CS1800 Discrete Structures

Solutions to Recitations 6

Strange & Varma *Date: October 29 & 30, 2026*

- This recitation handout contains practice problems. You do *not* need to submit anything.
- We do recommend handwritting your solutions to completion. Do ensure that you are aware of how and why each step in the solution exists.
- For best effects, read through the questions before your recitation section and if possible attempt them on your own. This way you can compare your perspective to the TAs during the recitation.
- If you get stuck on a problem, use your recitation section to get help. Of course, you can also use office hours or post on Campuswire!
- The solutions to this recitation will be posted after Friday. Use those to verify your own attempts and for any clarification.
- If problems are marked $(*)$, those are fundamental and the others are additional practice using concepts from the fundamental practice problems in more complex ways.

Problem 1 (Dice)

You roll two fair dice. One die has 6 faces numbered 1 through 6, and the other die has 3 faces, numbered 1 through 3.

(a) What is the sample space for this experiment?

Solution:

The sample space for this experiment is $S = \{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3\}$.
$|S| = 6 \cdot 3 = 18$. ■

(b) What is the probability of the event that we get 2 odd numbers?

Solution:

The subset of the sample space that contains pairs of odd numbers is $E = \{1, 3, 5\} \times \{1, 3\}$.

$$Pr[\text{Both numbers are odd}] = \frac{|E|}{|S|} = \frac{6}{18} = \frac{1}{3}$$

Alternatively, using d6 for 6-sided dice roll and d3 for 3-sided dice roll, we can say that $Pr[\text{Both numbers are odd}] = Pr[\text{d6 is odd and d3 is odd}] = Pr[\text{d6 is odd}] \cdot Pr[\text{d3 is odd}]$ via independence of the two separate dice rolls.

This gives us:

$$Pr[\text{Both numbers are odd}] = \frac{3}{6} \cdot \frac{2}{3} = \frac{2}{6} = \frac{1}{3}$$

■

Problem 2 (Bitcoin Mining)

Bitcoin 'mining' depends on generating a random 24-bit string where the first 8 digits are zero.[1] These strings are called 'valid' bit-strings. Each 24-bit string is equally likely to be generated by the hashing process.

What is the probability that the mining process generates a valid bit-string?

You may leave any counting formulas unsimplified in your final answer (e.g. feel free to leave $\binom{10}{2}$ instead of computing its value).

Solution:

The total number of possible passcodes is $|S| = 2^{24} = 1677721$.

Let the event $E \subset S$ be the set of valid bit strings. To count $|E|$, note that we must fix the first 8 out of the 24 locations as 0's leaving the other 16 locations: $2^{16} = 65536$ Because each passcode is equally likely, the probability of generating a passcode with all unique characters is:

$$Pr(E) = \frac{|E|}{|S|} = \frac{2^{16}}{2^{24}} = \frac{65536}{16777216} \approx 0.0039$$

[1] This process is called *hashing* and it used to be the reason why graphic cards were expensive (not compared to today though) about a decade ago. Nowadays they're expensive for a whole other reason.

■

Problem 3 (Conditioned)

(a) Suppose we have a bit string of length 5, randomly generated such that all bit strings are equally likely. What is the probability that the string contains at least three consecutive ones, given that the first bit is a zero?

Solution:

This is a conditional probability, $Pr(E|F) = \frac{Pr(E \cap F)}{Pr(F)}$, with E = the set of bit strings with at least three consecutive 1s and F = the set of bit strings where the first bit is a zero.

There are a total of $2^5 = 32$ bit strings of length 5.

$Pr(E \cap F)$ = the probability of three consecutive 1s AND the first bit is a zero. There are three of these (01111, 01110, 00111), giving us $\frac{3}{32}$.

$Pr(F)$ = the probability that our 5-bit string starts with a zero, which is $\frac{1}{2}$.

Finally, we put everything together to get...

$$Pr(E|F) = \frac{Pr(E \cap F)}{Pr(F)} = \frac{3/32}{1/2} = \frac{3}{16}$$

■

(b) *[Skippable during recitation session, use as practice after]* You have an urn which contains 11 black and 9 white balls. You pick upto 4 balls from the urn. You can stop early if you get 3 white balls. If the first ball you picked is white, what is the probability that you will keep picking until 4 balls are picked from the urn?

Bonus: What is the probability that you actually get 3 white balls given that you do pick 4 balls with the first ball being white?

Solution:

A = the first ball is white. B = there are 3 white balls in total. $P_n = n$ balls are picked.

