1 ADD to the section on Recursive

DONE

Early work in complexity theory drew on analagies to computable theory. P is analogous to Decidable. NP is analgous to computably enumerable which we now define.

Def 1.1 We give three definitions of computably enumerable (c.e.) We will leave it as an exercise to show they are equivalent.

- A set A is c.e. if there exists a Turing machine M such that

  $$A = \{x : (\exists y)[M(x) \downarrow = x].$$

  (So A is the *range* of a computable function. Note that M might not halt on some inputs.)

- A set A is c.e. if either $A = \emptyset$ or there exists a Turing machine M that halts on all inputs such that

  $$A = \{x : (\exists y)[M(x) \downarrow = x].$$

  (So A is empty or the *range* of a total computable function.)

- A set A is c.e. if there exists a Turing machine M such that

  $$A = \{x : [M(x) \downarrow].$$

- (So A is the *range* of a computable function.)

- A set A is c.e. if there exists a $B \in \mathsf{R}$ such that

  $$A = \{x : (\exists y)[B(x, y) = 1].$$

Note 1.2 What we call *computably enumerable (c.e.)* is also called *recursively enumerable (r.e.)*. Soare [25] has tried to get the community to change to c.e. and gives good arguments for the change.

Exercise 1.3 Show that all four definitions of c.e. in Definition 1.1 are equivalent.

The fourth definition of c.e. in Definition 1.1 looks like NP which is identical to Σ_1. The next exercise asks you to define Π_1, Σ_2, Π_2 in the context of decidability.

Exercise 1.4

1. In the Section **??** on the Polynomial Hiearchy we defined Σ_i and Π_i by adding alternating poly-bounded quantifiers to P. Define similar classes by adding alternating unbounded quantifiers to R. Call them Σ_i and Π_i also (they will only be used within this problem so there should be no confusion). We call this the *Arithmetic Hiearchy*. The Arithmetic Hierarchy was defined by Kleene [15].

2. Let $M_0, M_1, \ldots,$ be a list of all Turing Machines. Assume that there For each of the following sets find which Σ_i or Π_i it is in. Try to make i as low as possible.

$$\text{FIN} = \{e : P_e \text{ halts on an infinite number of inputs } \}$$

$$\text{TOT} = \{e : P_e \text{ halts on all inputs } \}$$

2 Another Solution

DONE

Given a Boolean formula ϕ and a satisfying assignment $\vec{b}$, is there another one? This is an example of the class *Another Solution Problem (ASP)*. This problem turns out to be NP-complete. This is not obviious: there are some NP-complete problems for which the ASP version is in P.

Yato [27] and Yato & Seta [28] studied this notion and showed that for several problems in NP, the ASP version is also NP-complete. We state one of them: Sudoko. More precisely: *given a Soduku instance, and an answer for it, does it have another solutions?* is NP-complete.

We study this notion in Section **??**.

3 Exercise

DONE

Exercise 3.1 For each of the following statements either prove it, disprove, or state that it is unknown to science.

1. If $A, B \in$ P then $A \cup B \in P$.
2. If $A, B \in$ P then $A \cap B \in P$.
3. If $A, B \in$ P then $A \cdot B \in P$.
4. If A then $\overline{A} \in P$.
5. If A then $A^* \in P$.
6. If $A, B \in$ NP then $A \cup B \in NP$.
7. If $A, B \in$ NP then $A \cap B \in NP$.
8. If $A, B \in$ NP then $A \cdot B \in NP$.
9. If A then $\overline{A} \in NP$.
10. If A then $A^* \in NP$.

We need the next definition for the next problem.

Problem 3.2 MINFML *INSTANCE: A formula $\phi(x_1, \ldots, x_n)$ and a number L. QUESTION: Is there a formula $\psi(x_1, \ldots, x_n)$ of length $\leq L$ that is equivalent to ϕ?*

Exercise 3.3

1. Show that if P = NP then MINFML $\in$ P.
2. Vary the notion of length in various ways (e.g., number of $\wedge$ number of $\neg$) and determine if P = NP implies that this variant of MINFML is in P.

4 The Complexity of Puzzles

Many puzzles have been studied with respect to complexity theory. We will pursue this direction throughout the book; however, for now we give a few examples. The following problems are from a survey by Kendall, et al. [14].

- Helmert [11] studied the complexity of solitaire games. There results were complicated so we do not state them here. Later Paul and Helmert [18] looked at actual algorithms for solitaire games.

- Friedman [8] looked at the complexity of Corral Puzzles. A *C*orral puzzle is defined as follows: (1) The input is a grid where some of the spaces have natural numbers; (2) the goal is to find a loop that contains all of the spaces with numbers such that, thinking of the border of the loop as a wall, if a square has number n then there are exactly n squares visible from that square. The decision problem, *i*s there a loop?, is NP-complete.

