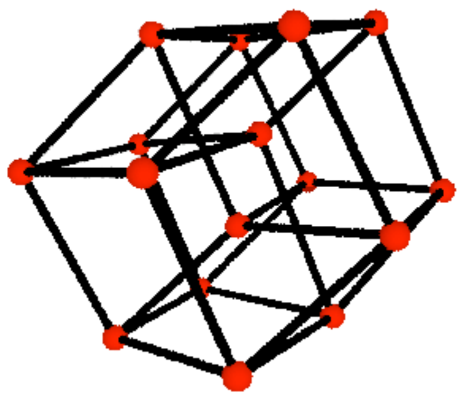
480 - April 28, 2008 -- combinatorics

Combinatorics, part 1

Roughly speaking, **combinatorics** is the study of discrete objects. It involves counting objects with properties, finding optimal objects, understanding algebraic structures on sets of objects, etc. It has deep connections with many other areas of mathematics, and is also heavily used in computer science.

```
show(graphs.CubeGraph(4).plot3d(), spin=True, viewer='tachyon')
```



Mupad-combinat is being actively ported to sage; project led by Mike Hansen. Involvement by Nicolas Theiry, Dan Bump, and others.

Sage Combinatorics

See the combinatorics section of the reference manual for an overview of what is available in Sage. The combinatorics functionality builds on core functionality offered by the following packages.

1. Symmetrica 2.0 is standard in Sage
2. Networkx for graph theory
3. GAP: Group, Algorithms and Programming: ``GAP is used in research and teaching for studying groups and their representations, rings, vector spaces, algebras, **combinatorial structures**, and more."

Combinatorial functions

There are many combinatorial functions in Sage:

This is a small sampling...

```
factorial(7)
```

```
bernoulli(10)
```

```
bell_number(10)
```

```
catalan_number(10)
```

```
euler_number(10)
```

```
fibonacci(100)
```

```
stirling_number1(10,5)
```

```
number_of_partitions(5)
```

```
for p in partitions(5):
   print p
```

```
@interact
def _(n=(2..15)):
   show('<h1>The %s Partitions of
%s</h1>'%(number_of_partitions(n), n))
   for p in partitions(n):
        print p
```

More details: Number of Partitions

Sage is the *best in the world* and computing the number of partitions of an integer. Sage has its own implementation mostly due to Jon Bobber.

```
time p = number_of_partitions(10^7)
```

```
len(str(p))
```

```
@interact
def _(xmax=(10..200)):
   show(plot(lambda x: number_of_partitions(int(x)), 1, xmax))
```

```
# Please read the source code!
#   SAGE_ROOT/devel/sage/sage/combinat/partitions_c.cc
# which implements
# Hans Rademacher, On the Partition Function p(n),
# *      Proceedings of the London Mathematical Society 1938
# *
 http://plms.oxfordjournals.org/cgi/content/citation/s2-43/4/241
```

More details: Bernoulli Numbers

The Bernoulli B_n numbers are defined by the equality

$$\frac{x}{e^x - 1} = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}$$

```
var('x')
f = x/(e^x - 1)
show(f.taylor(x,0,16))
```

So

$$\frac{B_6}{6!} = \frac{1}{30240}$$

hence

$$B_6 = \frac{6!}{30240} = \frac{1}{42}$$

```
factorial(6)/30240
```

```
for n in range(15):
  print '%-4s%-8s'%(n,bernoulli(n))
```

```
@interact
def _(n=(1..30)):
    print "Bernoulli number B_%s =\n\n\n"%(2*n)
    B = bernoulli(2*n)
    show(B)
    show(factor(B))
```

Sage (via PARI) is **very good** at computing Bernoulli numbers (better than anything else):

```
time k = bernoulli(20008)
```

```
len(str(k))
```

```
%mathematica
Timing[BernoulliB[20008];]
```

Compute **all** Bernoulli numbers up to B_{20011} but modulo the prime 20011. This uses a clever algorithm of Buhler et al. that David Harvey implemented.

```
time all = bernoulli_mod_p(20011)
```

```
all[-1]
```

```
bernoulli(2008) % 20011
```