We stop once event B happens. To pick 4 balls, it must be the case that after 3 balls, the event B has not happened. Since event A has happened, we are in a situation where the balls picked are WWB or WBW or WBB. If it was WWW, then there would not pick the 4th ball. So $Pr(P_4|A)$ is the probability of one of these three sequences of ball picks happening.

$$Pr(P_4|A) = \frac{Pr(P_4 \cap A)}{Pr(A)}$$

$$Pr(P_4|A) = \frac{Pr(WWB) + Pr(WBW) + Pr(WBB)}{Pr(A)}$$

$$= \frac{1}{Pr(A)} \cdot \left[\frac{9}{20} \cdot \frac{8}{19} \cdot \frac{11}{18} + \frac{9}{20} \cdot \frac{11}{19} \cdot \frac{8}{18} + \frac{9}{20} \cdot \frac{10}{19} \cdot \frac{10}{18}\right] = \frac{20}{9} \cdot \frac{9}{20} \cdot \frac{276}{342} = \frac{46}{57}$$

Bonus: Using the above, we can calculate:

$$Pr(P_4 \cap A) = Pr(P_4|A)Pr(A) = \frac{46}{57} \cdot \frac{9}{20}$$

The probability of getting 3 white balls, given that the first ball was white and we picked 4 balls is: $Pr(B|A \cap P_4)$ which can be calculated as follows:

$$Pr(B|A \cap P_4) = \frac{Pr(B \cap A \cap P_4)}{Pr(A \cap P_4)} = \frac{Pr(WWBW) + Pr(WBWW)}{\frac{46}{57} \cdot \frac{9}{20}}$$
$$= \frac{\frac{9}{20} \cdot \frac{8}{19} \cdot \frac{11}{18} \cdot \frac{7}{17} + \frac{9}{20} \cdot \frac{11}{19} \cdot \frac{8}{18} \cdot \frac{7}{17}}{\frac{46}{57} \cdot \frac{9}{20}}$$

■

Project Guidelines

Utilize the recitation session to get guidance from TAs on whether your project idea is well scoped or not.

We'll provide a few examples of what projects for CS1800 could look like.[2] In each case, the proposal is provided and the questions raised in the project proposal should be explained/analyzed/derived in sufficient detail during your project for a 10 minute presenation (including Q&A).

Title: Efficient of Binary Adder Circuit Using Different Gates

Proposal: *We will look at efficiency of circuits for addition of two unsigned binary numbers when built using different set of base logic gates. The relevant concepts from CS1800 will be circuits and combinatorics. Circuits will explain how addition actually works. Combinatorics will be used to count number of gates and wires required and hence understand how efficient binary addition is when using different gates as building blocks. Apart from the standard XOR gate version of full adders, we will also consider the AND, OR, NOT gate version. The third version we will consider is a NAND gate based version. In each version, we will consider how the number of gates and connections grows as the number of bits n being added grows. Utilizing this, we will estimate efficiency of the circuit assuming that fewer gates and wires results in a more efficient circuit. Expected conclusion: for each version of the circuit, we will provide a count for number of gates and wires dependant on the number of bits n in the binary numbers being added.*

Title: Sets and Logic for Solving Sudokus

Proposal: *We will look at how concepts from set theory and logic are helpful in understanding how to solve (standard) sudoku puzzles. Each square in the sudoku can be modelled as a set of possible values called the candidates set of that square. If the square is already given, then it is a singleton set since there is only one valid value which can be filled there. Otherwise, any of the nine possible numbers is in the candidate set.*
We will look at how to model each of the rules of sudoku using logical statements with these candidate sets as variables in the statements. The goal of solving a sudoku is to make each candidate set become a singleton set using inferences and implications. Various common sudoku strategies will be translated into logic implications which can be used to reduce the cardinality of candidate sets (via set operations).

[2] Clearly, these are very skewed towards things Akshar can think of and hence tend to be on the nerdier end. However, that is not a requirement whatsoever. I should've talked about food trucks though.

Expected conclusion: A collection of logical statements which can be used to model unsolved sudokus, the various rules of sudokus as well as common strategies to solve sudokus. Put together, these should sufficient to implement a program which solves (standard, easy) sudokus.

Title: Coloring Graphs as a Model for Scheduling

Proposal: *We will look at feasibility of vertex coloring of graphs as a tool to model feasibility of scheduling events. The relevant concepts from CS1800 will be boolean expressions (to represent scheduling constraints) and graph theory. We consider scheduling of n events $e_1, \ldots, e_n$ into m time slots $t_1, \ldots, t_m$ while utilizing k re-usable resources $r_1, \ldots, r_k$ and ℓ consumable resources $c_1, \ldots, c_\ell$. An example of these will be scheduling 100 exams into 20 time slots during Finals week, using 30 rooms (re-usable resources) and 60 proctors (consumable resources). For example, there might be limits on when certain exams can be held, which rooms have sufficient capacity to host any exam e_i, a limit on how many times proctors can actually proctor, and so on. Each of these can be represented as a collection of quantified boolean expressions. Further, we can model the problem of finding a satisfying schedule into a problem of finding a coloring of vertices on (a specially constructed) graph where two vertices cannot be colored the same color (or they'll violate some requirement of the scheduling). Understanding limitations of when/how the vertices can be colored can help us understand the scheduling problem.*
Expected Conclusion: Understand the relationship between scheduling problems and graph coloring and how various constraints of scheduling translate into graphs.

