Math 3012 Applied Combinatorics
Guidelines for Instructors

William T. Trotter

This document consists of three parts:

1. Part 1. Math 3012: Course Overview.
2. Part 2. Math 3012: Recommended Topics, Schedule and Comments on Presentation.
3. Part 3. Math 3012: Sample Syllabus.

The first two parts are included here, while the sample syllabus is available on-line at:

http://people.math.gatech.edu/ trotter/math-3012-Fa14/math-3012-syllabus.html

These guidelines are intended for graduate teaching assistants, postdocs and visiting faculty assigned to teach Math 3012. Necessarily, it reflects my personal perspective to some degree. However, I have made an effort to gather information that will be value to instructors, even if they chart a different path.

Part 1. Math 3012: Course Overview.

The Georgia Tech catalog provides the following concise statement concerning Math 3012:

> *Elementary combinatorial techniques used in discrete problem solving: counting methods, solving linear recurrences, graph and network models, related algorithms, and combinatorial designs.*

Students. Math 3012 is intended to serve as an introduction to combinatorial mathematics for students in computer science, engineering and mathematics. It is a required course for both undergraduate degrees in mathematics (B.S. in Applied Mathematics and B.S. in Discrete Mathematics). It is also a required course for undergraduate computer science and a recommended course for several engineering programs. The following table provides enrollment data in aggregate form for the three year period 2011-13.

Major	**Enrollment**	**Percentage**
Computer Science	1033	74%
Engineering	245	18%
Math & Discrete Math	119	9%
Total Enrollment	1387	100% (with usual round off errors)

As reflected by this data, instructors in Math 3012 should be aware that the bulk of their students are CS majors, so they must find the right balance between mathematical correctness and applications. In particular, Math 3012 is not a "theorem-proof" course in the spirit of traditional advanced courses in pure and applied mathematics. In our School, these courses carry a 4000 level designator.

All students enrolled in Math 3012 will have completed the calculus sequence, so they will be comfortable with basic concepts of linear algebra and they will know a bit about infinite series. However, in other areas, students' backgrounds will be more varied. Computer science students will have seen spanning trees and sorting algorithms. CS and engineering students will be relatively skilled programmers, and some are quite advanced, but math students often have no programming background whatsoever.

Tests, Final Exams and Grading Standards.

Syllabus. It is School policy that *all* instructors (faculty, postdocs, graduate teaching assistants and visiting faculty) distribute a written syllabus with clear statements regarding course content, grading policy and attendance policy, at a minimum. Many instructors prepare their syllabus in html format and post it on the course web page on T-Square. A sample syllabus for the Fall 2014 semester can be found at:

http://people.math.gatech.edu/ trotter/math-3012-Fa14/math-3012-syllabus.html

Instructors are welcome to download the source for this page and modify it as they feel appropriate. Of course, the text file can be readily converted into a LaTeX document by those who prefer to distribute hard copy.

Grading Policy. For fall and spring semesters, instructors are advised to give at least three mid-term tests. Two mid-term tests is considered as a minimum. Giving more than four is discouraged, as experience shows that the manpower associated with grading that many tests in a timely manner becomes a liability. During summer terms, instructors are advised to give two mid-term tests.

Mid-term tests are designed to be taken in a normal class period (50 minutes for MWF and 75 minutes for T-Th during regular semsesters and

70 minutes during summer semesters). Cumulatively, these mid-term exams should account for at least 50% and at most 75% of the final grade. The date for the final exam (two hours, fifty minutes in duration) is set by the registrar and this date should be listed on the course syllabus. The final exam should be weighted at least 15% and at most 35% on the final grade. However, many instructors giving a cumulative final use a grading system that allows a very strong performance on the final exam to count extra.

Homework should be assigned on a regular basis but how this should be incorporated into the grading scheme depends on several factors, including whether or not grading support is provided. Some instructors have had good success with a commitment that at least one question on each mid-term exam will come directly from homework assingments. Also, some instructors have incorporated special projects, which typically consist of a set of options, such as a programming project, reading and reporting on an appropriate mathematics paper, etc. Do not make a programming assignment for all students.

Grade Distributions. The School of Mathematics does not impose on instructors any requirements regarding the distribution of final grades in the course. Nevertheless, instructors are counseled to be mindful of standards and expectations that have been set for mathematics courses over an extended period of time. Moreover, the School collects and reports grading distributions in *every* section of *every* course we offer. These reports may be accessed at:

https://www.math.gatech.edu/academics/undergraduate/grade-distributions

A cumulative review of results for Math 3012 would show typical grading patterns of this nature:

A	B	C	D	F	W
25–35%	25–35%	25–40%	5–10%	0–5%	0–5%

If instructors anticipate that their grades may differ substantially from these patterns, they are advised to speak, as soon as possible, with their faculty mentors, or a member of the School admininstration (Director of Teaching Effectiveness, Director of Postdoctoral Teaching Effectiveness, Undergraduate Coordinator, Graduate Director).

