Lecture Notes

The topic for the next few weeks is determining the cardinality of a set. (The cardinality of a set is the number of elements that it contains.)

There are many ways of counting the elements in a set. For example, last week we used recurrences to count the number of comparisons used by a sorting algorithm and the number of computer science professors in a given year. In the week before, we used summations to count the number of dollars paid by an annuity.

Another good way to determine the cardinality of a set is to show that the items it contains can be correlated in some way with the items in another set whose cardinality is known. For example, suppose we want to determine the cardinality of the set of students' watches in the 6.042 classroom. The set of watches can be correlated with the set of students; specifically, for each student there is at most one watch. (At least, let's assume this.) Now we know something about the cardinality of the set of students, since there are only 180 people signed up for 6.042. So we can conclude that there are at most 180 student's watches in the classroom.

This type of argument is very simple, but also quite powerful. We will see how to use such simple arguments to prove results that are hard to obtain any other way.

1 Functions

The formal way that we establish a correlation between two sets is by special types of functions called surjections, injections, and bijections. We will define these terms in a minute, but let's start slowly and just recall the definition of a function.

Definition A *function* $f : A \mapsto B$ is a way of relating **every** item in a set A with **precisely one** item in a set B. In particular, the item $a \in A$ is related to an item in B denoted $f(a)$. The set A is called the *domain* of the function, and the set B is called the *range* of the function.

If the sets A and B are small, then we can describe the function with a diagram as in Figure 1. These diagrams are helpful in explaining what a function is and how injections, surjections, and bijections differ.

Definition The *image* of a function $f : A \mapsto B$ is the set $\{b \mid \exists a \; f(a) = b\}$ and is denoted $f(A)$.

In other words, the image is the set of values in the domain that the function f actually takes on. The image is always a subset of the range. For example, consider the function $f : \mathcal{R} \mapsto \mathcal{R}$ defined as follows:

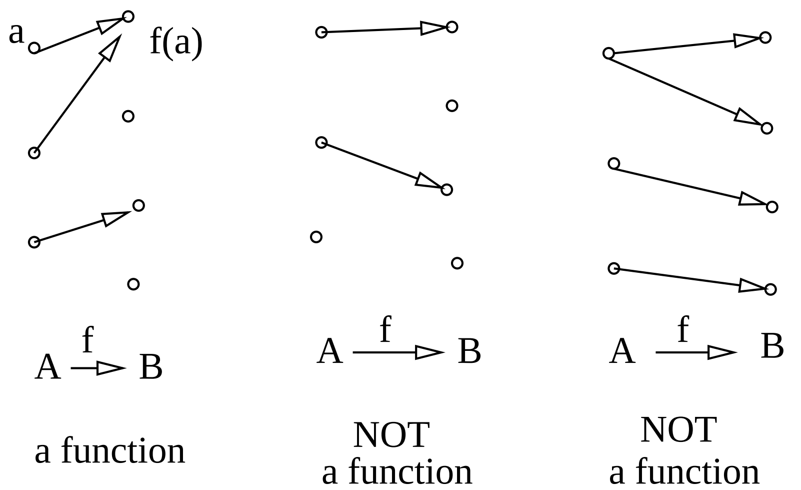

Figure 1: *In each of the three diagrams, A is the set of dots in the left column and B is the set of dots in the right column. The left diagram shows a function; every item in set A is mapped to precisely one item in set B. The center and right diagrams do not depict functions. In the center diagram, not every item in set A is mapped. In the right diagram, an item in set A is mapped to more than one item in set B.*

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

This defines the familiar absolute value function $f(x) = |x|$. Notice that while the domain and range of f are both the set of all real numbers, the image of f is only the set of non-negative real numbers.

As another example, consider the "function" $f : \mathcal{R} \mapsto \mathcal{R}$ defined by $f(x) = \sqrt{x}$. There are two issues here. First, for f to be a function, we must specify that $\sqrt{x}$ denotes just one of the two square roots of x; otherwise, elements in the domain would not be associated with **precisely one** element in the range. For example, 4 could be associated with both 2 and -2. Second, we have not related **every** element in the domain $\mathcal{R}$ with an element in the range $\mathcal{R}$. For example, -1 is an element in the domain not associated with any element in the range. For this reason, f is not a function.

