

1 Jelly Bean Factory

Note 10

A candy factory has an endless supply of red, orange and yellow jelly beans. The factory packages the jelly beans into jars of 100 jelly beans each, with each possible combination of colors in the jar being equally likely. (One possible color combination, for example, is a jar of 56 red, 22 orange, and 22 yellow jelly beans.)

Find N , the number of different possible color combinations of jelly beans in a single jar (note that color combinations are unordered).

Solution: The correct way to think of this problem is in terms of 100 unlabeled balls (the jelly beans) in 3 labeled bins (the colors red, orange, yellow). Recall from class that we have a formula for this: $\binom{n+k-1}{k}$, where k is the number of balls and n is the number of bins. Therefore, the correct answer is

$$N = \binom{102}{100} = 5151$$

2 Grids and Trees!

Note 10

Suppose we are given an $n \times n$ grid, for $n \geq 1$, where one starts at $(0,0)$ and goes to (n,n) . On this grid, we are only allowed to move left, right, up, or down by increments of 1.

- (a) How many shortest paths are there that go from $(0,0)$ to (n,n) ?
- (b) How many shortest paths are there that go from $(0,0)$ to $(n-1, n+1)$?

Now, consider shortest paths that meet the conditions where we can only visit points (x,y) where $y \leq x$. That is, the path cannot cross line $y = x$. We call these paths n -legal paths for a maze of side length n . Let F_n be the number of n -legal paths.

- (c) Compute the number of shortest paths from $(0,0)$ to (n,n) that cross $y = x$. (Hint: Let (i,i) be the first time the shortest path crosses the line $y = x$. Then the remaining path starts from $(i, i+1)$ and continues to (n,n) . If in the remainder of the path one exchanges y -direction moves with x -direction moves and vice versa, where does one end up?)
- (d) Compute the number of shortest paths from $(0,0)$ to (n,n) that do not cross $y = x$. (You may find your answers from parts (a) and (c) useful.)
- (e) A different idea is to derive a recursive formula for the number of paths. Fix some i with $0 \leq i \leq n-1$. We wish to count the number of n -legal paths where the last time the path

touches the line $y = x$ is the point (i, i) . Show that the number of such paths is $F_i \cdot F_{n-i-1}$. (Hint: If $i = 0$, what are your first and last moves, and where is the remainder of the path allowed to go?)

- (f) Explain why $F_n = \sum_{i=0}^{n-1} F_i \cdot F_{n-i-1}$.
- (g) Create and explain a recursive formula for the number of trees with n vertices ($n \geq 1$), where each non-root node has degree at most 3, and the root node has degree at most 2. Two trees are different if and only if either left-subtree is different or right-subtree is different.

(Notice something about your formula and the grid problem. Neat!)

Solution: Let $(x, y) \rightarrow (x+1, y)$ be a move 'right' command, and $(x, y) \rightarrow (x, y+1)$ be a move 'up' command.

- (a) There are $\binom{2n}{n}$ paths, as there are total number of $2n$ moves, and n of them must be move 'right' command, the rest of them must be the move 'up' command.
- (b) There are $\binom{2n}{n-1}$ paths as there are now $n-1$ move 'right' command.
- (c) We will argue that the set of all paths that cross $y = x$, form a bijection with the set of all paths from $(0, 0)$ to $(n-1, n+1)$. Suppose we have a path that crosses $y = x$. Once a path crosses $y = x$, we can flip the later portion of the path. Let the first time the invalid path crosses $y = x$ be at (i, i) and arrives at $(i, i+1)$. Then if we do not flip the path, it will arrive at (n, n) by taking $n-i$ "right" commands, and $n-1-i$ "up" commands. If we flip these commands, it will go to $(i + (n-1-i), (i+1) + (n-i)) = (n-1, n+1)$. So all invalid paths map to a path from $(0, 0)$ to $(n-1, n+1)$. Next we argue that all path from $(0, 0)$ to $(n-1, n+1)$ maps to a invalid path. Paths from $(0, 0)$ to $(n-1, n+1)$ must cross the line $y = x$, let it first cross the line at (i, i) and arrives $(i, i+1)$. Then it must take $(n-1) - i$ "right" commands, and $(n+1) - (i+1)$ "up" commands. We flip these commands and so we now have, $n-1-i$ "up" commands, $n-i$ "right" commands. Then the path will arrive $(i + (n-i), (i+1) + (n-1-i)) = (n, n)$ and this new path is considered as invalid path since it crosses $y = x$ at point (i, i) . So all paths from $(0, 0)$ to $(n-1, n+1)$ can be mapped to an invalid paths.

