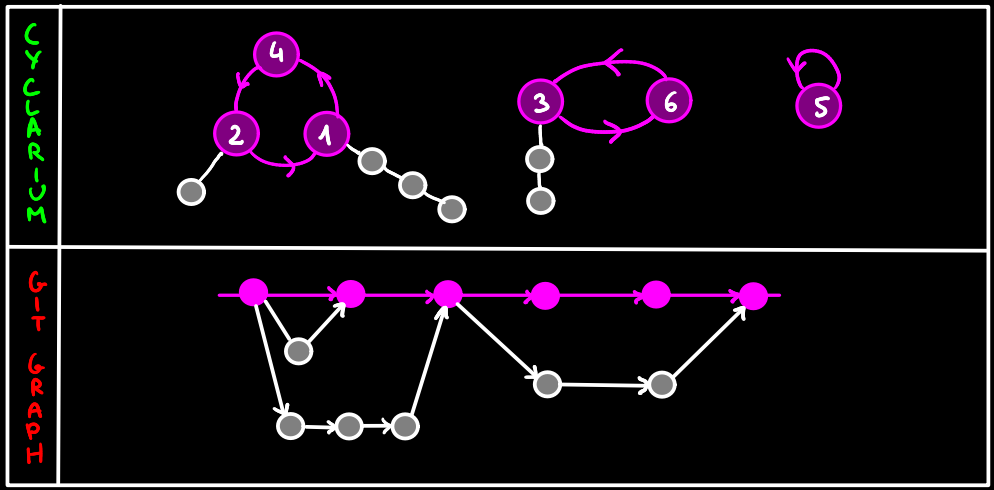
BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs: sending

vertices ⟶ vertices

magenta vertices ⟶ magenta vertices

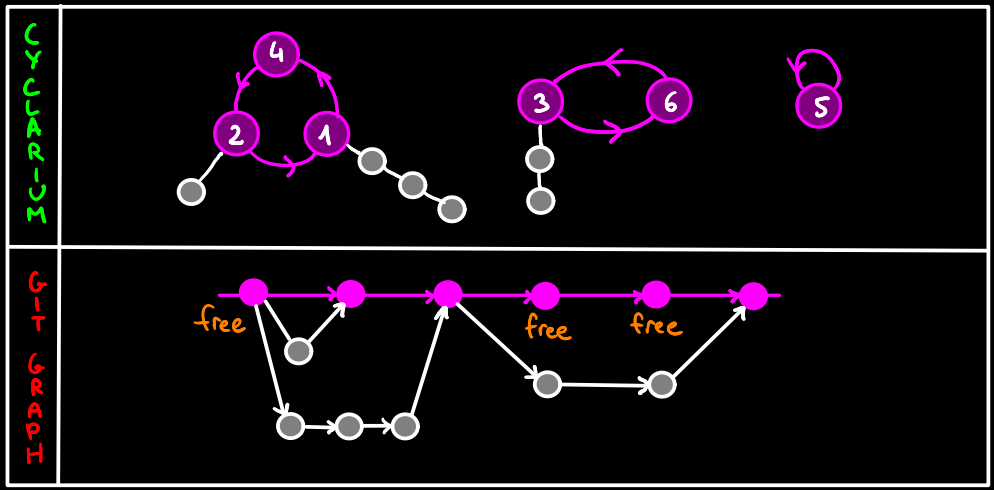


BIJECTION

Proposition

There is a bijection from cyclariums to Git graphs: sending

- vertices ⟶ vertices
- magenta vertices ⟶ magenta vertices
- cycles ⟶ free vertices / i.e magenta vertices of indegree ≤ 1
- cycle lengths ⟶ gaps between free vertices