- Robertson and Munro [20] looked at the following problem: Given n cubes where each face has a color chosen from a set of n colors, can the cubes be stacked so that each color appears exactly one time on each of the four sized of the tower? They showed this problem is NP-complete.

- Stuckman and Zhang [26] looked at the classic boardgame Mastermind. Given a set of guesses and the answers that the mastermind gave, is there a solution? They showed this problem is NP-complete. There has been a lot of work done on Mastermind where they try to optimize the number of queries needed in either the worst or average case. See Doerr et al. [7] for results with close-to-matching upper and lower bounds, and also many references to the past literature.

- Kaye [13] and A. Scott et al. [21] studied Minesweeper. (We assume the reader knows how to play minesweeper.) Consider the following decision problem: Given a board and an assignment of numbers to each square, can the mines be placed so that all of the numbers make sense? Kaye had a proof that this problem was NP-complete; however, his proof was incorrect. Later A. Scott et al. showed that it was co-NP-complete.

5 Complexity Classes that use Randomization

In this section we informally describe two randomized classes for, pardon the pun, completeness. Actually, there is an irony here since the in that these classes do not have complete problems.

5.1 Randomized Polynomial Time (RP)

Def 5.1 A set A is in *R*andomized Polynomial Time (RP) if there is a Turing Machine that flips coins such that

- If $x \in A$ then the probablility that $M(x) = 1$ is $\geq \frac{1}{2}$.
- If $x \notin A$ then the probablility that $M(x) = 0$ is 1.

Some facts about RP

1. Miller [17] obtained a poly time algorithm for PRIMES that depending on the Extended Riemann Hypothesis being true. Rabin [19] modified the algorithm to be in RP. For many years it was open as to whether PRIMES is in P until Agrawak et ak, [1] showed PRIMES $\in$ P.
2. There are very few problems that are in RP that are not known to be in P. We state one: given a polynomial $f(x_1, \ldots, x_n)$ and a prime p, is the polynomial identically zero of Z_p.
3. There are reasons to think that P = RP. Google *H*ardness versus Randomness to see the reason.

Exercise 5.2

1. Let $0 < \alpha < \frac{1}{2}$. In the definition of RP replace $\frac{1}{2}$ with α and call the class RP_α. Show that $\mathrm{RP}_{1/2} = \mathrm{RP}_\alpha$.
2. Let $\alpha(n)$ be a decreasing function from N to $(0, \frac{1}{2})$ In the definition of RP replace $\frac{1}{2}$ with $\alpha(|x|)$ and call the class $\mathrm{RP}_{\alpha(n)}$. Is $\mathrm{RP}_{\frac{1}{n^2}} = \mathrm{RP}$? Is $\mathrm{RP}_{\frac{1}{2^n}} = \mathrm{RP}$?

5.2 Bounded Probabilisitc Polynomial Time (BPP)

Def 5.3 A set A is in Bounded Probabilisitc Polynomial Time (BPP) if there is a Turing Machine that flips coins such that

- If $x \in A$ then the probablility that $M(x) = 1$ is $\geq \frac{3}{4}$.
- If $x \notin A$ then the probablility that $M(x) = 0$ is $\geq \frac{3}{4}$.

Some facts about BPP

1. There are no natural problems that are known to be in BPP but not known to be in RP. There aren't even any unnatural problems.
2. There are reasons to think that P = BPP. Google Hardness versus Randomness to see the reason.
3. It is not know if BPP $\subseteq$ NP.
4. Lautemann [16] and Sipser [24] proved that BPP $\subseteq \Sigma_2 \cap \Pi_2$. Lautemann's proof is simpler; however, Sipser's paper started the field of time-bounded Kolmogorov complexity.

Exercise 5.4

1. Let $\frac{1}{2} < \alpha < 1$. In the definition of BPP replace $\frac{3}{4}$ with α and call the class BPP_α. Show that $\text{BPP}_{3/4} = \text{BPP}_\alpha$.
2. Let $\alpha(n)$ be a decreasing function from N to $(0, \frac{1}{2})$ In the definition of RP replace $\frac{1}{2}$ with $\alpha(|x|)$ and call the class $\text{RP}_{\alpha(n)}$. Is $\text{BPP}_{\frac{1}{n^2}} = \text{BPP}$? Is $\text{BPP}_{\frac{1}{2^n}} = \text{BPP}$?

6 Add these to list of NPC problems

1. HAM CYCLE $\in$ NP: Given a graph, does it have a Hamiltonian cycle (A cycle that visits every vertex exactly once.)
2. TSP $\in$ NP: Given a weighted graph and a number k, is there a Hamiltonian cycle of weight $\leq k$. TSP stands for Travelings Salesperson's Problem. This problem is often stated as trying to $f$ind the optimal Hamiltonian cycle. Since NP is a set of sets, we look at the set-version of the find-problem.

