Discrete Mathematics – Toán Rời Rạc

Nguyễn Quản Bá Hồng*

Ngày 30 tháng 12 năm 2024

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *Discrete Mathematics – Toán Rời Rạc.*
  PDF: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/discrete_mathematics/NQBH_discrete_mathematics.pdf.
  TEX: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/discrete_mathematics/NQBH_discrete_mathematics.tex.

Mục lục

1 Basic Discrete Mathematics

- Reddit/mathematics books recommendations.
- Reddit/Good discrete math book recommendations?

Resources – Tài nguyên.

1. [GKP94]. Ronald L. Graham, Donald E. Knuth, Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. [241 Amazon ratings][1862 Goodreads ratings]

   Amazon review. This book introduces mathematics that supports advanced computer programming & analysis of algorithms. Primary aim of its well-known authors is to provide a solid & relevant base of mathematical skills – skills needed to solve complex problems, to evaluate horrendous sums, & to discover subtle patterns in data. An indispensable text & reference not only for computer scientists – authors themselves rely heavily on it! – but for serious users of mathematics in virtually every discipline.

   Concrete Mathematics is a blending of CONtinuous & disCRETE mathematics. "More concretely," authors explain, "it is the controlled manipulation of mathematical formulas, using a collection of techniques for solving probelms." Subject matter is primarily an expansion of Mathematical Preliminaries sect in Knuth's classic Art of Computer Programming, but style of presentation is more leisurely, & individual topics are covered more deeply. Several new topics have been added, & most significant ideas have been traced to their historical roots. Book includes >500 exercises, divided into 6 categories. Complete answers are provided for all exercises, except research problems, making book particularly valuable for self-study.

   Major topics include: *sum, recurrences, integer functions, elementary number theory, binomial coefficients, generating functions, discrete probability, asymptotic methods.* This 2e includes important new material about mechanical summation. In

*A Scientist & Creative Artist Wannabe. E-mail: nguyenquanbahong@gmail.com. Bến Tre City, Việt Nam.

response to widespread use of 1e as a ref book, bibliography & index have also been expanded, & additional nontrivial improvements can be found on almost every page. Readers will appreciate informal style of Concrete Mathematics. Particularly enjoyable are marginal graffiti contributed by students who have taken courses based on this material. Authors want to convey not only importance of techniques presented, but some of fun in learning & using them.

About the Author. Ronald L. Graham (1935–2020) was for many years Chief Scientist at AT&T Labs Research. He was also a Professor of Computer & Information Science at University of California, San Diego, & a former President of American Mathematical Society & Mathematical Association of America. He was the coauthor of 7 other mathematics books.

Donald E. Knuth is Professor Emeritus of The Art of Computer Programming at Stanford University. His prolific writings include 4 volumes on *The Art of Computer Programming*, & 5 books related to his TEX & METAFONT typesetting systems.

Oren Patashnik is a member of research staff at Center for Communications Research, La Jolla, California. He is also the author of BibTEX, a widely used bibliography processor.

Preface.

> *"Audience, level, & treatment – a description of such matters is what prefaces are supposed to be about."* – P. R. Halmos

This book is based on a course of same name that has been taught annually at Stanford University since 1970. ≈ 50 students have taken it each year – juniors & seniors, but mostly graduate students – & alumni of these classes have begun to spawn similar courses elsewhere. thus time seems ripe to present material to a wide audience (including sophomores).

A dark & stormy decade when Concrete Mathematics was born. Long-held values were constantly being questioned during those turbulent years; college campuses were hotbeds of controversy. College curriculum itself was challenged, & mathematics did not escape scrutiny. John Hammersley had just written a thought-provoking article "On enfeeblement of mathematical skills by 'Modern Mathematics' & by similar soft intellectual trash in schools & universities" [176]; other worried mathematicians [332] even asked, "Can mathematics be saved?" 1 of present authors had embarked on a series of books called *The Art of Computer Programming*, & in writing 1st volume he (DEK) had found: there were mathematical tools missing from his repertoire; mathematics he needed for a thorough, well-grounded understanding of computer programs was quite different from what he'd learned as a mathematics major in college. So he introduced a new course, teaching what he wished somebody had taught him.

> "People do acquire a little brief authority by equipping themselves with jargon: they can pontificate & air a superficial expertise. But what we should ask of educated mathematicians is not what they can speechify about, nor even what they know about existing corpus of mathematical knowledge, but rather what can they now do with their learning & whether they can actually solve mathematical problems arising in practice. In short, look for deeds not words." – J. Hammersley

Course title "Concrete Mathematics" was originally intended as an antidote to "Abstract Mathematics," since concrete classical results were rapidly being swept out of modern mathematical curriculum by a new wave of abstract ideas popularly called "New Math." Abstract mathematics is a wonderful subject, & there's nothing wrong with it: It's beautiful, general, & useful. But its adherents had become deluded that the rest of mathematics was inferior & no longer worthy of attention. Goal of generalization had become so fashionable that a generation of mathematicians had become unable to relish beauty in particular, to enjoy challenge of solving quantitative problems, or to appreciate value of technique. Abstract mathematics was becoming inbred & losing touch with reality; mathematical education needed a concrete counterweight in order to restore a healthy balance.

When DEK taught Concrete Mathematics at Stanford for 1st time, he explained somewhat strange title by saying that it was his attempt to each a math course that was hard instead of soft. He announced: contrary to expectations of some of his colleagues, he was *not* going to teach Theory of Aggregates, nor Stone's Embedding Theorem, nor even Stone-Čech compactification. (Several students from civil engineering depart got up & quietly left room.)

> *"The heart of mathematics consists of concrete examples & concrete problems."* – P. R. Halmos

Although Concrete Mathematics began as a reaction against other trend, main reasons for its existence were positive instead of negative. & as course continued its popular place in curriculum, its subject matter "solidified" & proved to be valuable in a variety of new applications. Meanwhile, independent confirmation for appropriateness of name came from another direction, when Z. A. Melzak published 2 volumes entitled *Companion to Concrete Mathematics*.

Material of concrete mathematics may seem at 1st to be a disparate bag of tricks, but practice makes it into a discipline set of tools. Indeed, techniques have an underlying unity & a strong appeal for many people. When another 1 of authors (RLG) 1st taught course in 1979, students had such fun that they decided to hold a class reunion a year later.

Concrete Mathematics is a bridge to abstract mathematics.

But what exactly is Concrete Mathematics? A blend of CONtinuous & disCRETE mathematics. More concretely, controlled manipulation of mathematical formulas, using a collection of techniques for solving problems. Once you, reader, have learned material in this book, all you will need is a cool head, a large sheet of paper, & fairly decent handwriting in order to evaluate horrendous-looking sums, to solve complex recurrence relations, & to discover subtle patterns in data. You will be so fluent in algebraic techniques that you will often find it easier to obtain exact results than to settle for approximate answers that are valid only in a limiting sense.

