Mathematics and Statistics Research Competition Topic S-01 Integer Construction

Jiamu Li & Frank Tang & Edward Wang

Scotch College

July 31, 2024

Contents

1 Problem 1 1

2 Problem 2 3

3 Problem 3 4

A Code for computation of coefficients 7

References 8

Let $\mathcal{N}_{10,2}$ be the set of positive integers whose digits in base-10 comprise only 0s and 1s. Examples of elements in $\mathcal{N}_{10,2}$ are: 1001, 110, and 11. Examples of elements not in $\mathcal{N}_{10,2}$ are: 4201, 690, and 12.

Consider a positive integer N. It can be constructed as the sum of elements in $\mathcal{N}_{10,2}$. For example, one construction of 1337 with 8 summands which are elements in $\mathcal{N}_{10,2}$ is as follows

$$1337 = 1000 + 111 + 111 + 111 + 1 + 1 + 1 + 1.$$

1 Problem 1

Problem. What are all of the constructions of 1337 using elements of $\mathcal{N}_{10,2}$?

We interpret the question as asking for how many unique ways there are to obtain 1337 as a sum of elements from $\mathcal{N}_{10,2}$. To do this, we look at the general case which seeks to find the number of unique ways to obtain a positive integer n as a sum of elements from $\mathcal{N}_{10,2}$.

Definition 1.1. For sake of convenience, we define a function $C(n)$ that counts the number of unique ways of constructing n as a sum of elements in $\mathcal{N}_{10,2}$. That is, $C(n)$ is the number of unique constructions such that

$$n = \sum_{a_i \in \mathcal{N}_{10,2}} a_i.$$

We start with an elementary example. Consider $n = 15$, and suppose we wish to find $C(15)$. Clearly, the only elements of $\mathcal{N}_{10,2}$ that are relevant here are 1, 10 and 11. With so few elements, we can easily calculate $C(15)$ manually. We find that there are three unique constructions of 15, which are

$$\begin{aligned} 15 &= 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 \\ 15 &= 1 + 1 + 1 + 1 + 1 + 10 \\ 15 &= 1 + 1 + 1 + 1 + 11. \end{aligned}$$

Hence $C(15) = 3$. However, this method quickly becomes impractical for large n, where the number of relevant elements of $\mathcal{N}_{10,2}$ increases with the length of the number. For example, there are 15 relevant elements of $\mathcal{N}_{10,2}$ for $n = 1337$, which are 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 and 1111. As such, we need a better way to count $C(n)$.

Let us consider a simpler case. Consider a fictional country Numberland whose currency system consists of only \$1 and \$10 coins. Suppose Dave wants to count the number of ways to make \$20 in this currency system. This is equivalent to asking what $C(20)$ is.

Take a power series $f(x) = \sum_{i=0}^{\infty} a_i x^i$, whose coefficients a_i denote the number of ways to make \$$i$ using only \$1 coins. Clearly,

$$f(x) = 1 + x + x^2 + x^3 + \dots,$$

because there is one way to make every integer using only 1s. In addition, take the power series $g(x) = \sum_{i=0}^{\infty} b_i x^i$ whose coefficients b_i represent the number of ways to make \$$i$ using only \$10 coins. Since one can only make multiples of \$10 using \$10 coins, we have

$$g(x) = \mathbf{1} + 0x + 0x^2 + \cdots + 0x^9 + \boldsymbol{x}^{10} + 0x^{11} + \cdots + 0x^{19} + \boldsymbol{x}^{20} + \cdots = 1 + x^{10} + x^{20} + \dots,$$

since there is exactly one way to make every multiple of 10. Now let us take the product $f(x)g(x)$ and find the coefficient of x^{20}. We have

$$f(x)g(x) = (1 + x + x^2 + x^3 + \dots)(1 + x^{10} + x^{20} + x^{30} + \dots),$$

and note that we can make x^{20} from $1 \times x^{20}$, $x^{10} \times x^{10}$ and $x^{20} \times 1$, meaning that the coefficient of x^{20} is 3. The key observation here is that each of the ways to make x^{20} corresponds to a way to make \$20 using \$1 or \$10 coins, and we can indeed verify that there are three ways to make \$20 using \$1 or \$10 coins, which are $20 = 1 + 1 + \cdots + 1 = 10 + 1 + \cdots + 1 = 10 + 10$.

