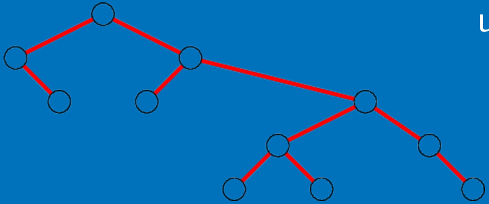
Viviane Pons

Assistant professor, Université Paris-Sud Orsay
viviane.pons@lri.fr – @PyViv

Experimental pure mathematics

using Sage



Viviane Pons
Assistant professor Paris-Sud Orsay
Computer scientist, mathematician
very serious about python
@PyViv

Experimental pure mathematics using Sage

What does pure mathematics look like?

What does pure mathematics look like?

L'équation de Boltzmann,

$$\frac{\partial f}{\partial t}+v\cdot\nabla_x f=\int_{\mathbb{R}^3}\int_{\mathbb{S}^2}|v-v_*|\Big[f(v')f(v'_*)-f(v)\,f(v_*)\Big]\,dv_*d\sigma,$$

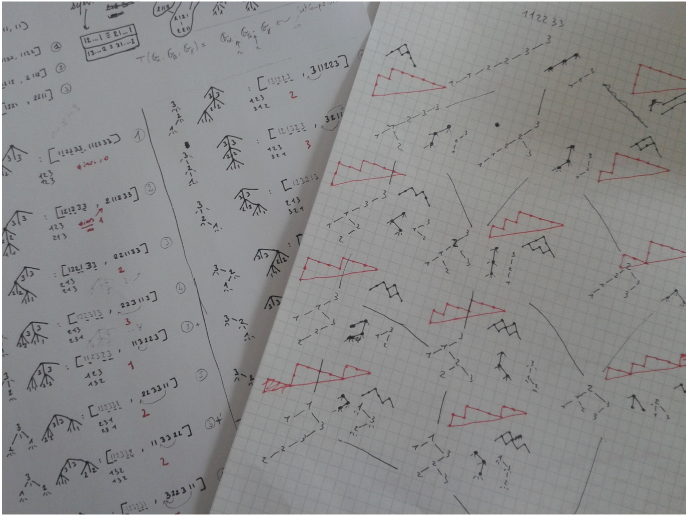
découverte aux alentours de 1870, modélise l'évolution d'un gaz raréfié, fait de milliards de milliards de particules, qui se cognent les unes contre les autres ; on représente la distribution statistique des positions et vitesses de ces particules par une fonction $f(t,x,v)$, qui au temps t indique la densité de particules dont la position est (environ) x et dont la vitesse est (environ) v.

Ludwig Boltzmann découvrit la notion statistique d'entropie, ou désordre, d'un gaz :

$$S=-\iint f\,\log f\,dx\,dv;$$

(from *Théorème Vivant* by Cédric Villani)

What does pure mathematics look like?



What does pure mathematics look like?

```python
def inf_mperms(p1,p2):
    m = len([i for i in p1 if i==1])
    return perm_to_mperm(inf_perms(mperm_to_perm(p1),mperm_to_perm(p2)),m)

def is_last(perm,i):
    for b in perm[i+1:]:
        if b == perm[i]:
            return False
    return True

def mperm_to_tree(perm):
    values = list(set(perm))
    values.sort()
    values.reverse()
    m = len(perm) / len(values)
    tree = MDecreasingTree(m+1,None)
    for v in values:
        tree = tree.insert_from_mperm(perm,v)
    return tree


def mperm_to_tree2(perm, mfor0 = 1):
    if len(perm)==0:
        return MDecreasingTree(mfor0,None)
    n = max(perm)
    posr = [i for i in xrange(len(perm)) if perm[i]==n]
    m = len(posr)
    children = [[] for i in xrange(m+1)]
    right = {a for a in perm if a!=n}
    for i in xrange(m):
        pos = posr[i]
        for j in xrange(pos-1,-1,-1):
            a = perm[j]
            if a!=n:
                if is_last(perm,j):
                    if a in right:
                        children[i].append(a)
                        right.remove(a)
                elif a in right:
                    right.update([aa for aa in children[i] if aa < a])
                    children[i] = [b for b in children[i] if b >a]
    children[-1] = list(right)
    children_trees = [mperm_to_tree2([a for a in perm if a in c], mfor0 =m) for c in children]
    return MDecreasingTree(m+1,children_trees, label=n)
```

What does pure mathematics look like?