1.1 Surjections, Injections, and Bijections

Now we'll look at three special types of function.

Definition A function $f : A \mapsto B$ is an *surjection* if for all $b \in B$, there exists an $a \in A$ such that $f(a) = b$.

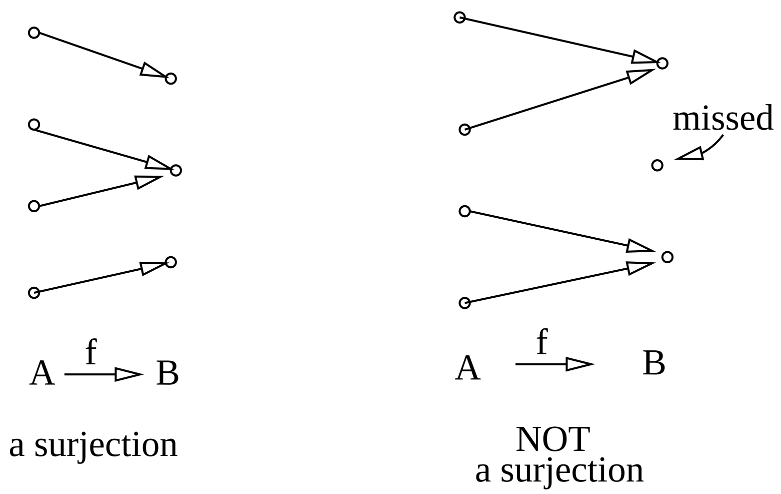

Figure 2: *The left diagram shows a surjection; every item in set B is mapped to by some item in A. The right diagram does not show a surjection, because one item is B is missed.*

In other words, a function f is a surjection if it takes on every value in its range. That is, the image of f is the entire range of f. Figure 2 shows the idea of a surjection with diagrams. A surjective function is also sometimes called an *onto* function.

Definition A function $f : A \mapsto B$ is a *injection* if for all $a, a' \in A$, $(a \neq a') \Rightarrow (f(a) \neq f(a'))$.

Informally, f is an injection if no two items in A are mapped to the same item in B. Note that f need not take on every value in its range; the only restriction is that f cannot take on the same value twice. This idea is shown pictorially in Figure 3. Injective functions are also called *one-to-one*.

Definition A function $f : A \mapsto B$ is a *bijection* if it is both a surjection and an injection.

Bijections are the nicest sort of function because they associate each element of A with exactly one element of B and vice versa. This is shown in Figure 4. Sometimes a bijection is also called a *1-to-1 correspondence* or a *one-to-one and onto* function.

1.2 Examples of Functions

Define $f : \mathcal{N} \mapsto \mathcal{N}$ by $f(x) = 2x$. This is a function, since it maps every number in the domain $\mathcal{N}$ to precisely one number in the range $\mathcal{N}$. The function is injective (or "one-to-one") because no two distinct integers are mapped to the same value. This fact is easy to see, but let's prove it formally:

Fact 1 The function f is one-to-one.

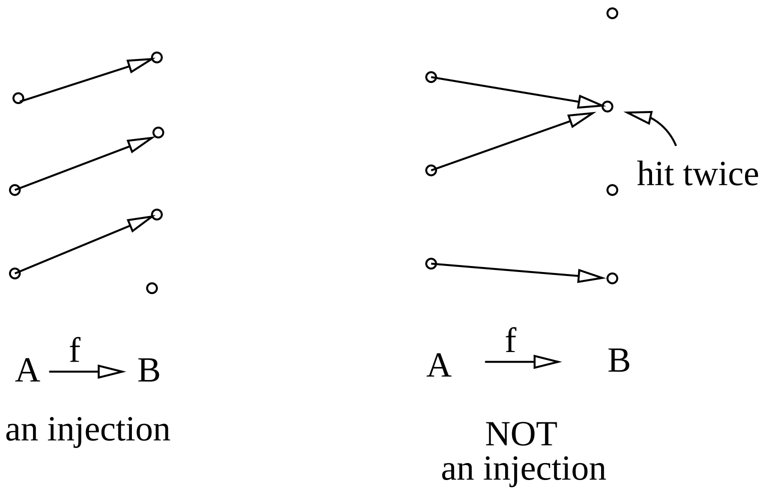

Figure 3: *The left diagram shows an injection; no two items in A are mapped to the same item in B. The right diagram does not show an injection, because two items in A are mapped to the same item in B.*

