Mathematics for Computer Science: Math essentials for Computer Science

Organizer:	ACM UTEC	**Time:**	13:00 – 16:00
Email:	acm@utec.edu.pe	**Place:**	A1001, A705, A806, A808 @UTEC

1 Course Pages

You can find all the resources that will be used or produced throughout the course in the GitHub Repository.

- https://github.com/acm-utec/CS1D01-CS1D02-Training-camp-Math-essentials-for-Computer-Science

2 Schedule

A lot of reading prior to each class accompanied with two three-hour reunions per week throughout the 2020-0 academic period. The desirable distribution of the 3 hours is one hour to discuss and talk about the information retrieved from the home readings and two hours to try and solve problems with your peers. It will take place from XX:00 to XX:00, but extra studying and reading by yourself will be required.

3 General objectives

- Go over some fundamental topics in Mathematics and Computer Science.
- Fill the knowledge gaps in students that have already taken the course.
- Raise awareness of the importance of some courses in Computer Science (CS1D01, CS1D02, CS2101, CS2102).
- Emphasize the significance of hard work, curiosity and intellectual vitality.

4 Prerequisites

No specific requirements are necessary, just the desire to learn. At the end of the course you will posses a fairly good understanding of CS1D01, CS1D02 and the basics of CS2101 and CS2102. Bear in mind, however, that Mathematics are key in the entirety of Computer Science so strengthening those basic skills will be of great use in the course of the following years.

5 Methodology

We will assume **no previous knowledge on the topic**, so we encourage all students in Computer Science (and from other careers, if interested) to participate. It's important to know that this is neither a **workshop nor class**: No one will *teach* anything in the traditional sense, but rather the focus is on learning from one's peers. This will be fairly intensive, but what is important is that if you don't understand something don't give up. Your friends and peers will be there to help.

We understand that 9 weeks is a very short amount of time to cover what seems to be 4 courses of topics, counting the aforementioned assumptions of level of comprehension, so we are tackling this problem by taking advantage of two main things: 1) the unique experience of individuals with different learning strategies and resources, and 2) the exchange of ideas between those unique individuals. In order to maximize these effects, the class will also be split in groups. The reason for this decision is that sometimes the amount of people that participate in exchange of ideas in traditional classes is very small. Those people tend to get a fairly good understanding by actively learning, but the remaining people don't. Dividing a class in small groups can lessen the pressure of asking questions or expressing one's own view while also simplifying the group dynamic, since helping a small group is much more manageable than helping a big one.

The specific details of the methodology can be summarized briefly in the following figure:

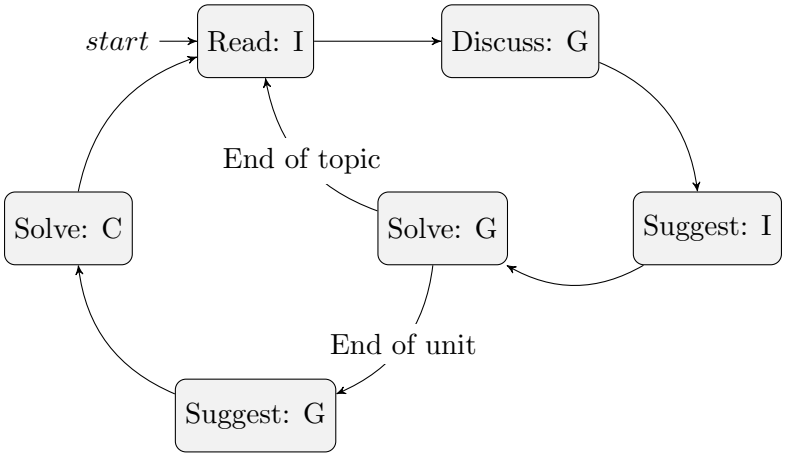

Figure 1: The cyclic process of peer learning.

I: Individual G: Group C: Class

(We just wanted to show off our LaTeX skills.)

