

November 1, 2007

1 Countable Sets

1.1

Reference: pp. 174-178 of the text.

An infinite set is *countable* if there is an infinite *list* containing all the elements of the set. (“List” here means a 1-1 correspondence with the positive integers.)

1.2

The set of all integers is countable:

$$0, 1, -1, 2, -1, 3, -3, 4, -4, \dots$$

1.3

The set of rational numbers is countable:

$$\frac{0}{1}, \frac{0}{2}, \frac{1}{1}, \frac{0}{3}, \frac{1}{2}, \frac{2}{1}, \frac{0}{4}, \dots$$

The list is made by first writing down all the fractions in which the sum of the numerator and denominator is 1, then those for which the sum is 2, then 3, etc. For each value of the sum, there are only finitely many fractions to list. Note that the above list enumerates only the nonnegative rational numbers, but we can get an enumeration of all the rationals by alternating negative and positive values as with the integers. Note also that there are many repeated values, but this is not a problem: If you strike off all but the first occurrence of each value on the list, you get a one-to-one enumeration of the rationals.

1.4

If Σ is a finite alphabet then Σ^* is countable. We illustrate with $\Sigma = \{a, b\}$:

$$\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, \dots$$

The idea is to enumerate first the string of length 0, then those of length 1, then those of length 2, etc. For each length there are only finitely many strings.

Problem: What if we had a *countable* alphabet $\Sigma = \{a_1, a_2, \dots\}$? The above argument does not work, because we have infinitely many strings of each length. But still the set is countable.

2 Uncountable Sets

In class and in the text we saw the proof that there is no enumeration r_1, r_2, \dots of all the real numbers between 0 and 1. The proof is the famous diagonal argument: If you had such a list, construct the number $s = 0.d_1d_2\dots$ (this is the decimal expansion). The i^{th} digit d_i of s is 1 if the i^{th} digit of r_i is 2, and 2 if the i^{th} digit of r_i is different from 2. s cannot appear on the list, because if it did, it would be r_k for some k . But s and r_k differ in the k^{th} digit. (This argument avoids the problem of numbers like $0.200000000\dots$ which has an alternative representation as $0.19999\dots$)

3 Computable Numbers

This is a real number whose infinite decimal expansion is generated by a Java program like

```
public static void main(String[] args)
{
    int x;
    //section without I/O statements
    while(true)
    {
        //section without I/O statements
        if(x==0)
            System.out.print('0');
        else if(x==1)
            //more like this
        else if(x==8)
            System.out.print('8');
        else
            System.out.print('9');
    }
}
```

As we saw in class, $\sqrt{2}$ is computable (although irrational), as are numbers like π , $\log_{10} 2$, etc.

The set of computable numbers is countable, since each program is a string over the ASCII character set, and we saw that the set of all strings is countable.

4 A Paradox

We can go farther: There is a Java method

```
String enumerate(int i)
```

that returns the i^{th} ASCII string in the enumeration of Strings. Certainly we can determine with a Java program if a String is a correct Java program (that's part of what compilers do!), so there is a Java method

```
boolean isValidNumberComputer(String s)
```

that determines this. An interpreter will not only determine if a String is a correct Java program, but simulate the Java code as well. This means that there is a Java method

```
int simulate(String numberComputer, int step)
```

that takes a String, determines if it is a valid number computer, and then simulates it for `step` times through the while loop, returning the value it would have printed on the last step.

This means that we can produce the following program, which simulates the diagonal argument:

```
public static void main(String[] args)
{
    String s="";
    int stringnum=0;
    int stepnum=0;
    while(true)
    {
        stepnum++;
        while(!isValidNumberComputer(s))
            s=enumerate(++stringnum);
        x=simulate(s,stepnum);
        if(x==2)
            x=1;
        else
            x=2;
        if(x==0)
            System.out.print('0'); //etc.
    }
}
```

Note that the methods called above can be expanded inline, so that the whole thing is just a single program without method calls. But now we've produced a program that computes a number different from every computable number. It's computable, but it's...not computable?