"The advanced reader who skips parts that appear too elementary may miss more than the less advanced reader who skips parts that appear too complex." – G. Pólya

Major topics treated in this book include sums, recurrences, elementary number theory, binomial coefficients, generating functions, discrete probability, & asymptotic methods. Emphasis is on manipulative technique rather than on existence theorems or combinatorial reasoning; goal: for each reader to become as familiar with discrete operations (like greatest-integer function & finite summation) as a student of calculus is familiar with continuous operations (like absolute-value function & infinite integration).

Notice: this list of topics is quite different from what is usually taught nowadays in undergraduate courses entitled "Discrete Mathematics." Therefore subject needs a distinctive name, & "Concrete Mathematics" has proved to be as suitable as any other. (*We're not bold enough to try Discontinuous Mathematics*)

Original textbook for Stanford's course on concrete mathematics was "Mathematics Preliminaries" sect in *The Art of Computer Programming*. But presentation in those 110 pages is quite terse, so another author (OP) was inspired to draft a lengthy set of supplementary notes.

Present book is an outgrowth of those notes; an expansion of, & a more leisurely introduction to, material of Mathematical Preliminaries. Some of more advanced parts have been omitted; on other hand, several topics not found there have been included here so that story will be complete.

"... a concrete life preserver thrown to students sinking in a sea of abstraction." – W. Gottschalk

Authors have enjoyed putting this book together because subject began to jell & take on a life of its own before our eyes; this book almost seemed to write itself. Somewhat unconventional approaches we have adopted in several places have seemed to fit together so well, after these years of experience, that we can't help feeling that this book is a kind of manifesto about our favorite way to do mathematics. So think book has turned out to be tale of mathematical beauty & surprise, & hope: readers will share at least ε of pleasure we had while writing it.

Since this book was born in a university setting, have tried to capture spirit of a contemporary classroom by adopting an informal style. Some people think: mathematics is a serious business that must always be cold & dry; but think mathematics is fun, & we aren't ashamed to admit fact. Why should a strict boundary line be drawn between work & play? Concrete mathematics is full of appealing patterns; manipulations are not always easy, but answers can be astonishingly attractive. Joys & sorrows of mathematical work are reflected explicitly in this book because they are part of our lives.

Math graffiti: Kilroy wasn't Haar. Free the group. Nuke the kernel. Power to the n. $N = 1 \Rightarrow P = NP$.

Students always know better than their teachers, so we have asked 1st students of this material to contribute their frank opinions, as "graffiti" in margins. Some of these marginal markings are merely corny, some are profound; some of them warn about ambiguities or obscurities, others are typical comments made by wise guys in back row; some are positive, some are negative, some are zero. But they all are real indications of feelings that should make text material easier to assimilate. (Inspiration for such marginal notes comes from a student handbook entitled *Approaching Stanford*, where official university line is counterbalanced by remarks of outgoing students. E.g., Stanfords says, "There are a few things you cannot miss in this amorphous shape which is Stanford"; margin says, "Amorphous ... what the h*** does that mean? Typical of pseudo-intellectualism around here." Stanford: "There is o end to potential of a group of students living together." Graffito: "Stanford dorms are like zoos without a keeper.")

This was the most enjoyable course I've ever had. But it might be nice to summarize material as you go along.

Margins also include direct quotations from famous mathematicians of past generations, giving actual words in which they announced some of their fundamental discoveries. Somehow it seems appropriate to mix words of Leibniz, Euler, Gauss, & others with those of people who will be continuing the work. Mathematics is an ongoing endeavor for people everywhere; many strands are being woven into 1 rich fabric.

I see: Concrete mathematics means drilling.

Homework was tough but I learned a lot. It was worth every hour.

Take-home exams are vital – keep them.

Exams were harder than homework led me to expect.

Cheaters may pass this course by just copying answers, but they're only cheating themselves.

Difficult exams don't take into account students who have other classes to prepare for.

This book contains > 500 exercises, divided into 6 categories:

- **Warmups**: exercises that every reader should try to do when 1st reading material.
- **Basics**: exercises to develop facts that are best learned by trying one's own derivation rather than by reading somebody else's.
- **Homework exercises** are problems intended to deepen an understanding of material in current chapter.
- **Exam problems** typically involve ideas from ≥ 2 chaps simultaneously; they are generally intended for use in take-home exams (not for in-class exams under time pressure).

- **Bonus problems** go beyond what an average student of concrete mathematics is expected to handle while taking a course based on this book; they extend text in interesting ways.
- **Research problems** may or may not be humanly solvable, but ones presented here seem to be worth a try (without time pressure).

Answers to all exercises appear in Appendix A, often with additional information about related results. (Of course, "answers" to research problems are incomplete; but even in these cases, partial results or hints are given that might prove to be helpful.) Readers are encouraged to look at answers, especially answers to warmup problems, but only after making serious attempt to solve problem without peeking.

Have tried in Appendix C to give proper credit to sources of each exercise, since a great deal of creativity &/or luck often goes into design of an instructive problem. Mathematicians have unfortunately developed a tradition of borrowing exercises without any acknowledgment; believe: opposite tradition, practiced e.g. by books & magazines about chess (where names, dates, & locations of original chess problems are routinely specified) is far superior. However, have not been able to pin down sources of many problems that have become part of folklore. If any reader knows origin of an exercise for which our citation is missing or inaccurate, we would be glad to learn details so that we can correct omission in subsequent editions of this book.

Typeface used for mathematics throughout this book is a new design by HERMANN ZAPF, commissioned by AMS & developed with help of a committee that included `[list of names]`. Underlying philosophy of ZAPF's design: capture flavor of mathematics as it might be written by a mathematician with excellent handwriting. A handwritten rather than mechanical style is appropriate because people generally create mathematics with pen, pencil, or chalk. (E.g., 1 o trademarks of new design is symbol for zero, '0', which is slightly pointed at top because a handwritten zero rarely closes together smoothly when curve returns to its starting point.) Letters are upright, not italic, so that subscripts, superscripts, & accents are more easily fitted with ordinary symbols. This new type family has been named *AMS Euler*, after great Swiss mathematician LEONHARD EULER (1707–1783) who discovered so much of mathematics as we know it today. Alphabets include Euler text, Euler Fraktur, & Euler Script Capitals, as well as Euler Greek, & special symbols e.g. ℵ. Especially pleased to be able to inaugurate Euler family of typefaces in this book, because LEONHARD EULER's spirit truly lives on every page: Concrete mathematics is Eulerian mathematics.