At the start of each unit, the class will be divided into random groups of 3 to 4 people. Before going through the topics estimated to cover for that unit, it is highly recommended you discuss briefly how each member of your group usually studies. The common structure is that of Figure 1, but your group might prefer to tweak it to your own needs (and who knows, it might suit you better!). At any rate, this is how Figure 1 should be understood:

1) **Read I(ndividually)**: Pick a book of the References section or some other source (YouTube, ...) and start to read. You should be taking notes of the concepts you find most important or exercises you find interesting or hard to solve (a ton of exercises can also be found on the online courses cited in the References section, don't be afraid to see what MIT takes as homework!).

2) **Discuss in G(roup)**: After getting a decent understanding of the topic, try to discuss it with your group. How does your understanding differ from that of other members? This is what we call *homogenizing* the group - but really, that's only a fancy way of saying that the entire group should have around the same level of understanding (Note that the unique experience of the individual has now become the unique experience of the group!).

3) **Suggest I(ndividually)**: It's time to list all the exercises and concepts you've found important throughout `step 1`. Try to build a comprehensive list - use Markdown, LaTeX or plain old pen and paper.

4) **Solve in G(roup)**: Try to solve the exercises and questions left in the previous two steps. Don't take too much time on things you don't understand. If something just doesn't work, take note of it

(this will be important later). If you've reached the end of the unit, you can review some exercises or concepts that remain unclear (while waiting for others to finish). Once everyone is finished, go to `step 4`. Otherwise, if you've reached the end of a topic, go to `step 1` with the next topic.

5) **Suggest in G(roup)**: Now, pick the questions and concepts that you think were most useful for your learning (that means you solved and understood them) **and** also those that you, as a group, couldn't. **We cannot stress enough how important this is**.

6) **Solve in C(lass)**: The entire class will try to solve all the exercises and review the concepts that every group deemed important. Hopefully there's a group that specialized in something that yours didn't - that's good! No one can understand everything perfectly, so don't be embarrassed. Everyone now benefits from the unique experience of each group. If everything gets cleared up, the unit is finished, groups are be shuffled randomly (don't worry, you can go for a coffee with your new friends after the session ends) and the whole process starts again at `step 1`. If something is missing, we'll add it to the list of *topics-that-no-one-understood-but-should-be-understood* and schedule a visit with Prof. José Miguel Renom to solve those doubts. Time and date can be out of the regular schedule so we'll make the appropriate arrangements and let you know.

At the end, the overall success will depend on the exchange of ideas of each participant, so don't be shy. **Your perspective is important**.

6 Course Outline

The subjects will be separated in Home Hours (HH) and Classroom Hours (CH). As explained before, in this course you won't be taught directly, so it's important to highlight the differences between those two. Home hours are the hours of work you give to studying outside of the scheduled in a classroom. These and the classroom hours are equally important and have to be taken into consideration when organizing your week, as its expected of every student to do the specific readings before the "class". The amount of Home hours depends on the subject, as you will see in this section. This is the minimum amount of weekly hours you are expected to study, but of course you are not limited to it. The classroom hours take place inside a classroom at UTEC with the rest of your peers, where questions about the readings can be made and exercises will be resolved and discussed. We will have 6 hours of weekly classroom time.

The organization will be as follows:

I: Logic and proofs (18 hours; 4 HH and 12 CH) Week 1 and 2

- Examine and explore the concepts of propositions, logical deduction and axioms, along with some basic ways of organizing proofs.
- Master the expression of propositions using logical formulas.
- Understand how logic propositions can be used to model situations or applications in life.
- Describe the structure of different proof techniques.
- Be able to determine which type of demonstration is more suitable for a given problem.
- Understand the relationship between strong and weak induction.