A NICE FORMULA

Proposition

There is a bijection from cyclariums to Git graphs:
sending

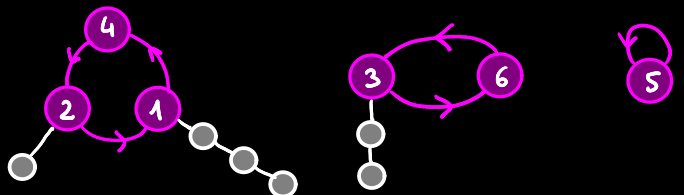
vertices ⟶ vertices

magenta vertices ⟶ magenta vertices

cycles ⟶ free vertices
i.e magenta vertices of indegree ≤ 1

cycle lengths ⟶ gaps between free vertices

CYCLARIUM



Corollary

$$g_{n,k} = \sum_{f=1}^{k-1} \begin{bmatrix} k \\ f \end{bmatrix} \binom{n-k-1}{k-f-1} \qquad (k < n)$$

where $g_{n,k}$ = number of Git graphs counted by vertices & magenta vertices
and $\begin{bmatrix} : \\ : \end{bmatrix}$ = (unsigned) Stirling number of 1st kind

BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

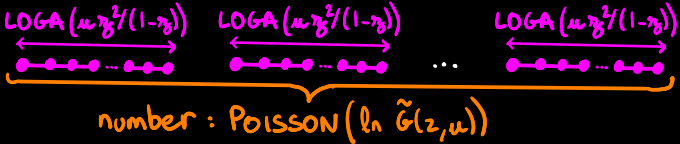
Output: A Git graph $\mathcal{G}$ with proba $z^{\text{size}(\mathcal{G})}\, u^{\#\text{magental}(\mathcal{G})} / (\#\text{magenta}(\mathcal{G})!\, \widetilde{G}(z,u))$

BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

Output: A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

Step 1. Sample



Ex:

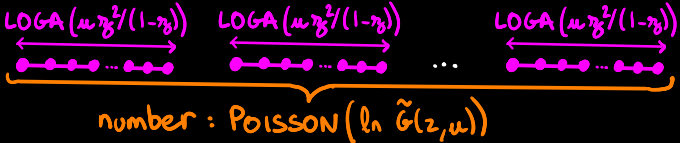
POISSON = 3

BOLTZMANN SAMPLER

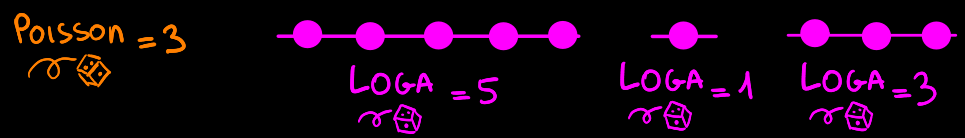
Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

Step 1. Sample



Ex :

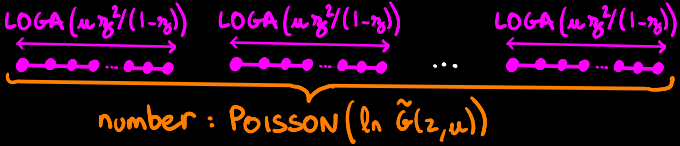


BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

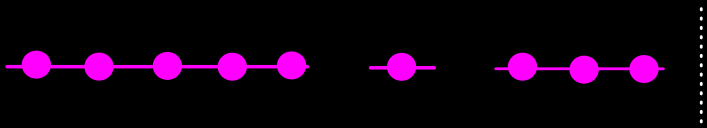
Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \widetilde{G}(z,u))$

Step 1. Sample



Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Ex :



BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

Step 1. Sample

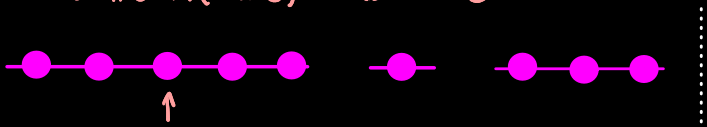
$\text{LOGA}(u z^2/(1-z))$ $\text{LOGA}(u z^2/(1-z))$ ... $\text{LOGA}(u z^2/(1-z))$

number : $\text{POISSON}(\ln \tilde{G}(z,u))$

Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Ex :