3. $0-1-P \in$ NP.: Given a matrix M of of integers and a vector $\vec{b}$ or integers, is there a vector of 0's and 1's, $\vec{x}$ such that $M\vec{x} \leq \vec{b}$? The problem where instead of $0-1$ any integer vector is allowed is also in NP but this is harder to show since the entires could be large. (The P in $0-1-P$ stand for Programming.)

4. COL $\in$ NP: Given a graph and a number k, is the graph k-colorable? This means, is there a way to assign vertices to elements of $\{1, \ldots, k\}$ such that no two adjacent vertices have the same color. $3-$COL is the same problem but with $k = 3$.

5. IS $\in$ NP: Given a graph $G = (V, E)$ and a number k, is there a set $I \subseteq V$ such that $(\forall x, y \in I)[(x, y) \notin E]$ of size k. (Such a set is called an Independent Set.

6. VC $\in$ NP: Given a graph $G = (V, E)$ and a number k, is there a set $C \subseteq V$ such that $(\forall x, y \in C[(x, y) \in C]$. (Such a set is called a Clique.

7. VC $\in$ NP: Given a graph $G = (V, E)$ and a number k, is there a set $C \subseteq V$ such that $(\forall (x, y) \in E][x \in C \vee y \in C]$. (Such a set is called a Vertex Cover.

7 Add this to the sectiong on Strong NPC

The following problems are strongly NP-complete.

1. IS, VC, CLIQ. Garey and Johnson [9] showed that IS is strongly NP-complete. There are very easy reductions of IS to CLIQ and to VC hence they are also NP-complete.

2. *3*-Partition Problem: Given a list of integers, can they be partitioned into three groups with identical sums? This was originally shown to be NP-complete by Garey and Johnson using a reduction from 3-dimensional matching [10]. The 3-partition problem is used in many reductions to show problems on strongly NP-hard. We will study this more in Section **??**.

3. Bin Packing Problem: Given a set of items (positive rationals ≤ 1) and k bins of volume 1, can the items fit in the k bins? This problem can

be shown to be NP-complete using a reduction from the 3-partition problem. Bin packing has many interesting approximate algorithms, as surveyed by Coffman et al. [6].

8 Add to the Intermediary Problems

DONE

1. DL (Discrete Logarithm). We describe this as a function and leave it to the reader to give the set-version. We will use DL for both the function and set version.

   DL: Given prime p, generator g, and $a \in \mathsf{Z}_p$, find x such that $g^x \equiv a \pmod{p}$.

   Much like factoring (1) DL is important since, it if was easy to solve, the Diffie-Helman key exchange protocol in cryptography would be cracked, (2) DL is not known to be in P, (3) DL is not known to be NP-complete, (4) there are algorithms for DL that are better than the naive one, though still exponential (see the Wikipedia page on Discrete Log), (5) there has not been a better algorithm since 1998, (6) Shor [22, 23] has shown there is a quantum polynomial time algorithm for it, and (7) If DL is NP-complete then NP = co-NP; hence DL is unlikely to be NPC.

2. *Minimum Circuit Size Problem (*MCSP*)* Given the truth table of a boolean function f and an integer s, does f have a circuit with at most s logic gates?

   MCSP is not known to be in P, nor has it been proven to be NP-complete. Papers by Kabanets & Cai [12] and Allender & Hirahara [3] have provided arguments for why the problem is intermediary. Also see Allender [2] for a survey.

9 Add to Undecidable Problem Example

DONE

1. *Mortal Matrix Problem* Given a set of m $n \times n$ matrices over Z, determine whether they can be multiplied in some order (with repetition

allowed) to obtain the all-zero matrix. Cassaigne et al. [5] have shown the problem is undecidable for: (1) 6 3×3 matrices, (2) 4 5×5 matrices, (3) 3 9×9 matrices, (4) 2 15×15 matrices. Bournez & Branicky have shown the problem is decidable for 2 2×2 matrices the problem is decidable. Bell et al. [4] have shown that the problem for arbitrary large 2×2 matrices is NP-hard. Note that even the case of 2 3×3 matrices is open.

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES in p. *Annals of Mathematics*, 160:781–793, 2004.
`http://www.cs.umd.edu/~gasarch/BLOGPAPERS/primesinP.pdf`.

[2] Eric Allender. The new complexity landscape around circuit minimization. In Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, editors, *Language and Automata Theory and Applications - 14th International Conference, LATA 2020, Milan, Italy, March 4-6, 2020, Proceedings*, volume 12038 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2020.
`https://people.cs.rutgers.edu/~allender/papers/jones.pdf`.