- Propositional logic.
- Logical connectors.
- Truth tables.
- Normal forms (conjunctive, disjunctive).
- Predicates and quantifiers.
- Inference, equivalence, conversion, inverse, contrapositive, negation and contradiction.
- Proof methods and strategies (by cases, contradiction, direct proofs, induction).
- Induction variants (weak, strong, structural, prefix)

Readings: [Ros19] (Ch. 1), [Leh17] (Ch. 1, 2, 3), [URSS1] (Ch. 1), [Gri18] (Ch. 2, 15), [Epp18] (Ch. 2, 3, 4)

II: Basic structures (18 hours; 4 HH and 12 CH) Week 3 and 4

- Explain with examples the basic terminology of functions, relations and sets
- Perform operations associated with sets, functions and relationships
- Be familiar with mathematical concepts such as sets and functions
- Explore proofs on infinite sets
- Be able to find and express closed-form expressions for commonly-occurring sums and products

- Sets
- Functions
- Definition and properties of relations
- Methods of representation
- Closures of relations
- Equivalence relations
- Partial ordering relations
- Sequences and sums
- Matrices

Readings: [Ros19] (Ch. 2, 9), [Leh17] (Ch. 4, 9), [URSS1] (Ch. 1, 3), [Gra94] (Ch. 2), [Epp18] (Ch. 1, 6, 7, 8)

III: Counting (18 hours; 4 HH and 12 CH) Week 5 and 6	
• Apply counting arguments such as the sum and product rules, inclusion-exclusion principle and arithmetic/geometric progressions. • Apply generating functions to solve counting problems using algebra formula simplification. • Be able to compute permutations and combinations in a given set and interpret the results. • Apply counting arguments, including rules of the product and of the sum, principle of inclusion, exclusion and arithmetical/geometric progressions	• Basic counting techniques – Counting and cardinality of a set – Sum and product rule – Inclusion-exclusion principle – Arithmetic and Geometric progressions • The Pigeonhole Principle • Permutations and combinations – Basic definitions – Pascal's identity – Binomial Theorem • Generating functions
Readings: [URSS1] (Ch. 1, Ch. 4), [Ros19] (Ch. 6, 8), [Leh17] (Ch. 13, 14, 15), [Gra94] (Ch. 5, 7), [Epp18] (Ch. 9)	

IV: Recursion (12 hours; 6 HH and 6 CH) Week 7	
• Be able to solve recurrences using counting arguments and other proof techniques. • Recursively examine defined data types. • Analyze the performance of recursive algorithms.	• Recursive definitions • Applications of recurrence relations • Solving linear recurrence relations • Divide-and-Conquer Algorithms and Recurrence Relations
Readings: [Ros19] (Ch. 5), [Leh17] (Ch. 5, 6), [URSS1] (Ch. 1, 4), [Epp18] (Ch. 5)	

V: Graphs (9 hours; 3 HH and 6 CH) Week 8	
• Illustrate by means of examples the basic terminology of graph theory, and some of the properties and special cases of each type of graphs • Understand the difference between each type of graphs.	• Basic Definitions – Types of Graphs – Paths and Cycles – Graph Connectivity – Subgraphs – Complete Graph and Graph Complement – Graph Isomorphism • More Graph Structures – Euler Trails and Circuits – Hamiltonian Paths – Planar Graphs – Graph Coloring • Applications – Graph Representations – Shortest Path Algorithms
Readings: [Ros19] (Ch. 10), [Leh17] (Ch. 9, 11, 12), [Epp18] (Ch. 10)	

VI: Trees (9 hours; 3 HH and 6 CH) Week 9	
• Demonstrate how graphs and trees concepts appear on different data structures, algorithms, and proof and counting methods. • Explain how to build an expansion tree of a graph. • Demonstrate various methods of walking through trees and graphs. • Understand the most-known algorithms related to graphs and trees.	• Trees and Forests • Rooted Trees • Minimum-Spanning Trees • Arborescence • Applications – Traversal Algorithms on Trees – Breadth First Search (BFS) and Depth First Search (DFS.)
Readings: [Ros19] (Ch. 11), [Leh17] (Ch. 11), [Epp18] (Ch. 10)	

7 Class Policy

- Full attendance is not expected, but is desirable. This is a fast-paced study group and although we care about the full understanding of each and every member, we won't hold back for those that don't

come.

