Texas A&M University Kingsville
Department of EECS
CSEN 5303 Foundations of Computer Science
Project 2 Balanced Ternary

Mengxiang Jiang
Professor Habib Ammari

October 8, 2022

Contents

Chapter 1

Introduction

A common problem given to math students during their introduction to number theory is as follows:

> Suppose you are given a balance scale and were tasked to come up with a collection of weights such that every integral weight from 1 pound to 40 pounds are measurable with the scale. What is the collection with the least number of weights needed?

The trivial answer is simply to use all 1 pound weights, but one would need 40 of them, so this is obviously not ideal. After thinking for a little bit, a solution that commonly comes up is to use powers of two weights (1, 2, 4, 8, 16, 32). This solution is pretty good and would be optimal if we can only place weights on one side of the scale. However, when weights can be placed on both side of the scale, there is a better solution: to use powers of three weights (1, 3, 9, 27). On first glance, it is not entirely obvious how just these 4 weights can measure every one of the 40 different weights, and to show that it does, we are going to use the *balanced ternary* number system.

Called "perhaps the prettiest number system of all" in volume 2 of *The Art of Computer Programming* by Donald Knuth, balanced ternary is just ternary (or base 3) but instead of 0, 1, and 2, the digits used are -1, 0, and 1 in each position.[1] Since $-1 \equiv 2 \pmod 3$, the two number systems are equivalent and can be converted between each other by adding ...111 and then subtracting ...111.[1] Because there is a bijection between every decimal number and every ternary number, and a bijection between every ternary number and every balanced ternary number, therefore there is a bijection between every decimal number and every balanced ternary number, so every integer has a unique balanced ternary representation.

Now looking back at the weights problem, we see that the weights and the balance scale can be considered a balanced ternary system and with the given weights, we can represent every balanced ternary number from 0 to 1111_3 which is $1 + 3 + 9 + 27 = 40$ pounds.

In the rest of this report, I will detail two algorithms to convert from decimal to balanced ternary, as well as Python code for it.

Chapter 2

Design

In the project statement, the recommended approach is to use intervals to decompose a decimal number into its balanced ternary representation. As such, I have implemented such an algorithm as shown below:

```
procedure BalancedTernaryInterval(i: integer; var bt: queueInt);
    var
        n: integer;
    begin
        bt := new queueInt();
        while (i <> 0) do
            for n := 0 to sizeof(INTERVALS) - 1 do
                if INTERVALS[n].start <= abs(i) and abs(i) <= INTERVALS[n].end then
                    if i > 0 then
                        bt.enqueue(pow(3, n));
                        i := i - pow(3, n);
                    else
                        bt.enqueue(-pow(3, n));
                        i := i + pow(3, n);
```

A second approach is to convert the decimal number to ternary and then into balanced ternary. However, after writing this approach I found that I can combine the two steps to convert from decimal to balanced ternary directly. The algorithm of this is shown below:

```
procedure BalancedTernary(i: integer; var bt: stackInt);
    var
        n, j, rem: integer;
    begin
        bt := new stackInt();
        j := abs(i);
        n := 0;
        while (j > 0) do
            rem := j % 3;
            j := j // 3;
            if (rem = 2) then
                if (i > 0) then
                    bt.push(-pow(3, n));
                else
                    bt.push(pow(3, n));
                j := j + 1;
            if (rem = 1) then
                if (i > 0) then
                    bt.push(pow(3, n));
                else
                    bt.push(-pow(3, n));
            n := n + 1;
```

Both these algorithms have O(log(n)) performance since they both reduce the given input number by a factor 3 every iteration. The second algorithm is slightly more complicated, but can compute values outside the range of the intervals in the first.

