Representation Change

The idea behind a representation change approach to a problem is to restate the problem or transform it into some alternative representation that will yield insights into the solution, make the problem easier to solve, and/or reduce the problem into one where the solution is known or where other methods can be applied.

Usually, such a representation will make use of principles associated with discrete mathematics (e.g. graphs, binary trees), or number theory (e.g. modular arithmetic, change of base). Some familiarity with those topics will go a long way towards both recognizing representation changes that will be useful, and for suggesting possible representation changes.

Note that many representation changes will actually add another layer of abstraction on top of an existing problem. This can often be beneficial, as it may allow you to apply more effective tools to reach a solution. For example, consider a simple algebraic word problems, such as "What number, added to 4, gives the result 7?" A brute-force searching algorithm might try many different numbers until a solution is found, but the *representation change* into the equation $x + 4 = 7$ allows us to apply simple algebraic rules to quickly yield the solution of $x = 3$.

The general problem-solving strategy of finding solutions to simple base cases before generalizing can be of particular use in developing a representation change. The patterns that appear when solving simple cases can reveal facets of the problem that suggest certain types of representation change.

Examples of Representation Change

- State Machines
- Graphs
- Change of Base / Modular Arithmetic

1 Problems

Problem 1: Missionaries and Cannibals

Three missionaries and three cannibals must cross a river. Their boat can only hold two people, and it cannot cross the river by itself with no people on board. Each missionary and each cannibal can row the boat. If present, missionaries cannot be outnumbered by cannibals. How can all six get across the river with the fewest crossings?

Solution: Missionaries and Cannibals

First, we can establish some lower bounds: each traversal of the river can transport (up to) 2 people from one side to the other. If there are more than 2 people to transport in total, the boat somehow needs to be brought back to the other side by someone. Thus, the first 2 people take 1 crossing, and each additional person takes 2 more crossings (once for someone to bring the boat

back, and once to bring the additional person over). Thus, the minimum number of crossings for 6 people is 9.

We can choose to represent the state of the problem as an ordered quadruplet: (M_0, C_0, M_1, C_1), where M_0 and C_0 are the numbers of missionaries and cannibals on the starting side, respectively, and M_1 and C_1 are the numbers of missionaries and cannibals on the ending side, respectively. So, we begin with $(3, 3, 0, 0)$ and wish to reach an end state of $(0, 0, 3, 3)$.

Exploring some possible moves, we notice immediately that the first crossing must consist of two people. (Otherwise, there is only one person to bring the boat back, and we are at the initial state again, squandering two crossings.) Furthermore, the first crossing must consist of a cannibal and one other person. (Otherwise, it would leave behind 3 cannibals and 1 missionary, which is not allowed.)

If we try 1 cannibal and 1 missionary on the first trip: $(3, 3, 0, 0) \to (2, 2, 1, 1)$. Then the missionary must bring the boat back, or there would be 3 cannibals and 2 missionaries on one side. So $(2, 2, 1, 1) \to (3, 2, 0, 1)$. The next crossing over cannot take 2 missionaries (would leave 1 missionary and 2 cannibals behind), and cannot take 1 missionary and 1 cannibal (would put 1 missionary and 2 cannibals on the far side), so it must take over 2 cannibals, leaving only the option for any of the cannibals to bring the boat back. $(3, 2, 0, 1) \to (3, 0, 0, 3) \to (3, 1, 0, 2)$.

Now we observe that the next trip must bring over at least 2 missionaries (any less and they will be outnumbered on the other side). $(3, 1, 0, 2) \to (1, 1, 2, 2)$. The return trip here must consist of 1 missionary and 1 cannibal (or else one or the other side will have more cannibals than missionaries. $(1, 1, 2, 2) \to (2, 2, 1, 1)$. Observing the symmetry here, we note that the remainder of the crossings can be performed in reverse (as though swapping sides): $(2, 2, 1, 1) \to (0, 2, 3, 1) \to (0, 3, 3, 0) \to (0, 1, 3, 2) \to (1, 1, 2, 2) \to (0, 0, 3, 3)$. Thus, the final sequence is:
$(3, 3, 0, 0) \to (2, 2, 1, 1) \to (3, 2, 0, 1) \to (3, 0, 0, 3) \to (3, 1, 0, 2) \to (1, 1, 2, 2) \to$
$(2, 2, 1, 1) \to (0, 2, 3, 1) \to (0, 3, 3, 0) \to (0, 1, 3, 2) \to (1, 1, 2, 2) \to (0, 0, 3, 3)$. With 12 total states, that means that it took 11 crossings.