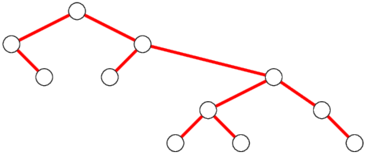
```
AUTHORS:

- Florent Hivert (2010-2011): initial implementation.

REFERENCES:

.. [LodayRonco] Jean-Louis Loday and Maria O. Ronco.
   *Hopf algebra of the planar binary trees*,
   Advances in Mathematics, volume 139, issue 2,
   10 November 1998, pp. 293-309.
   http://www.sciencedirect.com/science/article/pii/S0001870898917595

.. [HNT05] Florent Hivert, Jean-Christophe Novelli, and Jean-Yves Thibon.
   *The algebra of binary search trees*,
   :arxiv:`math/0401089v2`.

.. [CP12] Gregory Chatel, Viviane Pons.
   *Counting smaller trees in the Tamari order*,
   :arxiv:`1212.0751v1`.
"""
#*****************************************************************************
#       Copyright (C) 2010 Florent Hivert <Florent.Hivert@univ-rouen.fr>,
#
#  Distributed under the terms of the GNU General Public License (GPL)
#  as published by the Free Software Foundation; either version 2 of
#  the License, or (at your option) any later version.
#                  http://www.gnu.org/licenses/
#*****************************************************************************
from sage.structure.list_clone import ClonableArray
from sage.combinat.abstract_tree import (AbstractClonableTree,
                                         AbstractLabelledClonableTree)
from sage.combinat.ordered_tree import LabelledOrderedTrees
from sage.rings.integer import Integer
from sage.misc.classcall_metaclass import ClasscallMetaclass
from sage.misc.lazy_attribute import lazy_attribute, lazy_class_attribute
from sage.combinat.combinatorial_map import combinatorial_map

class BinaryTree(AbstractClonableTree, ClonableArray):
    """
    Binary trees.

    Binary trees here mean ordered (a.k.a. plane) finite binary
    trees, where "ordered" means that the children of each node are
    ordered.

    Binary trees contain nodes and leaves, where each node has two
```

What is combinatorics?

We study **mathematical properties** of structures **from computer science**.

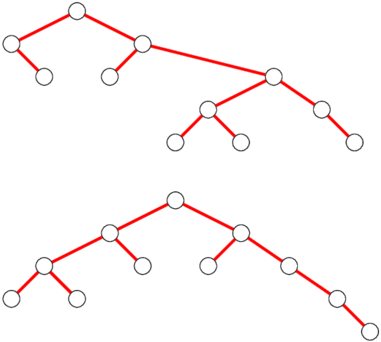
Examples: graphs, trees, binary words, etc.

Example: binary trees



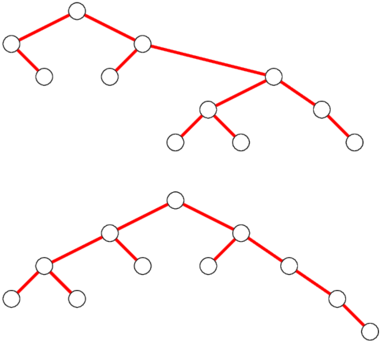
Questions:

Example: binary trees



Questions:

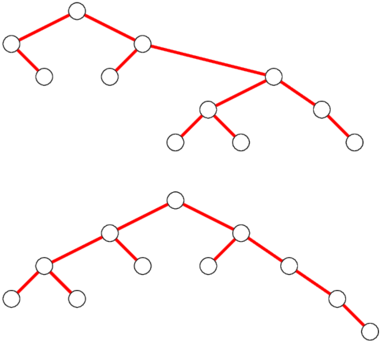
Example: binary trees



Questions:

- How many binary trees with 11 nodes?

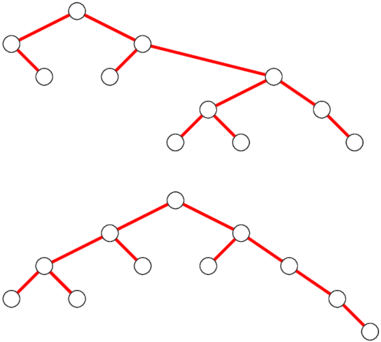
Example: binary trees



Questions:

- How many binary trees with 11 nodes? 58786

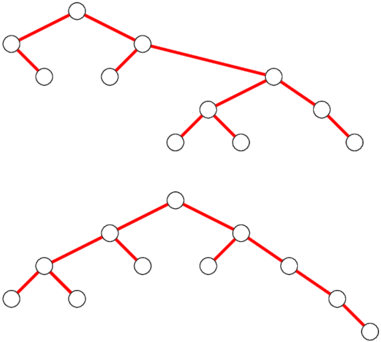
Example: binary trees



Questions:

- How many binary trees with 11 nodes? 58786
- What does a "random" binary tree look like?

Example: binary trees



Questions:

- How many binary trees with 11 nodes? 58786
- What does a "random" binary tree look like?
- Are there other combinatorial objects somehow linked to binary trees?

Download the demo on http://www.lri.fr/~pons

More about Sage

Sage Days 67 at Pycon: Monday – Thursday, at UQAM and PyCon

http://wiki.sagemath.org/days67