Functions like f and g are called *generating functions* that are defined as the series $p(x) = \sum_{i=1}^{\infty} a_i x^i$. In general, let the coefficients a_i describe the number of ways to construct \$$i$ using coins of value \$$a$. Now let another generating function be defined as $q(x) = \sum_{i=1}^{\infty} b_i x^i$, where the coefficients b_i represent the number of ways to construct \$$i$ using coins of value \$$b$. If we multiply these functions together, we get another generating function h where

$$h(x) = p(x)q(x) = (a_0 + a_1 x + a_2 x^2 + \dots)(b_0 + b_1 x + b_2 x^2 + \dots) = \sum_{i=1}^{\infty} c_i x^i.$$

Now, each coefficient c_i represents the number of ways to make i using coins of value \$$a$ or \$$b$. We now return to the problem of making amounts using coins in Numberland, with our tools of generating functions.

Theorem 1.1. *The generating function for coins of value a is*

$$g_a(x) = \frac{1}{1 - x^a}.$$

Proof. Given coins of value a, we can only make amounts of value ka, where k is an integer. Moreover, we can make these amounts in exactly one way, with k coins. Thus the generating function, which can be rewritten as a geometric series, must be

$$g_a(x) = x^{0a} + x^a + x^{2a} + \cdots = \frac{1}{1 - x^a}.$$ □

Theorem 1.2. *For positive integers n, $C(n)$ is equal to the coefficient of x^n in*

$$\prod_{a_i \in \mathcal{N}_{10,2}} \frac{1}{1 - x^{a_i}}.$$

Proof. Recall that we can multiply generating functions together, and the resulting coefficients of each term x^n represents the number of ways to form n. Thus we multiply the generating functions for all the numbers in the set $\mathcal{N}_{10,2}$ to get the function

$$\prod_{a_i \in \mathcal{N}_{10,2}} \frac{1}{1 - x^{a_i}},$$

where the coefficient of x^n denotes the number of ways to form n, which is equivalent to $C(n)$. □

Unfortunately, it is unknown whether there exists a closed form expression to determine $C(n)$, which in the literature is referred to as a restricted partition function. Hardy and Ramanujan proved in 1918 [1] that the partition function asymptotically approaches

$$C(n) \sim \frac{a}{n} e^{b\sqrt{n}}$$

for constants a and b.

We can, however, use computer software like Mathematica to calculate the coefficients in the generating function, and thus we can calculate $C(n)$ to get $C(1337) = 1448684$, which is the answer to Problem 1. We

may graph the function $C(n)$, and although it looks exponential (Figure 1), the function is sub-exponential (as expected due to Hardy's asymptotic result). We can graph $C(n)$ using a logarithmic scale to showcase this, as seen in Figure 2.