If we try a different initial trip (2 cannibals), we get: $(3, 3, 0, 0) \to (3, 1, 0, 2) \to (3, 2, 0, 1)$, which turns out identically as above. Thus, it appears as though the minimum is 11 crossings, regardless of how we begin. (Similarly, the ending sequence can vary as well, from the $(0, 1, 3, 2)$ state.)

Note that this is more than the absolute minimum of 9 (ignoring any no constraints regarding cannibals outnumbering missionaries). We observe that an extra 2 crossings occur because we reach an intermediate state, $(1, 1, 2, 2)$, where the only valid return trip is 1 missionary and 1 cannibal. This effectively uses 2 extra turns, because the 2-trip cycle that includes this trip does not increase the total number of people on the other side. If we step through the state diagram to see what alternative choices there might be, however, we see that these extra 2 trips are necessary due to the constraints of the problem, and therefore 11 moves is optimal.

Problem 2: Penny Distribution Machine

A "machine" consists of a row of (infinite) boxes. To start, one places n pennies in the leftmost box. The machine then redistributes the pennies as follows. On each iteration, it replaces a pair of pennies in one box with a single penny in the next box to the right. The iterations stop when there is no box with more than one coin. For example, Figure 1 shows the work of the machine in distributing six pennies by always selecting a pair of pennies in the leftmost box with at least two coins.

(a) Does the final distribution of pennies depend on the order in which the machine processes the coin pairs?

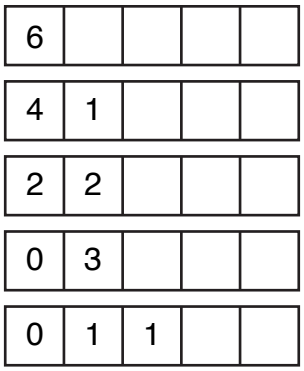

Figure 1: Example of penny distribution.

(b) What is the minimum number of boxes needed to distribute n pennies?

(c) How many iterations does the machine make before stopping?

Solution: Penny Distribution Machine

1.1 Problem Restatement

One useful strategy for restating the problem is to separate it into different components. This ensures that we understand all aspects of the problem and that we aren't leaving behind any information. It also allows to establish a framework to distinguish between fundamentally different aspects of the problem. In this case, we can identify three separate pieces:

1. There is an initial state, which consists of n pennies in the leftmost box.
2. We are allowed to do "redistribution" actions, which replace 2 pennies in one box with 1 penny in the next box to the right.
3. We stop when each box has 0 or 1 pennies.

Let us number the boxes, from left to right with $0, 1, \ldots$.

1.2 Brainstorming

We can begin brainstorming by first listing several corollaries of the problem statement. While some of these may be "obvious", just writing these down and mulling them over may reveal useful approaches, often involving several of the "obvious" statements.

- If $n > 1$, some redistribution must occur before we reach the "end" state.
- If the end state has a penny in some box that is not the leftmost box, it must have been placed there by some redistribution action.

- Each redistribution decreases the total penny count by 1. Therefore, if we begin with n pennies, we reach an end state in fewer than n steps. (By definition, if 1 penny is left, it must be an end state.)

Now, we have some useful tools with which to approach the first problem: "Does the final distribution of pennies depend on the order in which the machine processes the coin pairs?"

At first glance, it may not be clear whether the answer is "yes" or "no". Because we may have multiple redistribution actions to choose from at some intermediate state, there may be many different ways in which an initial state can be brought to an end state. However, the regularity of the redistribution action may make us think that ordering doesn't matter: if $n = 2k$ (or $2k + 1$), then there must be k redistributions from box 0 into box 1. This puts k total pennies into box 1, though possibly at different times depending on how we choose to perform redistribution actions.