Authors are extremely grateful to ANDREI BRODER, ERNST MAYR, ANDREW YAO, & FRANCES YAO, who contributed greatly to this book during years that they taught Concrete Mathematics at Stanford. Offer 1024 thanks to teaching assistants who creatively transcribed what took place in class each year & who helped to design examination questions; their names are listed In Appendix C. This book, essentially a compendium of 16 years' worth of lecture notes, would have been impossible without their 1st-rate work.

Many other people have helped to make this book a reality. E.g., wish to commend students at Brown, Columbia, CUNY, Princeton, Rice, & Stanford who contributed choice graffiti & helped to debug 1st drafts. [...]

Have tried to produce a perfect book, but we are imperfect authors. Therefore solicit help in correcting any mistakes that we've made. A reward of $2.56 will gratefully be paid to 1st finder of any error, whether it is mathematical, historical, or typographical.

> *I'm unaccustomed to this face.*
> *Dear prof: Thanks for (1) the puns, (2) the subject matter.*
> *I don't see how what I've learned will ever help me.*
> *I had a lot of trouble in this class, but I know it sharpened my math skills & my thinking skills.*
> *I would advise casual student to stay away from this course.*

A Note on Notation. Some of symbolism in this book has not (yet?) become standard. A list of notations that might be unfamiliar to readers who have learned similar material from other books, together with page numbers where these notations are explained. (See general index, at end of book, for refs to more standard notations.)

- $\ln x$: natural logarithm: $\log_e x$
- $\lg x$: binary logarithm: $\log_2 x$
- $\log x$: common logarithm: $\log_{10} x$
- $x \mod y$: remainder: $x - y\lfloor \frac{x}{y} \rfloor$
- $\{x\}$: fractional part: $x \mod 1$
- $\sum f(x)\delta x$: indefinite summation
- $\sum_a^b f(x)\delta x$: definite summation
- $x^{\underline{n}}$: falling factorial power: $\frac{x!}{(x-n)!}$
- $x^{\overline{n}}$: rising factorial power: $\frac{\Gamma(x+n)}{\Gamma(x)}$
- subfactorial: $\frac{n!}{0!} - \frac{n!}{1!} + \cdots + (-1)^n \frac{n!}{n!}$
- $\Re z$: real par: x, if $z = x + iy$

- $\Im z$: imaginary part: y, if $z = x + iy$
- H_n: harmonic number $\sum_{i=1}^{n} \frac{1}{i}$
- $H_n^{(x)}$: generalized harmonic number $\sum_{i=1}^{n} \frac{1}{i^x}$
- $f^{(m)}(z)$: mth derivative of f at z
- $\begin{bmatrix} n \\ m \end{bmatrix}$: Stirling cycle number 1st kind
- $\begin{Bmatrix} n \\ m \end{Bmatrix}$: Stirling subset number 2nd kind
- $\left\langle \begin{matrix} n \\ m \end{matrix} \right\rangle$: Eulerian number
- $\left\langle\left\langle \begin{matrix} n \\ m \end{matrix} \right\rangle\right\rangle$: 2nd-order Eulerian number
- $(a_m, \ldots, a_0)_b$: radix notation for $\sum_{k=0}^{m} a_k b^k$
- $K(a_1, \ldots, a_n)$: continuant polynomial
- $F\left(\begin{matrix} a, b \\ c \end{matrix} | z\right)$: hypergeometric function
- $\#A$: cardinality: number of elements in set A
- $[z^n]f(z)$: coefficient of z^n in $f(z)$
- $[\alpha..\beta]$: closed interval: set $\{x | \alpha \le x \le \beta\}$
- $[m = n]$: 1 if $m = n$, otherwise 0

  Remark 1. *In general, if S is any statement that can be true or false, bracketed notation $[S]$ stands for* 1 *if S is true,* 0 *otherwise.*
- $[m \backslash n]$: 1 if m divides n, otherwise 0
- $[m \backslash\backslash n]$: 1 if m exactly divines n, otherwise 0
- $[m \perp n]$: 1 if m is relatively prime to n, otherwise 0

Throughout this text, use single-quote marks ('...') to delimit text as it is *written*, double-quote marks ("...") for a phrase as it is *spoken*. Thus, string of letters 'string' is sometimes called a "string." An expression of form 'a/bc' means same as 'a/(bc)'. Moreover, $\log x / \log y = (\log x)/(\log y)$ & $2n! = 2(n!)$.

> *Prestressed concrete mathematics is concrete mathematics that's preceded by a bewildering list of notations.*
> *Also 'nonstring' is a string.*

- 9.6. Final Summations.
- Exercises.

2. [WR21]. Ryan T. White, Archana Tikayat Ray. *Practical Discrete Mathematics: Discover math principles that fuel algorithms for computer science & machine learning with Python.* [Amazon 44 ratings]

Amazon review. A practical guide simplifying discrete math for curious minds & demonstrating its application in solving problems related to software development, computer algorithms, & DS. Key Features:

- Apply math of countable objects to practical problems in computer science
- Explore modern Python libraries e.g. scikit-learn, NumPy, & SciPy for performing mathematics
- Learn complex statistical & mathematical concepts with help of hands-on examples & expert guidance

Book Description. Discrete mathematics deal with studying countable, distinct elements, & its principles are widely used in building algorithms for computer science & data science. Knowledge of discrete math concepts will help understand algorithms, binary, & general mathematics that sit at core of data-driven tasks.

Practical Discrete Mathematics is a comprehensive introduction for those who are new to mathematics of countable objects. This book will help you get up to speed with using discrete math principles to take your computer science skills to a more advanced level.

As learn language of discrete mathematics, also cover methods crucial to studying & describing computer science & ML objects & algorithms. Chaps will guide through how memory & CPUs work. In addition to this, understand how to analyze data for useful patterns, before finally exploring how to apply math concepts in network routing, web searching, & data science.

By end of this book, have a deeper understanding of discrete math & its applications in computer science, & be ready to work on real-world algorithm development & ML.

What You Will Learn.

- Understand terminology & methods in discrete math & their usage in algorithms & data problems
- use Boolean algebra in formal logic & elementary control structures
- Implement combinatorics to measure computational complexity & manage memory allocation
- Use random variables, calculate descriptive statistics, & find average-case computational complexity
- Solve graph problems involved in routing, pathfinding, & graph searches, e.g. depth-1st search
- Perform ML tasks e.g. data visualization, regression, & dimensionality reduction

Who this book is for. This book is for computer scientists looking to expand their knowledge of discrete math, core topic of their field. University students looking to get hands-on with computer science, mathematics, statistics, engineering, or related disciplines will also find this book useful. Basic Python programming skills & knowledge of elementary real-number algebra are required to get started with this book.

About the Authors. Ryan T. White, Ph.D. is a mathematician, researcher, & consultant with expertise in ML & probability theory along with private-sector experience in algorithm development & data science. Dr. White is an assistant professor of mathematics at Floria Institute of Technology, where he leads an active academic research program centered on stochastic analysis & related algorithms, heads private-sector projects in ML, participates in numerous scientific & engineering research projects, & teaches courses in ML, neural networks, probability, & statistics at undergraduate & graduate levels.