Test Archives. There is a strong tradition at Georgia Tech of providing students with access to an archive of tests from previous semesters. Instructors

who teach Math 3012 more than once should build such an archive of their own tests. However, many instructors encourage their students to consult archives maintained by others, and any Math 3012 instructor is more than welcome to point their students to my archive which can be accessed at:

http://people.math.gatech.edu/ trotter/math-3012-Fa14/toppage.html

You may note that some of the solution sets are in an "expanded" form and as such they also provide supporting material for the course. These solutions go into far greater depth than students are expected to provide on a test. On the other hand, some solution sets are handwritten and more closely match what a student would actually do. Also, note that this archive is maintained outside of T-Square, so that any student or instructor can access it. Normally, access to T-Square is limited to students enrolled during the current semester.

Recommended Text. In recent years, most instructors have used *Applied Combinatorics* by Keller and Trotter. This text is available online and is free for Georgia Tech students. Understandably, I think it is a good choice. Students can print sections or entire chapters if they wish to do so, and students who prefer a complete bound copy can purchase one at the bookstore. The on-line version can be accessed here:

http://people.math.gatech.edu/ trotter/book.pdf

Other instructors have used *Discrete and Combinatorial Mathematics: An Applied Introduction, Fifth Edition* by Grimaldi. However, the detailed syllabus guidance given below will work with either choice.

Part 2. Math 3012: Recommended Topics, Schedule and Comments on Presentation.

The outline which follows has been developed for the Fall 2014 semester. Classes start on Monday, August 18 and end on Friday, December 5. Accordingly, there are 16 weeks of classes. Here are the key dates for the Fall 2014 semester, recognizing that obvious changes will be necessary for other terms.

1. August 22. Last date to add class or drop without penalty.
2. September 1. Labor Day holiday.
3. September 18. Test 1 (September 19 for M-W-F).
4. October 13–14. Fall recess.
5. October 23. Test 2 (October 24 for M-W-F).
6. November 25. Test 3 (November 24 for M-W-F).
7. November 27–28. Thanksgiving holiday.
8. December 1–5. Dead week (no exams).

Week-by-week schedule.

1. Week 1. Quick treatment of material from Chapter 1, including an overview of combinatorics using easily understood problems, some of which will be solved during the term. Chapter 2, including strings, functions and distributions, distinct and non-distinct objects and cells.
2. Week 2. Chapter 2 continued. Permutations and combinations, subsets and binomial coefficients. Combinatorial proofs, identities and lattice paths, for example. Binomial theorem and multnomial coefficients.
3. Week 3. Chapter 3. First look at induction and recursion. Recursive formulas considered as solutions. Euclidean algorithm. Sorting. Proofs by induction.
4. Week 4. Chapter 4. Pigeon-hole principle. Notion of a verifiable certificate and first look at measuring complexity of an algorithm.

5. Week 5. Chapter 5 on graph theory. Paths, cycles, components, trees, labelled and unlabelled graphs, subgraphs and induced subgraphs. Euler circuits. Hamiltonian cycles. Dirac's theorem. Test 1 at end of the week.

6. Week 6. Graph theory continued. Chromatic number, clique size, planarity, Euler's formula, interval graphs, perfect graphs. Number of labelled trees. Test 1 at end of the week.

7. Week 7. Chapter 6 on posets. Cover graphs and comparability graphs. height and width, Dilworth's theorem and dual form. Linear extensions and connections with sorting.

8. Week 8. Posets continued. Subset lattices and Sperner's theorem. Interval orders and representations.

9. Week 9. Chapter 7 on inclusion-exclusion. Emphasis on three examples: derangements, surjections and euler ϕ-function.

10. Week 10. Chapter 8 on generating functions with emphasis on ordinary generating functions. Examples illustrating use of "real functions" where calculus techniques are used and others where convergence is ignored. Partitions of an integer. Test 2 at end of the week.

11. Week 11. Chapter 9 on recurrence equations. Solution of constant coefficient advancement operator equations. General and particular solutions. Analogy with partial fractions from calculus and constant coefficient linear differential equations.