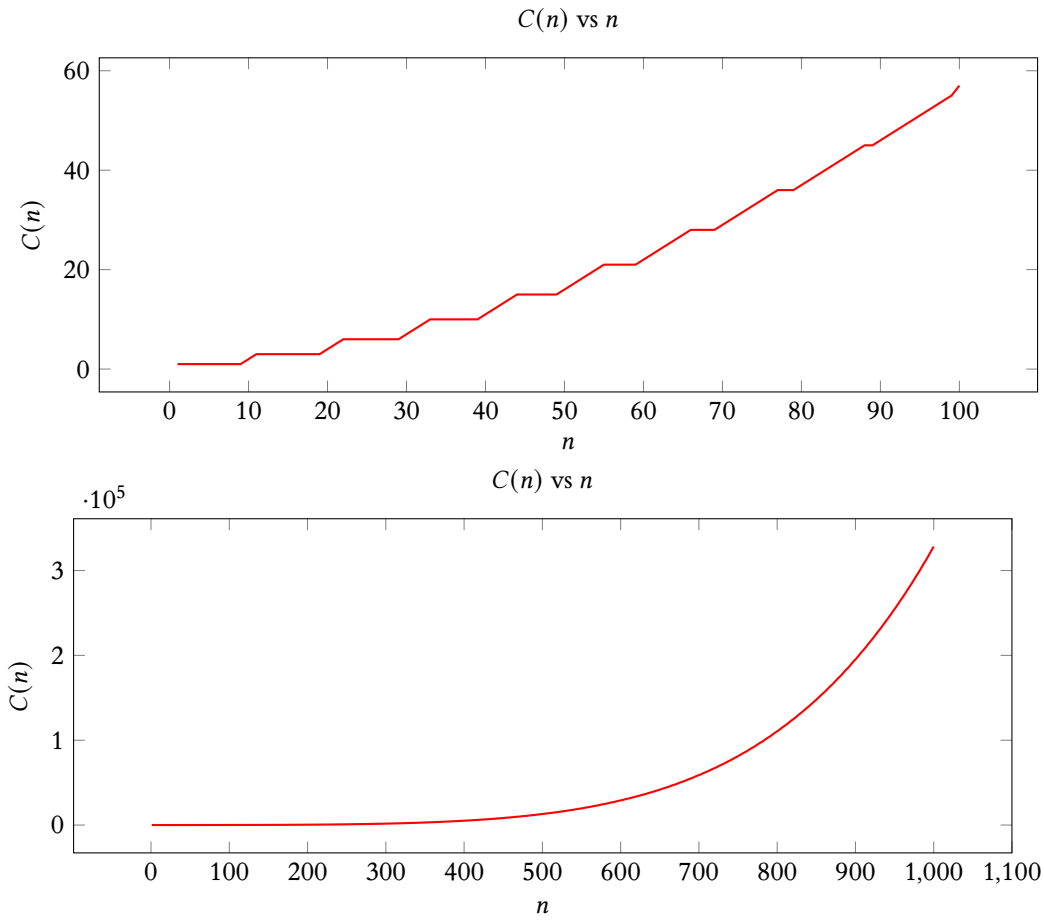

Figure 1: The number of ways $C(n)$ to make n increases rapidly as n increases.

2 Problem 2

Definition 2.1. Let $\mathcal{L}_{10,2}(n)$ be the minimum number of summands needed to construct n using elements in $\mathcal{N}_{10,2}$.

Problem. Determine $\mathcal{L}_{10,2}(n)$ for the following values of n:

- 1337
- 12345
- 190274876

For sake of convenience, we define $d_{n,i}$ to be the ith digit of n.

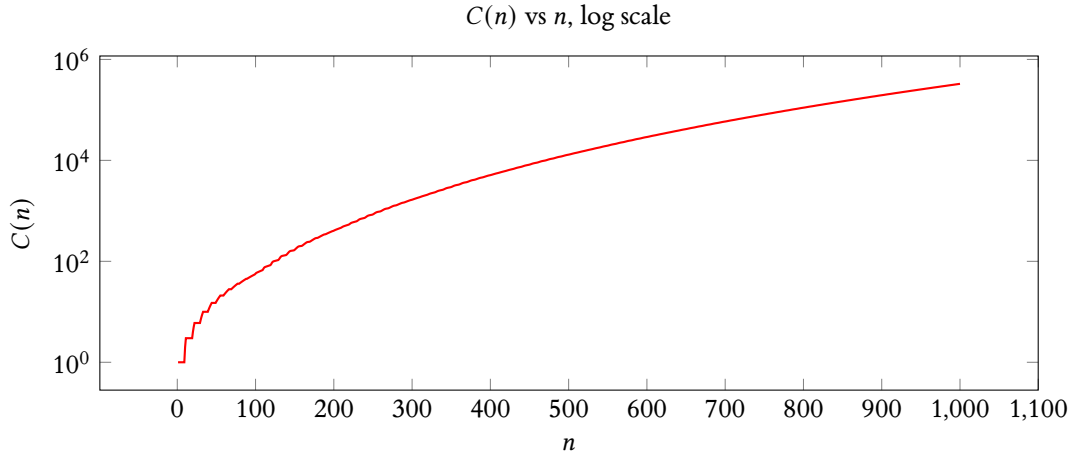

Figure 2: The number of ways, $C(n)$, grows sub-exponentially.

Definition 2.2. For an integer n, the ith digit of n, with the leftmost digit being the first, is denoted $d_{n,i}$.

For example, $d_{1337,4} = 7$.

Theorem 2.1. *For a positive integer N, $\mathcal{L}_{10,2}(n) = \max\{d_{n,i}\}$.*

Proof. Firstly, we recognise that $\mathcal{L}_{10,2}(n)$ is equivalent to the minimum number of subtractions to make 0 from n, using elements of $\mathcal{N}_{10,2}$. Intuitively, it is obvious that an optimal strategy would be to subtract 1 from every non-zero digit in n, which decreases the largest digit by 1 every time. This means that the total number of subtractions is simply equal to the largest digit of n. We prove that this is the optimal strategy as part of a more general theorem in Theorem 3.1. □

3 Problem 3

In general, let $\mathcal{N}_{10,k}$ be the set of positive integers whose digits in base-10 are strictly less than k, where k is a given positive integer. Let $\mathcal{L}_{10,k}(n)$ be the minimum number of summands needed to construct n using elements in $\mathcal{N}_{10,k}$.