Archana Tikayat Ray s a Ph.D. student at Georgia Institute of Technology, Atlanta, where her research work is focused on ML & Natural Language Processing (NLP) applications. She has a master's degree from Georgia Tech as well, & a bachelor's degree in aerospace engineering from Florida Institute of Technology.

About the reviewer. Valeriy Babushkin is senior director of data scientist at X5 Retail Group, where he leads a team of $>$ 80 people in fields of ML, data analysis, computer vision, NLP, R&D, & A/B testing. Valeriy is a Kaggle competition grandmaster & an attending lecturer at National Research Institute's Higher School of Economics & Central Bank of Kazakhstan.

Valeriy served as a technical reviewer for books *AI Crash Course & Hands-On Reinforcement Learning with Python*, both published by Packt.

Preface. *Practical Discrete Mathematics* is a comprehensive introduction for those who are new to mathematics of countable objects. This book will help you get up to speed with using discrete math principles to take your computer science skills to another level. Learn language of discrete mathematics & methods crucial to studying & describing objects & algorithms from computer science & ML. Complete with real-world examples, this book covers internal workings of memory & CPUs, analyzes data for useful patterns, & shows how to solve problems in network routing, web searching, & data science.

Who this book is for. This book is for computer scientists looking to expand their knowledge of core of their field. University students seeking to gain expertise in computer science, mathematics, statistics, engineering, & related disciplines will also find this book useful. Knowledge of elementary real-number algebra & basic programming skills in any language are only requirements.

To get most out of this book. Knowledge of elementary real-number algebra & Python SPACE basic programming skills: main requirements for this book.

Will need to install Python – latest version, if possible – to run code in book. Will also need to install Python libraries listed in following table to run some of code in book. All code examples have been tested in JupyterLab using a Python 3.8 environment on Windows 10 OS, but they should work with any version of Python 3 in any OS compatible with it & with any modern integrated development environment, or simply a command line.

Python libraries: NumPy, matplotlib, pandas, scikit-learn, SciPy, seaborn. More information about installing Python & its libraries can be found in following links:

- Python: https://www.python.org/downloads/
- matplotlib: https://matplotlib.org/3.3.3/users/installing.html
- NumPy: https://numpy.org/install/
- pandas: https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html
- scikit-learn: https://scikit-learn.org/stable/install.html
- SciPy: https://www.scipy.org/install.html
- seaborn: https://seaborn.pydata.org/installing.html

If use digital version of this book, advise to type code yourself or access code via GitHub repository. Doing so will help avoid any potential errors related to copying & pasting of code.

Download example code files. Can download example code files for this book from GitHub at https://github.com/PacktPublishing/Practical-Discrete-Mathematics. Also have other code bundles from rich catalog of books & videos available at https://github.com/PacktPublishing/.

Download color images. provide a PDF file that has color images of screenshots/diagrams used in this book. Can download via https://static.packt-cdn.com/downloads/9781838983147_ColorImages.pdf.

Part I: Basic Concepts of Discrete math.

- 1. Key Concepts, Notation, Set Theory, Relations, & Functions. an introduction to basic vocabulary, concepts, & notations of discrete mathematics.

  This chap is a general introduction to main ideas of discrete mathematics. Alongside this, go through key terms & concepts in field. After that, cover set theory, essential notation & notations for referring to collections of mathematical object & combining or selecting them. Also think about mapping mathematical objects to 1 another with functions & relations & visualizing them with graphs. Topics covered in this chap:

  - What is discrete mathematics?
  - Elementary set theory
  - Functions & relations

  By end of chapter, should be able to speak in language of discrete mathematics & understand notation common to entire field.

  - What is discrete mathematics? Discrete mathematics is study of countable, distinct, or separate mathematical structures. A good example is a pixel. From phones to computer monitors to televisions, modern screens are made up of millions of tiny dots called *pixels* lined up in grids. Each pixel lights up with a specified color on command from a device, but only a finite number of colors can be displayed in each pixel.

    Millions of colored dots taken together form intricate patterns & give our eyes impression of shapes with smooth curves, as in boundary of following circle Fig. 1.1: boundary of a circle. But if zoom in & look closely enough, true "curves" are revealed to be jagged boundaries between differently colored regions of pixels, possibly with some intermediate colors: Fig. 1.2: A zoomed-in view of circle. Some other examples of objects studied in discrete mathematics are logical statements, integers, bits & bytes, graphs, trees, & networks. Like pixels, these too can form intricate patterns that we will try to discover & exploit for various purposes related to computer & data science throughout course of book.

    In contrast, many areas of mathematics that may be more familiar, e.g. elementary algebra or calculus, focus on continuums. These are mathematical objects that take values over continuous ranges, e.g. set of numbers $x \in (0, 1)$, or mathematical functions plotted as smooth curves. These objects come with their own class of mathematical methods, but are mostly distinct from methods for discrete problems on which we will focus.

    In recent decades, discrete mathematics has been a topic of extensive research due to advent of computers with high computational capabilities that operate in "discrete" steps & store data in "discrete" bits. This makes it important for us to understand principles of discrete mathematics as they are useful in understanding underlying ideas of software development, computer algorithms, programming languages, & cryptography. These computer implementations play a crucial role in applying principles of discrete mathematics to real-world problems.

    Some real-world applications of discrete mathematics:

    * **Cryptography.** Art & science of converting data or information into an encoded form that can ideally only be decoded by an authorized entity. This field makes heavy use of number theory, study of counting numbers, & algorithms on base-n number systems. Will learn more about these topics in *Chap. 2: Formal Logic & Constructing Mathematical Proofs*.

* **Logistics.** This field makes use of graph theory to simplify complex logistical problems by converting them to graphs. These graphs can further be used to find best routes for shipping goods & services, & so on. E.g., airlines use graph theory to map their global airplane routing & scheduling. Investigate some of these issues in Chaps of *Part II: Implementing Discrete Mathematics in Data & Computer Science*.
* **Machine Learning.** Area that seeks to automate statistical & analytical methods so systems can find useful patterns in data, learn, & make decisions with minimal human intervention. Frequently applied to predictive modeling & web searches, see in *Chap 5: Elements of Discrete Probability*, & most of chaps in *Part III: Real-World Applications of Discrete Mathematics*.
* **Analysis of Algorithms.** Any set of instructions to accomplish a task is an algorithm. An effective algorithm must solve problem, terminate in a useful amount of time, & not take up too much memory. To ensure 2nd condition, often necessary to count number of operations an algorithm must complete in order to terminate, which can be complex, but can be done through methods of combinatorics. 3rd condition requires a similar counting of memory usage. Encounter some of these ideas in *Chap. 4: Combinatorics Using SciPy, Chap. 6: Computational Algorithms in Linear Algebra, & Chap. 7: Computational Requirements for Algorithms*.
* **Relational Databases.** They help to connect different traits between data fields. E.g., in a database containing information about accidents in a city, "relational feature" allows user to link location of accident to road condition, lighting condition, & other necessary information. A relational database makes use of concept of set theory in order to group together relevant information. See some of these ideas in *Chap. 8: Storage & Feature Extraction of Trees, Graphs, & Networks*.