So there is a bijective mapping between invalid paths and paths from $(0, 0)$ to $(n-1, n+1)$. Hence, the number of invalid paths is $\binom{2n}{n-1}$.

- (d) The number of paths that don't cross $y = x$ is given by the number of paths minus the number of paths that do cross $y = x$, or $\binom{2n}{n} - \binom{2n}{n-1}$.
- (e) Let F_n be the total number of different ways from $(0, 0)$ to (n, n) satisfies the condition above. We know $F_1 = 1$. Let (i, i) be the last point on line $y = x$ that a path touches except for (n, n) . Then total number of such path is $F_i \cdot F_{n-1-i}$ where F_i is the total number of paths from $(0, 0)$ to (i, i) . Since (i, i) is the last boundry point it touches, so for all later steps, it must not cross the line $y = x - 1$, it's equivalent to say the total number of paths from $(i+1, i)$ to $(n, n-1)$, it's F_{n-1-i} .
- (f) From the previous part, the number of n -legal paths where the last time the path touches

the line $y = x$ at the point (i, i) is $F_i \cdot F_{n-1-i}$. To get the total number of paths that cross the $y = x$, we simply need to sum across every possible point (i, i) that the path could have passed through. Hence, $F_n = \sum_{i=0}^{n-1} F_i \cdot F_{n-1-i}$.

- (g) Let T_n be the total number of different trees with n nodes. The number of different trees when the left subtree has size i and right subtree has size $n - i - 1$ is $T_i \cdot T_{n-i-1}$. If we sum over all possible sizes of left subtrees, we can get the total number of different trees: $T_n = \sum_{i=0}^{n-1} T_i T_{n-i-1}$, with the base cases $T_0 = 1, T_1 = 1$. Note that a similar counting argument captures totally different objects (mazes and trees)!

If you are interested in why these problems are related, check out the [Catalan numbers](#).

3 Is This CS 61C?

Note 10

XOR (\oplus) is a function that takes in two integers that are each either 0 or 1 (also known as **one-bit integers**), and returns 0 if they have the same value, and 1 otherwise. For example, $1 \oplus 0 = 1$, whereas $0 \oplus 0 = 0$. Note that we can generalize XOR for k one-bit integers to say $x_1 \oplus \dots \oplus x_k = (x_1 + \dots + x_k) \pmod{2}$.

- (a) Show that the number of one-bit integer solutions (x_1, \dots, x_k) for $x_1 \oplus \dots \oplus x_k = 0$ is 2^{k-1} .

We can extend XOR to an n -bit string, which is a length n string where each integer in the string is an one-bit integer. If we have two n -bit strings $s = s_1 \dots s_n$ and $t = t_1 \dots t_n$, we define $s \oplus t$ to be $u_1 \dots u_n$, where for all i , $u_i = s_i \oplus t_i$. We can similarly define XOR for k n -bit strings $y_1 = z_{1,1} \dots z_{1,n}, \dots, y_k = z_{k,1} \dots z_{k,n}$, and $y_1 \oplus \dots \oplus y_k = w_1 \dots w_n$, where for all i , $w_i = z_{1,i} \oplus \dots \oplus z_{k,i}$.

- (b) Show that for an arbitrary n -bit string x , the number of n -bit string solutions (y_1, \dots, y_k) for $y_1 \oplus \dots \oplus y_k = x$ is $2^{(k-1)n}$. (Hint: You may find your answer from part (a) useful.)

Solution:

- (a) For every $(x_1, \dots, x_{k-1}) \in \{0, 1\}^{k-1}$, there exists a unique x_k that satisfies $x_1 \oplus \dots \oplus x_k = 0$ as we can rewrite it as $x_k = (x_1 + \dots + x_{k-1}) \pmod{2}$. Thus, the number of solutions for $x_1 \oplus \dots \oplus x_k = 0$ is the same as the number of distinct tuples (x_1, \dots, x_{k-1}) , and since for each x_i , x_i can either be 0 or 1, so in total, there are 2^{k-1} solutions.
- (b) Observe that bits in different positions do not interfere with each other, so we can reduce to an one-bit integer case. Let's focus on bits on position $1 \leq i \leq n$. We have $z_{1,i} \oplus \dots \oplus z_{k,i} = x_i$. If $x_i = 0$, then we know the number of solutions $(z_{1,i}, \dots, z_{k,i})$ is 2^{k-1} , and if $x_i = 1$, then we know it is $2^k - 2^{k-1} = 2^{k-1}$, which is the same as the $x_i = 0$ case. Thus, for each bit position $1 \leq i \leq n$, there are 2^{k-1} possibilities, so in total, we have $2^{(k-1)n}$ solutions for the equation $y_1 \oplus \dots \oplus y_k = x$.