Title: Subgroups of a Group

Proposal: *In many situations, there is a need to form smaller sized groups with people/entities chosen from a larger group while satisfying certain requirements. This can be a set of players to field from an overall roster. It could be a subunit of K-pop group to hold further activities. It could be choosing a subset of TAs to help with behind-the-scenes tasks out of all TAs in a course. In each of these situations, the people/entities usually have specializations and there are requirements for what the subgroups need to have. For example, out of n players in the roster, maybe k need to be chosen such that there are 2–3 attaching players, 3–4 defending players, 3–4 support players, which the others being arbitrary roles. Given knowledge about the specializations of each player we are interested in determining how many ways subgroups can be formed. In the K-pop example, the specialization may look like dancers vs. vocalists vs. rappers and so on. Subunit requirements here might look quite different since a fully vocal focused subunit might also work out. TA specializations might be less about specializations and more about what aspect of behind-the-scenes work they are more interested in.*
Expected Conclusion: Form a general method to count number of possible subgroups given constraints and information about the full group. The relevant concepts from CS1800 are counting, as well as set theory (subsets for specializations, cartesian products for subgroups with certain specializations).

Some Project Ideas from a previous version of this course

A brief list of some of the project ideas from an old version of the course. The requirements were slighlty different so the details are less useful as reference. Each comes with a quick comment.

1. *Title: Optimizing the Dodgers Batting Order.* Looked at expected number of run based on different order of batters from a baseball team. Used concepts from counting (possible permutations) and probability (expected value, conditional probability).

2. *Title: Optimizing Poker Strategies.* Used combinatorics (counting possible hands) and probability (Bayes rule, expectation) to model how players might play based on what cards are known and how card probabilities and their choices affect strategies to use in response.

3. *Title: Progression of Video Games.* Uses bit representations/binary numbers as well as set theory to look at the range of color palattes in use in video games and how this changes as capabilities improve.

4. *Title: Breadth First Search in Social Networks.* Lots of graph theory concepts. **In our course you'll need to combine with another CS1800 concept.** [There was another about using Dijstra's algorithm to find routes inside Northeastern's campus. Same issue as the other project. Another looked at the A-star algorithm. All of these are nice ideas, but focus too much on one area from CS1800.]

5. *Title: The probability of a perfect March Madness bracket.* Probability and counting.

6. *Title: Song recommendations.* Set theory as well as graph theory to construct models for recommending songs to people based on known interests and liked songs. [There was another about movie similarity, which used similar concepts, but utilized them slightly differently. Another used set operations and conditional probability for the same purpose.]

7. *Title: How to beat blackjack.* Deep dive into just probability. **In our course you'll need to combine with another CS1800 concept.**

8. *Title: Password security via counting:* **Not a great idea since this is too limited in scope. Only counting shows up and most of the counting we've seen in recitations.**

9. *Title: Film Script Writing.* Cool idea about how logical motivation is necessary for a "good" script and discussed how implications and chaining implications is relevant. **In our course you'll need to combine with another CS1800 concept.** So this is not directly usable but might give ideas.

10. *Title: Set building for card games.* Used set builder notation to define a set which included as many of our cards and as few of the opponents cards. By itself, only uses one concept. However, this could be combined with a round of drawing a card and then looking at the probability of how the score changes based on new cards. That'll make it both sets and probability.

11. *Title: Lottery Counting.* Using combinatorics and probability to find the expected payout given how the lottery works and what is costs to buy vs. payoff. So expectations from probability tells you what your average earning is. Would be cool to make-up your own lottery and explain why you made such choices.

12. *Title: When to shop.* Using a little of set theory and counting but mostly probability, determine what's the best time to go shopping.

13. *Title: Spotify's shuffle.* Why it isn't just about counting. Set theory comes in, and compare different "shuffles". Can also bring in probability (Bayes rule) to improve the shuffle so that there are fewer skips.

14. *Title: Counting Subway Orders.* **Lacking another concept from CS1800. Only uses counting. So not a good example for this course if used as is.**

15. *Title: Medical diagnosis.* Using logical expressions and probability to determine likely diagnosis.

16. *Title: Set theory in dating app filters.* As is, only used set theory, but might be possible to extend to include some probability to find "likely matches".