8 Main References

This is only a limited list of various interesting and useful books and online resources that will be touched upon during the course. You should consult them periodically. They are in order of relevance in the course syllabus structure, but each and every one of them provides rich insight into the topics.

8.1 Books

`[NOTE: You can find the books in the link in the GitHub Course Page. They are encrypted, so ask one of the organizers for the indecipherable password.]`

[Ros19] Rosen, K. (2019). *Discrete Mathematics and its Applications* (8th ed.). New York, NY: McGraw-Hill.

[Leh17] Lehman, E., Leighton, F. & Meyer, A. (2017). *Mathematics for Computer Science.* Hong Kong: Samurai Media Limited.

[Gri18] Grimaldi, R. (2018). *Discrete and Combinatorial Mathematics: An Applied Introduction.* New York, N.Y: Pearson.

[Gra94] Graham, R., Knuth, D. & Patashnik, O. (1994). *Concrete Mathematics: A Foundation for Computer Science* (2nd ed.). Reading, Mass: Addison-Wesley.

[Tao06] Tao. T. (2006). *Analysis I.* Hindustan Book Agency.

[Cor09] Cormen, T. (2009). *Introduction to algorithms* (3rd ed.). Cambridge, Mass: MIT Press.

[Kle06] Kleinberg, J. & Tardos, E. (2006). *Algorithm design.* Boston: Pearson/Addison-Wesley.

[Epp18] Epp, S. (2018). *Discrete Mathematics with Applications.* Delhi, India: Cengage Learning India Private Limited.

[URSS1] Evnin, Alexandr Y., et al (2015). *Problemas de matemática discreta: más de 400 problemas con soluciones detalladas* Moscu: Krasan, Print.

[URSS2] Miélnikov O. I., Carlos D. Hernández, and Juan E. Pérez. *Teoría de grafos: En problemas recreativos resueltos.* Moscu: URSS, 2011. Print.

[URSS3] Beriézina L. Y. (2013) *Grafos y sus aplicaciones: una introducción a la teoría de grafos.* Moscu: URSS. Print.

[Gal11] Gallier, J. (2011). Discrete mathematics. New York: Springer.

8.2 Courses

- Tom Leighton, and Marten Dijk. 6.042J *Mathematics for Computer Science. Fall 2010.* Massachusetts Institute of Technology: MIT OpenCourseWare. Recovered at: `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/`
- José Fiestas, and José Miguel Renom. CS1D01 *Estructuras Discretas I* Universidad de Ingeniería y Tecnología: UTEC. Recovered at: `https://clei2004.spc.org.pe/Peru/CS-UTEC/Plan%202018/syllabi/CS1D01-ES.pdf`
- José Fiestas, and José Miguel Renom. CS1D02 *Estructuras Discretas II* Universidad de Ingeniería y Tecnología: UTEC. Recovered at: `https://clei2004.spc.org.pe/Peru/CS-UTEC/Plan%202018/syllabi/CS1D02-ES.pdf`

- Alistair Sinclair, and Yun S. Song. CS70 *Discrete Mathematics and Probability Theory*: Berkeley. Recovered at: `http://inst.eecs.berkeley.edu/~cs70/fa16/`
- Amy Liu. CS103: *Mathematical Foundations of Computing* Recovered at: `http://web.stanford.edu/class/cs103/`

9 Collaborators

- Claudia Noche `cnoche@acm.org`
- Stephano Württele, `swurtteleigari@acm.org`
- Diego Linares, `dlinares@acm.org`
- Giordano Alvitez, `giordano.alvitez@utec.edu.pe`
- Diego Cánez, `dcanez@acm.org`
- Jorge Rios, `jorgerios3@acm.org`
- Rodrigo Morales, `rodrigomorales5@acm.org`
- César Salcedo, `cesarsalcedo@acm.org`
- Alejandro Tang, `alejandrotang@acm.org`

Special thanks to PhD. Yamilet Serrano, PhD. José Miguel Renom,PhD. Nancy Camarena Espinoza and ICPC World Finalist Miguel Miní for all the feedback that went into the creation of this work.