Assignment 3 Option 2

Christian Diangco

Math 381 A

Let’s consider a set of integers *A* = {1, …, *n*}.

Suppose that we want to divide *A* into subsets where for each subset, the sum or difference of unique elements *i* and *j* cannot be a square or cube.

We want to find the smallest number of subsets that have this property for a given *n*.

We can actually think of this problem as a graph coloring problem in order to solve it.

Let’s say we have a graph *G* = (*V*, *E*) where each vertex represents an element in *A*.

Therefore, we have *n* vertices in our graph, *V* = {.

If the sum or difference of elements *i* and *j* in *A* are equal to a square or cube, there will be an edge connecting their corresponding vertices and .

For the case where *n* = 10, we end up with a graph like this:



We can think of the color of each vertex as the subset that it belongs to.

As a reminder, if the sum or difference of elements *i* and *j* in *A* are equal to a square or cube, there will be an edge connecting their corresponding vertices and

Therefore, if there is an edge connecting two vertices, they must not have the same color, i.e. not be in the same subset.

We can solve for the coloring of this graph using an LP.

We will have *n* colors to color the graph, but we want to use as few as possible.

We will define variable where only if we use color *i*.

We will define variable where only if has color *k*.

We want to use as few colors as possible, so the function we are trying to minimize is

We will also require the following constraints:

for all E and

The first constraint requires that each vertex is colored with only one color.

For example, if a vertex is colored with color 1 , they cannot be colored with color 2 .

The second constraint requires that a vertex cannot be colored with a color that hasn’t been used.

For example, if we color a vertex with color 1 then we are using color 1 .

The third constraint requires that vertices connected by an edge cannot have the same color.

Looking at our graph, and are connected by an edge and thus cannot have the same color.

If , then .

The fourth constraint limits the number of colors we use in order to speed up calculations.

For example, if we have not used color 1 then we do not use color 2 .

So instead of choosing from *n* colors, we are only choosing from the minimum needed.

To generate the LPSolve input file to solve for *n =* 10, the following Python code will be used:

# This is the input file for LPSolve.

f = open("Assignment3.txt", "w")

# Checks whether a positive integer n is a square or cube.

def is\_square\_or\_cube(n):

return (n\*\*(1/2) % 1).is\_integer() or n\*\*(1/3) % 1 == 0 or n\*\*(1/3) % 1 > 0.999

# n is the number of integers in A = {1, 2, 3, ..., n}.

n = 10

# This string contains the function we are trying to minimize, this case being the number of colors (subsets of A).

obj\_func = "min: "

# This string contains the constraint requiring each vertex to be colored with exactly one color.

one\_color = ""

# This string contains the constraint requiring that a vertex cannot be colored with an unused color.

used\_color = ""

# This string contains the constraint requiring adjacent vertices have different colors.

different\_color = ""

# This string contains the constraint that stops us from using color k if k-1 is not used.

limit\_color = ""

# This string contains all of our variables.

variables = "bin "

# This loop creates the objective function and constraints.

for i in range(1, n + 1):

obj\_func += "+y\_" + str(i)

limit\_color += "y\_" + str(i) + "<= y\_" + str(i-1) + ";\n"

variables += "y\_" + str(i) + ","

for k in range(1, n + 1):

one\_color += "+x\_" + str(i) + "\_" + str(k)

used\_color += "x\_" + str(i) + "\_" + str(k) + " <= y\_" + str(k) + ";\n"

variables += "x\_" + str(i) + "\_" + str(k) + ","

temp = i - 1

# For each integer in A, checks if the sum or difference of each of the numbers before it and itself are equal

# to a square or cube. If so, they cannot be in the same subset/cannot have the same color.

while temp > 0:

if is\_square\_or\_cube(i + temp) or is\_square\_or\_cube(i - temp):

different\_color += "+x\_" + str(i) + "\_" + str(k) + "+x\_" + str(temp) + "\_" + str(k) + " <= 1;\n"

temp -= 1

one\_color += " = 1;\n"

obj\_func += ";\n"

limit\_color = limit\_color[11:]

variables = variables[:-1] + ";"

f.writelines([obj\_func, one\_color, used\_color, different\_color, limit\_color, variables])

f.close()

The generated LPSolve input file looks like this:

min: +y\_1+y\_2+y\_3+y\_4+y\_5+y\_6+y\_7+y\_8+y\_9+y\_10;

(10 lines of the following type:

requires that each vertex is colored with only one color)

+x\_1\_1+x\_1\_2+x\_1\_3+x\_1\_4+x\_1\_5+x\_1\_6+x\_1\_7+x\_1\_8+x\_1\_9+x\_1\_10 = 1;

.

.

(100 lines of the following type:

requires that a vertex cannot be colored with a color that hasn’t been used)

x\_1\_1 <= y\_1;

.