$\text{UNIFORM}(1 \dots 9)$ = 3

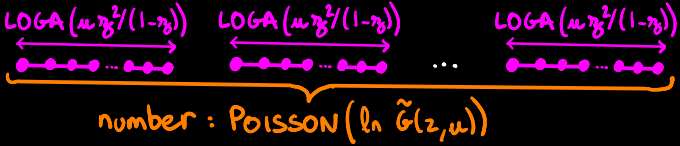


BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

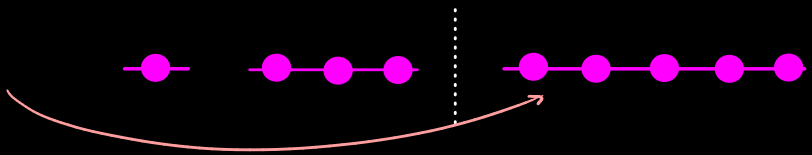
Output: A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

Step 1. Sample



Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Ex:

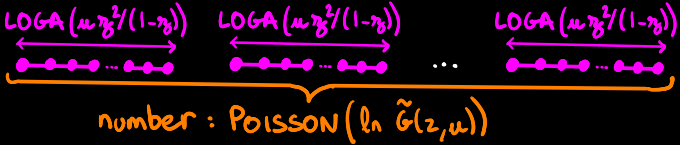


BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

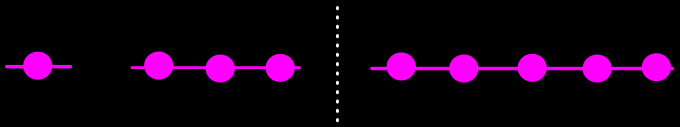
Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \widetilde{G}(z,u))$

Step 1. Sample



Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Ex :

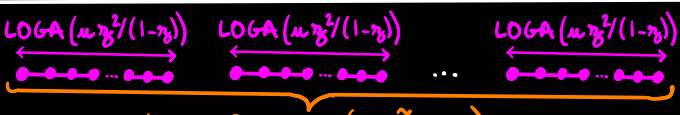


BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

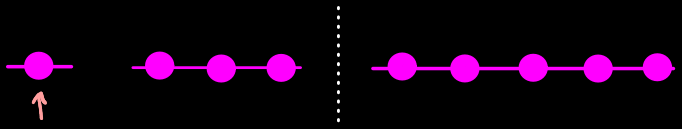
Step 1. Sample



Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Ex :

UNIFORM(1...4) = 1

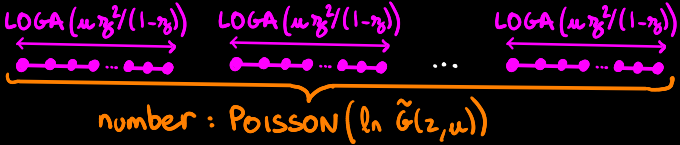


BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \widetilde{G}(z,u))$

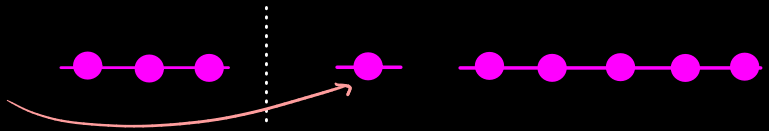
Step 1. Sample



Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Ex :

UNIFORM(1...4) = 1

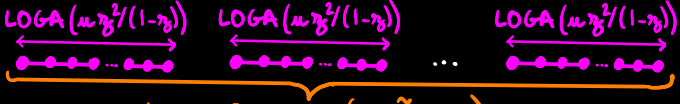


BOLTZMANN SAMPLER

Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

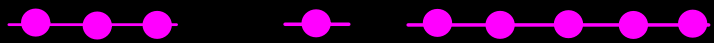
Output: A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