Problem. Given the definitions,

- Explore the relationship between k and $\mathcal{L}_{10,k}(n)$.
- Define $L_{j,k}(n)$ and investigate the nature of $L_{j,k}(n)$ for positive integers j, k.

We expand on the ideas from Theorem 2.1 and prove a generalised formula for $\mathcal{L}_{10,k}(n)$.

Theorem 3.1. *For a positive integers n, k, we have*

$$\mathcal{L}_{10,k}(n) = \left\lceil \frac{\max\{d_{n,i}\}}{k-1} \right\rceil.$$

Proof. Recall that in $\mathcal{N}_{10,2}$, the largest available digit is $k-1$. Again, the optimal strategy is to subtract $k-1$ from the largest digit in n at every subtraction, until the largest digit is less than $k-1$, at which point it is

obvious that it takes one more subtraction to reach zero. This gives us a total number of

$$\left\lceil \frac{\max\{d_{n,i}\}}{k-1} \right\rceil$$

subtractions. We must now prove that it is impossible to do better than this.

Firstly, notice that if we subtract using numbers that are less than the largest digit every time, then the largest digit decreases by less than $k-1$ each subtraction, meaning that we cannot beat the previous minimum number of subtractions. However, if we use subtractions that use digits larger than the largest digit in n, such as in $123-4$, we must be more careful.

Consider the representation of a 4-digit number n as concatenated digits $a|b|c|d$. Note that we can 'group' digits to make an equivalent representation $(10a+b)|c|d$, in other words, we will allow 'digits' that have two numbers.

Now consider the subtraction $123-4$. The usual way we subtract numbers is that we 'carry' a 1 to the 3 to do the subtraction $1|1|13-4=119$. Thus we see that the representation $a|b|c|d$ is equivalent to the representation $a|b|(c-1)|(d+10)$. This only occurs when we subtract using a digit that is greater than the largest digit in n. Notice that if we do not use any 'alternate' representations, with the digits in their usual place, then every digit will be between 0 and 9, inclusive. This means that the *maximum* number of subtractions must be

$$\left\lceil \frac{9}{k-1} \right\rceil.$$

However, note that if we subtract using digits larger than the largest digit in n, then we must have some pattern of $(c-1)|(d+10)$ in the representation of n. This means that the largest digit of n now ranges from 10 to 19, inclusive. Importantly, this means that the *minimum* number of subtractions is now

$$\left\lceil \frac{10}{k-1} \right\rceil \geq \left\lceil \frac{9}{k-1} \right\rceil.$$

Thus it always takes more (or the same) number of subtractions if we subtract using digits that are larger than the largest digit in n. Hence, the optimal strategy of subtracting using digits that are equal to the largest digit, is indeed the best possible strategy. □

Using this, we can easily determine the answers to Problem 2, which are in Table 1.

n	$\mathcal{L}_{10,2}(n)$
1337	7
12345	5
190274876	9

Table 1: The minimum number of summands is equal to the largest digit.

We now turn our attention to the second part of the problem.

Definition 3.1. For integers j, k, we define $\mathcal{N}_{j,k}$ to be the set of numbers whose digits in base j are less than k.

Definition 3.2. For integers j, k, we define $\mathcal{L}_{j,k}(n)$ to be the minimum number of summands to construct n using elements from $\mathcal{N}_{j,k}$.

This is not so different from the previously defined $\mathcal{N}_{10,2}$, apart from the fact that we now allow bases greater than 10, such as allowing digits like F = 15 as in hexadecimal.

In fact, Theorem 3.1 remains unchanged for $\mathcal{L}_{j,k}$, that is, we have that

$$\mathcal{L}_{j,k}(n) = \left\lceil \frac{\max\{d_{n,i}\}}{k-1} \right\rceil$$

for all positive integers n, j, k. The proof of this fact will follow identical reasoning to that of the previous theorem.