Chapter 3

Code

```
"""
class for converting an integer from decimal to balanced ternary
"""
# used to read the lookup csv file
import csv

class BalancedTernary:
    INTERVALS = ((1, 1), (2, 4), (5, 13), (14, 40), (41, 121))
    lookup_table = {}
    with open('tests/lookup.csv') as csvfile:
        csvreader = csv.reader(csvfile)
        for row in csvreader:
            lookup_table[int(row[0])] = list(map(int, row[1:]))

    def __init__(self, dlist, alg_flag='interval'):
        self.dlist = dlist
        self.btlist = []
        self.convert(alg_flag)

    # converts the decimal integers in dlist to balanced ternary in btlist
    def convert(self, alg_flag='interval'):
        for i in self.dlist:
            # itv_flag tells method whether to use interval method
            match alg_flag:
                case 'interval':
                    self.btlist.append(BalancedTernary.
    decimal_to_balanced_ternary_interval(i))
                case 'ternary':
                    self.btlist.append(BalancedTernary.
    decimal_to_balanced_ternary(i))
                case 'lookup':
                    self.btlist.append(BalancedTernary.lookup_table[i])

    # converts an integer to a balanced ternary list using pure math
    @staticmethod
    def decimal_to_balanced_ternary(i):
        bt = []
        j = abs(i)
```

```
        n = 0
        while (j > 0):
            j, rem = divmod(j, 3)
            if rem == 2:
                if i > 0:
                    bt.append(-3**n)
                else:
                    bt.append(3**n)
                j += 1
            elif rem == 1:
                if i > 0:
                    bt.append(3**n)
                else:
                    bt.append(-3**n)
            n += 1
        # since we want the largest power of three first, we reverse the order
        bt.reverse()
        return bt

    # converts an integer to a balanced ternary list using intervals
    @staticmethod
    def decimal_to_balanced_ternary_interval(i):
        bt = []
        # interval method only supports integers in the range -121 to 121
        if (abs(i) > 121):
            raise ValueError('number out of range (must be between -121 and 121)')
        while (i != 0):
            for n, (a, b) in enumerate(BalancedTernary.INTERVALS):
                if a <= abs(i) <= b:
                    if i > 0:
                        bt.append(3**n)
                        i -= 3**n
                    else:
                        bt.append(-3**n)
                        i -= -3**n
        return bt

def main():
    raw_input = input("Please enter a list of integers between -121 and 121, each separated by a space(i.e. 36 -74 13):")
    dlist = list(map(int, raw_input.split()))
    bt = BalancedTernary(dlist, alg_flag='ternary')
    for n in range(len(dlist)):
        result = ''
        for m in range(len(bt.btlist[n])):
            if m == 0:
                result += str(bt.btlist[n][m])
            else:
                if bt.btlist[n][m] < 0:
                    result += ' - ' + str(-bt.btlist[n][m])
                else:
                    result += ' + ' + str(bt.btlist[n][m])
```

```
88          print(f'{dlist[n]} = {result}')
89
90  if __name__ == '__main__':
91      main()
```

Listing 3.1: balanced_ternary.py

Chapter 4

Tests

```
import unittest
from balanced_ternary import BalancedTernary

class TestBalancedTernary(unittest.TestCase):
    # test interval algorithm works for all numbers in range
    def test_interval(self):
        for i in range(-121, 122):
            interval_bt = BalancedTernary.
   decimal_to_balanced_ternary_interval(i)
            lookup_bt = BalancedTernary.lookup_table[i]
            self.assertEqual(interval_bt, lookup_bt)

    # test interval algorithm fails with ValueError when given number out
   of range
    def test_interval_out_of_range(self):
        with self.assertRaises(ValueError):
            BalancedTernary.decimal_to_balanced_ternary_interval(122)

    # test lookup algorithm fails with KeyError when given number out of
   range
    def test_lookup_out_of_range(self):
        with self.assertRaises(KeyError):
            BalancedTernary.lookup_table[122]

    # test ternary algorithm works for all numbers in range
    def test_ternary(self):
        for i in range(-121, 122):
            ternary_bt = BalancedTernary.decimal_to_balanced_ternary(i)
            lookup_bt = BalancedTernary.lookup_table[i]
            self.assertEqual(ternary_bt, lookup_bt)

    # test ternary algorithm still works for numbers outside interval
   range
    def test_ternary_out_of_range(self):
        ternary_bt = BalancedTernary.decimal_to_balanced_ternary(122)
        ground_truth_bt = [243, -81, -27, -9, -3, -1]
        self.assertEqual(ternary_bt, ground_truth_bt)

