Math 640: EXPERIMENTAL MATHEMATICS Spring 2008 (Rutgers University) Webpage

http://sites.math.rutgers.edu/~zeilberg/math640_08.html

Last Update: Oct. 11, 2008.

Added Oct. 11, 2008: Here are the awsome final projects done by the students

---

- Teacher: *Dr. Doron ZEILBERGER ("Dr. Z")*
- Classroom: Allison Road Classroom Building [Busch Campus], IML Room 119 [Inside computer lab]
- Time: Mondays and Thursdays , period 3 (12:00noon-1:20pm)
- "Textbook": handouts.
- Dr. Zeilberger's Office: Hill Center 704 (Phone: (732) 445-1326)
- Dr. Zeilberger's E-mail: zeilberg at math dot rutgers dot edu
- Dr. Zeilberger's Office Hours: MTh 10:00-10:01 and by appointment

Description

Experimental Mathematics used to be considered an oxymoron, but the future of mathematics is in this direction. In addition to learning the philosophy and methodology of this budding field, students will become computer-algebra wizards, and that should be very helpful in whatever mathematical specialty they'll decide to do research in.

We will first learn Maple, and how to program in it. This semester we will explore Automated (symbolic!) Enumeration, that consists of teaching the computer how to find explicit formulas, and/or general algorithms, for enumerating combinatorial objects. But the actual content is not that important, it is mastering the methodology of computer-generated and computer-assisted research that is so crucial for your future.

There are no prerequisites, and no previous programming knowledge is assumed. Also, very little overlap with previous years. The final projects for this class may lead to journal publications.

Here are the suggested final projects .

Diary and Homework

Programs done on Thurs., Jan. 24, 2008

- jan24.txt , a program that inputs an integer p and outputs true or false according to whether or not a^p mod p = a for all a between 1 and p-1.

Homework for Thurs., Jan. 24, class (due Jan. 28, 2008)

1. Read and do all the examples, plus make up similar ones, in the first 30 pages of Frank Garvan's awesome Maple booklet.
2. Write a Maple program that inputs positive integers a and K and prints out all the integers n, less than K that are **not** prime, yet satisfy a^n mod n = a .

Programs done on Mon., Jan. 28, 2008

- jan28.txt , contains the program choose1(S,k) that inputs a set S and a non-negative integer k, and outputs the set of subsets of S with exactly k elements, and Fnk(n,k) the inputs non-negative integers n and k and outputs the set of lists of length k, whose entries are 1 or 2, and that add up to n.

Homework for Mon., Jan. 28, class (due Thurs., Jan. 31, 2008)

- For Newcomers:
  1. Read and do all the examples, plus make up similar ones, in pages 30-60 of Frank Garvan's awesome Maple booklet. Print out a few sample examples, that you made up on your own, and hand them in.
  2. Modify Fnk(n,k) to write a Maple program, call it Gnk(n,k) that inputs non-negative integers n and k and outputs the set of lists of length k, whose entries are from the set {1,2,3}.
  3. Using Fnk(n,k) as a subroutine write a program Fn(n) that inputs a non-negative integer n, and outputs the set of lists whose entries are drawn from {1,2} any length, that sum-op to n.
  4. **Without** using Fnk(n,k) as a subroutine, write a program Fn1(n) that inputs a non-negative integer n, and outputs the set of lists whose entries are drawn from {1,2} any length, that sum-op to n.
  5. What is the sequence nops(Fn(n)), n=1,2,3, ... ?
- For Old-timers: (updated Jan. 29, 2008, correcting errors pointed out by Lara Pudwell)
  1. Make sure that your mentee read and does all the examples, plus make up similar ones, in pages 30-60 of Frank Garvan's awesome Maple booklet. Also, make sure

that they can do the homework. If he or she have any problems, help them, but don't do it for them.
2. Modify Fnk(n,k) to write a Maple program, call it FSnk(S,n,k) that inputs a set of integers, S, and non-negative integers n and k and outputs the set of lists of length k, whose entries are from the set S, and that sum to n.
3. Using FSnk(S,n,k) as a subroutine write a program FSn(S,n) that inputs a set of **positive** integers and a non-negative integer n, and outputs the set of lists whose entries are drawn from the set S, of any length, that sum-op to n.
4. **Without** using FSnk(S,n,k) as a subroutine, write a program FSn1(S,n) that inputs and a set of **positive** integers S, and a non-negative integer n, and outputs the set of lists whose entries are drawn from the set S, of any length, that sum-op to n.
5. What is the (ordinary) generating function for the sequence nops(FSn(S,n)), n=0,1,2,3, ... ?

Programs done on Thurs., Jan. 31, 2008

- jan31.txt , in addition to the previous programs in jan28.txt, contains programs for computing the powerset of a set S, as well as Fn(n), that outputs the set of sequences of 1's and 2's that adds up to n.

Homework for Thurs. Jan. 31, class (due Mon., Feb. 4, 2008)

- For Newcomers:
  1. Read and do all the examples, plus make up similar ones, in pages 61-90 of Frank Garvan's awesome Maple booklet. Print out a few sample examples, that you made up on your own, and hand them in.
  2. Write a program Seq01(n), that inputs a non-negative integer n, and outputs the set of all sequences of length n, whose entries are drawn from {0,1}. For example, Seq01(2); should yield {[0,0],[0,1],[1,0],[1,1]};
  3. Using Seq01(n), write a program, G01(n), that inputs a non-negative integer n, and outputs the subset of Seq01(n) of those sequences that never have two 0's in a row. For example, G01(3); should give {[0,1,0],[1,0,1],[0,1,1],[1,1,0],[1,1,1]}.
- For Old-Timers:
  1. Make sure that your mentee does all the examples, and completely masters pages 61-90 of Frank Garvan's awesome Maple booklet.
  2. Write a program SeqS(S,n), that inputs a set S and a non-negative integer n, and outputs all sequences of length n whose entries are drawn from S. For example, SeqS({0,1},2); should yield {[0,0],[0,1],[1,0],[1,1]};
  3. Using SeqS(S,n), write a program SeqG(S,T,n) that inputs a set of integers S, a set of lists T (in the alphabet S), and a non-negative integer n, and outputs the subset of SeqS({0,1},n) of those sequences that do not have any "factors" (i.e. contiguous

subwords) in T. For example, SeqG({0,1},{[0,0]}, 3); should give {[0,1,0],[1,0,1], [0,1,1],[1,1,0],[1,1,1]}.

Programs done on Feb. 4

- feb4.txt , contains brute-force programs for computing the set of sequences whose entries are from S or their negatives, and such that all partial sums (except the 0-th) are never zero.

Homework for Feb. 4 class (due Feb. 7)

- For Newcomers:
  1. Read and do all the examples, plus make up similar ones, in pages 91-120 of Frank Garvan's awesome Maple booklet. Print out a few sample examples, that you made up on your own, and hand them in.
  2. I just realized that it was stupid to have the input consist of a set S of positive integers, and then invite their negatives in. Modify TieLessGamesC(S,n), so that the input is any set of numbers S (do not have to be integers), and a non-negative integer, n, and outputs the set of lists of length n, whose elements are drawn from S, such that no partial sum (except the 0-th, of course) is ever zero.
  3. Using the new TieLessGamesC(S,n) that you have just written, modify it to write a program fSTn(S,T,n), that inputs two sets of numbers, S, and T, and a non-negative integer n, and outputs the set of lists of length n, whose entries are drawn from the set S, and such that no partial sum (except possibly the 0-th) belongs to T.
- For Old-Timers:
  1. Make sure that your mentee reads and does all the examples, and makes up similar ones, in pages 91-120 of Frank Garvan's awesome Maple booklet.
  2. I just realized that it was stupid to have the input consist of a set S of positive integers, and then invite their negatives in. Modify TieLessGamesC(S,n), so that the input is any set of numbers S (that do not have to be integers), and a non-negative integer, n, and outputs the set of lists of length n, whose elements are drawn from S, such that no partial sum (except the 0-th, of course) is ever zero.
  3. (Version of Feb. 5, 2008, correcting the previous version that it didn't make sense, thanks to Emilie Hogan!) Write a slightly more general program then the above TieLessGamesC(S,n), call it GenTieLessGamesC(S,n,k), that inputs a set of numbers S, a number k, and a non-negative integer n, and outputs the set of lists of length n, whose entries are drawn from S, none of whose partial sums are 0, and **in addition** the (full) sum of their entries equals k. Note that TieLessGamesC(S,n)=Union(GenTieLessGamesC(S,n,k); k from n*min(S) to n*max(S), and k NOT 0)
  4. By mimicking the above program, write an enumeration program, let's call it fSkn(S,n,k), that finds the number of elements of GenTieLessGamesC(S,n,k).