This suggests that we may be able to pursue some kind of backwards reasoning: given a penny in some box i, we should be able to "retrace" its steps to the initial state: From some end state with a penny in box i ($i > 0$), it must have come from two pennies in box $i - 1$. Similarly, if $i - 1 \neq 0$ (i.e. those two pennies did not come from the n pennies in the first box), each of the two pennies in box $i - 1$ must have come from 2 pennies each (4 total) in box $i - 2$. This leads us to the following useful statement:

For any penny in box i of the end state, it must have been placed there through a series of distribution actions beginning with 2^i pennies in the initial state (i.e. from 2^i pennies in box 0).

1.3 Solution (Question 1)

Let b_i be the number of pennies in box i of the ending state. Then $n = 2^0 \cdot b_0 + 2^1 \cdot b_1 + 2^2 \cdot b_2 + \cdots + 2^i \cdot b_i + \ldots$.

For those familiar with changing numerical bases, it is clear that the end state is the binary (base 2) representation of n. In such case, the answer to the first question is "no", because each integer has a unique (finite) binary representation. However, it may be useful to verify that this is the case.

Since we wish to show that each integer has a unique binary representation, this suggests a proof by contradiction: suppose n has two different binary representations ($b_0, b_1, b_2, \ldots$, and $a_0, a_1, a_2, \ldots$, where b_i is digit i of one representation, and a_i is digit i of the other representation). Because n is finite, there is some largest j such that $a_j \neq b_j$. Then we have:

$$\begin{aligned} n = \; & 2^0 \cdot a_0 + 2^1 \cdot a_1 + 2^2 \cdot a_2 + \cdots + 2^j \cdot a_j + \ldots = 2^0 \cdot b_0 + 2^1 \cdot b_1 + 2^2 \cdot b_2 + \cdots + 2^j \cdot b_j + \ldots \\ = \; & 2^0 \cdot a_0 + 2^1 \cdot a_1 + 2^2 \cdot a_2 + \cdots + 2^j \cdot a_j = 2^0 \cdot b_0 + 2^1 \cdot b_1 + 2^2 \cdot b_2 + \cdots + 2^j \cdot b_j \end{aligned}$$

Without loss of generality, let $a_j = 1 > b_j = 0$. Even if we set $a_i = 0$ and $b_i = 1$ for all $i < j$ (that is, the remaining digits are such that minimizes the value of a_i while maximizing the value of b_i) to try and maintain equality, it is still the case that $2^j \cdot 1 > 2^0 \cdot 1 + 2^1 \cdot 1 + 2^2 \cdot 1 + \cdots + 2^{j-1} \cdot 1 + 2^j \cdot 0$ (this is a special case of $x^j - 1 = (x - 1)(x^{j-1} + x^{j-2} + \cdots + 1)$ for $x = 2$). Then the two binary representations are actually for different numbers, and we have a contradiction.

Thus, there is exactly one unique end state for each initial state.

1.4 Solution (Question 2)

Now that we know that the end state is the binary representation of n, the second question reduces to the number of digits in the binary representation. If we are aware of the relationship between a

number and the size of its representation, we can derive that this answer should involve the log function. Otherwise, we may approach this question more systematically:

Let k be the largest value such that b_k is 1 in the end state of n. Then the smallest value of n occurs when all other b_i are 0. Thus, $n \geq 2^k$. Similarly, the largest value of n occurs when all other b_i (for $i < k$) are 1. Thus $n \leq 1 + 2 + 4 + \cdots + 2^k = 2^{k+1} - 1$. Combining these two expressions, we have $2^k \leq n \leq 2^{k+1} - 1$ for an integer n with $k+1$ binary digits. In other words, for each additional power of 2, the number of digits increases by 1. (This should also suggest the use of the logarithm function.)

Otherwise, we can manipulate the expression as follows:

$$\begin{aligned} 2^k \leq n \leq 2^{k+1} - 1 < 2^{k+1} \\ k \leq \log_2(n) \leq \log_2\left(2^{k+1} - 1\right) < \log_2 2^{k+1} \\ k \leq \log_2 n < k + 1 \end{aligned}$$

To deal with the fact that k is an integer, and $\log_2 n$ is usually not (except when n is a power of 2), we can apply the floor ($\lfloor x \rfloor$) and ceiling ($\lceil x \rceil$) functions, which are defined in the following way: $\lfloor x \rfloor = \max\{i : i \leq x, i \in \mathbb{Z}\}$ (the largest integer less than or equal to x), $\lceil x \rceil = \min\{i : i \geq x, i \in \mathbb{Z}\}$ (the smallest integer greater than or equal to x). It is useful to rephrase these definitions as follows: $\lfloor x \rfloor = i \Leftrightarrow i \leq x < i + 1, i \in \mathbb{Z}$ and $\lceil x \rceil = i \Leftrightarrow i - 1 < x \leq i, i \in \mathbb{Z}$.