Step 1. Sample



Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Ex:

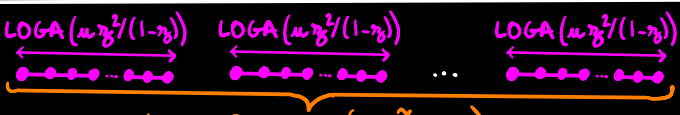


BOLTZMANN SAMPLER

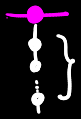
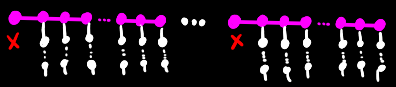
Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

Step 1. Sample

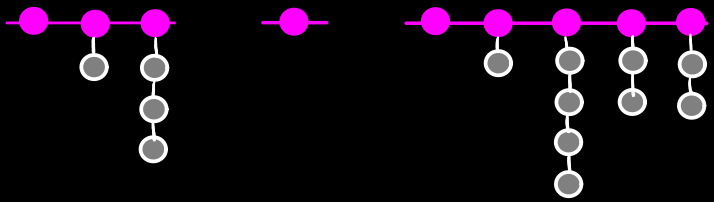


Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Step 3 -  $1 + \text{GEOM}(z)$ for 

Ex :

GEOM  x6

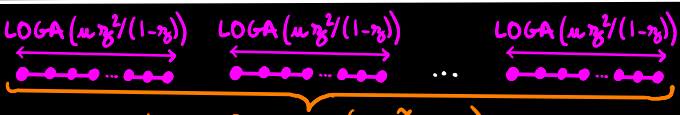


BOLTZMANN SAMPLER

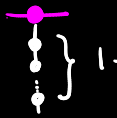
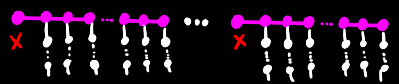
Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

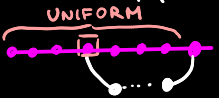
Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

Step 1. Sample

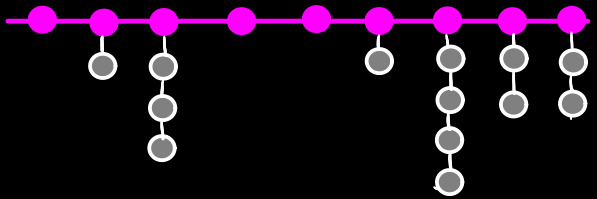


Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Step 3.  $1 + \text{GEOM}(z)$ for 

Step 4. Glue + for each  , 

Ex:

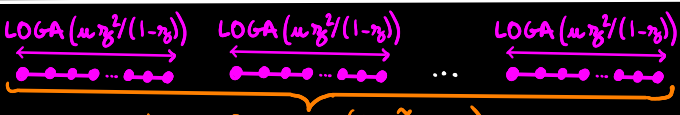


BOLTZMANN SAMPLER

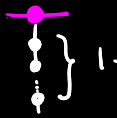
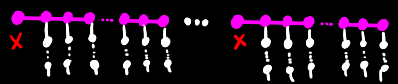
Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

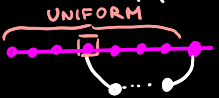
Output : A Git graph γ with proba $z^{\text{size}(\gamma)}\, u^{\#\text{magenta}(\gamma)} / (\#\text{magenta}(\gamma)!\, \tilde{G}(z,u))$

Step 1. Sample

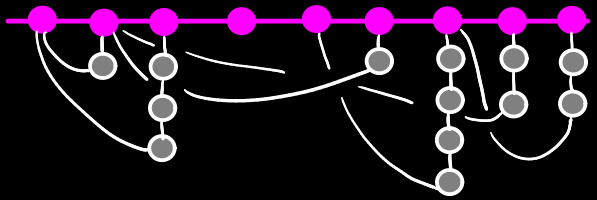


Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Step 3.  $1 + \text{GEOM}(z)$ for 