Compare the output to nops(GenTieLessGamesC(S,n,k)), for small n and k.

5. For the following games, how many tie-less games, with 50 scoring "events" are there
    - Soccer (S={1,-1})
    - Old-Time Basketball (S={1,2,-1,-2})
    - Contemporary Basketball (S={1,2,3,-1,-2,-3})
    - [American] Football (S=?, I forgot, look it up, or ask Lara) (If it runs for too long, replace 50 by a smaller number)

Programs done on Feb. 7 class

- feb7.txt , contains fSn(S,n): a program that inputs a set of integers S and outputs the **number** of n-letter words in the alphabet S, none of whose partial sums (except the 0th, of course) are 0, and fSnSeq(S,N) that gives the first N terms of this sequence.
- FindRec.txt , a program that guesses recurrences and analyzes sequences, do ezra(); to find out.

Homework for the Feb. 8, class (due Feb. 11, 2008)

- For Newcomers
    1. Read and do all the examples, plus make up similar ones, from page 121 all the way to the end, of Frank Garvan's awesome Maple booklet. Print out a few sample examples, that you made up on your own, and hand them in.
    2. Modify fSn(S,n) to write a program fSTn(S,T,n) that inputs two sets of integers S and T, and an integer n, and outputs the number of n-letter words in the alphabet S, none of whose non-empty partial sums has an element of T. Note that fSTn(S,{0},n) equals fSn(S,n) .
    3. Using Findrec(L,n,N,12); in FindRec.txt , find linear recurrences for various S's and T's.
    4. I roll a standard die 100 times. If it shows 1,2,3, I lose -3,-2,-1 dollars respectively. If it shows 4,5,6, I win 1,2,3, dollars respectively. What is the exact probability that I never had in my possession neither 2 dollars, nor 5 dollars, nor, -3 dollars?
- For Old-Timers:
    1. Make sure that your mentee reads and does all the examples, and makes up similar ones, pages 120 all the way to the end, of Frank Garvan's awesome Maple booklet.
    2. Write a program, WSTn(S,T,n) that inputs a set of steps in 2 dimensions given as lists of length 2, (for example, for the simple random walk it is {[-1,0],[1,0],[0,-1],[0,1]}), as well as a set of lattice points T, and an integer n, and outputs the number of n-step walks, with steps drawn from the set S, that never ever visit any of the points of T.

Programs done on Feb 11 class

- *feb11.txt* , An algebraic approach to finding the number of sequences of length n in an alphabet of integers S, none of whose (non-trivial) partial sums belong to a set of integers T.
- *WalkPlot*, Andrew Baxter's neat program that plots both 2D and 3D walks.

Homework for Feb 11 class (due Feb 14)

- For Newcomers:
  1. A generous beggar, who also donates to other beggars, receives or gives away, one coin at a time, that could be either a penny, a nickel, a dime, or a quarter. If there were 1000 events, what is the exact probability that, he was never in the red (in other words, he never had -1, -2, ..., -25 cents) .?
  2. Let a(n) be the number of old-time basketball games with n scoring events that ended in a tie, and where the home team was never behind. Find, empirically, a recurrence for a(n).
  3. What is the probability of a football games with 20 scoring events ending with a tie? (assume, unrealistically, that each scoring event is equally likely).
  4. Write a program that inputs an integer r, and outputs the numbers a_r and b_r such that

     $\Sigma_{0 \le k \le n}$ binomial(n,k)^r

     is asymptotic to $(a_r)^n n^{b_r}$. Can you conjecture what a_r and b_r are in general (as expressions in r?)
  5. Write a program that inputs a positive integer n, and outputs the number of integer partitions of n. For example p(3)=3, since 3 can be written as 1+1+1, 2+1, 3. Do you notice anything about p(5n+4) ?
- For Old-Timers:
  1. A walk is self-avoiding if it never visits the same place twice. Write a program that inputs a set of integers S, and a non-negative integer n, and outputs the number of one-dimensional self-avoiding walks of length n, using steps from S.
  2. Generalize the above for two-dimensional walks. Find the first 10 terms (or whatever you can) terms in the sequence of 2D **simple** self-avoiding walks, i.e., where S= {[-1,0],[1,0],[0,-1],[0,1]};

Programs done on Feb. 14, 2008, class

- feb14.txt

Homework for Feb 14 class (due Feb. 21)

- For Newcomers:
  1. Modify fAFxt(A,F,x,t), to write a program fAFt(A,F,t), that inputs a set of of letters A, and as set F of "dirty" words (lists) in the alphabet A, and a variable t, and outputs

a rational function in t, whose Maclaurin expansion is such that its coeff. of t^n equals the number of n-letters words that avoid (IN THE STRONG SENSE) the "dirty words" of F.

2. Use the above-mentioned fAFt(A,F,t) to write a program, SeqAFN(A,F,N), that inputs A and F as above, as well as a positive integer N, and outputs the first N terms of the sequence whose n-th term is the number of n-letter words in A avoiding the dirty words of F (in the strong sense).
3. Write a program IsSubWord(w,v) that inputs a word w, and another word v, and outputs true if and only of v is a subword of w.

   (Hint:

   with(combinat):
   L:=choose(nops(w),nops(v)):
   and use the fact that the subword of w from the places i1,i2, ..., ik is the word [w[i1],w[i2], ..., w[ik]]:)

4. Write a program that inputs a word w, and a set of "dirty" words, and outputs true if and only if w does not contain any of these dirty words.
5. Write a brute-force program CleanWords(A,F,n) that inputs an alphabet A, and a set of dirty words F, and an integer n, and outputs the set of n-letter words in A, that avoid the dirty words of F (in the strong sense).
6. By taking nops of the above, write SeqAFNstupid(A,F,N), a stupid analog of SeqAFN(A,F,N).
7. Test that you get the same output for six different random choices of A,F, and N (N between 7 and 11).

- For Old-Timers:

  All the above problems for Newcomers *PLUS*

  1. (A real challenge!) A word w avoids another word v in the weak sense if it does not contain v as a consecutive subword. For example Middlesexy does not avoid sex, but ASAEAX does avoid SEX in the weak sense (but not in the strong sense). Write an analogous program to fAFxt(A,F,x,t) that outputs the generating function for words that avoid F in the weak sense. (Hint: Now you have to consider two kinds of restrictions, avoiding in general, that is always the same, the set F, and avoiding at the very end, that changes).

Programs done on Feb 18, 2008

(Guest Lecturer: Eric Rowland, made a comparative study of Maple vs. Mathematica)

feb18.nb

Homework (To be handed-in to Eric, due Feb. 25, 2008)

1. Take a simple function that you've written recently, and rewrite it (in your language of choice) using functional (rather than procedural) constructs.
2. At 3am early Monday morning you discover that the Riemann hypothesis reduces to the statement that all triangle-free Hamilton-connected graphs with 7 vertices satisfy a certain simple condition. Use GraphData[ ] to find all graphs that you need to check. Will you be able to get eight hours of sleep and still make it to Experimental Math on time?
3. Make something cool with Manipulate[ ], and if it's sufficiently cool then upload it to the Demonstrations Project.