.

x\_10\_10 <= y\_10;

(250 lines of the following type:

requires that vertices connected by an edge cannot have the same color)

+x\_2\_1+x\_1\_1 <= 1;

.

.

(ensure all variables are binary)

bin y\_1, …, y\_10, x\_1\_1, …, x\_10\_10;

After running LPSolve with this input file, we receive this output:

Value of objective function: 4.00000000

Actual values of the variables:

y\_1 1

y\_2 1

y\_3 1

y\_4 1

x\_1\_1 1

x\_2\_2 1

x\_3\_3 1

x\_4\_1 1

x\_5\_2 1

x\_6\_1 1

x\_7\_4 1

x\_8\_2 1

x\_9\_3 1

x\_10\_4 1

All other variables are zero.

For n = 10, we used 4 colors.

In other words, we have 4 subsets with the following elements: {1, 4, 6}, {2, 5, 8}, {3, 9}, {7, 10}.

From our LPSolve output, the coloring of our graph looks like this:



We can solve the same problem for increasing values of *n* by just changing the value assigned to our *n* variable in the Python code and running LPSolve with the generated input file.

The table below shows the number of subsets, values in each subset, and computation times for increasing values of *n*.

| Number of integers *n* in  *A*={1, …, n} | Number of Subsets | Values of Subsets | Computation Time (s) |
| --- | --- | --- | --- |
| 2 | 2 | {1},  {2} | 0.10 |
| 3 | 3 | {1},  {2},  {3} | 0.25 |
| 4 | 3 | {1},  {2, 4},  {3} | 0.41 |
| 5 | 3 | {1, 4},  {2, 5},  {3} | 0.67 |
| 6 | 3 | {1, 4, 6},  {2, 5},  {3} | 0.88 |
| 7 | 4 | {1, 4, 6},  {2, 5},  {7},  {3} | 2.02 |
| 8 | 4 | {1, 6},  {2, 4},  {3, 5, 8},  {7} | 2.70 |
| 9 | 4 | {1, 4, 6},  {2, 5, 8},  {3, 9},  {7} | 2.97 |
| 10 | 4 | {1, 4, 6},  {2, 5, 8},  {3, 9},  {7, 10} | 4.22 |
| 15 | 4 | {1, 4, 6, 11},  {2, 8, 13, 15},  {3, 9, 14},  {5, 7, 10, 12} | 8.08 |
| 16 | 5 | {1, 4, 6, 16},  {2, 5, 8, 15},  {3, 9, 14},  {7, 10, 12},  {11, 13} | 92.69 |
| 17 | 5 | {1, 4, 6, 11},  {2, 5, 8, 15},  {3, 9},  {7, 12, 14, 17},  {10, 13, 16} | 108.65 |
| 18 | 5 | {1, 4, 6, 11, 18},  {2, 5, 8},  {3, 9, 15},  {7, 12, 14, 17},  {10, 13, 16} | 131.05 |
| 19 |  | {1, 4, 6, 11, 18},  {2, 5, 9, 16, 19},  {3, 15},  {7, 8, 12, 14, 17},  {10, 13} | 113.76 |
| 20 | 5 | {1, 4, 14, 16, 19},  {2, 5, 12, 17},  {3, 8, 15, 18, 20},  {7, 10, 13},  {6, 9, 11} | 160.72 |
| 21 | 5 | {1, 4, 14, 16, 19},  {2, 5, 12, 17},  {3, 9, 15, 18, 20},  {6, 8, 11, 13, 18},  {7, 10, 21} | 234.12 |
| 22 | 5 | {1, 4, 14, 16, 19},  {2, 5, 9, 15, 20, 22},  {3, 8, 18, 21},  {5, 7, 10, 12, 13, 17},  {6, 11, 13, 18} | 298.62 |
| 23 | 5 | {1, 6, 11, 13, 18},  {2, 4, 9, 15, 22},  {3, 8, 10, 18, 20, 23},  {5, 7, 12, 13, 17},  {14, 16, 19, 21} | 189.21 |
| 24 | 5 | {1, 4, 13, 14, 16, 19},  {2, 9, 15, 20, 22},  {3, 8, 10, 21, 23},  {5, 7, 12, 17},  {6, 11, 13, 18} | 253.15 |
| 25 | 5 | {1, 14, 16, 19, 21},  {2, 4, 9, 15, 22},  {3, 8, 10, 20, 23, 25},  {5, 7, 11, 12, 17},  {6, 13, 18, 24} | 317.30 |
| 26 | 6 (sub-optimal) |  | Terminated after 600 |

I started at *n* = 2 and increased by 1 until the computation time exceeded 10 minutes.

This occurred at *n* = 26.

So the largest *n* I solved for was *n* = 25, which had 5 subsets.

Here is the graph for *n* = 25:

