CS 70 Discrete Mathematics and Probability Theory
Spring 2015 Vazirani HW 8

Reminder: Deadline for HW7 self-grade is Thursday, March 12 at noon.

Reading: Note 10 and Note 11.

Due Monday March 16

1. **Infinity**

   For each of the following sets, indicate whether it is finite, countably infinite, or uncountable. Give a one sentence justification for your choice:

   (a) $\mathbb{Q}$ (the set of all rational numbers)

   (b) $\mathbb{R}$ (the set of all real numbers)

   (c) $\mathbb{C}$ (the set of all complex numbers)

   (d) $\{0,1\}^*$ (the set of all finite-length binary strings)

   (e) $\{0,1,2\}^*$ (the set of all finite-length ternary strings)

   (f) $\mathbb{N}^4 = \{(a,b,c,d) : a,b,c,d \in \mathbb{N}\}$ (the set of quadruples of natural numbers)

   (g) $\mathbb{Z}^n = \{(a_1,a_2,\ldots,a_n) : a_i \in \mathbb{Z}\}$ for $n > 0$ (the set of n-tuples of integers)

   (h) $S = \{p(x) : p(x) = ax^2 + bx + c$, where $a,b,c \in \mathbb{Z}\}$ (the set of all polynomials of degree at most 2 with integer coefficients)

   (i) $T = \{p(x) : p(x) = a_n x^n + \cdots + a_1 x + a_0$, where $n \in \mathbb{N}$ and $a_0, a_1, \ldots, a_n \in \mathbb{Z}\}$ (the set of all polynomials with integer coefficients, of *any* degree)

   (j) $U = \{\frac{p(x)}{q(x)} : p(x) = a_n x^n + \cdots + a_1 x + a_0$ and $q(x) = b_n x^n + \cdots + b_1 x + b_0$, where $n \in \mathbb{N}$ and $a_0, a_1, \ldots, a_n, b_0, b_1, \ldots, b_n \in \mathbb{Z}\}$ (the set of all rational functions with integer coefficients, of *any* degree)

   (k) The set of all primes.

   Answer:

   (a) Countably infinite. Note 10 shows a simple injection from $\mathbb{Q}$ to $\mathbb{N}$, defined using a spiral on $\mathbb{Z}^2$ (p. 3 of Note 10).

   (b) Uncountable. Shown in Note 10 using Cantor's diagonalization method.

   (c) Uncountable. Obvious because $\mathbb{C}$ contains $\mathbb{R}$ as a subset.

   (d) Countably infinite. We can count finite-length binary strings by listing them in increasing order of length and then in lexicographic order.

   (e) Countably infinite. We can list them in increasing order of length and then in lexicographic order.

(f) Countably infinite. We can list them in increasing order of $a+b+c+d$ and then in lexicographic order.

(g) Countably infinite. Since $\mathbb{Z}$ is countable, it suffices to show that $\mathbb{N}^n$ is countable. We can list them in increasing order of the sum of all entries and then in lexicographic order, as in previous part.

(h) Countably infinite. The cardinality of S is the same as the cardinality of $\mathbb{Z}^3$.

(i) Countably infinite. We can think of T as $\mathbb{Z} \cup \mathbb{Z}^2 \cup \mathbb{Z}^3 \cup \cdots$. Since $\mathbb{Z}^i$ for any $i > 0$ is countable, we can index every element of T by first specifying i and then the index j of that element within $\mathbb{Z}^i$. In other words, there exists a bijection between T and $\{(i,j) : i,j \in \mathbb{N}\} = \mathbb{N}^2$. Hence, the cardinality of T is the same as that of $\mathbb{N}^2$.

(j) Countably infinite. The cardinality of U is at most the cardinality of T^2, and we know that T is countable.

(k) Countably infinite. We know that there are infinitely many primes. Countable because it is a subset of $\mathbb{N}$.

2. **Subsets**

Consider the set S of all (possibly infinite) subsets of $\mathbb{N}$.

(a) Show that there is a bijection between S and $T = \{f : \mathbb{N} \to \{0,1\}\}$ (the set of all functions that map each natural number to 0 or 1).

(b) Prove or disprove: S is countable.

Answer:

(a) We note that any subset A of $\mathbb{N}$ can be regarded as a function f_A from $\mathbb{N}$ to $\{0,1\}$ and vice versa; simply set $f_A(x) = 1$ if and only if $x \in A$.

(b) Note that each $f \in T$ can be viewed as a binary encoding of a real number between 0 and 1, where $f(n)$ corresponds to the $(n+1)$-th digit after the dot. (E.g. if the real number is $0.1011\ldots$ in binary, then $f(0) = 1$, $f(1) = 0$, $f(2) = 1$, $f(3) = 1$, and so on.)
This exhibits an onto map from T to the real interval $[0,1]$. Hence, the cardinality of T is at least that of $[0,1]$, but $[0,1]$ is uncountable by Cantor's diagonalization. By (a), S is also uncountable.

3. **Rationality**

(a) Show that if A and B are countably infinite sets, then $A \cup B$ is also countably infinite.

(b) A real number which is not a rational number is said to be *irrational*. Using (a), prove or disprove that the set of all irrational numbers is countable.

Answer:

(a) If we have two countable sets A and B, we can count their union by counting A with even numbers and B with odd numbers.
Since A and B are countably infinite, there exist bijections $f_A : \mathbb{N} \to A$ and $f_B : \mathbb{N} \to B$. We construct $f : \mathbb{N} \to A \cup B$ as follows:

$$f(n) = \begin{cases} f_A(\frac{n}{2}) & \text{if } n \text{ is even;} \\ f_B(\frac{n-1}{2}) & \text{if } n \text{ is odd.} \end{cases}$$

Observe that f hits every element of $A \cup B$, because f_A and f_B hit every element of A and B respectively. Therefore $A \cup B$ is countably infinite.

(b) Assume for the sake of contradiction that the set of irrational numbers $\mathbb{I}$ is countable. Then, since the set of rational numbers $\mathbb{Q}$ is also countable, by part (a), we have that $\mathbb{I} \cup \mathbb{Q}$ should also be countable. However, note that the union of $\mathbb{I}$ and $\mathbb{Q}$ is precisely the set of all real numbers, which we know to be uncountable by Cantor's diagonalization. We have come to a contradiction, and therefore our initial assumption that $\mathbb{I}$ is countable must have been false.

4. **Computable or Uncomputable**

A function $f: \mathbb{N} \to \mathbb{N}$ is said to be computable if there exists a program that takes x as input and produces $f(x)$ as output.

(a) Prove or disprove: every increasing function $f: \mathbb{N} \to \mathbb{N}$ (i.e., if $x \geq y$, then $f(x) \geq f(y)$) is computable.

(b) Prove or disprove: every decreasing function $f: \mathbb{N} \to \mathbb{N}$ (i.e., if $x \geq y$, then $f(x) \leq f(y)$) is computable.

Answer:

(a) No, there exists an increasing function that is uncomputable.

We prove this by an explicit reduction of the halting problem to this problem. The trick is simple; we know that the set of all programs is countable (Problem 1(d)) and therefore we can list all programs and index them by natural numbers. Thus we simply define an increasing function f such that $f(x)$ is equal to $f(x-1)$ if and only if program #x halts. If this function were computable, we can determine whether program #i halts by computing $f(i)$ and $f(i-1)$ and comparing them. This is a contradiction to the uncomputability of the halting problem, and therefore f must be uncomputable.

(Note: there is another approach to this problem that uses a counting argument. To do this, we first show that there are uncountably many increasing functions. In fact, we can map each increasing function f to a (possibly infinite) subset $A_f \subseteq \mathbb{N}$ as follows: $x \in A_f$ if and only if $f(x) < f(x+1)$. Noting that this map is onto, we conclude that there are at least as many increasing functions as there are subsets of $\mathbb{N}$. The latter is uncountable by Problem 2, and therefore the former is also uncountable.

On the other hand, the set of all programs is countable. Since each program can compute at most one function, there are only countably many computable functions. Hence, there must be increasing functions that are not computable.)

(b) Yes. For any decreasing function $f: \mathbb{N} \to \mathbb{N}$, there is always a program that computes it. To see this, let $m = f(0)$. Since the value of f cannot be negative, this means that f can decrease at at most m places, after which it will be constant. Therefore, we can easily compute f by writing a program that knows m along with where those descents are located.