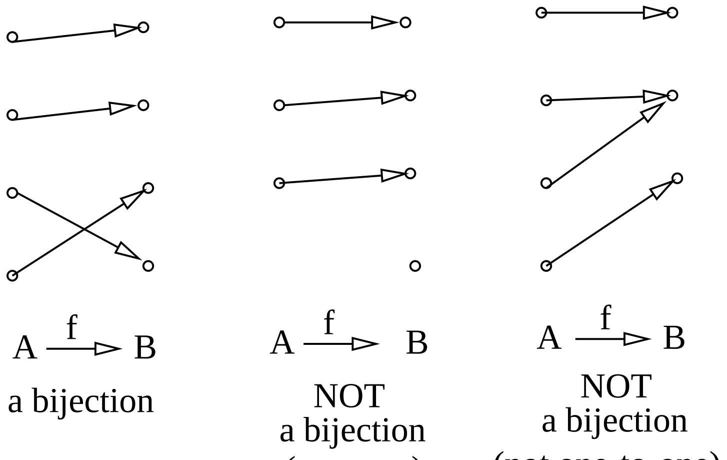

Figure 4: *The left diagram shows a bijection; each element of A is associated with exactly one element of B and vice versa. The center diagram does not show a bijection because an element of B is not associated with any element of A; f is not onto. The right diagram also does not show a bijection. In this case, the reason is that one element of B is associated with two elements of A; f is not one-to-one.*

Proof. We must show that for all $x, x' \in \mathcal{N}$, $(x \neq x') \Rightarrow (f(x) \neq f(x'))$. We will prove the contrapositive; that is, for all $x, x' \in \mathcal{N}$, $(f(x) = f(x')) \Rightarrow (x = x')$. For all $x, x' \in \mathcal{N}$, we have:

$$\begin{aligned} f(x) = f(x') \quad &\Rightarrow \quad 2x = 2x' \\ &\Rightarrow \quad x = x' \end{aligned}$$

The first equality follows by substituting the definition of $f(x)$, and the remainder is simplification. ∎

The function f is not surjective (or "onto"). In other words, f does not take on all values in the range $\mathcal{N}$.

Fact 2 The function f is not onto.

Proof. The proof is by contradiction. Assume for the purpose of contradiction that f is onto. Then there exists an $x \in \mathcal{N}$ such that $f(x) = 3$. Substituting the definition of f gives $2x = 3$ or $x = 3/2$. But this contradicts the assumption that $x \in \mathcal{N}$. ∎

2 Counting

Injections, surjections, and bijections are useful in counting the number of items in a set.

Suppose there is an injection from A to B. For example, let A be a set of balls, and let B be a set of boxes. Then an injection from A to B is a way of assigning balls to boxes so that no box gets multiple balls. If such an assignment exists, then intuitively there can be no more balls than boxes. We can state this as a theorem.

Theorem 2.1 If A and B are finite sets and $f : A \mapsto B$ is an injection, then $|A| \leq |B|$.

Proof. Suppose A is the n-element set $\{a_1, a_2, \ldots, a_n\}$. Since f maps A to B, the set $\{f(a_1), f(a_2), \ldots f(a_n)\}$ is a subset of B. Because f is an injection, all the $f(a_i)$ are different and so B also contains at least n elements. Therefore, $|A| \leq |B|$. ∎

The condition in the theorem that sets A and B are finite is important. For example, suppose $A = \{0, 1, 2, 3, \ldots\}$ and $B = \{0, 2, 4, 6, \ldots\}$. The function $f : A \mapsto B$ defined by $f(x) = 2x$ is an injection. This would seem to imply that A is no larger than B. However, in fact A contains all the natural numbers, but B contains only the even natural numbers! We will resolve this apparent paradox in the next lecture; for today we'll avoid the problem by sticking with finite sets.