Theorem 3.2. *For positive integers n, j, k, we have*

$$\mathcal{L}_{j,k}(n) = \left\lceil \frac{\max\{d_{n,i}\}}{k-1} \right\rceil.$$

Proof. We adapt our previous argument to work for digits of base j. Again, we consider the analogous problem of finding the minimum number of subtractions from n to make 0.

Like previously, the optimal strategy of subtracting $k-1$ from the largest digit until the largest digit is less than $k-1$ works, and the maximum number of subtractions using this method is

$$\left\lceil \frac{j-1}{k-1} \right\rceil.$$

The case where we subtract less than $k-1$ from the largest digit is trivially less optimal, and takes more subtractions, so we move on to the case where we subtract using digits larger than the largest digit.

Consider a 4-digit number in base j, which we will denote $a|b|c|d$, where a, b, c, d are digits in base j, and suppose d is the largest digit. Since we are in base j, all the digits are less than j, where j is an arbitrary integer. Now suppose we want to subtract e where $e > d$. This is equivalent to the subtraction $a|b|c-1|d+j-e$. Critically, notice that $d + j - e > d$, since $j > e$. Thus the largest digit has now increased, meaning that we must make more subtractions. That is, a lower bound for the number of subtractions is now

$$\left\lceil \frac{j}{k-1} \right\rceil.$$

Again, this shows that the optimal strategy indeed provides the minimal answer. □

A Code for computation of coefficients

The following Mathematica code provides the function `c[n]` that returns the value of $c(n)$, for $n < 9999$.

```
f[x_] := 1/((1 - x^(1)) (1 - x^(10)) (1 - x^(11)) (1 - x^(100)) (1 - x^(101)) (1 - x^(110)) (1 - x^(111)) (1 - x^(1000)) (1 - x^(1001)) (1 - x^(1010)) (1 - x^(1011)) (1 - x^(1100)) (1 - x^(1101)) (1 - x^(1110)) (1 - x^(1111)))
c[n_] := SeriesCoefficient[f[x], {x, 0, n}]
```

It does this by calculating the coefficient of x^n in the expansion of

$$\prod_{a_i \in \mathcal{N}} \frac{1}{1 - x^{a_i}},$$

where $\mathcal{N}$ consists of the first 15 terms in $\mathcal{N}_{10,2}$, which are 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, due to numerical limitations. This means that $C(n)$ is only accurate up to 9999.

The following C++ code provides the functions `countTotal(n)` which returns $C(n)$ and `countMin(n)`, which returns $\mathcal{L}_{10,2}(n)$, which are accurate for $n \leq 33896$, at which point integer overflow prevents further calculation (although arbitrary integer precision can be used to circumvent this).

```
#include <bits/stdc++.h>

std::vector<long long> getNumbers(int n) {
  std::vector<long long> out;
  for (long long i = 1; i < (1 << std::to_string(n).length()); i++) {
    out.push_back(stoi(std::bitset<sizeof(n) * 8>(i).to_string()));
  }
  return out;
}

long long dpTotal(std::vector<long long> &coins, long long n, long long sum) {
  std::vector<std::vector<long long>> dp(n + 1, std::vector<long long>(sum + 1, 0));
  dp[0][0] = 1;
  for (long long i = 1; i <= n; i++) {
    for (long long j = 0; j <= sum; j++) {
      dp[i][j] += dp[i - 1][j];
      if (j - coins[i - 1] >= 0) {
        dp[i][j] += dp[i][j - coins[i - 1]];
      }
    }
  }
  return dp[n][sum];
}

long long countTotal(long long n) {
  std::vector<long long> coins = getNumbers(n);
  return dpTotal(coins, coins.size(), n);
}

long long countMin(long long n) {
  long long dp[n + 1];
```

```
  dp[0] = 0;
  for (long long i = 1; i <= n; i++) {
    dp[i] = INT_MAX;
    for (auto c : getNumbers(n)) {
      if (i - c >= 0) {
        dp[i] = std::min(dp[i], dp[i - c] + 1);
      }
    }
  }
  return dp[n];
}

int main() {

  printf("%lld\n", countTotal(1337));
  printf("%lld\n", countMin(12345));

  return 0;
}
```

This outputs

```
1448684
5
```

which matches the expected values of $C(1337)$ and $\mathcal{L}_{10,2}(12345)$.

References

[1] G. H. Hardy and S. Ramanujan. “Asymptotic Formulae in Combinatory Analysis”. In: *Proceedings of the London Mathematical Society* 1 (1918), pp. 75–115.