Step 4. Glue t for each  , 

Ex:

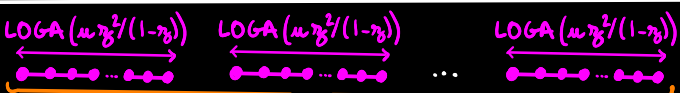


BOLTZMANN SAMPLER

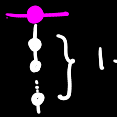
Input $z > 0$ and $u > 0$, tuned to target a size n & a # magenta vertices k

Output : A Git graph δ with proba $z^{\text{size}(\delta)}\, u^{\#\text{magenta}(\delta)} / (\#\text{magenta}(\delta)!\, \tilde{G}(z,u))$

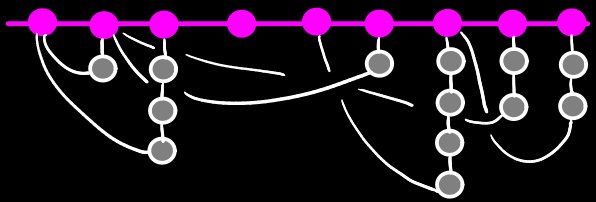
Step 1. Sample



Step 2. Biased shuffle (Pick a • unif. at random & put the segment on the right & repeat)

Step 3.  $1 + \text{GEOM}(z)$ for 

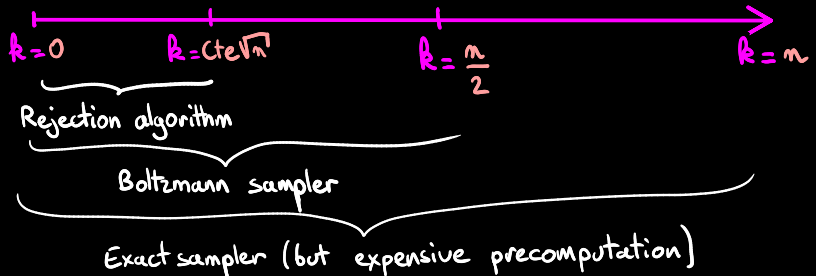
Step 4. Glue + for each  , 

Ex: 

COMPLEXITY: $O(n)$ (in average)