Bernoulli numbers are useful for

1. Giving an explicit closed formula for sums of powers (Bernoulli introduced them for this reason):

$$\sum_{k=1}^{n} k^r = \frac{1}{r+1}\sum_{j=0}^{r}\binom{r+1}{j} B_j n^{r+1-j},$$

   but with $B_1 = 1/2$ instead of $-1/2$ (another convention). E.g., the sums of the squares of the integers with $r = 2$ we get:

$$\sum_{k=1}^{n} k^2 = \frac{1}{3}(B_0 n^3 + 3B_1 n^2 + 3B_2 n + B_3) = \frac{1}{3}(n^3 + \frac{3}{2}n^2 + \frac{1}{2}n) = \frac{2n^3 + 3n^2 + r}{6}$$

``I have found in less than a quarter of an hour that the tenth powers (or the quadrate- sursolids) of the first thousand numbers beginning from 1 added together equal
91,409,924,241,424,243,424,241,924,242,500,
from which it is apparent how useless should be judged the works of Ismael Bullialdus, recorded in the thick volume of his Arithmeticae Infinitorum, where all he accomplishes is to show that

with immense labor he can sum the first six powers–part of what we have done in a single page." -- Bernoulli (somewhere around 1700)

With that salvo, Bernoulli makes no further mention, in his treatise, of the numbers we will be concentrating on, and turns his attention to other things. (Mazur)

2. Recall the famous identity

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

Bernoulli numbers star in a generalization of this formula to arbitrary even powers. The formula is that for integers $k \geq 1$, we have

$$\zeta(2k) = \frac{2^{2k-1}|B_{2k}|\pi^{2k}}{(2k)!}$$

3. Bernoulli numbers also appear as constant terms in the theory of modular forms (fairly advanced) and yield deep information about class groups of number fields.

```
sum(n^10 for n in [1..1000])
```

```
k = 1
print zeta(2*k),
float(2^(2*k-1)*abs(bernoulli(2*k))*pi^(2*k)/factorial(2*k))
```

```
@interact
def _(k=(1..20)):
  a = 2^(2*k-1)*abs(bernoulli(2*k))*pi^(2*k)/factorial(2*k)
  show('2k = %s'%(2*k))
  html('<font size=+1>$$\\sum_{n=1}^{\\infty} \\frac{1}{n^{%s}} =
%s$$</font>'%(2*k, latex(a)))
```

Project Idea: There is a notion of generalized Bernoulli numbers associated to Dirichlet characters and I co-authored an unfinished paper to efficiently compute them. Implementing that algorithm and/or helping finish that paper is possible student project. Another project would be just to reimplement in Cython the PARI algorithm Sage uses to compute `bernoulli(n)`, since honestly the comments in that code do *not* inspire confidence.

See Barry Mazur's nice paper "Bernoulli numbers and the unity of mathematics" available on the math 480 website.

Combinatorial Structures

Sage supports dozens of combinatorial ways to form sets of objects from other sets. These can be very useful in enumeration problems, e.g., if you are a writer trying to think of ideas for a TV show:

```
C = Combinations(['Jim', 'Pam', 'Dwight', 'Angela', 'Michael'], 2)
```

```
C
```

```
len(C)
```

```
C.list()
```

```
# Ordered lists of positive integers that sum to 5.
C = Compositions(5)
C
```

```
C.list()
```

```
len(Compositions(30))
```

```
P = Partitions(5)
P
```

```
len(P)
```

```
P.list()
```

```
for p in P:
   print p
   print ferrers_diagram(p)
   print
```

```
len(Partitions(100))
```

```
p = Permutations(['Jim', 'Pam', 'Dwight'])
p
```

```
list(p)
```

```
len(p)
```

```
S = SetPartitions(['Jim', 'Pam', 'Dwight'])
S
```

```
S.list()
```

```
S = Subsets(['Jim', 'Pam', 'Dwight'])
S
```

```
S.list()
```

```
S.count()
```