Have a rough idea of what discrete mathematics is & some of its applications, discuss set theory, which forms basis for this field.

- Elementary set theory.

"A set is a Many that allows itself to be thought of as a One." – Georg Cantor

In mathematics, set theory is study of collections of objects, which is prerequisite knowledge for studying discrete mathematics. `SKIP FAMILIARS`

With knowledge of set theory, can now move on to learn about relations between different sets & functions, which help us to map each element from a set to exactly 1 element in another set.

- Functions & relations.

"Gentlemen, mathematics is a language." – Josiah Willard Gibbs

We are related to different people in different ways; e.g., relationship between a father & his son, relationship between a teacher & their students, & relationship between co-workers, to name just a few. Similarly, relationships exist between different elements in mathematics.

* Example: Functions in elementary algebra. Elementary algebra courses tend to focus on specific sorts of functions where domain & range are intervals of real number line. Domain values are usually denoted by x & values in range are denoted by y because set of ordered pairs (x, y) satisfying equation $y = f(x)$ plotted on Cartesian xy-plane form graph of function, as can be seen in Fig. 1.10: Cartesian xy-plane. While this typical type of functions may be familiar to most readers, concept of a function is more general than this. 1st, input or output is required to be a number. Domain of a function could consist of any set, so members of set may be points in space, graphs, matrices, arrays or strings, or any other types of elements.

  In Python & most other programming languages, there are blocks of code known as "functions," which programmers give names & will run when you call them. These Python functions may or may not take inputs (referred to as "parameters") & return outputs, & each set of input parameters may or may not always return same output. As such, important: Python functions are not necessarily functions in mathematical sense, although some of them are.

  An example of conflicting vocabulary in fields of mathematics & computer science. Next example will discuss some Python functions that are, & some that are not, functions in mathematical sense.

* Example: Python functions vs. mathematical functions. Consider `sort()` Python function, which is used for sorting lists. See this function applied to 2 lists – 1 list of numbers & 1 list of names:

```
numbers = [3, 1, 4, 12, 8, 5, 2, 9]
names = ['Wyatt', 'Brandon', 'Kumar', 'Eugene', 'Elise']
# Apply the sort() function to the lists
numbers.sort()
names.sort()
# Display the output
print(numbers)
print(names)
```

  In each case, `sort()` function sorts list in ascending order by default (w.r.t. numerical order or alphabetical order). Furthermore, can say: `sort()` applies to any lists & is a function in mathematical sense. Indeed, it meets all criteria:

  (a) domain is all lists that can be sorted.

  (b) range is set of all such lists that have been sorted.

  (c) `sort()` always maps each list that can be inputted to a unique sorted list in range.

Consider now Python function random `shuffle()`, which takes a list as an input & puts it into a random order. Just like shuffle option on your favorite music app!

```
import random
# Set a random seed so the code is reproducible
random.seed(1)
# Run the random.shuffle() function 5 times and display the outputs
for i in range(0,5):
  numbers = [3, 1, 4, 12, 8, 5, 2, 9]
  random.shuffle(numbers)
  print(numbers)
```

This code runs a loop where each iteration sets list numbers to `[3,1,4,12,8,5,29]`, applies shuffle function to it, & prints output.
In each iteration, Python function `shuffle()` takes same input, but output is different each time. Therefore, Python function `shuffle()` is not a mathematical function. It is, however, a relation that can pair each list with any ordering of itself.

- Summary. Discussed meaning of discrete mathematics & discrete objects. Furthermore, provided an overview of some of many applications of discrete mathematics in real world, especially in computer & data sciences, which will be discussed in depth in later chaps.
  Have established some common language & notation of importance for discrete mathematics in form of set notation, which will allow us to refer to mathematical objects with ease, count size of sets, represent them as Venn diagrams, & much more. Beyond this, learned about a number of operations that allow us to manipulate sets by combining them, intersecting them, & finding complements. These give rise to some of foundational results in set theory in De Morgan's laws, which we will make use of in later chaps.
  Took a look at ideas of functions & relations, which map mathematical objects e.g. numbers to 1 another. While certain types of functions may be familiar to reader from high school or secondary school, these familiar functions are typically defined on continuous domains. Since focus on discrete, rather than continuous, sets in discrete mathematics, drew distinction between familiar idea & a new one we need in this field. Similarly, showed difference between functions in mathematics & functions in Python & saw: some Python "functions" are mathematical functions, but others are not.
  In remaining 4 chaps of *Part I: Core Concepts of Discrete Mathematics*, will fill our discrete mathematics toolbox with more tools, including logic in *Chap. 2: Formal Logic & Constructing Mathematical Proofs*, numerical systems, e.g. binary & decimal, in *Chap. 3: Computing with Base n Numbers*, counting complex sorts of objects, including permutations & combinations, in *Chap. 4: Combinatorics Using SciPy*, & dealing with uncertainty & randomness in *Chap. 5: Elements of Discrete Probability*. With this array of tools, will be able to consider more & more real-world applications of discrete mathematics.

- 2. Formal Logic & Constructing Mathematical Proofs. cover formal logic & binary & explain how to prove mathematical results.

  This chap is an introduction to formal logic & mathematical proofs. 1st introduce some primary results of formal logic & prove logical statements with use of truth tables. In remainder of chap, consider most common methods of mathematical proofs (direct proof, proof by contradiction, & proof by mathematical induction) to build skills that you will need for more complex problems to come later. Cover topics:

  - Formal logic & proofs by truth tables
  - Direct mathematical proofs
  - Proof by contradiction
  - Proof by mathematical induction

  By end of chap, will have a grasp of how formal logic provides a grounding for deductive thought, will have learned how to model logical problems with truth tables, will have proved claims with truth tables, & will have learned how to construct mathematical proofs using several methods: direct proof, proof by contradiction, & proof by mathematical induction. This short introduction to mathematical proofs will help you to learn how to think like a mathematician, use powerful deductive thought, & learn later material in book.

  - Formal Logic & Proofs by Truth Tables.
  - Direct Mathematical Proofs.
  - Proof by Contradiction.
  - Proof by Mathematical Induction.
  - Summary. Introduced primary results of formal logic & proved logical statements by using truth tables. Also learned about constructing mathematical proofs using several methods, e.g. direct proofs, proofs by contradiction, & proofs by mathematical induction. In addition, these different methods for constructing mathematical proofs were accompanied by simple step-by-step examples to help you think like a mathematician & use deductive thought, which will be helpful for rest of chaps in this book. In next chap, learn about numbers in base n & perform some arithmetic operations with them. Will also learn about binary & hexadecimal numbers & their uses in computer science.