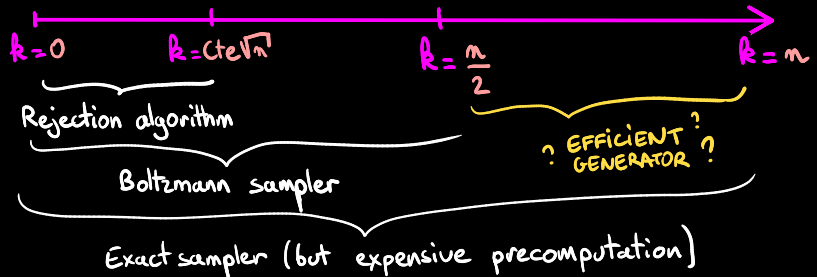
CONCLUSION

Sampling with respect to ratio $\frac{\#\text{ magenta vertices}}{\text{size}}$



CONCLUSION

Sampling with respect to ratio # magenta vertices / size

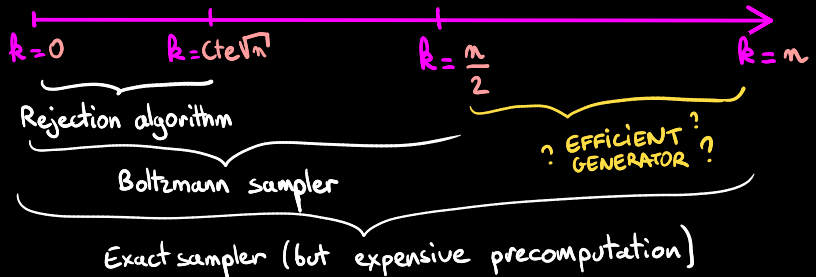


PERSPECTIVES →

→

CONCLUSION

Sampling with respect to ratio # magenta vertices / size



PERSPECTIVES

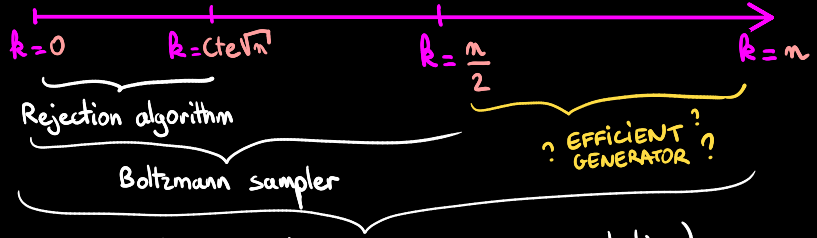
→ Asymptotic behaviour

- Asymptotic equivalent of # Git graphs of size n?
- Limit Laws
- Phase transition?

→

CONCLUSION

Sampling with respect to ratio # magenta vertices / size

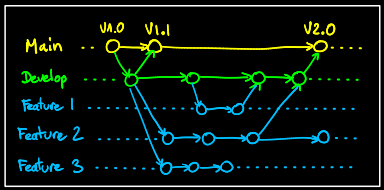


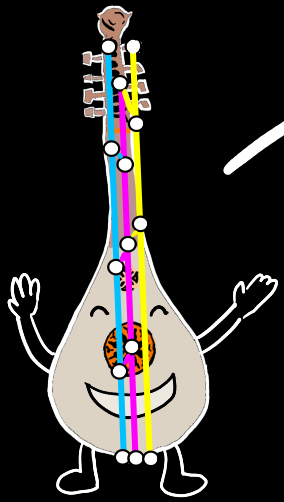
PERSPECTIVES

→ Asymptotic behaviour

- Asymptotic equivalent of # Git graphs of size n?
- Limit Laws
- Phase transition?

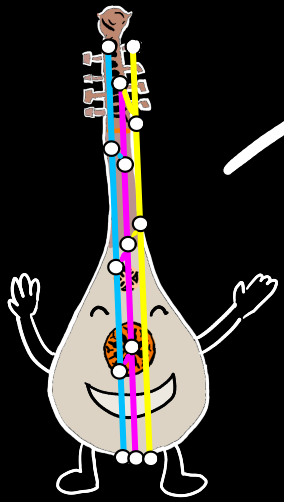
→ Other workflows?





THANK You!

Answer to the riddle :



THANK You!

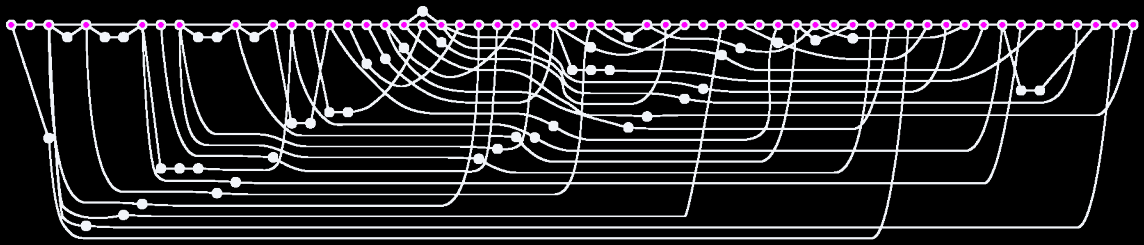
Answer to the riddle:
GITTERN

MOST GIT GRAPHS LOOK ALIKE

Why not just sampling given a size n ?

Theorem

In a Git graph of size n taken uniformly at random, the number of magenta vertices is $\frac{n}{2} + o(n)$



random Git graph of size 100

Sampling a Git graph of size n uniformly means aiming a number of magenta vertices $k \simeq \frac{n}{2}$