Program done in Feb. 21, 2008 class

feb21.txt (Under construction, to be completed next time)

Homework for Feb 21, 2008 (due Feb. 25, 2008)

- For Newcomers
  1. Write a program IsFactor(u,w) that inputs two words (given as lists) and outputs true if and only if u is a factor of w (which means that there exist i < j such that w[i]w[i+1]...w[j]=u
  2. Write a program AllWords(A,n) that inputs an alphabet A, and a non-neg. integer, and outputs all $|A|^n$ n-letter words (lists) in the alphabet A.
  3. Write a program, GoodWords(A,F,n), that inputs a set A, a set F of lists in A, and a non-neg. integer n, and outputs all the n-letter words in A, that do not contain (as a factor) any of the words of F.
  4. using GoodWords(A,F,n), and nops, write a program, SeqGood(A,F,n), that inputs A,F,n and outputs a list of integers, of length n, whose i^{th} term is the number of n-letter words in the alphabet A that do not contain, as factors, any of thw words of F.
- For Old-timers: All the above for newcomers, PLUS
  Complete, on your own, feb21.txt , and compare notes with SeqGood(A,F,n) .

Program done on Feb. 25, 2008 class

Recommended Reading: Enumerating Totally Clean Words by Dr. Z., that outlines the algorithm for counting words that omit (as subwords) a given set of "dirty" words.

feb25.txt, contains

- the function fAFt(A,F,t), that inputs a set of letters ("alphabet"), and a set of "dirty words", F, (using the letters of A), and a variable t, and outputs the rational function in t, that is the generating function for the sequence "number of n-letter words in the alphabet A, not containing as "factors", any of the words of F."

- the function , SeqC(A,F,N), that inputs a set of letters ("alphabet"), and a set of "dirty words", F, (using the letters of A), and a positive integer N, and outputs the list of length N, whose n^{th} term is the number of n-letter words in the alphabet A, not containing as "factors", any of the words of F.

Homework for Feb. 25, 2008 (due Feb. 28, 2008)

- For Newcomers:
  1. If you toss a fair coin 100 times, what is the probability that you have neither 6 Heads-in-a-row nor 6 Taills-in-a-row?
  2. What is the generating function for the number of n-letter words in the alphabet {1,2,3,4} that never contain four consecutive letters that are all different? Calling the counting sequence a(n), find constants C and μ such that a(n) is asymptotic to $C\mu^n$.
  3. Adapt fAFt(A,F,t) to write a program fAFtx(A,F,t,x) that has the same input as fAFt(A,F,t) and in addition another symbol x, and outputs the rational function in (t,x[A[1]],x[A[2]], ...) whose Maclaurin expansion in t has for its coefficient of t^n the POLYNOMIAL in (x[A[1]],x[A[2]], ...) that is the weight-enumerator of the set of n-letter words in A, avoiding F, where the *weight* of a word w is x[w[1]]x[w[2]]... (for example, the weight of ESSEX is x[E]x[S]x[S]x[E]x[X]= $x[E]^2x[S]^2x[X]$).
     For example, fAFtx({1,2},{},t,x); should be 1/(1-t*(x[1]+x[2]));
  4. Read carefully pages 1-9 in the highly readable and entertaining article, by Dr. Z. and John Noonan.
- For Old-timers: ALL the above plus:
  1. Write a new version of fAFt(A,F,t), call it, gAFt(A,F,t), implementing the Goulden-Jackson method. Do not peek at any Maple packages that may be available on-line. Test that your gAFt(A,F,t) yields the same output as fAFt(A,F,t), for various randomly-chosen A and F.

Program done on Feb. 28, 2008, class

feb28.txt, contains

- the function gAFt(A,F,t), that does exactly what fAFt(A,F,t) does in feb25.txt, but using the Goulden-Jackson cluster approach.

Homework for Feb. 28, 2008 (due March 3, 2008)

For everyone (Note: from now on everybody has the same homework, but I will understand if new-comers won't do everything).

1. Adapt gAFt(A,F,t) to write a program gAFtx(A,F,t,x) that has the same input as gAFt(A,F,t) and in addition another symbol x, and outputs the rational function in (t,x[A[1]],x[A[2]], ...) whose Maclaurin expansion in t has for its coefficient of t^n the POLYNOMIAL in (x[A[1]],x[A[2]], ...) that is the weight-enumerator of the set of n-letter words in A, avoiding F, where the *weight* of a word w is x[w[1]]x[w[2]]... (for example, the weight of ESSEX is x[E]x[S]x[S]x[E]x[X]= $x[E]^2x[S]^2x[X]$).
   For example, gAFtx({1,2},{},t,x); should be 1/(1-t*(x[1]+x[2]));
2. Adapt gAFt(A,F,t) to write a program gAFts(A,F,t,s) that has the same input as gAFt(A,F,t) and in addition another symbol s, and outputs the rational function in (t,s) whose Maclaurin expansion in (t,s) has for its coefficient of $t^n s^m$ the number of n-letter words in the alphabet A, that have exactly m mistakes. Note that gAFts(A,F,t,0) should equal gAFt(A,F,t).
   For example, gAFts({1,2},{[1,2]},t,s); should be 1/(1-2*t-(s-1)*t^2);
   Hint: replace -1 by (s-1) in gAFt(A,F,t) (and/or its subroutines)
3. Write a test program Test(r,m,k), that inputs integers r and m and k, that verifies that fAFt and gAFt yield the same output, for A={1,2,...,r}, and for **EVERY** possible k-element set of m-lettered "dirty words" and see who is faster
   For A={1,...,r} each such F
   t0:=time(): read `feb25.txt`: a:=fAFt(A,F,t); time()-t0;
   t0:=time(): read `feb28.txt`: b:=gAFt(A,F,t); time()-t0; evalb(a=b);
   Actually run Test(2,3,3); and if you can Test(2,3,4);
4. Adapt gAFt(A,F,t), to write a program gPFt(P,F,t), that inputs a probability vector P (whose length is the length of the alphabet) and where P[i] is the prob. that the i^{th} letter shows up, and outputs the rational function whose Maclaurin expansion, yields, as coeff. of t^n, the prob. that a random n-letter word will not contain any occurrence of the "dirty words" of F.
5. Look up the frequencies of English letters, and determine the probability that a 200-letter random "word" in English is SEX-less.
6. (5 dollar reward for the champion). Find an example of A and F where gAFt(A,F,t) beats, in running time fAFt(A,F,t) by as big a factor as possible (or vice versa).

Programs done on March 3, 2008, class

mar3.txt, contains

- G(n): a program that inputs a pos. integer n, and outputs the set of simple labelled graphs on {1, ..., n}
- Cc(G,v): a program that inputs a graph G, and a vertex v, and outputs the set of vertices that are connected, via some path, to v.
- IsConnected(G,n): inputs a graph G and a pos. integer n, and outputs true iff the graph G (on the set of vertices {1, ..., n}) is connected.

- CG(n): inputs a pos. integer n, and outputs the set of labelled connected graphs on {1, ..., n} .

Homework for March 3, 2008 (due March 6, 2008)

1. Print out, and carefully read Dr. Z.'s crash course on enumerative combinatorics
2. Modify G(n) to write a program Gnk(n,k), that inputs pos. integers n and k, and outputs the set of simple labelled graphs on {1, ..., n} with exactly k edges.
3. Modify CG(n) to write a program CGnk(n,k), that inputs pos. integers n and k, and outputs the set of simple connected labelled graphs on {1, ..., n} with exactly k edges.
4. Compute [seq(nops(CGnk(n,n-1)),n=1..7)], and conjecture an explicit expression for the number of connected labelled graphs with n vertices and n-1 edges.
5. Write a program, called Image(G,n,pi) that inputs a labelled grap G on the set of vertices {1, ..., n} and a permutation pi on {1, ..., n} and outputs the graph on {1, ..., n} obtained by renaming i by pi[i] (for all i=1...n).
6. Using permute(n) of the combinat package, write a program Images(G,n), that inputs a labelled graph G on {1, ..., n} and outputs the set of all images of G under permutations of {1, ..., n}
7. An unlabeled graph on n vertices may be viewed as an equivalence class of labelled graphs on {1, ..., n} under the equivalence relation "being an image under a permutation". So you can represent an unlabeled graph as a set of labelled graphs. For example the equivalence class of
   {[1,2],[1,3]} is
   {{[1,2],[1,3]}, {[1,2],[2,3]}, {[1,3],[2,3]}} .
   Write a program ULCGnk(n,k), the inputs pos. integers n and k, and outputs all unlabeled connected graphs on n vertices and k edges.
8. Compute [seq(nops(ULGCnk(n,n-1)),n=1..7)]: and search for it in Sloane.