12. Week 12. Chapter 10 on probability. Brief treatment of basic ideas. Conditional probability and independent events, Bernoulli trials, discrete random variables and linearity of expectation. May have to omit more advanced topics, such as variance, standard deviation and measures of central

13. Week 13. Chapter 11 on applications of probability. Ramsey theory. Probabilistic method.

14. Week 14 Chapter 12 on graph algorithms. Key lemma on spanning trees. Linear algebra (and matroid analogies). Kruskal's and Prim's algorithms. Depth and breadth first search. Djikstra's algorithm.

15. Week 15. Chapter 13 on network flows. Linear programming basics: integer problems have rational solutions. Network flow problems. Cuts and flows with min-max overview. Ford-Fulkerson algorithm.

16. Week 16. Chapter 14. Combinatorial applications of network flow problems. Emphasis on how 0–1 constraints imply combinatorial implications of solutions. Menger's theorem. Bipartite matching. Algorithm for Dilworth's theorem.

Part 3. Suggestions on Course Organization and Presentation.

1. Strings are a good starting point. I like to emphasize attributes and properties of strings that can be effectively computed, and throughout the course, I make frequent comments about whether a task can be carried out—with the assistance of a state-of-the-art computer. Even with steady advances in computing power (and even with a real breakthrough), we will live with a boundary of what can be done and what can not.

2. Induction and recursion are fundamentally important topics, both from a theoretical perspective and for applications. For the CS types, I use language and examples which is suggestive of computer programs and I encourage them to look at the sample programs and header files developed as supporting material. Even with no programming back ground, students should be comfortable with $n * 1 = n$ and $n * (k + 1) = n * k + n$ as a definition of multiplication. Although I never assign a programming task to the entire class, I do pause and comment on data structure or programming sublteties, such as pointing out that Kruskal's spanning tree algorithm seems to call for a sorting subroutine, while Prim's algorithm requires heaps. Also, I comment on avoiding recursive calls such as solving the diaphantine equation $am + bn = \gcd(m, n)$ with a loop and avoiding recursive calls on memory in general.

3. I think it is important for students to be able to set up recursive solutions for problems. Note that this chapter comes well before we study closed form solutions.

4. The next time I teach the course, I'm going to expand a bit the material on combinatorial proofs. The insights students gain with this approach pays off later.

5. I frequently use examples to illustrate how one defends the correctness of an answer with a "certificate." Of course, this is done in a very informal setting but students should understand that an algorithm can be proven to be correct just as they should understand that an answer can be shown to be correct—without knowing how it was obtained. So an affirmative answer to the question of whether a graph is 3-colorable or whether it has a hamiltonian cycle can be done via a certificate. I always stress that nobody knows an effective strategy for settling these

two questions—as a contrast to the euler circuit issue. So naturally, I think that the material from the Chapter on Combinatorial Basics is a small but essential fraction of the course.

6. I see the material on graph theory as one area where instructors can exercise some personal choice. I like to do euler circuits, some sufficient conditions for hamiltonian cycles, chromatic number, planar graphs (including Euler's formula) and Turán's theorem. I also do interval graphs because I want to set up the greedy coloring algorithm as a precursor for Dilworth's theorem. But typically, I add a few topics on graph theory. I might do Brooks' theorem (with a sketch of Lovász's proof), and since I do interval graphs, I might also do interval numbers of graphs, showing that $i(T) \leq 2$ when T is a tree, or even $i(K_{m,n}) = \lceil (mn+1)/(m+n) \rceil$ (at least the lower bound part). Kuratowski's theorem is always discussed but this is one of very few theorems that I don't make any attempt to prove. In the discussion of planar graphs, I always prove Euler's formula. Sometimes I do the full proof for counting labelled graphs and sometimes not. But I always give at least one proof of triangle-free graphs with large chromatic number, usually via shift graphs, but sometimes also via the classic constructions.

7. Admittedly, this is a personal perspective, but I feel that posets are important discrete structures and very worthy of study by students at this level. I always introduce them to the concept of a comparability graph and show them how one can determine whether a graph is a comparability graph. I do Dilworth's theorem (Perles' proof) to illustrate how on the surface we have an existence proof without a clear approach to an algorithm. On the other hand, Mirsky's theorem (dual Dilworth) has a proof which *is* the algorithm. I introduce them to interval orders and the problems of finding the representation using the least number of end points. This is very good algorithmic stuff. They learn about Fishburn's characterization (excluding $\mathbf{2}+\mathbf{2}$) and how this test is efficiently made. Via the connection with comparability graphs, you then have an effective algorithm for recognizing interval graphs. In turn, the link with interval graphs provides a solution for the Dilworth problem in the case of interval orders.