```

```
if __name__ == '__main__':
    unittest.main()
```

Listing 4.1: balanced_ternary_tests.py

```
-121,-81,-27,-9,-3,-1
-120,-81,-27,-9,-3
-119,-81,-27,-9,-3,1
-118,-81,-27,-9,-1
-117,-81,-27,-9
-116,-81,-27,-9,1
-115,-81,-27,-9,3,-1
-114,-81,-27,-9,3
-113,-81,-27,-9,3,1
-112,-81,-27,-3,-1
-111,-81,-27,-3
-110,-81,-27,-3,1
-109,-81,-27,-1
-108,-81,-27
-107,-81,-27,1
-106,-81,-27,3,-1
-105,-81,-27,3
-104,-81,-27,3,1
-103,-81,-27,9,-3,-1
-102,-81,-27,9,-3
-101,-81,-27,9,-3,1
-100,-81,-27,9,-1
-99,-81,-27,9
-98,-81,-27,9,1
-97,-81,-27,9,3,-1
-96,-81,-27,9,3
-95,-81,-27,9,3,1
-94,-81,-9,-3,-1
-93,-81,-9,-3
-92,-81,-9,-3,1
-91,-81,-9,-1
-90,-81,-9
-89,-81,-9,1
-88,-81,-9,3,-1
-87,-81,-9,3
-86,-81,-9,3,1
-85,-81,-3,-1
-84,-81,-3
-83,-81,-3,1
-82,-81,-1
-81,-81
-80,-81,1
-79,-81,3,-1
-78,-81,3
-77,-81,3,1
-76,-81,9,-3,-1
-75,-81,9,-3
-74,-81,9,-3,1
-73,-81,9,-1
```

```
-72,-81,9
-71,-81,9,1
-70,-81,9,3,-1
-69,-81,9,3
-68,-81,9,3,1
-67,-81,27,-9,-3,-1
-66,-81,27,-9,-3
-65,-81,27,-9,-3,1
-64,-81,27,-9,-1
-63,-81,27,-9
-62,-81,27,-9,1
-61,-81,27,-9,3,-1
-60,-81,27,-9,3
-59,-81,27,-9,3,1
-58,-81,27,-3,-1
-57,-81,27,-3
-56,-81,27,-3,1
-55,-81,27,-1
-54,-81,27
-53,-81,27,1
-52,-81,27,3,-1
-51,-81,27,3
-50,-81,27,3,1
-49,-81,27,9,-3,-1
-48,-81,27,9,-3
-47,-81,27,9,-3,1
-46,-81,27,9,-1
-45,-81,27,9
-44,-81,27,9,1
-43,-81,27,9,3,-1
-42,-81,27,9,3
-41,-81,27,9,3,1
-40,-27,-9,-3,-1
-39,-27,-9,-3
-38,-27,-9,-3,1
-37,-27,-9,-1
-36,-27,-9
-35,-27,-9,1
-34,-27,-9,3,-1
-33,-27,-9,3
-32,-27,-9,3,1
-31,-27,-3,-1
-30,-27,-3
-29,-27,-3,1
-28,-27,-1
-27,-27
-26,-27,1
-25,-27,3,-1
-24,-27,3
-23,-27,3,1
-22,-27,9,-3,-1
-21,-27,9,-3
-20,-27,9,-3,1
-19,-27,9,-1
```

```
-18,-27,9
-17,-27,9,1
-16,-27,9,3,-1
-15,-27,9,3
-14,-27,9,3,1
-13,-9,-3,-1
-12,-9,-3
-11,-9,-3,1
-10,-9,-1
-9,-9
-8,-9,1
-7,-9,3,-1
-6,-9,3
-5,-9,3,1
-4,-3,-1
-3,-3
-2,-3,1
-1,-1
0
1,1
2,3,-1
3,3
4,3,1
5,9,-3,-1
6,9,-3
7,9,-3,1
8,9,-1
9,9
10,9,1
11,9,3,-1
12,9,3
13,9,3,1
14,27,-9,-3,-1
15,27,-9,-3
16,27,-9,-3,1
17,27,-9,-1
18,27,-9
19,27,-9,1
20,27,-9,3,-1
21,27,-9,3
22,27,-9,3,1
23,27,-3,-1