Suppose again that A is a set of balls and B is a set of boxes. If there is a surjection from A to B, then we can assign balls to boxes so that every box gets at least one ball. This suggests that there must be as many balls as there are boxes.

Theorem 2.2 If A and B are finite sets and $f : A \mapsto B$ is a surjection, then $|A| \geq |B|$.

The proof is similar to that of the preceding theorem. Taken together, these two theorems have an important corollary.

Corollary 2.3 If A and B are finite sets and $f : A \mapsto B$ is a bijection, then $|A| = |B|$.

Proof. Theorem 2.1 implies that $|A| \leq |B|$. Theorem 2.2 implies that $|A| \geq |B|$. Therefore, $|A| = |B|$. ■

This corollary is used all the time. Suppose that we want to know the cardinality of some set A and already know the cardinality of a set B. If we can find a bijection between A and B, then the corollary implies that $|A| = |B|$. This idea is very simple, but amazingly useful.

2.1 Example of Bijection: Subsets of a Set

Here is an example of a counting problem that can be solved by finding a bijection. Suppose we have a set with n elements; how many subsets are there?

The following definition lets us restate this as a question about the cardinality of a set.

Definition The *power set* of a set $\mathcal{U}$, denoted $2^{\mathcal{U}}$, is the set of subsets of $\mathcal{U}$:

$$2^{\mathcal{U}} = \{S \mid S \subseteq \mathcal{U}\}$$

For example, if $\mathcal{U} = \{u_1, u_2\}$, then $2^{\mathcal{U}} = \{\{u_1\}, \{u_2\}, \{u_1, u_2\}, \phi\}$. In this case, $\mathcal{U}$ has four subsets and, equivalently, the power set $2^{\mathcal{U}}$ has four elements.

Our problem, therefore, is to determine the cardinality of the power set of an n-element set. Our method will be to establish a bijection between the power set and the set of all n-bit binary strings. Since there are 2^n such binary strings, Corollary 2.3 will imply that the power set has cardinality 2^n. (We could formally prove that the number of n-bit binary strings is 2^n using induction, but this is practically an axiom for computer science.) The details are given in the following theorem and proof.

Theorem 2.4 Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$. Then $|2^{\mathcal{U}}| = 2^n$.

Proof. We will establish a bijection from $2^{\mathcal{U}}$ to the set of all n-bit binary strings, $B_n = \{b_1 b_2 \ldots b_n \mid b_i \in \{0, 1\}\}$. This is sufficient to prove the theorem; since $|B^n| = 2^n$, Corollary 2.3 will imply that $|2^{\mathcal{U}}| = 2^n$.

Let $f : 2^{\mathcal{U}} \mapsto B_n$ be the function defined by $f(S) = b_1 b_2 \ldots b_n$ where:

$$b_i = \begin{cases} 1 & \text{if } u_i \in S \\ 0 & \text{if } u_i \notin S \end{cases}$$

(This is a digression from the proof to give some intuition about the bijection f. As an example, if $\mathcal{U} = \{u_1, u_2\}$, then the values of f are as shown below.

$$\begin{aligned} f(\{u_1, u_2\}) &= 11 \\ f(\{u_1\}) &= 10 \\ f(\{u_2\}) &= 01 \\ f(\{\phi\}) &= 00 \end{aligned}$$

In this example, f is certainly a bijection; every subset is matched with a unique binary string and vice-versa. To complete the theorem, we must show that f is a bijection in general.)

To prove that f is a bijection, we must show f is a one-to-one and onto function. First, f is a well-defined function, since every element of the domain $2^{\mathcal{U}}$ is mapped to an element of the range B_n.