4 Code Reachability

Note 12

Consider triplets (M, x, L) where

- M is a Java program

- x is some input
- L is an integer

and the question of: if we execute $M(x)$, do we ever hit line L ?

Prove this problem is undecidable.

Solution: Suppose we had a procedure that could decide the above; call it `Reachable(M, x, L)`. Consider the following example of a program deciding whether $P(x)$ halts:

```
def Halt(P, x):
    def M(t):
        run P(x) # line 1 of M
        return    # line 2 of M
    return Reachable(M, 0, 2)
```

Program M reaches line 2 if and only if $P(x)$ halted. Thus, we have implemented a solution to the halting problem — contradiction.

5 Computations on Programs

Note 12

- (a) Is it possible to write a program that takes a natural number n as input, and finds the shortest arithmetic formula which computes n ? For the purpose of this question, a formula is a sequence consisting of some valid combination of (decimal) digits, standard binary operators ($+$, \times , the “ $^$ ” operator that raises to a power), and parentheses. We define the length of a formula as the number of characters in the formula. Specifically, each operator, decimal digit, or parentheses counts as one character.

(Hint: Think about whether it’s possible to enumerate the set of possible arithmetic formulas. How would you know when to stop?)

- (b) Now say you wish to write a program that, given a natural number input n , finds another program (e.g. in Java or C) which prints out n . The discovered program should have the minimum execution-time-plus-length of all the programs that print n . Execution time is measured by the number of CPU instructions executed, while “length” is the number of characters in the source code. Can this be done?

(Hint: Is it possible to tell whether a program halts on a given input within t steps? What can you say about the execution-time-plus-length of the program if you know that it does not halt within t steps?)

Solution:

- (a) Yes it is possible to write such a program.

We already know one way to write a formula for n , which is to just write the number n (with no operators). Let the length of this formula in characters be l . In order to find the *shortest* formula we simply need to search among formulae that have length at most l .

Since there are a finite number of formulas of length at most l , we can write a program that iterates over all of them. For example, if we treat each character as a byte or an 8-bit number, the whole formula becomes a binary integer of length at most $8l$, so we can simply iterate over all binary numbers up to 2^{8l} and for each one check if it is a valid formula.

For each formula that we encounter we can compute its value in finite time (since there are no loop/control structures in formula). Therefore we can check whether it computes n , and then among those that do compute n we find the smallest one.

(b) Yes. Again it is possible to write such a program.

As before, given a number n , there is one program that we know can definitely write n , which is the program that prints the digits of n one by one. Let the length plus running time of this program be l . We only need to check programs that have a length of at most l and a running time of at most l , since otherwise their running time plus length would be bigger than l .

Similar to the previous part, we can iterate over all programs of length at most l (by treating each one as a large binary integer and checking each one's validity by e.g. compiling it). For each such program, we then run it for at most l steps. If it takes more time, we stop executing it and go to the next program, otherwise in at most l steps we see its output and we can check whether it is equal to n or not.

Now among all programs that have length at most l and execute for at most l steps and print n we find the one that has the shortest length plus execution time.

6 Countability: True or False

Note 11

(a) The set of all irrational numbers $\mathbb{R} \setminus \mathbb{Q}$ (i.e. real numbers that are not rational) is uncountable.

(b) The set of real solutions for the equation $x + y = 1$ is countable.

For any two functions $f : Y \rightarrow Z$ and $g : X \rightarrow Y$, let their composition $f \circ g : X \rightarrow Z$ be given by $f \circ g = f(g(x))$ for all $x \in X$. Determine if the following statements are true or false.

(c) f and g are injective (one-to-one) $\implies f \circ g$ is injective (one-to-one).

(d) f is surjective (onto) $\implies f \circ g$ is surjective (onto).

Solution:

(a) **True.** Proof by contradiction. Suppose the set of irrationals is countable. From Lecture note 10 we know that the set \mathbb{Q} is countable. Since union of two countable sets is countable, this would imply that the set \mathbb{R} is countable. But again from Lecture note 10 we know that this is not true. Contradiction!

(b) **False.** Let $S \subset \mathbb{R} \times \mathbb{R}$ denote the set of all real solutions for the given equation. For any $x' \in \mathbb{R}$, the pair $(x', y') \in S$ if and only if $y' = 1 - x'$. Thus $S = \{(x, 1 - x) : x \in \mathbb{R}\}$. Besides, the mapping x to $(x, 1 - x)$ is a bijection from \mathbb{R} to S . Since \mathbb{R} is uncountable, we have that S is uncountable too.