Congratulations to Eric Rowland for winning the $10 prize for the longest Goulden-Jackson-style cluster

---

Programs done on March 6, 2008, class

- mar6.txt, (finishing up mar3.txt, adding procedures for rapid counting of connected labelled graph and connected labelled graph with a given number of edges, and the "stupid" way to count unlabeled connected graphs.
- mar6a.txt, direct counting of labelled trees (under construction, to be continued by you!, see homework).

Homework for March 6, 2008 (due March 10, 2008)

1. Modify UCG(n) of mar6.txt, to write a program UCGe(n,e) that inputs pos. integers n and e, and outputs the "set" of unlabeled graphs on n vertices and n-1+e edges. For example, UCGe(n,0) should give the set of unlabeled trees on n vertices (or rather one representative labelled tree from each equivalence class)
2. Modify NuUCGs(n) in mar6.txt, to write a program NuUCGse(n,e) that inputs pos. integers n and e, and outputs the counting sequence, from i=1 to i=n for the number of unlabeled trees on n vertices and n-1+e edges. For example, NuUCGse(n,0) should give the sequence for the number of unlabeled trees on i vertices for i from 1 to n .
   Look up NuCGse(6,0); in Sloane
3. Finish up RT(S) of mar6a.txt, as follows.
4. Write a procedure, SetPartition(S, i), that inputs a set S and a pos. integer i, and outputs the set of set-partitions of S into i disjoint (non-empty sets). For example, SetPartition({a,b,c},1); should give {{{a,b,c}}}, SetPartition({a,b,c},2); should give { {{a,b},{c}},{{a,c},{b}},{{b,c},{a}} }, while SetPartition({a,b,c},3); should give { {{a}, {b},{c}} }.
   Hint: As usual, use recursion, (with option remember!) Look at the largest element (n:=max(op(S))). It is either a singleton, so you take a member of SetPartition(S\n,i-1) and add to it the singleton {n}, or it can be joined to one of the already existing sets of any set-partition in SetPartion(S\n,i);
5. Write a program Combine(L) that inputs a list of "sets of sets" L=[S1,S2, ..., Sk] and outputs the set of all possible sets of the form
   s1 union s2 union ... union sk
   for all possible s1 in S1, s2 in S2, ..., sk in Sk
   For example Combine([{{a,b},{c,d}},{{e,f},{g,h}}]) should output { {a,b,e,f},{a,b,g,h}, {c,d,e,f},{c,d,g,h}}
   Hint: Use recursion, i.e. do Combine([S1,S2,...,Sk]) by using Combine([S1,S2,...,S(k-1)]) and combining with Sk.
6. Use the above to write RT(S)
7. Write a procedure RTn(n) that inputs a pos. integer n and outputs all the rooted labelled trees on {1, ..., n}
8. Find
   [seq(nops(TRn(n),n=1..7))];

Programs done on March 10, 2008, class

mar10.txt, contains RT(S): a program that inputs a set S and outputs the set of rooted trees on S (completed by Dr. Z. after class).

Homework for March 10, 2008 (due March 13, 2008)

Note: This is the version of 9:55am, March 11, 2008. (thanks to Paul Raff for pointing an error in the previous version). Please discard the previous version of the homework (in case you already downloaded it).

1. **Very Important:** Read carefully Dr. Z's masterpiece, especially the introduction, section 1 and section 4.
2. Look carefully at Combine(L,r) of mar10.txt, (finished by me after class), and convince yourself that it indeed does what it is supposed to do, and hence that RT(S) gives all rooted labelled trees on S. Note that the edges are now lists of length 2, and we consider them **directed edges** so [i,j] means i->j and [j,i] means j->i. By convention, in a rooted tree, all the edges point towards the root.
3. Write a procedure Weight(a,T) that inputs a tree, and a letter a, outputs the product of a[op(edge)] over all the edges of T. For example,
   Weight(a,{[1,2],[2,3]});
   should yield
   a[1,2]a[2,3]
   (Hint: use mul):
4. Write a program TW(n,i,a) that inputs a pos. integer n, another pos. integer i in {1,2, ..., n}, and a letter a, and outputs the sum of Weight(a,t) over all rooted trees [i,t], rooted at i. For example,
   TW(2,1,a);
   should yield a[2,1],
   while TW(2,2,a);
   should yield a[1,2],
   TW(3,1,a) should yield
   a[2,1]a[3,1]+a[3,2]a[2,1]+a[2,3]a[3,1] ;
5. (Correcting an error pointed out by Paul Raff, this is the corrected version of 3/11/08, 9:55am) Write a program P(n,i,a): that inputs pos. integers n, and i, with i ≤ n, and a synbol (letter) a, and outputs the polynomial (in the a[i,j]'s)

   (a[i,1]+a[i,2]+ ... +a[i,n])*TW(n,i,a)-
   (a[1,i]*TW(n,1,a)+a[2,i]*TW(n,2,a)+ ... + a[n,i]*TW(n,n))

   (Note that in the ... there is no a[i,i]), Don't forget to expand at the end.

6. Using the methodogy of experimental math, of generating data, looking for patterns, and making a conjecture, Conjecture an explicit expression for P(n,i,a).
7. ($5 prize, to be divided among all solvers by March 13, 2008) Prove your conjectured expression for P(n,i,a), using combinatorics! (no money for other methods).
   [Added March 13, 2008: Congratulations to Lara Pudwell, Andrew Baxter, Paul Raff, Beth Kupin, Emilie Hogan, and Brian Nakamura for sharing the $5 prize. They all got it right!. Here are Lara's solution and Baxter's solution .

Programs done on March 13, 2008, class

mar13.txt, contains the following nifty programs

- Sol(n,a): Solves the generic google PageRank's equation for n webpages and arbitrary probabilities of going from one webpage to another. (Warning: do not try to apply it to the full internet!)
- MTT(n,a): (The Matrix Tree Theorem). Inputs a pos. integer n, and outputs the weight enumerator of all labelled trees on {1, ..., n} rooted at 1, in case a is a letter, or inputs a numerical list of lists of size n, that is the adjanency matrix of the graph, and outputs the number of so-called spanning trees (trees whose vertices are all the vertices of the graph ({1,2, ..., n}) and whose n-1 edges are drawn from the edges of the graph.
- Arthur(n): Implements MTT(n,a) for the complete graph.
- Cayley(n): Spells out Arthur(n) for the complete graph, getting the determinant of a certain matrix, and then evaluating it.
- Cayleynx(n,x): a generalization of Cayley(n) that is easier to prove (see homework problem below)

Homework for March 13, 2008 (due March 24, 2008)

1. Find and a prove a closed-form expression, in n and x, for Cayleynx(n,x). Then plug-in x=n and deduce Cayley's formual.
2. Use the Matrix Tree Theorem to write a short program SPBP(m,n), that inputs two integers m and n, and outputs the number of spanning trees for the complete bi-partite graph $K_{m,n}$ (this is a graph with m+n vertices, m of which are called men, and n of which are called women, and there are all the possible mn edges between men and women, but no edges between men and no edges between women).
3. [I don't the answer for that one, I didn't try] Output many values for different m and n for SPBP(m,n) and see if they happen to be in Sloane. Try to conjecture a closed form formula for SPBP(m,n), if possible, or at least for SPBP(m,1),SPBP(m,2),SPBP(m,3), ... as far as you can.
4. Consider the graph G(n,i) whose vertices are {0, ..., n-1} and any vertex v is connected to v+1, v+2, ..., v+i, v-1, .., v-i (mod n) (so there are ni edges). Write a program,SPW(n,i), that finds the number of spanning trees of G(n,i).
5. [I don't the answer for that one, I didn't try] Output many values for different m and n for SPW(n,i) and see if they happen to be in Sloane. Try to conjecture a closed form formula for SPW(n,i), if possible, or at least for SPW(n,1),SPW(n,2),SPW(n,3), ... as far as you can.
6. Carefully read my writeup on the Lagrange Inversion Formula
7. Write a Maple program LIF(PHI,t,N) that inputs a function Φ of t, and outputs the first N coefficients of the formal power series that satisfies the functional equation:

   $u(t)=t\,\Phi(u(t))$

8. Using Generatingfunctionology, convince yourself that the exponential generating function, T(t), for labelled rooted trees satisfies the functional equation

   $T(t)=t\ e^{T(t)}$.