8. The last few times I've taught Math 3012, I've used the Fulkerson proof of Dilworth's theorem as a "capstone" event. To understand this result, they have to have mastered network flows, the combinatorial conversion when capacities are 0–1, the extraction of information from

the halting condition including the construction of the chain partition and identifying a maximum antichain. To my thinking, any student who has this bundle of skills and concepts in their toolkit is ready for more challenging topics in future courses.

9. I do inclusion-exclusion very quickly. I use the three standard examples. Surjections and derangements illustrate the case when the coefficients depend only on the number of properties, while the Euler ϕ-function illustrates the case where the formula can be interpreted and rewritten in a way that collapses an otherwise exponential number of terms. Of course, this also allows me to digress and talk about factoring, one of my favorite side streets.

10. The treatment of ordinary generating functions in our book is satisfactory but probably should be expanded. In presenting this material, I try to be faithful to the concept that generating functions use convergence when it helps and ignore it when it doesn't. So I always do at least one example using Taylor series, evaluating coefficients using derivatives, etc. However, I spend most of the time treating generating functions in the formal algebraic setting without regard to any issue of convergence. The central example of proving that the number of partitions into odd parts is equal to the number of partitions into distinct parts always makes a good example. I've challenged students with finding a bijective proof and some have just gone on-line and found one. But I've yet to get a student who did it on their own.

11. I take advantage of Georgia Tech students' background in treating advancement operator equations (some prefer the term difference equations) and always speak about the solution space. I draw analogies with partial fractions and differential equations (on the last topic, not all students have seen this material, so you must exercise some care). I give a full proof for the solution for $p(A) = f$, working over the field of complex numbers.

12. Over the years, I've gone back and forth as to how much probability was included in the course. This is another topic where some variation is to be expected.

13. In some years, I've included a brief treatment of Ramsey's theorem (just the graph version) with a hint as to the difficulty of computing $R(n, n)$. And I typically sketch the classic probabilistic lower bound—

although only a small fraction of my students really understand this part.

14. My treatment of spanning trees is mathematically correct. I first prove the basic lemma involving partial trees and specified components and then show how Kruskal and Prim follow. Also, in the presentation I at least set up the matroid framework, speaking of bases, independent sets and exchange properties.

15. I also prove correctness for Dijkstra and I emphasize the breadth-first search aspect of the algorithm.

16. In my presentation of network flows and the Ford-Fulkerson algorithm, I emphasize the "primal-dual" relationship with Dijkstra. Here, I am setting the stage for students who will study optimization at a more advanced level. Also, I emphasize the role of Dijstra in insuring that the number of iterations depends only on the size of the network and not the capacities. And I emphasize the notion of a class of LP problems posed in integers which has integer solutions. Again, this is setting framework for future courses.

17. I think the material on combinatorial applications of network flows is absolutely essential. As mentioned above, I use the bipartite matching problem and its application to Dilworth as a capstone event.

18. I have rarely done much with Polya counting. Also, I don't do anything with block designs or combinatorial geometry. All three are nice topics but in my view, they are not as central for a course at this level and for this group of students.

19. In most semesters, I take two or three lectures at random points in the term, and discuss something just for "fun". I've done game coloring, on-line algorithms, stable matchings, extremal set theory, among others. I've also presented a discussion of recent research results, without proofs, in instances where the import of the result is accessible. These topics are typically not covered on exams.

20. While my classroom presentations are mathematically correct, I am very prudent in what I ask Math 3012 students to prove on exams. In recent years, I have had good results from asking true-false questions to draw out whether they really understand what a theorem is saying. For example, why a graph with 1028 vertices and 4291 edges non-planar. I've found that students don't like True-False questions, at

least they resist them. But if carefully designed, they can help to draw out details.

21. I've settled into a pattern of composing *very* comprehensive tests and final exams. I don't go in for surprises and admit that some might look over my test archive and find that I am a bit predictable and not very exciting. This is by choice and not by accident. If we've covered it in class, it *will* be covered on the exam—with the obvious exception of the occasional bonus topic. My thinking is that I want to be certain that a student who does well does indeed know the material.

    It is of course possible to design a test which covers only a fraction of the material and be certain that a student who does well is a strong student. But I am less comfortable in making judgments on a student who just doesn't get to the bottom of a subtle problem. For this reason, my tests are lengthy but straightforward.

22. I try hard to identify and challenge top students. They can usually be spotted from class comments and questions, which I strongly encourage and even go to some length to promote. But sometimes a quiet talent goes unnoticed until the first test. In class, when we come to a subtle point, I will sometimes stop and give a "challenge" problem. I'll explain that this is not part of the regular homework, but students who find a solution are asked to discuss it with me before or after class or during office hours. I've had some success in drawing out the best students in this way.