- (c) **True.** Recall that a function $h : A \rightarrow B$ is injective iff $a_1 \neq a_2 \implies h(a_1) \neq h(a_2)$ for all $a_1, a_2 \in A$. Let $x_1, x_2 \in X$ be arbitrary such that $x_1 \neq x_2$. Since g is injective, we have $g(x_1) \neq g(x_2)$. Now, since f is injective, we have $f(g(x_1)) \neq f(g(x_2))$. Hence $f \circ g$ is injective.
- (d) **False.** Recall that a function $h : A \rightarrow B$ is surjective iff $\forall b \in B, \exists a \in A$ such that $h(a) = b$. Let $g : \{0, 1\} \rightarrow \{0, 1\}$ be given by $g(0) = g(1) = 0$. Let $f : \{0, 1\} \rightarrow \{0, 1\}$ be given by $f(0) = 0$ and $f(1) = 1$. Then $f \circ g : \{0, 1\} \rightarrow \{0, 1\}$ is given by $(f \circ g)(0) = (f \circ g)(1) = 0$. Here f is surjective but $f \circ g$ is not surjective.

7 Counting Shapes

Note 11

Suppose scaled and shifted copies of a shape S are embedded into the plane \mathbb{R}^2 . Let \mathcal{C} denote the collection of all these copies. Thus each element in \mathcal{C} determines the scaling and the position of that copy. Suppose further that the embedding is such that no two copies intersect. For example in the case of filled squares, if there is any overlap between two squares, then they intersect. In the case of the (non-filled) square, two copies intersect if and only if their boundaries intersect. Similarly, in the case of the halved-square, if either the boundary or the middle line of one square intersects with either the boundary or the middle line of some other square, then these two squares intersect.

Can \mathcal{C} be uncountable if S is

- (a) the filled square:  ?
- (b) the square:  ?
- (c) the halved square:  ?

If no uncountable \mathcal{C} exists, prove that all \mathcal{C} must be countable.

Solution:

- (a) There can be at most countably many filled squares: We will show that there is an injection from \mathcal{C} to $\mathbb{Q} \times \mathbb{Q}$, which we know to be countable. To do so, we argue that for any filled square $F \in \mathcal{C}$, we can find a point (x_F, y_F) with rational coordinates (i.e. $p_F, q_F \in \mathbb{Q}$) inside F : We know that F has some center (x_F, y_F) and side length $2 \cdot r_F$, and whence $F = [x_F - r_F, x_F + r_F] \times [y_F - r_F, y_F + r_F]$. But any interval must contain a rational number, so in particular there is a rational $p_F \in [x_F - r_F, x_F + r_F]$ and $q_F \in [y_F - r_F, y_F + r_F]$, so that indeed $(p_F, q_F) \in F \cap (\mathbb{Q} \times \mathbb{Q})$.

Now let us define the function f from \mathcal{C} to $\mathbb{Q} \times \mathbb{Q}$ as $f(F) = (p_F, q_F)$ and show that it is injective: If $f(F) = f(F')$, then F and F' share the point $(p_F, q_F) = (p_{F'}, q_{F'})$, so F must be F' , for otherwise they would contradict the assumption that any distinct F and F' do not intersect.

- (b) We give an example of a \mathcal{C} that is uncountable: Let all squares $Q \in \mathcal{C}$ share the same center (e.g. $(0, 0)$), but be of different side lengths $r_Q \in \mathbb{R}$. Then no two squares intersect, and we

have \mathbb{R} many of them. But we know that \mathbb{R} is uncountable, and so we have uncountably many squares.

- (c) Now \mathcal{C} can be at most countable again. This time, we construct an injection from \mathcal{C} to $(\mathbb{Q} \times \mathbb{Q}) \times (\mathbb{Q} \times \mathbb{Q})$, which we know to be countable. As before, for any $H \in \mathcal{C}$ we can pick a rational point $(p_H, q_H) \in \mathbb{Q} \times \mathbb{Q}$ in the upper half of H and $(s_H, t_H) \in \mathbb{Q} \times \mathbb{Q}$ in the lower half of H . Defining f from \mathcal{C} to $(\mathbb{Q} \times \mathbb{Q}) \times (\mathbb{Q} \times \mathbb{Q})$ by $f(H) = ((p_H, q_H), (s_H, t_H))$, we see that f must once more be injective, for if $f(H) = f(H')$, then neither H nor H' can be fully contained in each other's halves. Therefore, either $H = H'$, or they intersect.