5. **Hello World**

(a) You want to determine whether a program P on input x outputs "Hello World" before executing any other statement. Is there a computer program that can perform this task? Justify your answer.

(b) You want to determine whether a program P on input x outputs "Hello World". Is there a computer program that can perform this task? Justify your answer.

Answer:

(a) There is a computer program that can perform this task. Simply run the first operation of program P, and check if that operation outputs "Hello World".

(b) There is no computer program that can perform this task, because we can reduce the halting problem to this problem.

To do this, suppose for sake of contradiction that we could determine whether any program outputs "Hello World". Then we could solve the halting problem as follows: for any program Q, modify it to obtain another program Q' such that:

1. Q' emulates Q except that it suppresses all printing instructions.
2. If Q halts, Q' prints "Hello World" before halting.

Then determining whether Q' prints "Hello World" is equivalent to determining whether an arbitrary program Q halts, which is exactly the halting problem. Since the halting problem is uncomputable, this is a contradiction. Thus we conclude that there is no computer program that can determine whether an arbitrary program eventually outputs "Hello World".

6. **Gilbert's Hotel**

Gilbert opens a new hotel with a countably infinite set of rooms, numbered $1, 2, 3, \ldots$. As people arrive, they are assigned rooms, one at a time. After a particularly busy week, Gilbert's hotel is full – every room has a person inside.

(a) Another person arrives and asks for a room. How would Gilbert be able to accommodate this person in his hotel? (Gilbert can ask his guests to assist by moving to different rooms.)

(b) A bus arrives with a countably infinite set of people. How would Gilbert be able to accommodate all of these people in his hotel?

Answer:

(a) Ask each person in room n to move to room $n+1$. Every person moves to a unique room, and this opens up room number 1, so the new guest can be placed there.

(b) Ask each person in room n to move to room $2n$. This opens up every odd numbered room, and we can place the new arrivals into those rooms.

7. **Extra Credit**

In class we discussed how programs can print their own description. In this question you will prove the recursion theorem, which goes further — not only are there programs that can obtain their own description, they can also use their descriptions to perform an arbitrary computation.

(a) The recursion theorem states that given any program $P(x,y)$, we can construct another program $Q(x)$ such that $Q(x) = P(x,Q)$, i.e., Q behaves exactly as P would if its second input was the description of the program Q. In this sense Q may be thought of as having access to its own description. Explain how you can construct Q. Justify your answer.

(b) The recursion theorem can be used to give very simple proofs of uncomputability. Show that (a) yields a simple proof of the uncomputability of the halting problem.

Answer:

(a) We desire $Q(x) = P(x,Q)$, where the difficulty is that Q cannot have an explicit self-reference in its description. How do we circumvent this problem? Suppose we have a program A that takes one input. By default A may not know its own description, but what if we run $A(A)$, i.e. we run A on its own description as input?

To use this idea, suppose we define $Q(x) = P(x,A(A))$. We want $A(A)$ to produce the description of Q, i.e. the program that maps x to $P(x,A(A))$. Since we already know that A will be given its own description as input, we can simply define A as follows:

$A(y)$ is the program that outputs the description of the program that maps x to $P(x,\mathbf{y}(y))$ ($\mathbf{y}$ denotes that y is being interpreted as a program).

To see that this construction works, we ask what $A(A)$ is. Indeed, it follows by definition of A that $A(A)$ returns the description of the program that maps x to $P(x,A(A))$, which is nothing but Q! Therefore, we have $Q(x) = P(x,Q)$ as desired.

(b) If the halting problem was computable, one could write a program $P(x,y)$ that does the following:

i. Using the algorithm for the halting problem, decide whether program y halts on input x.

ii. If program y does not halt on input x, then $P(x,y)$ halts. Otherwise, $P(x,y)$ gets into an infinite loop.

Now, using (a), we construct $Q(x)$ such that $Q(x) = P(x,Q)$ and ask whether $Q(x)$ halts. If $Q(x)$ halts, then $P(x,Q)$ does not halt, and therefore $Q(x)$ does not halt, which is a contradiction. If $Q(x)$ does not halt, then $P(x,Q)$ halts, and therefore $Q(x)$ halts, which is again a contradiction.

Hence we conclude that the halting problem is uncomputable.