   Use the Lagrange Inversion Formula (by purely human means) to give yet another proof of Cayley's n^{n-2} formula.

9. Use Generatingfunctionology to find the number of ways of partitioning {1, ..., 100} into a disjoint union of sets, each of which has a size that is a perfect square.
10. Use the Lagrange Inversion Formula to find the number of labelled rooted trees on 100 vertices where the root has two neighbors, and each other vertex has exactly three neighbors, or exactly one neighbor.
11. Enjoy the Spring Break, but don't forget to do the homework!

Added March 17, 2008: Pick a final project ,

March 17, and March 20 2008

Spring Break.

Programs done on March 24, 2008, class

1. mar24.txt, implements the Lagrange Inversion Formula. It contains the following nifty programs
   - NoLIFexp(PHI,z,N): inputs an expression PHI in a variable z, and an integer N, and outputs the first N coeffs. of the t^n/n! of the formal power series u(t) that satisfies the Functional Equation

     u(t)=t&Phi(u(t))

   - NoLIF(PHI,z,N): inputs an expression PHI in a variable z, and an integer N, and outputs the first N coeffs. of the t^n of the formal power series u(t) that satisfies the Functional Equation

     u(t)=t&Phi(u(t))

   - LIF(PHI,z,N): Uses the Lagrange Inversion Formula to find exactly what NoLIF(PHI,z,N) does.
2. mar24a.txt, Uses the symbolic (and numeric) Matrix Tree Theorem. It contains the following nifty programs
   - LT(n): inputs a pos. integer n ($n \geq 3$), and outputs the set of labelled trees on {1, ..., n} (all n^{n-2} of them!). It uses the Matrix Tree Theorem in its symbolic manifestation,

and the fact that algebra is combinatorics (and vice versa)
- ST(n,G): inputs a pos. integer n (n ≥ 3) and a set of edges G (where an edge connecting i and j is denoted by {i,j}), representing a labeled graph on {1, ..., n}, and outputs the set of spanning trees of E.
- NuST(n,G): inputs a pos. integer n and a set of edges G (where an edge connecting i and j is denoted by {i,j}), representing a labeled graph on {1, ..., n}, and outputs the NUMBER of spanning trees of E.
- Cycle(n): inputs a pos. integer n and outputs the cycle graph on {1, ..., n}
- Wheel(n):inputs a pos. integer n and outputs the wheel graph on {1, ..., n,n+1}, where the center is labeled n+1.

Homework for March 24, 2008 (due March 27, 2008)

1. Write a program a(r,N), that inputs pos. integers r and N and outputs the first N terms of the counting sequence for labelled rooted trees where the root has degree ≤ r and every other vertex had degree ≤ r+1 (i.e., viewed from the top, it has ≤ r children). Look at the sequences for r=1,2,3, ... in Sloane, and see which exist.
2. An ordered tree is an unlabelled tree where there is a distinguished vertex r, and it has children, ordered from left to right, each of whom may have no children, or start its own dynasty.
   Write a program b(r,N), that inputs pos. integers r and N and outputs the first N terms of the counting sequence (i.e. the list whose n-th term is the number of such trees with n vertices) for ordered trees where every vertex has ≤ r children. Look at the sequences for r=1,2,3, ... in Sloane, and see which exist.
3. Write a program c(r,N), that inputs pos. integers r and N and outputs the N-term sequence (list) whose n-th entry is the number of ordered trees with n vertices, and where every vertex has either no children (i.e. is a leaf) or has **exactly** r children. Can you conjecture a closed-form formula for c(r,N)? Can you prove it? (using the Lagrange Inversion Formula or otherwise).
4. Conjecture a closed-form expression for NuST(n+1,Wheel(n));
5. Define a generalized wheel, Gwheel(n,r), with rn+1 vertices where
   the outer rim is labelled 1...n,
   the next inner rim is labelled n+1, ..., 2n
   ...
   the innermost rim is labelled (r-1)n+1, ..., rn
   the center is labelled rn+1.
   Assume that within each rim, m has an edge to its immediate neighbors (so there is an edge between m and m+1 if m is not a multiple of n and an edge between jn+n and jn+1), and in addition, the center has an edge to each the innermost rim, and each other vertex ,m, not on the inner rim is connected to m+n. Write a program Gwheel(n,r) that outputs this graph as a set of edges.

6. For r=1,2,3, ... (as far as you can go), compute sequences for the number of spanning trees of Gwheel(n,r) and look them up in Sloane. If possible, conjecture "nice" expressions for them and prove them.
7. Read about Prüfer's bijection in Dr. Z.'s crash course on enumerative combinatorics, or Wikipedia, or, in more detail, in almost any combinatorics text, and program it, first the "easy" direction (from labelled trees to codes), and then the reverse. Call them S, and T resp .. Check empirically that ST and TS are the identity mappings on their respective domains.

Programs done on March 27, 2008, class

mar27.txt, to construct and count integer partitions. It contains the following procedures.

1. Par(n): the set of integer-partitions of n
2. Par1(n,k): the set of partitions of n whose largest part is k
3. par1(n,k): the number of partitions of n whose largest part is k
4. par(n): the number of integer-partitions of n, using a 2-parameter recurrence (i.e. adding up par1(n,k) for k from 1 to n)
5. P(N): the first N terms of the partition function (the stupid way, using par(n))
6. Pf(N):the first N terms of the partition function (using the generating function 1/(1-q)/(1-q^2)/(1-q^3)/...)
7. p(n): the number of partitions of n using the Euler-Pentagonal recurrence
8. Pff(N):the first N terms of the partition function (using the Euler-pentagonal recurrence)

Homework for March 27, 2008 (due March 31, 2008)

1. Write a program, call it Par2(n,k), that inputs pos. integers n and k and outputs the set of integer partitions with exactly k parts.
2. Write a program, call it par2(n,k), that inputs pos. integers n and k and outputs the number of integer partitions with exactly k parts.
3. Prove that par1(n,k) is always equal to par2(n,k). Construct an explicit bijection between Par1(n,k) and Par2(n,k), and vice versa.
4. Modify Par(n) to write a program cPar(n,d,S) that inputs a pos. integer n, a pos. integer d, and a set S of integers between 0 and d-1 and outputs the set of partitions of n **all** of whose parts are congruent to some element s in S modulo d. For example cPar(n,2,{1}); is the set of partitions of n all whose parts are odd, cPar(n,5,{1,4}); is the set of partitions of n all whose parts are congruent to 1 or 4 modulo 5.
5. Modify par(n) to write a program cpar(n,d,S) that inputs a pos. integer n, a pos. integer d, and a set S of integers between 0 and d-1 and outputs the number of partitions of n **all** of whose parts are congruent to some element s in S modulo d. For example cpar(n,2,{1}); is the number of partitions of n all whose parts are odd, cpar(n,5,{1,4}); is the number of

partitions of n all whose parts are congruent to 1 or 4 modulo 5.
(In other words, cpar(n,d,S) is the number of elements of cPar(n,d,S).)