Next, we must show that f is one-to-one. In other words, for all $S, S' \in 2^{\mathcal{U}}$, if $S \neq S'$, then $f(S) \neq f(S')$. We will prove the contrapositive; that is, for all $S, S' \in 2^{\mathcal{U}}$, if $f(S) = f(S') = b_1 b_2 \ldots b_n$, then $S = S'$. This is true because $u_i \in S \Leftrightarrow b_i = 1 \Leftrightarrow u_i \in S'$. Every element in S is also in S' and vice-versa.

Finally, we must show that f is onto; that is, for all $b_1 b_2 \ldots b_n \in B$, there exists an $S \in 2^{\mathcal{U}}$ such that $f(S) = b_1 b_2 \ldots b_n$. To prove this, we can construct such an S explicitly: $S = \{u_i \mid b_i = 1\}$. Then $f(S) = b_1 b_2 \ldots b_n$ as desired. ■

This is a simple example of the use of bijection to find the cardinality of a set; we'll see examples later that are more interesting. In more complicated problems, finding a bijection can require considerable ingenuity.

3 The Pigeonhole Principle

The theorem about injections, Theorem 2.1, has another important corollary:

Corollary 3.1 Let A and B be finite sets. If $|A| > |B|$ and $f : A \mapsto B$, then there exist distinct elements a and a' in A such that $f(a) = f(a')$.

Proof. The contrapositive of Theorem 2.1 is that if $|A| > |B|$, then f is not an injection. The corollary then follows from the definition of an injection. ■

This corollary is a very famous result known as the Pigeonhole Principle. The name derives from the following, more colorful statement of the corollary:

Pigeonhole Principle: If $n + 1$ pigeons are put into n holes, then at least one hole contains two or more pigeons.

Proof. Let A be the set of "pigeons", let B be the set of "holes", and let the function $f : A \mapsto B$ define the assignment of pigeons to holes. Since $|A| > |B|$, Corollary 3.1 implies that there exist two distinct pigeons assigned to the same hole. ■

As a trivial application of the Pigeonhole Principle, suppose that there are three people in a room. The pigeonhole principle implies that two have the same gender. In this case, the "pigeons" are the three people and the "pigeonholes" are the two possible genders, male and female. Since there are more pigeons than holes, two pigeons must be in the same pigeonhole; that is, two people must have the same gender.

The pigeonhole principle seems too obvious to be really useful, but the next two examples show how it gives short proofs of results that are difficult to obtain by other means.

3.1 Pigeonhole Principle Example: A Final Exam Question

A problem on an old final exam was to prove the following claim:

Claim In every set of 1000 integers, there are two integers x and y such that $573 \mid (x - y)$.

At first glance, this looks very hard! Those 1000 numbers could be anything! Since there are no less than 1000 integer-valued variables here, even our old standby, induction, seems hopeless. Surprisingly, however, there is a short proof using the Pigeonhole Principle.

To apply the Pigeonhole Principle, we must identify two things: pigeons and holes. Furthermore, to prove anything with the Pigeonhole Principle, we must have more pigeons than holes. Since there are only two numbers mentioned in this problem, a natural thing to try is 1000 pigeons and 573 holes.

Under this interpretation, a pigeon is an integer, but what is a hole? Ideally, the existence of two pigeons and in the same hole should be equivalent to the existence of two numbers x and y such that $573 \mid (x - y)$. This suggests numbering the holes $0, 1, \ldots 572$ and putting in hole n all integers congruent to n modulo 573. Now we can construct a proof:

Proof. Let S be a set of 1000 integers. Let $M = \{0, 1, \ldots 572\}$. Let $f : S \mapsto M$ be the function defined by $f(n) = n \bmod 573$. Since $|S| > |M|$, Corollary 3.1 implies that there exist distinct elements x and y in S such that $f(x) = f(y)$. This means $(x \bmod 573) = (y \bmod 573)$ and so $573 \mid (x - y)$. ■

Really there was nothing special about the numbers 1000 and 573 other than the fact that $1000 > 573$. We could have made a stronger claim: if $n > m$, then in every set of n integers, there are two integers x and y such that $m \mid (x - y)$.