Thus, it should be clear that $k = \lfloor \log_2 n \rfloor$. However, note that we start indexing the boxes at 0, such that $k = 1$ means that two boxes are used. So the number of boxes used is $\lfloor \log_2 n \rfloor + 1$.

1.5 Solution (Question 3)

Recall that each redistribution reduces the total number of pennies by 1. Thus, the number of redistributions is equal to the difference between the initial and final number of pennies: $n - \sum_i b_i$.

1.6 Variants and Generalizations

1. Instead of replacing 2 pennies with 1 penny, we can replace b pennies with 1 penny. You may want to verify that this achieves the base b representation.

2. What if we replace 3 pennies with 2 pennies? Does this end result behave like a base 1.5 representation?

3. What if the boxes extend in both directions, and each redistribution simply replaces 2 pennies in a box with 1 penny in each box to the left and the right? Clearly, the total number of pennies would not change, but what would the final distribution look like? Would there still be a unique final state for the same initial value of n?

2 Homework

Problem 3: Digit Sum

Without the help of a computer or calculator, find the total sum of the *digits* in all integers from 1 to 1 million, inclusive.

Problem 4: Four Alternating Knights

There are four knights on a 3×3 chessboard: the two white knights are at the two bottom corners,

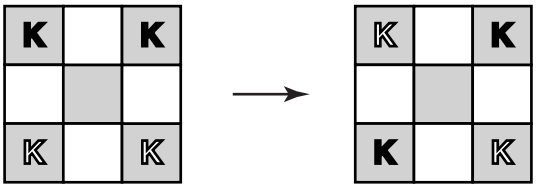

Figure 2: Desired Goal of the Four Alternating Knights puzzle.

and the two black knights are at the two upper corners of the board. Find the shortest sequence of moves to achieve the position shown on the right of Figure 2 or prove that no such sequence exists. Of course, no two knights can ever occupy the same square of the board.

Problem 5: Turning on a Light Bulb

A light bulb is connected to n switches in such a way that it lights up only when all the switches are closed. Each switch is controlled by a push button; pressing the button toggles the switch, but there is no way to know the state of the switch.

(a) Devise a sequence of button pushes that is guaranteed to turn on the light bulb when $n = 3$. What is the minimum number of button pushes in such a sequence?

(b) Design an algorithm to turn on the light bulb with the minimum number of button pushes needed in the worst case.

3 Advanced Problems

Problem 6: Poisoned Wine

An evil king is informed that exactly one of his 1000 wine barrels has been poisoned. The poison is so potent that a miniscule amount of it, no matter how diluted, kills a person on the 30th day after consumption. The king is prepared to sacrifice 10 of his slaves to determine the poisoned barrel.

(a) Can this be done before a feast scheduled in 5 weeks?

(b) Can the king achieve his goal with just 8 slaves?

Problem 7: Catching a Spy

In a computer game, a spy is located on a one-dimensional line. At time 0, the spy is at location a. With each time interval, the spy moves b units to the right if $b \geq 0$, and $|b|$ units to the left if $b < 0$. Both a and b are fixed integers, but they are unknown to you. Your goal is to identify the spy's location by asking at each time interval (starting at time 0) whether the spy is currently at some location of your choosing. For example, you can ask whether the spy is currently at location 19, to which you will receive a truthful yes/no answer. If the answer is "yes", you reach your goal; if the answer is "no", you can ask the next time whether the spy is at the same or another location of your choice.

(a) For $|a| \leq 4$ and $|b| \leq 4$, find a pattern of questions that is guaranteed to catch the spy in fewer than 100 questions.

(b) Devise an algorithm that will find the spy after a finite number of questions, with the only restrictions being that a and b are fixed integers.