- 3. Computing with Base-n Numbers. discuss arithmetic in different numbering systems, including hexadecimal & binary.

  We are all accustomed to decimal (base-10) numbers. Introduce numbers in other bases, describe arithmetic operations with those numbers, & convert numbers from 1 base to another. Then move to binary digits (base-2), which are foundation on which all computer operations are built, develop an approach to efficient arithmetic with them, & look at some of core uses of binary, including Boolean algebra. Lastly, discuss hexadecimal (base-16) numbers & their uses in computer science. Use Python code to do some computations e.g. converting decimal numbers to binary & hexadecimal & use Boolean operators to select & view data that satisfies a certain criterion. Cover following topics:

  - Base-n numbers
  - Converting between bases
  - Binary numbers & their application
  - Boolean algebra
  - Hexadecimal numbers & their application

  By end of this chap, should be able to write numbers in different bases & convert numbers from 1 base to another. E.g., 123 is a base-10 number that can be converted into other bases, depending on need. Also learn about importance of binary & hexadecimal number systems along with their applications in computer science.

  - Understanding base-n numbers.
  - Converting between bases.
  - Binary numbers & their applications.
  - Hexadecimal numbers & their application.
  - Summary. Learned about numbers in different bases (decimal, binary, hexadecimal) & how can convert between bases. Binary numbers are a base-2 number system, whereas decimal numbers are base-10 & hexadecimal numbers are base-16, resp. Also learned about 1 very crucial application of binary number system – Boolean algebra & Boolean operators. Next chap, will learn about combinatorics, which includes study of permutations & combinations that will enable you to calculate amount of memory required to store certain kinds of data. Will learn about hashing & efficacy of brute force algorithms.

- 4. Combinatorics Using SciPy. explain how to count elements in certain types of discrete structures.

  This chap is about counting (or combinatorics), which seems simple, but rapidly gains complexity when counting number of ways to combine, order, or select various finite sets. This includes study of permutations & combinations, which can be applied to determining memory required to store various types of data.

  Apply these ideas to measure efficacy of brute-force algorithms applied to cryptography & traveling salesman problem. Cover following topics:

  - Fundamental counting rule
  - Counting permutations & combinations of objects
  - Applications to memory allocation
  - Efficacy of brute-force algorithms

  By end of chap, will be able to count various mathematical structures, distinguish between combinations & permutations, & be able to count them. Also be able to apply these ideas to practical problems in memory allocation & measure effectiveness of brute-force algorithms in code-breaking in cryptology, traveling salesman problem, & beyond. SciPy Python library as well as standard Python `math` library will be used in this chap.

  - Fundamental counting rule.
  - Counting permutations & combinations of objects. This sect is dedicated to counting orderings, or permutations, of objects in a set, as well as subsets of specified cardinalities, or combinations, of elements of some wider set. Def [permutation]: A *permutation* is a rearrangement of elements of a set.
    - Growth of factorials. Factorials grow extremely quickly. Number of permutations of just 10 elements is $10! = 3628800 > 3$ million. By time reach 20 elements, number of permutations is $20! \approx 2.43 \cdot 10^{18}$, over 2 quintillion. Many computations tools, e.g. calculators & programming languages, cannot (by default) calculate permutations if number of elements gets too high, but factorial function from math module in Python `math` does not experience much trouble, as it can calculate large factorials efficiency using some mathematical tricks, as seen:

      ```
      import math
      print(math.factorial(20))
      print(math.factorial(100))
      ```

      `math` module will be used frequently in this book. See official documentation for `math` module at https://docs.python.org/3/library/math.html.

      Sometimes, wish to count a slightly different type of permutation; e.g., example with playlists, suppose we want to randomly play only half playlist of 20 songs. Then, how many distinct permutations of a subset of 10 of 20 ares are there?

      Theorem 1. *Number of permutations of k out of n distinct elements from a set, or k-permutations:* $P_k = \frac{n!}{(n-k)!}$.

- Applications to memory allocation.
- Efficacy of brute-force algorithms.
- Summary.

- 5. Elements of Discrete Probability. cover measuring chance & basics of Google's PageRank algorithm.
  - Basics of discrete probability.
  - Conditional probability & Bayes' theorem.
  - Bayesian spam filtering.
  - Random variables, means, & variance.
  - Google PageRank I.
  - Summary.

Part II: Implementing Discrete Mathematics in Data & Computer Science.

- 6. Computational Algorithms in Linear Algebra. explain how to solve algebra problems with Python using NumPy.
  - Understanding linear systems of equations.
  - Matrices & matrix representations of linear systems.
  - Solving small linear systems with Gaussian elimination.
  - Solving large linear systems with NumPy.
  - Summary.
- 7. Computational Requirements for Algorithms. give tools to determine how long algorithms take to run & how much space they require.
  - Computational complexity of algorithms.
  - Understanding Big-O Notation.
  - Complexity of algorithms with fundamental control structures.
  - Complexity of common search algorithms.
  - Common classes of computational complexity.
  - Summary.
- 8. Storage & Feature Extraction of Graphs, Trees, & Networks. cover storing graph structures & finding information about them with code.
  - Understanding graphs, trees, & networks.
  - Using graphs, trees, & networks.
  - Storage of graphs & networks.
  - Feature extraction of graphs.
  - Summary.
- 9. Searching Data Structures & Finding Shortest Paths. explain how to traverse graphs & figure out efficient paths between vertices.
  - Searching Graph & Tree data structures.
  - Depth-1st search (DFS).
  - Shortest path problem & variations of problem.
  - Finding Shortest Paths with Brute Force.
  - Dijkstra's Algorithm for Finding Shortest Paths.
  - Python Implementation of Dijkstra's Algorithm.
  - Summary.

Part III: Real-World Applications of Discrete Mathematics.

- 10. Regression Analysis with NumPy & Scikit-Learn. a discussion on prediction of variables in datasets containing multiple variables.
  - Dataset.
  - Best-fit lines & least-squares method.
  - Least-squares lines with NumPy.
  - Least-squares curves with NumPy & SciPy.
  - Least-squares surfaces with NumPy & SciPy.
  - Summary.
- 11. Web Searches with PageRank. show how to rank results of web searches to find most relevant web pages.
  - Development of Search Engines over time.

- Google PageRank II.
- Implementing PageRank algorithm in Python.
- Applying Algorithm to Real Data.
- Summary.