[3] Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. *ACM Trans. Comput. Theory*, 11(4):27:1–27:27, 2019.
`https://doi.org/10.1145/3349616`.

[4] Paul C. Bell, Mika Hirvensalo, and Igor Potapov. Mortality for 2×2 matrices is np-hard. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings*, volume 7464 of *Lecture Notes in Computer Science*, pages 148–159. Springer, 2012.

[5] Julien Cassaigne, Vesa Halava, Tero Harju, and François Nicolas. Tighter undecidability bounds for matrix mortality, zero-in-the-corner problems, and more, 2014.
`http://arxiv.org/abs/1404.0644`.

[6] Edward Coffman, Michael Garey, and David Johnson. Approximation algorithms for bin packing: A survey. In Dorit Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 46–93. PWS, 1996.

[7] Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. Playing mastermind with many colors. *Journal of the Association of Computing Machinery*, 63(5):42:1–42:23, 2016.
https://doi.org/10.1145/2987372.

[8] Erich Friedman. Corral puzzles are NP-complete, 2002.
http://www.cs.umd.edu/gasarch/BLOGPAPERS/corral.pdf.

[9] Michael Garey and David Johnson. "Strong" NP-completeness results: Motivation, Examples, and Implications. *Journal of the Association of Computing Machinery*, 25(3):499–508, 1978.
https://doi.org/10.1145/322077.322090.

[10] Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal of Computing*, 4(4):397–411, 1975.
https://doi.org/10.1137/0204035.

[11] Malte Helmert. Complexity results for standard benchmark domains in planning. *Artif. Intell.*, 143(2):219–262, 2003.
https://doi.org/10.1016/S0004-3702(02)00364-8.

[12] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79. ACM, 2000.
https://doi.org/10.1145/335305.335314.

[13] Richard Kaye. Minesweeper in NP-complete. *The Mathematical Intelligencer*, 22:9–15, 2000.
https://link.springer.com/content/pdf/10.1007/BF03025367.pdf.

[14] Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A survey of NP-complete puzzles. *J. Int. Comput. Games Assoc.*, 31(1):13–34, 2008.
http://www.cs.umd.edu/gasarch/BLOGPAPERS/corral.pdf.

[15] Stephen Kleene. Recursive predicates and quantifiers. *Transactions of the the American Mathematics Society*, 53:41–73, 1943.

[16] Clemens Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.
https://doi.org/10.1016/0020-0190(83)90044-3.

[17] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computing and System Science*, 13(3):300–317, 1976.
https://doi.org/10.1016/S0022-0000(76)80043-8.

[18] Gerald Paul and Malte Helmert. Optimal solitaire game solutions using a* search and deadlock analysis. In Jorge A. Baier and Adi Botea, editors, *Proceedings of the Ninth Annual Symposium on Combinatorial Search, SOCS 2016, Tarrytown, NY, USA, July 6-8, 2016*, pages 135–136. AAAI Press, 2016.
http://aaai.org/ocs/index.php/SOCS/SOCS16/paper/view/13968.

[19] Michael Rabin. A probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.
http://www.cs.umd.edu/~gasarch/BLOGPAPERS/rabinprime.pdf.

[20] Eric Robertson and Ian Munro. NP-completneess, puzzles, and games. *Utilas Mathematica*, 13:99–116, 1978. Not available online.

[21] Allan Scott, Ulrike Stege, and Iris van Rooij. Minesweeper may not be NP-complete but it is hard nontheless. *The Mathematical Intelligencer*, 33:5–17, 2011.
http://www.cs.umd.edu/~gasarch/BLOGPAPERS/msnotnpc.pdf.

[22] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE Computer Society, 1994.
https://doi.org/10.1109/SFCS.1994.365700.

[23] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41(2):303–332, 1999.
https://doi.org/10.1137/S0036144598347011.

[24] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 330–335. ACM, 1983.
https://doi.org/10.1145/800061.808762.

[25] Robert Soare. Computability and recursion. *The Bulletin of Symbolic Logic*, 2(4), 1996.
http://www.people.cs.uchicago.edu/~soare/History/compute.pdf.

[26] Jeff Stuckman and Guo-Qiang Zhang. Mastermind is NP-complete, 2005.
http://arxiv.org/abs/cs/0512049.

[27] Takayuki Yato. Complexity and completeness of finding another solution and its application to puzzles. Master's thesis, The University of Tokyo, 2003.
http://www.cs.umd.edu/users/gasarch/BLOGPAPERS/msasp.pdf.

[28] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles, 2003.
http://www.cs.umd.edu/users/gasarch/BLOGPAPERS/ansol.pdf.