3.2 Pigeonhole Principle Example: Weighing Balls

In the first tutorial you were given the following problem. There are 12 balls of which 11 have the same weight and one has a different weight. With three weighings on a balance scale, you must identify the odd ball and determine whether it is heavier or lighter than the rest. (A balance scale has a left pan and a right pan. In a weighing you put some balls in each pan. The scale then reveals whether the left pan is heavier, the right pan is heavier, or the two are equal.)

The problem is solvable using a tricky algorithm represented by a ternary tree as shown in Figure 5. Each internal node in the tree represents a weighing such as, "put balls 1, 3, and 5 on one side and 2, 8, and 10 on the other side". Each leaf node represents a result such as, "the 11th ball is heavier than the rest". A run of the algorithm corresponds to a path from the root to a leaf. At each internal node, we perform the weighing associated with the node. Based on the result, we descend to one of the three children. When we reach a leaf, we output the indicated result.

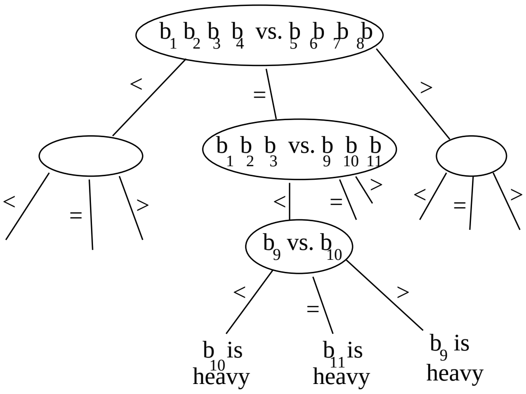

Figure 5: *A strategy for the weighing problem can be represented by a ternary tree. Each internal node represents a weighing. Each leaf node represents a result. A run of the algorithm corresponds to a path from the root to a leaf. At each internal node, we perform the weighing associated with the node and descend to a child based on the result. At a leaf, we output the indicated result.*

Suppose that we wanted to solve the same problem with more balls, but still with only three weighings. The case with 13 balls is complicated and we'll put it off a bit. However, if there are 14 balls, then we can prove that no solution exists. This would seem to require an elaborate case analysis to rule out every possible strategy. Actually, we can use the Pigeonhole Principle to give a short proof that no weighing strategy exists for 14 balls.

Claim 3.2 The weighing problem can not be solved with 14 balls and 3 weighings.

Proof. The pigeons are the possible situations. Since any one of the 14 balls could be the odd one and the odd ball could be either heavier or lighter than the rest, there are 28 possible situations and so 28 pigeons. The holes are the 27 leaves of the depth-3 ternary tree used to represent the weighing strategy. Since there are more pigeons than holes, the Pigeonhole Principle implies that some hole must have two pigeons; that is, some leaf is not associated with a unique situation. Therefore, for every weighing strategy, there is some pair of situations that the strategy cannot distinguish. In at least one of these situations, the strategy must give the wrong answer. ■

What if we had four weighings— could we identify the odd ball from among 41? In this case, there are $2 \cdot 41 = 82$ possible situations or pigeons. Every strategy can be represented by a depth-4 ternary tree with $3^4 = 81$ leaves or pigeonholes. Once again, the Pigeonhole Principle implies that in every weighing strategy, some leaf of the tree must be associated with two or more situations. The strategy cannot distinguish these situations and so must sometimes give the wrong answer.

In general, if there are n balls, we can compute a lower bound on the number of weighings necessary. In this case, there are $2n$ possible situations. If there are w weighings, then every strategy can be represented by a ternary tree with 3^w leaves. For a correct weighing strategy to exist, there must be as many leaves as situations. That is, we must have $3^w \geq 2n$ or, equivalently, $W \geq \log_3 2n$.

Now we can address the tricky case of 13 balls and 3 weighings. In this case, we can not use the Pigeonhole Principle directly to prove that there is no solution, because there are 26 situations and a depth-3 tree has 27 leaves. There may be a solution! But note that just because our Pigeonhole Principle argument does not rule out a weighing strategy, does not imply that a such a strategy exists! In fact, a closer analysis shows that there is no solution.

Claim 3.3 The weighing problem can not be solved with 13 balls and 3 weighings.