- 12. Principal Component Analysis with Scikit-Learn. explain how to reduce dimensionality of high-dimensional datasets to save space & speed up ML.
  - Understanding eigenvalues, eigenvectors, & orthogonal bases.
  - Principal component analysis approach to dimensionality reduction.
  - Scikit-learn implementation of PCA.
  - An application to real-world data.
  - Summary.

2 Miscellaneous

3 Wikipedia's

3.1 Wikipedia/discrete mathematics

"*Discrete mathematics* is study of mathematical structures that can be considered "discrete" (in a way analogous to discrete variables, having a bijection with $\mathbb{N}$) rather than "continuous" (analogously to continuous functions). Objects studied in discrete mathematics include integers, graphs, & statements in logic. By contrast, discrete mathematics excludes topics in "continuous mathematics" e.g. real numbers, calculus or Euclidean geometry. Discrete objects can often be enumerated by integers; more formally, discrete mathematics has been characterized as branch of mathematics dealing with countable sets (finite sets or sets with same cardinality as $\mathbb{N}$). However, there is no exact definition of term "discrete mathematics".

Set of objects studied in discrete mathematics can be finite or infinite. Term *finite mathematics* is sometimes applied to parts of field of discrete mathematics that deals with finite sets, particularly those areas relevant to business.

Graphs e.g. these are among objects studied by discrete mathematics, for their interesting mathematical properties, their usefulness as models of real-world problems, & their importance in developing computer algorithms.

Research in discrete mathematics increased in latter half of 20th century partly due to development of digital computers which operate in "discrete" steps & store data in "discrete" bits. Concepts & notations from discrete mathematics are useful in studying & describing objects & problems in branches of computer science, e.g. computer algorithms, programming languages, cryptography, automated theorem proving, & software development. Conversely, computer implementations are significant in applying ideas from discrete mathematics to real-world problems.

Although main objects of study in discrete mathematics are discrete objects, analytic methods from "continuous" mathematics are often employed as well.

In university curricula, discrete mathematics are discrete objects, analytic methods from "continuous" mathematics are often employed as well.

In university curricula, discrete mathematics appeared in 1980s, initially as a computer science support course; its contents were somewhat haphazard at time. Curriculum has thereafter developed in conjunction with efforts by ACM & MAA into a course that is basically intended to develop mathematical maturity in 1st-year students; therefore, it is nowadays a prerequisite for mathematics majors in some universities as well. Some high-school-level discrete mathematics textbooks have appeared as well. At this level, discrete mathematics is sometimes seen as a preparatory course, like precalculus in this respect.

Fulkerson Prize is awarded for outstanding papers in discrete mathematics.

3.1.1 Topics

1. Theoretical computer science. Complexity studies time taken by algorithms, e.g. this quick sort. Theoretical computer science includes areas of discrete mathematics relevant to computing. It draws heavily on graph theory & mathematical logic. Included within theoretical computer science is study of algorithms & data structures. Computability studies what can be computed in principle, & has close ties to logic, while complexity studies time, space, & other resources taken by computations. Automata theory & formal language theory are closely related to computability. Petri nets & process algebras are used to model computer systems, & methods from discrete mathematics are used in analyzing VLSI electronic circuits.

   Computational geometry applies computer algorithms to representations of geometrical objects. Computational geometry applies algorithms to geometrical problems & representations of geometrical objects, while computer image analysis applies them to representations of images. Theoretical computer science also includes study of various continuous computational topics.

2. Information theory. ASCII codes for word "Wikipedia", given here in binary, provide a way of representing word in information theory, as well as for information-processing algorithms. Information theory involves quantification of information. Closely related is coding theory which is used to design efficient & reliable data transmission & storage methods. Information theory also includes continuous topics e.g.: analog signals, analog coding, analog encryption.

3. Logic. Mathematical logic is study of principles of valid reasoning & inference, as well as of consistency, soundness, & completeness. E.g., in most systems of logic (but not in intuitionistic logic) Peirce's law $(((P \to Q) \to P) \to P)$ is a theorem. For classical logic, it can be easily verified with a truth table. Study of mathematical proof is particularly important in logic, & has accumulated to automated theorem proving & formal verification of software.

   Logical formulas are discrete structures, as are proofs, which form finite trees or, more generally, directed acylic graph structures (with each inference step combining 1 or more premise branches to give a single conclusion). Truth values of logical formulas usually form a finite set, generally restricted to 2 values: true & false, but logic can also be continuous-valued, e.g., fuzzy logic. Concepts e.g. infinite proof trees or infinite derivation trees have also been studied, e.g., infinitary logic.

4. Set theory. Set theory is branch of mathematics that studies sets, which are collections of objects, e.g. {blue, white, red} or (infinite) set of all prime numbers. Partially ordered sets & sets with other relations have applications in several areas.

   In discrete mathematics, countable sets (including finite sets) are main focus. Beginning of set theory as a branch of mathematics is usually marked by George Cantor's work distinguishing between different kinds of infinite set, motivated by study of trigonometric series, & further development of theory of infinite sets is outside scope of discrete mathematics. Indeed, contemporary work in descriptive set theory makes extensive use of traditional continuous mathematics.

5. Combinatorics. Combinatorics studies ways in which discrete structures can be combined or arranged. Enumerative combinatorics concentrates on counting number of certain combinatorial objects – e.g., 12fold way provides a unified framework for counting permutations, combinations, & partitions. Analytic combinatorics concerns enumeration (i.e., determining number) of combinatorial structures using tools from complex analysis & probability theory. In contrast with enumerative combinatorics which uses explicit combinatorial formulae & generating functions to describe results, analytic combinatorics aims at obtaining asymptotic formulae. Topological combinatorics concerns use of techniques from topology & algebraic topology/combinatorial topology in combinatorics. Design theory is a study of combinatorial designs, which are collections of subsets with certain intersection properties. Partition theory studies various enumeration & asymptotic problems related to integer partitions, & is closely related to q-series, special functions, & orthogonal polynomials. Originally a part of number theory & analysis, partition theory is now considered a part of combinatorics or an independent field. Order theory is study of partially ordered sets, both finite & infinite.

6. Graph theory. Graph theory has close links to group theory. This truncated tetrahedron graph is related to alternating group A_4. Graph theory, study of graphs & networks, is often considered part of combinatorics, but has grown large enough & distinct enough, with its own kind of problems, to be regarded as a subject in its own right. Graphs are 1 of prime objects of study in discrete mathematics. They are among most ubiquitous models of both natural & human-made structures. They can model many types of relations & process dynamics in physical, biological & social systems. In computer science, they can represent networks of communication, data organization, computational devices, flow of computation, etc. In mathematics, they are useful in geometry & certain parts of topology, e.g. knot theory. Algebraic graph theory has close links with group theory & topological graph theory has close links to topology. There are also continuous graphs; however, for most part, research in graph theory falls within domain of discrete mathematics.