6. Modify Par(n) to write a program dPar(n,d) that inputs a pos. integer n, a pos. integer d, and outputs the set of partitions of n where the difference between consecutive (and hence any) two parts is ≥ d. For example, dPar(n,0) is just Par(n), while dPar(n,1) is the set of partitions of n into **distinct** parts.
7. Modify par(n) to write a program dpar(n,d) that inputs a pos. integer n, a pos. integer d, and outputs the number of partitions of n where the difference between consecutive (and hence any) two parts is ≥ d. For example, dpar(n,0) is just par(n), while dpar(n,1) is the number of partitions of n into **distinct** parts.
   (In other words, dpar(n,d,S) is the number of elements of dPar(n,d,S).)
8. Prove, empirically, for 0 ≤ n ≤ 200, that
   cpar(n,2,{1})=dpart(n,1)
9. (5 dollars to be divided between all those who get it right by March 31, 2008, 12:00noon, and who have not seen it before). Find a bijective proof of the above fact, and program it.
   In other words find two mappings
   S: cPar(n,2,{1})->dPar(n,1)
   T: dPar(n,1)-> cPar(n,2,{1})
   such that ST and TS are both the identity mappings.
10. Prove, empirically, for 0 ≤ n ≤ 200, that
    cpar(n,5,{1,4})=dpart(n,2)
11. (100 dollars to the first person to get it, offer expires last day of classes). Find a *nice* bijective proof of the above fact, and program it. In other words find two mappings
    S: cPar(n,5,{1,4})->dPar(n,2)
    T: dPar(n,2)-> cPar(n,5,{1,4})
    such that ST and TS are both the identity mappings.

Handout and Programs given on March 31, 2008, class

- Handout: Dave Bressoud and Dr. Z's "proof from the book", of Euler's pentagonal recurrence.
- mar31.txt, generating function for the number of partitions with bounded number of parts and bouded larget part. It contains procedures, G(M,N,q), that inputs pos. integers M and N and a variable q, and outputs the polynomial in q, whose coeff. of q^n gives the number of partitions of n into at most M parts, each of which is ≤ N, and G1(M,N,q), the Gaussion polynomial
  $[(1-q^{M+1})(1-q^{M+2})... (1-q^{M+N})]/ [(1-q)(1-q^2) ...(1-q^N)]$
  (that is allegedly equal to it).
- Courtesy of Andrew Baxter: Baxter's neat Ferrers Diagrams package, that would help visualize partitions and plane-partitions.

First Homework Set for March 31, 2008 class, Due April 1, 2008 [No extensions!]

- Recall that for any set of non-negative integers A, mex(A) is the smallest non-negative integer **not** in A. For example, mex({2,4,5})=0, mex({0,1,2,5,8})=3, etc.
  Define a sequence a_i recursively by $a_1=2$, and for $i \geq 1$ by:
  $a_i=\text{mex}(\{0,1\} \cup \{ j a_r \ , \ j \geq 1 \ , 1 \leq r < i\})$,
  Prove the following properties of a_i
  1. There are infinitely many i such that $a_{i+1}-a_i=2$
  2. Every even intger $n \geq 6$ can be written as a_i+a_j, for some pos. integers i and j.
  3. Define a sequence F(n) by, $F(a_{i_1} a_{i_1} a_{i_2} ...a_{i_r})=(-1)^r$ if n can be expressed as a product of distinct a_i's , and 0 otherwise. Let
     G(n)=add(F(i), i=0..n)
     Prove that $|G(n)| \leq Cn^{.999}$, for some fixed constant C.
- Remember that Euler's pentagonal product
  $\eta(q)=(1-q)(1-q^2)(1-q^3) \ldots$,
  when expanded, has lots of 0-coefficients and the rest are 1 or -1. Condider the 24-th power of that
  $\eta(q)^{24}=[(1-q)(1-q^2)(1-q^3) \ldots]^{24}$,
  and let's call the coeff. of q^n, $\tau(n)$. Prove that $\tau(n)$ is never zero.

Second Homework Set for March 31, 2008 class, Due April 3, 2008

1. Write a Maple implementation of the Bressoud-Z involution from the book.
2. Write a procedure, PP2a(a1,a2,N,q), that inputs two non.neg. integers a1,a2 such that a1 ≥ a2 ≥ 0, and outputs the generating function for N-columned and 2-rowed plane partitions whose left-most column is [a1,a2].
3. Write a procedure, PP2(M,N,q), that inputs two non.neg. integers M and N and outputs the generating function for all 2-rowed and N-columned plane partitions, all of whose parts are ≤ M.
4. Write a procedure, PP3a(a1,a2,a3, N,q), that inputs three non.neg. integers a1,a2,a3, such that a1 ≥ a2 ≥ a3 ≥ 0, and outputs the generating function for N-columned and 3-rowed plane partitions whose left-most column is [a1,a2,a3].
5. Write a procedure, PP3(M,N,q), that inputs two non.neg. integers M and N and outputs the generating function for all 3-rowed and N-columned plane partitions, all of whose parts are ≤ M.
6. Conjecture explicit expressions for PP2(M,N,q) and PP3(M,N,q) .
7. (5 dollars to be divided among the people who would get it right). Conjecture an explicit expression for PPK(M,N,q), the generating function for K-rowed, N-columned Plane Partitions each of whose parts is ≤ M .

Programs done on April 3, 2008, class

apr3.txt contains the following procedures:

- GuessPolq(L,q,z): inputs a list L of polynomials in the variable q, and a variable z, and tries to guess a polynomial P(z) such that P(q^n)=L[n]
- PP2(M,N,q): The generating function for plane partitions whose 3D Ferrers graph is contained in an M by N by 2 box. Equivalently, the weight-enumerator of 2 by M arrays a[i,j], i=1..2, j=1..M, of non-negative integers, weakly decreasing along rows and columns (according to the weight $q^{\text{sum of entries}}$)
- PP3(M,N,q): The generating function for plane partitions whose 3D Ferrers graph is contained in an M by N by 3 box. Equivalently, the weight-enumerator of 3 by M arrays of non-negative integers a[i,j], i=1..3, j=1..M weakly decreasing along rows and columns (according to the weight $q^{\text{sum of entries}}$)

Homework for April 3, 2008 class, Due April 7, 2008

1. Using GuessPolq, conjecture a closed form formula (as a product) for PP2(M,N,q) and PP3(M,N,q). Generalize this to conjecture the generating function for "PPK(M,N,q)", i.e. the weight-enumerator for plane partitions whose 3D Ferrers diagram is inside an M by N by K box. formula (as some kind of product)
2. Write PPK(M,N,K,q) that generalizes PP2 and PP3 to plane partitions with K rows. Check its output for K=4,5 against your conjecture.
3. Plug-in M=N=K=infinity in the above prodcut-formula and deduce MacMahon's expression for ALL plane partitions
   $1/[(1-q)(1-q^2)^2(1-q^3)^3 \ldots]$
4. Write a program Solid22(N,M,q) that inputs integers M and N and outputs the generating function for all solid partitions whose 4D Ferrers graph fits inside a 2x2xMxN (four-dimensiona) box. In other words, all 3D 2x2xM arrays
   a[i,j,k], $1 \le i,j \le 2$, $1 \le k \le M$
   with $0 \le$ a[i,j,k] $\le N$ such that a[i,j,k] is weakly decreasing in every one of the three direction.
   Hint: First write a program Soild22a(a11,a12,a21,a22,M,q) that computes the generating function according to the information about the top layer, and then, to get Solid22(N,M,q) simply do
   Soild22a(N,N,N,N,M+1,q)/q^{4N}
   Note that a11 ≥ a12, a11 ≥ a21, a21 ≥ a22 in addition to them being between 0 and N.

Programs done on April 7, 2008, class

apr7.txt contains the following procedures:

- A(n,S): inputs a pos. integer n and a set of patterns S and outputs the set of permutations on {1, ..., n} that do not contain any of the patterns of S. For example, A(6,{[1,2,3], [1,3,2]}); gives the set of permutation pi of {1,2,3,4,5,6} such that you never have
  pi[i1] < pi[i2] < pi[i3]
  and never
  pi[i1] < pi[i3] < pi[i2]
  for 1 ≤ i1 < i2 < i3 ≤ 6
- SeqSN(S,N): Inputs a set of patterns S and a pos. integer N, and outputs the sequnce of length N whose n-th element is nops(A(n,S)).