To prove the claim, we must do some of the hairy case analysis that the Pigeonhole Principle lets us circumvent in the case of 14 balls. Nevertheless, appeals to the Pigeonhole Principle do greatly simplify the proof.

Proof. Let n be the number of balls placed in each pan of the balance scale on the first weighing. Since there are 13 balls, n could be 1, 2, 3, 4, 5, or 6. We can actually reduce these 6 cases to just two.

In the first case, suppose that $n \leq 4$. This means that at least 5 balls are not used in the first weighing. As a result, there are at least 10 situations in which the scale will be balanced on the first weighing. (The odd ball could be any of the unweighed balls, and it could be either heavy or light.) By the Pigeonhole Principle, we can not distinguish between these 10 or more cases with only two additional weighings.

In the second case, suppose that $n \geq 5$. Now there are $2n \geq 10$ situations in which the right pan is heavier: n when the odd ball is heavy and in the right pan and n more when the odd ball is light and in the left pan. Again, by the Pigeonhole Principle, we can not distinguish these 10 or more cases with only two weighings. ∎

3.3 20 Questions and Binary Search

Here is a game. I think of an animal. You can ask me 20 questions that take a yes/no answer such as, "Is the animal bigger than a breadbox?" To win the game, you must ask a question like, "Is the animal a walrus?" or "Is the animal a zebra?" and receive a "yes" answer. In effect, you have 19 questions to determine which animal I am thinking of, and then you must use 1 question to confirm your guess.

Suppose that I know a million animals. Can you always determine which animal I am thinking of? Any questioning strategy you use can be represented by a depth-19 binary tree as shown in Figure 6. As in the weighing problem, each internal node in the tree represents a question, and each leaf represents a final guess at my animal. A depth-19 binary tree can have at most $2^{19} = 524,233$ leaves, and I can use any of a million animals. By the pigeonhole principle, at least two animals must be associated with some leaf in the tree; this implies that you can not always determine which animal I am thinking of with only 19 questions.

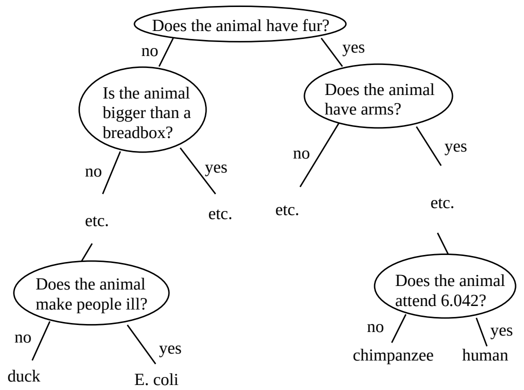

Figure 6: *A strategy for the animal game can be represented by a depth-19 binary tree. Each internal node represents a yes/no question such as, "Does the animal have fur?" Each leaf node represents a final guess. A run of the algorithm corresponds to a path from the root to a leaf.*

More generally, if I know n animals, then $\lceil \log_2 n \rceil$ questions are necessary to always identify the one I'm thinking of; a binary tree of lower depth must have fewer than n keaves, and so some animals can not be distinguished.

A similar argument applies to the binary search algorithm. In a binary search, we are looking for a particular item in a *sorted* list. We begin by comparing the middle element in the list to the item we are looking for. If our item precedes the middle element, then we continue the search recursively in the first half of the list. Similarly, if our item follows the middle element, then we search recursively in the second half of the list. For example, binary search could be applied to the animal game. You could sort the list of a million animals, pick out the middle one, and begin with a question like, "Does the animal alphabetically precede marmot?" Not surprisingly, given the similarity between the animal game and binary search, a Pigeonhole Principle argument shows that binary search requires $\Omega(\log n)$ steps to find an item in an n-element list in the worst case.

(There is a small difference between the animal game and a binary search. A "yes" in the animal game corresponds to "less than" in a binary search comparison, and a "no" corresponds to "greater than" (or vice versa). However, in a binary search, a comparison may return "equal" and this has no analogue in the animal game. This extra possibility reduces the time required for a binary search, but only by a constant factor.)