7. Number theory. Ulam spiral of numbers, with black pixels showing prime numbers. This diagram hints at patterns in distribution of prime numbers. Number theory is concerned with properties of numbers in general, particularly integers. It has applications to cryptography & cryptanalysis, particularly with regard to modular arithmetic, diophantine equations, linear & quadratic congruences, prime numbers & primality testing. Other discrete aspects of number theory include geometry of numbers. In analytic number theory, techniques from continuous mathematics are also used. Topics that go beyond discrete objects include transcendental numbers, diophantine approximation, p-adic analysis & function fields.

8. Algebraic structures. Main article: Wikipedia/abstract algebra. Algebraic structures occur as both discrete examples & continuous examples. Discrete algebras include: Boolean algebra used in logic gates & programming; relational algebra used in databases; discrete & finite versions of groups, rings, & fields are important in algebraic coding theory; discrete semigroups & monoids appear in theory of formal languages.

9. Discrete analogues of continuous mathematics. There are many concepts & theories in continuous mathematics which have discrete versions, e.g. discrete calculus, discrete Fourier transforms, discrete geometry, discrete logarithms, discrete differential geometry, discrete exterior calculus, discrete Morse theory, discrete optimization, discrete probability theory, discrete probability distribution, difference equations, discrete dynamical systems, & discrete vector measures.

   - Calculus of finite differences, discrete analysis. In discrete calculus & calculus of finite differences, a function defined on an interval of integers is usually called a sequence. A sequence could be a finite sequence from a data source or an infinite sequence from a discrete dynamical system. Such a discrete function could be defined explicitly by a list (if its domain is finite), or by a formula for its general term, or it could be given implicitly by a recurrence relation or difference equation. Difference equations are similar to differential equations, but replace differentiation by taking difference between adjacent terms; they can be used to approximate differential equations or (more often) studied in their own right. Many questions & methods concerning differential equations have counterparts for difference equations. E.g., where there are integral transforms in harmonic analysis for studying continuous functions for analogue signals, there are discrete transforms for discrete functions or digital signals. As well as discrete metric spaces, there are more general discrete topological spaces, finite metric spaces, finite topological spaces.

Time scale calculus is a unification of theory of difference equations with that of differential equations, which has applications to fields requiring simultaneous modeling of discrete & continuous data. Another way of modeling such a situation is notion of hybrid dynamical systems.

- Discrete geometry. Discrete geometry & combinatorial geometry are about combinatorial properties of *discrete collections* of geometrical objects. A long-standing topic in discrete geometry is tiling of plane.

  In algebraic geometry, concept of a curve can be extended to discrete geometries by taking spectra of polynomials rings over finite fields to be models of affine spaces over that field, & letting subvarieties or spectra of other rings provide curves that lie in that space. Although space in which curves appear has a finite number of points, curves are not so much sets of points as analogues of curves in continuous settings. E.g., every point of form $V(x-c) \subset \mathrm{Spec}K[x] = \mathbb{A}^1$ for K a field can be studied either as $\mathrm{Spec}K[x]/(x-c) \cong \mathrm{Spec}K$, a point, or as spectrum $\mathrm{Spec}K[x]_{(x-c)}$ of local ring at $(x-c)$, a point together with a neighborhood around it. Algebraic varieties also have a well-defined notion of tangent space called Zariski tangent space, making many features of calculus applicable even in finite settings.

- Discrete modeling. In applied mathematics, discrete modeling is discrete analogue of continuous modeling. In discrete modeling, discrete formulae are fit to data. A common method in this form of modeling is to use recurrence relation. Discretization concerns process of transferring continuous models & equations into discrete counterparts, often for purposes of making calculations easier by using approximations. Numerical analysis provides an important example.

3.1.2 Challenges

Much research in graph theory was motivated by attempts to prove: all maps can be colored using only 4 colors so that no areas of same color share an edge. Kenneth Appel & Wolfgang Haken proved this in 1976.

History of discrete mathematics has involved a number of challenging problems which have focused attention within areas of field. In graph theory, much research was motivated by attempts to prove 4 color theorem, 1st stated in 1852, but not proved until 1976 (by Kenneth Appel & Wolfgang Haken, using substantial computer assistance).

In logic, 2nd problem on David Hilbert's list of open problems presented in 1900 was to prove: axioms of arithmetic are consistent. Gödel's 2nd incompleteness theorem, proved in 1931, showed: this was not possible – at least not within arithmetic itself. Hilbert's 10th problem was to determine whether a given polynomial Diophantine equation with integer coefficients has an integer solution. In 1970, Yuri Matiyasevich proved: this could not be done.

Need to break German codes in World War II led to advances in cryptography & theoretical computer science, with 1st programmable digital electronic computer being developed at England's Bletchley Park with guidance of Alan Turing & his seminal work, *On Computable Numbers*. Cold War meant: cryptography remained important, with fundamental advances e.g. public-key cryptography being developed in following decades. Telecommunication industry has also motivated advances in discrete mathematics, particularly in graph theory & information theory. Formal verification of statements in logic has been necessary for software development of safety-critical systems, & advances in automated theorem proving have been driven by this need.

Computational geometry has been an important part of computer graphics incorporated into modern video games & computer-aiddd design tools.

Several fields of discrete mathematics, particularly theoretical computer science, graph theory, & combinatorics, are important in addressing challenging bioinformatics problems associated with understanding tree of life.

Currently, 1 of most famous open problems in theoretical science is P = NP problem, which involves relationship between complexity classes P & NP. Clay Mathematics Institute has offered a $1 million USD prize for 1st correct proof, along with prizes for 6 other mathematical problems." – Wikipedia/discrete mathematics

3.2 Wikipedia/outline of discrete mathematics

"Discrete is study of mathematical structures that are fundamentally discrete rather than continuous. In contrast to real numbers that have property of varying "smoothly", objects studied in discrete mathematics – e.g. integers, graphs, & statements in logic – do not vary smoothly in this way, but have distinct, separated values. Discrete mathematics, therefore, excludes topics in "continuous mathematics" e.g. calculus & analysis.

Included below are many of standard term used routinely in university-level courses & in research papers. This is not, however, intended as a complete list of mathematical terms; just a selection of typical terms of art that may be encountered.

- Logic: Study of correct reasoning.
- Modal logic – Type of formal logic
- Set theory – Branch of mathematics that studies sets

3.2.1 Discrete mathematical disciplines

3.2.2 Concepts in discrete mathematics

3.2.3 Mathematicians associated with discrete mathematics

" – Wikipedia/outline of discrete mathematics

Tài liệu

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. 2nd edition. Addison-Wesley Publishing, 1994, p. 678.

[WR21] Ryan T White and Archana Tikayat Ray. *Practical Discrete Mathematics: Discover math principles that fuel algorithms for computer science & machine learning with Python*. Packt Publishing, 2021, p. 330.