Homework for April 7, 2008 class, Due April 10, 2008

1. Prove rigorously (but humanly) that |A(n,{[1,2,3],[1,3,2]})|=2^{n-1} .
2. Try to prove that both |A(n,{[1,2,3]})| and |A(n,{[1,3,2]})| are given by the Catalan numbers C_n=(2n)!/(n!(n+1)!) .
3. Look at all the binomial(6,3)=20 sets that have exactly three patterns of length three . try to conjecture expressions for A(n,S) for as many S as you can.
4. Extend A(n,S) and write a procedure B(L,S) that inputs a list L of non-neg. integers and outputs all words (i.e. lists) with L[1] 1's, L[2] 2's, ... L[n] n's that avoid the set of patterns S. Note that B([1$n],S) should equal A(n,S).
   Warning, this is very slow, so only try it for L whose sum is ≤ 7 .

Programs done on April 10, 2008, class

Today was such a nice day, so the class was on the lawn, using our biological computers between our shoulders, aided by memory-storage device called pencil-and-paper. We talked about Young tableaux and how to count them. The following short program: apr10.txt, contains programs Par(n) to generate all integer-partitions of n, and GuessPol(L,n) that inputs a list L of numbers and a symbol n, and outputs a polynomial P, of degree nops(L)-4, if it exists, such that P(i)=L[i], for i=1..nops(L) (or returns FAIL). These are needed for the homework problems.

Homework for April 10, 2008 class, Due April 14, 2008

1. Write a program: Children(L) that inputs a partition of L, and outputs the of partitions obtained by reducing by one any of its parts, such that the remaining list is still a partition. For example, Children([3,2,2,1]); should give the set {[2,2,2,1],[3,2,2]}, but does not include [3,1,2,1]). Rememeber that if you get [3,2,2,0] you must kick out the 0 at the end.
2. Write a program f(L) that inputs a partition L and outputs the number of Standard Young tableaux of shape L.
   Hint: Using the "going down" recurrence f(L)=Sum(f(L-), L- in Children(L));, together with the the initial condition f([])=1 .

3. Verify up to n=13, the Humberto conjecture that
   Sum(f(L)^2, L in Par(n)) =n!
4. Write a procedure for
   a(n):=Sum(f(L), L in Par(n))
   and look up
   seq(a(n),n=1..9)
   in Sloane.
5. Using GuessPol in apr10.txt, conjecture explicit expressions for
   - f([a,1]); (a ≥ 1)
   - f([a,2]); (a ≥ 2)
   - f([a,3]); (a ≥ 3)
   - f([a,4]); (a ≥ 4)

   (Warning: the lists that GuessPol takes start at the argument being 1, here the argument starts later.
6. Glancing at the above output, conjecture an explicit expression for f([a,b]), with a ≥ b ≥ 0.
7. (5 dollars to be divided among all correct solvers). Using the same methodololgy as above conjecture an explicit expression for f([a,b,c]) for a ≥ b ≥ c ≥ 0 . Then Rigorously prove it!
8. (6 dollars to be divided among all correct solvers). Conjecture an expression for f([a1,a2, ..., ak]) for arbitrary k.
9. (7 dollars to be divided among all correct solvers) Prove the above conjecture.

Programs done on April 14, 2008, class

apr14.txt, contains the following procedurs (in addition to Par(n) from last time and GuessPol(L,n) described last time)

- Mamas(L): inputs a partition L and outsputs all its "parents" (what we called children last time), i.e. all the shapes obtained from L by nibbling (legally) one box.
- f(L) : inputs a partition L and outputs the number of Standard Young Tableaux of shape L.
- Guess2(a,b1) : inputs a symbol a and a pos. integer b1, and outputs the guessed polynomial, in a, for f([a,b1]);
- Guess3(a,b1,c1) : inputs a symbol a and a pos. integers b1,c1, with b1>=c1>0 and outputs the guessed polynomial, in a, for f([a,b1,c1]);

Homework for April 14, 2008 class, Due April 17, 2008

1. Recall that a **rational function** f(x) is a quotient of polynomials f(x)=P(x)/Q(x). Write a program GuessRat1(L,x,d) that inputs a list L, a variable x, and a non.neg. integer d such that 2*d+5 ≤ nops(L), and outputs a rational function P(x)/Q(x) with degree(P,x), degree(Q,x) ≤ d, such that P(i)/Q(i)=L[i] for all i from 1 to nops(L), if one exist, and FAIL otherwise.

2. By trying d=1..(nops(L)-5)/2, write a program GuessRat(L,x) that guesses a rational function P(x)/Q(x) that fits L.
3. By using GuessRat, conjecture a rational function for g2(a,b):=f([a,b])/((a+b)!/(a!b!)) . Hint, first, freeze b, and get rational functions g(a,b1) for various b, and then use GuessRat with respect to that list to conjecture an expression for g(a,b).
4. By using GuessRat, conjecture a rational function for g3(a,b,c):=f([a,b,c])/((a+b+c)!/(a!b!c!)) .
   Hint, first, freeze c, and b, to get rational functions g(a,b1,c1) for various b1, and c1. etc.
5. Write a procedure Children(L) (with the new meaning of children), that inputs the set of partiotions obtained from L by legally adding one box (i.e. the set of partitions L1 such that L is a member of Mamas(L1)).
6. What can use say about
   h(L):=Sum(f(L1), L1 in Children(L))/f(L) .

Programs done on April 17, 2008, class

apr17.txt, contains the following procedurs (in addition to those of apr14.txt described above)

- Children(L): Inputs a partition L and outputs the set of partitions obtained from L my legally adding one box to each qualified row.
- Y(n,k): Inputs positive integers n and k and outputs Sum(f(L)^k, L in Par(n)), where f(L) is the number of Standard Young Tableaux of shape L. Y(n,2) should equal n!
- SeqY(N,k): the sequence of Y(n,k) for n=1.. N
- Nes(n): verifying the "going-up recurrence" satisfied by L for all L in Par(n)
  (n+1)f(L)=Sum(f(L1), L1 in Children(L))
  Yr(n,k,r): Inputs positive integers n, k, and r, and outputs Sum(f(L)^k, L in Par(n,k)), where f(L) is the number of Standard Young Tableaux of shape L.

Homework for April 17, 2008 class, Due April 21, 2008

1. Write a procedure, SYT(L), that inputs a partition L, and outputs the set of Standard Young Tableaux of shape L. For example Y([2,2]); should yield {[[1,2],[3,4]],[1,3],[2,4]]}.
2. A **Semi-Standard** tableau of shape L is a way of filling-in L with positive integers such that it is strictly increasing along the columns but only weakly increasing along the rows. Write a program SSYT(L,r) that inputs a shape L and a pos. integer r, and outputs the set of Semi-Standard Young tableaux of that shape that only uses the integers {1, ..., r}. For example SSYT([2],2); should return {[1,1],[1,2],[2,2]}; while SSYT([1,1],2); should return {[1,2]};
   Hint: write the analog of Mamas, by considering those tableaux obtained by deleting the largest integer, r.

3. (10 dollars to be divided among those that get it right). Use human ingenuity to prove the "Going-Up recurrence",
(n+1)f(L)=Sum(f(L1), L1 in Children(L)) by using induction, together with the "Going-Down recurrence" (that is obvious):
f(L)=Sum(f(L1), L1 in Mamas(L))

Programs done on April 21, 2008, class

apr21.txt, contains the following procedurs (in addition to those of apr17.txt described above)

- F(L): inputs a partition L (given as a list of weakly decreasing pos. integers) and outputs the set of Standard Young Tableaux of shape L. For example, F([2,2]); should give {[[1,2],[3,4]],[[1,3],[2,4]]} .
- PY(Y): inputs a tableaux and prints it nicely
- PSY(S): prints out nicely all the tableaux of the set S
- YFF(L): The Rowland conjectured explicit formula for the number of SYT of shape L (unfortunately already proved, about one hundred years ago, by Young, Frobenius, and MacMahon)
- ProveYFF(k): inputs a pos. integer k, and outputs true or false according to whether YFF(L) is true for **all** (infinitely many!) Young tableaux with ≤ k rows. (Warning for k ≥ 7 it is starting to get slow, and for general k, you still need humans).

