COMS 3003A
Tutorial 2: Modelling data and computational problems

Dmitry Shkatov

22 February, 2024

Reading:
Boaz Barak. Introduction to Theoretical Computer Science. Sections 2.1, 2.2, 2.3, and 2.6.

1. Which one of the following types of object can be represented by a binary string?
   (1) integers;
   (2) rational numbers;
   (3) matrices;
   (4) finite directed graphs;
   (5) finite undirected graphs;
   (6) files containing Python source code;
   (7) all of the above.

   Solution: All of the above.

2. Write a Python program that converts English words given as input into binary strings. You will need to devise your own scheme for converting letters of the Latin alphabet into binary strings.

3. We're representing rational numbers in the alphabet $\Sigma = \{0, 1, \#\}$ using the following scheme:
   (1) to represent an integer, we use the first bit to indicate a sign—0 means 'positive', 1 means 'negative'—and the rest of the bits to represent a natural number;
   (2) if n and m are integers, we represent the rational number $\frac{n}{m}$ as $\langle n\rangle\#\langle m\rangle$, where $\langle x\rangle$ is the representation of the integer x according to (1);
   (3) if a string does not represent any rational number according to (1) and (2), we take it to be a representation of 0.

   (a) Which rational numbers are represented by the following strings of Σ?
      (i) 0101#01000;
      (ii) 0011#11010;

(iii) 0111#11000;
(iv) 1101#1011.

Solution:

(i) 5/8;
(ii) 0;
(iii) −7/8;
(iv) 0.

(b) For each string s from Question (a), determine its length $|s|$.

Solution:

(i) 10;
(ii) 10;
(iii) 10;
(iv) 9.

(c) Give at least 5 different representations of 0.

Solution: Here are a few possibilities: 0#0, 10#01, 00#011, 110#0011, 11011#1001.

(a) Write a Python program than converts a representation of a rational number given as input in the form $\pm n / \pm m$, where n and m are decimal representations of natural numbers, into the representation defined in (1) through (3).

4. (a) If we represent natural numbers in binary, what is the length of the representation of the number n?

Solution: The exact length is $\lfloor \log_2 n \rfloor + 1$. For most calculations in this course, it is going to be enough to know that it is $\approx \log_2 n$.

(b) If we represent natural numbers in k-ary, with $k > 1$, what is the length of the representation of the number n?

Solution: The exact length is $\lfloor \log_k n \rfloor + 1$. For most calculations in this course, it is going to be enough to know that it is $\approx \log_k n$.

(c) What is the ratio between representations of natural numbers in binary and in k-ary if $k > 1$?

Solution: We know (this is the "change of base"formula) that

$$\log_k n = \frac{\log_2 n}{\log_2 k}.$$

Hence,

$$\frac{\log_2 n}{\log_k n} = \log_2 k.$$

Let $\langle n \rangle_2$ and $\langle n \rangle_k$ be the encodings of the natural number n in, respectively, binary and k-ary. Since, as we have seen, $|\langle n \rangle_2| \approx \log_2 n$ and $|\langle n \rangle_k| \approx \log_k n$, it follows that

$$\frac{|\langle n \rangle_2|}{|\langle n \rangle_k|} \approx \log_2 k.$$

What is important to us is that this ratio, $\log_2 k$, is a constant, i.e., it does not depend on n.

(d) What is the ratio between representations of natural numbers in binary and in unary?

Solution: As we have seen, $|\langle n \rangle_2| \approx \log_2 n$. Clearly, $|\langle n \rangle_1| \approx n$ (why?). So, the ratio is $\approx 2^n$, i.e., an exponent in n.

(e) What is the ratio between representations of natural numbers in k-ary, with $k > 1$, and in unary?

Solution: By an argument similar to (d), we see that this is $\approx k^n$.

5. Consider the following algorithm, written here in Python-style pseudo-code (an input is a string $\langle n \rangle$ representing a natural number n):

```
if n < 2:
    return False

for factor in range(2,n):
    if n % factor == 0:
        return False

return True
```

(a) Determine the running time of this algorithm under the assumption that input is represented in unary.

Solution: $\mathcal{O}(n)$.

(b) Determine the running time of this algorithm under the assumption that input is represented in binary.

Solution: $\mathcal{O}(2^n)$.

(c) Determine the running time of this algorithm under the assumption that input is represented in k-ary, with $k > 1$.

Solution: $\mathcal{O}(k^n)$.

6. Use questions 4. and 5. to work out why using unary representation of numbers might be undesirable in theoretical computer science.

Solution: There are two observations to make here. First, representation of numbers in unary is not efficient: as we have seen in Question 4, the representation in unary is exponentially longer than in k-ary, with $k > 1$. Notice that, on the other hand, that if $k, k' > 1$, then conversion from k-ary to k'-ary takes constant time; hence, from the point of view of analysing the performance of algorithms—done using the big-Oh notation, which ignores the constant factors—it does not matter what size alphabet we are working with as long as it's not unary. Second, as a consequence, if we encode numbers in unary, then the running times of all our algorithms become exponentiall faster compared to the calculations that assume that input is represented in k-ary, with $k > 1$. This misleads us when thinking about efficiency of algorithms: they seem to be efficient only because inputs are encoded so that they become artificially long.

7. Consider the following algorithm, written here in Python (input is given as a binary string s representing a natural number):

```
def isEven( s ):

  if s[-1] == '0':
      return True


  else
      return False
```

Does this algorithm solve the problem of testing a natural number for evenness?

Solution: No. An algorithm solves a problem if it gives a correct answer on every input. Suppose our input is the string 0011. This string represents the number 0. Since 0 is even, we expect the answer "True". This algrithm, however, returns "False".

8. Consider the following algorithm, written here in Python (input is given as a binary string s representing a natural number):

```
def isProductOfPrimes( s ):

  if s[0] == '0':
      return False

  elif s == '1':
      return False

  else
      return True
```

Does this algorithm solve the problem of finding out if a natural number can be writting as a product of prime numbers?

Solution: Yes. Every number greater than 1 is a product of primes, so this algorithm always returns a correct answer.