24,27,-3
25,27,-3,1
26,27,-1
27,27
28,27,1
29,27,3,-1
30,27,3
31,27,3,1
32,27,9,-3,-1
33,27,9,-3
34,27,9,-3,1
35,27,9,-1
```

```
36,27,9
37,27,9,1
38,27,9,3,-1
39,27,9,3
40,27,9,3,1
41,81,-27,-9,-3,-1
42,81,-27,-9,-3
43,81,-27,-9,-3,1
44,81,-27,-9,-1
45,81,-27,-9
46,81,-27,-9,1
47,81,-27,-9,3,-1
48,81,-27,-9,3
49,81,-27,-9,3,1
50,81,-27,-3,-1
51,81,-27,-3
52,81,-27,-3,1
53,81,-27,-1
54,81,-27
55,81,-27,1
56,81,-27,3,-1
57,81,-27,3
58,81,-27,3,1
59,81,-27,9,-3,-1
60,81,-27,9,-3
61,81,-27,9,-3,1
62,81,-27,9,-1
63,81,-27,9
64,81,-27,9,1
65,81,-27,9,3,-1
66,81,-27,9,3
67,81,-27,9,3,1
68,81,-9,-3,-1
69,81,-9,-3
70,81,-9,-3,1
71,81,-9,-1
72,81,-9
73,81,-9,1
74,81,-9,3,-1
75,81,-9,3
76,81,-9,3,1
77,81,-3,-1
78,81,-3
79,81,-3,1
80,81,-1
81,81
82,81,1
83,81,3,-1
84,81,3
85,81,3,1
86,81,9,-3,-1
87,81,9,-3
88,81,9,-3,1
89,81,9,-1
```

```
90,81,9
91,81,9,1
92,81,9,3,-1
93,81,9,3
94,81,9,3,1
95,81,27,-9,-3,-1
96,81,27,-9,-3
97,81,27,-9,-3,1
98,81,27,-9,-1
99,81,27,-9
100,81,27,-9,1
101,81,27,-9,3,-1
102,81,27,-9,3
103,81,27,-9,3,1
104,81,27,-3,-1
105,81,27,-3
106,81,27,-3,1
107,81,27,-1
108,81,27
109,81,27,1
110,81,27,3,-1
111,81,27,3
112,81,27,3,1
113,81,27,9,-3,-1
114,81,27,9,-3
115,81,27,9,-3,1
116,81,27,9,-1
117,81,27,9
118,81,27,9,1
119,81,27,9,3,-1
120,81,27,9,3
121,81,27,9,3,1
```

Listing 4.2: lookup.csv

Chapter 5

Lessons Learned

When I started this project, I had a partner, but he kept failing to show up for meetings or showing up late. I told him this was not ok, and told him if he shows up late or not at all during our next planned meeting, that I would kick him off from the project. He showed up 30 minutes late again, and to show that I was serious I did what I said. After he left, I was actually able work more efficiently since I no longer had to wait for his work (which was rarely done on time if at all). This was the most important lesson I learned from this project.

Chapter 6

Bibliography

[1] Donald Ervin Knuth. *The art of computer programming, Volume II: Seminumerical Algorithms, 3rd Edition.* Addison-Wesley, 1998.