Homework for April 21, 2008 class, Due April 24, 2008

1. A skew-shape is a pair of partions [L1,L2] such that the Young diagram of L1 is a subset of the Young diagram of L2. A skew-standard-Young tableau, is a way of filling-in the integers 1 through nops(L2)-nops(L1) such that they are increasing along rows and along columns. Modify f(L) and F(L) to write procedures fSkew(L1,L2) and FSkew(L1,L2) that computes their numbers and the set itself, respectively.
2. Given a partition (shape) L, label its boxes like you would the entries of a matrix, i.e. box (i,j) is the j-th box of the i-th row. The **hook-length** of a box (i,j) in a shape L, is the number of boxes to its right (in the i-th row) plus the number of boxes below it (in the j-th column) plus 1 (for itself). For example, the hook length of box (2,3) in the shape [6,5,4,3,3,3] is 7. Write a procedure Hook(L,i,j) that inputs a partition L and pos. integers i and j, and outputs the hook-length of box (i,j) in the shape L. For example,
Hook([6,5,4,3,3,3],2,3);
should return 7. It should return FAIL if box (i,j) is not a member of the shape L.
3. Write a prodedure HLF(L), that inputs a partition L and outputs n! divided by the product of **all** the hook-lengths. (where n is the number of boxes in L (i.e. L[1]+L[2]+ ...+L[nops(L)]).

4. Verify empirically that HLF(L)=YFF(L) for all partions L of n ≤ 10 .
5. [5 dollars to be divided between all correct answers]. Prove that HLF(L)=YFF(L) for **every** shape.
6. [7 dollars to be divided between all correct answers] Prove that f(L)=YFF(L) for shape with k rows, with k arbitrary.
   Hint: You may want to use a variant of the Lagrange Interpolation Formula.

Programs done on April 24, 2008, class

apr24.txt, contains the following procedurs (in addition to those of apr21.txt and sooner described above)

- HL(L,i,j): inputs a partition L and pos. integers i and j and outputs the hooklengths of box (i,j) in L
- HLF(L): The hooklength formula for the number of Standard Young Tableaux of shape L (equals YFF(L)).
- Ins1(Y,i,row1): one step of one step of the Robinson-Schenstead algorithm. Inputs a Young tableau Y, and an integer i, and a row number row1, and outputs another Young tableau and a bumpee.
-
- Ins(Y,i): one step of the Robinson-Schenstead algorithm. Inputs a Young tableau Y, and an integer i, not in Y, and outputs another Young tableau with one extra box added, and the row-number of that added box. [corrected, 4/25/08 by Slava and Humberto]

Homework for April 24, 2008 class, Due April 28, 2008

1. Complete the Robinson-Schenstead algorithm, writing a procedure, RS(pi), that inputs a permutation pi of {1, ..., n} and outputs a pair of Standard Young Tableaux of the same shape (each with n boxes). Do it by setting Y:=[[pi[1]]], and then doing for each i=2..n, in turn, Y1:=Ins(Y1,pi[i]). The final tableau is the Insertion tableau. To get the second tableau (the so-called template tableau) keep inserting, in, turn, 1, 2, 3, ..., n, where i goes to the box that was added right after the insertion of pi[i].
2. Write the inverse of Ins(Y,i), call it Del(Y,bumpee). (Hint: first write the inverse of Ins1)
3. Using the above, write a procedure, IRS(Y1,Y2), that inputs two Standard Young tableaux Y1,Y2, of the same shape (first check that fact, and return FAIL if it fails), and outputs a permutation pi, such that RS(IRS(Y1,Y2))=(Y1,Y2).
4. Write a procedure that inputs a permutation and outputs its inverse.
5. How are RS(pi) and RS(inverse(pi)) related?

Programs done on April 28, 2008, class

apr28.txt, contains the following procedurs (in addition to those of apr24.txt and sooner described above)

- RS(pi): performs the Robinson-Schenstead Correspondence to a permutation pi
- TestRS(n): tests that the RS(pi) ranging over all pi, does indeed gives the set of pairs of SYT of the same shape with n boxes.

Homework for April 28, 2008 class, due May 1, 2008

1. A semi-standard Young tableaux (also called column-strict) is a way of filling an r-rowed shape with the integers {1, ..., n} (n ≥ r), such that the entries are weakly increasing along the rows and strictly increasing along the columns. Write a program SSYT(L,n) that inputs a shape L and a pos. integer n, and outputs the set of semi-standard Young tableaux.
2. The **weight** of a tableaux (a[i,j]) is the product of x[a[i,j]] over all entries. For example, the weight of [[1,2,2,3],[2,3,3,4],[3,4,5]] is (x[1]x[2]x[2][x[3]) (x[2]x[3]x[3]x[4])(x[3]x[4]x[5]). Let S(L,n,x) be the sum of the weights of all members of SSYT(L,n). Write a procedure Sslow(L,n,x) to find the polynomial (in x[1], ..., x[n]) S(L,n,x)
3. Write a faster program . Sfast(L,n,x), with the same output as Sslow(L,n,x), but without first literally constructiong the set SSYT(L,n). Do it by establishing a recurrence that expresses S(L,n,x) in terms of S(L',n-1,x) for smaller L' s.
4. [2 dollars to be divided by the k solvers]. Conjecture an explicit expression for S([a,b],2,x)*(x[1]-x[2]), as a polynomial in (with powers that involve a,b), for a ≥ b > 0
5. [3 dollars to be divided by the k solvers]. Conjecture an explicit expression for S([a,b,c],3,x)*(x[1]-x[2])*(x[1]-x[3])*(x[2]-x[3]), as a polynomial in x[1],x[2],x[3] (with powers that involve a,b,c) a ≥ b ≥ c > 0
6. [10 dollars to be divided by the k solvers]. Conjecture an explicit expression for S([a1,a2, ..., an],n,x)*Product(Product(x[i]-x[j], j=i+1..n),i=1..n)
   as a polynomial in x[1],x[2],x[3], ..., x[n] (with powers that involve a1,a2, ..., an) a1 ≥ a2 ≥ ... ak > 0
7. In preparation for Polya theory, to be covered next time, carefully read pages 10 and 11 of: Dr. Z.'s Crash course on enumerative combinatorics

Programs done on May 1, 2008, class

may1.txt, contains the following procedurs, starting Polya-Redfield theory.

- AvF(G): The average number of fixed points of the group of permutations G.
- kefel(pi,sig): permutation pi times permutation sig (as permutations)

- GenG(S): Given a set of permutations S all of the same size, outputs the group of permutations generated by them

Homework for May 1, 2008 class, due May 5, 2008

1. Figure out the group of rotations of the three dimensional cube that send faces to faces (hint: there 24 of them). Label the top 1, the bottom 2, the left 3, the right 4, the front 5, and the back 6.
2. Write a procedure Necklace(n), that inputs an integer n, and that outputs the group of rotations of an n-bead necklace, as a group of permutations of 1, 2, ...,n , where the beads are arranged clockwise.
3. Write a procedure Gp(G,n), that inputs a graph G with vertices {1, ...n}, given in terms of a set of edges, and outputs the set of permutations pi of {1,...,n} that when pi is applied to G, the set of edges remain the same. For example Gp(3, {{1,2},{1,3}}); should output {[1,2,3],[1,3,2]}.
4. Write a procedure Cycle(pi), that inputs a permutation pi, and outputs the set of its cycles. A cycle i1->i2->i3->...->ik->i1 should be denoted by [i1,i2, ..., ik] For example Cycles([1,2,3]); should return {[1],[2],[3]}, and Cycles([3,2,1]); should return {[1,3],[2]}.

Programs done on May 5, 2008, class

may5.txt, Finishing Polya-Redfield theory.

Have a great summer!

Dr. Z.'s teaching page