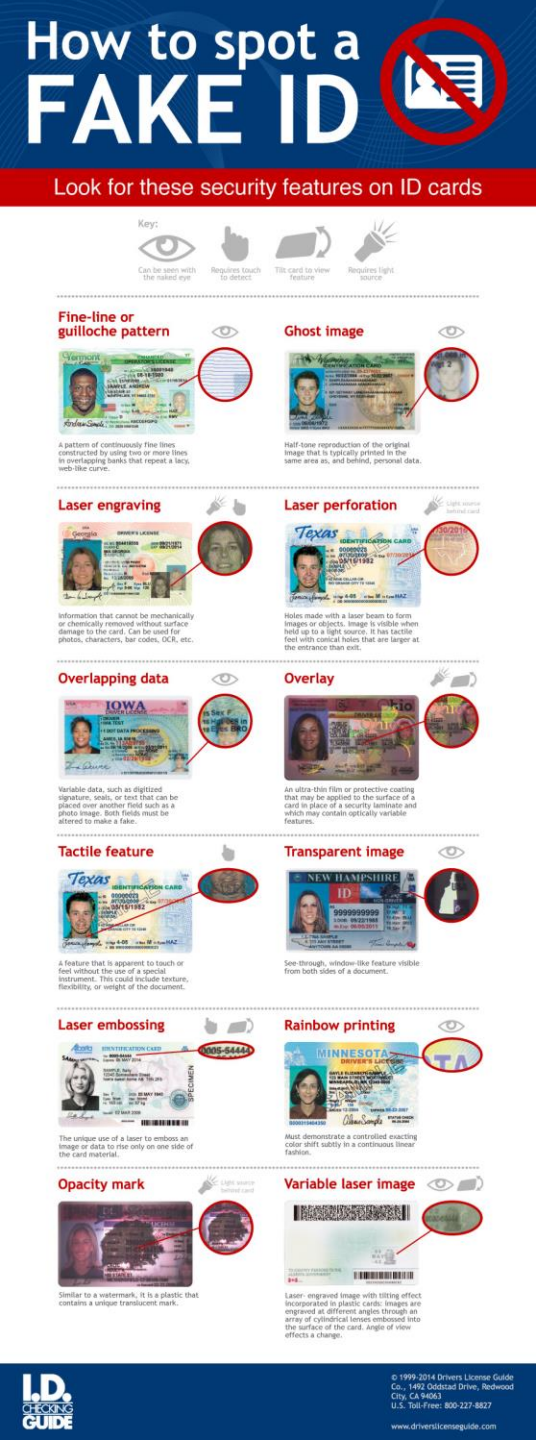
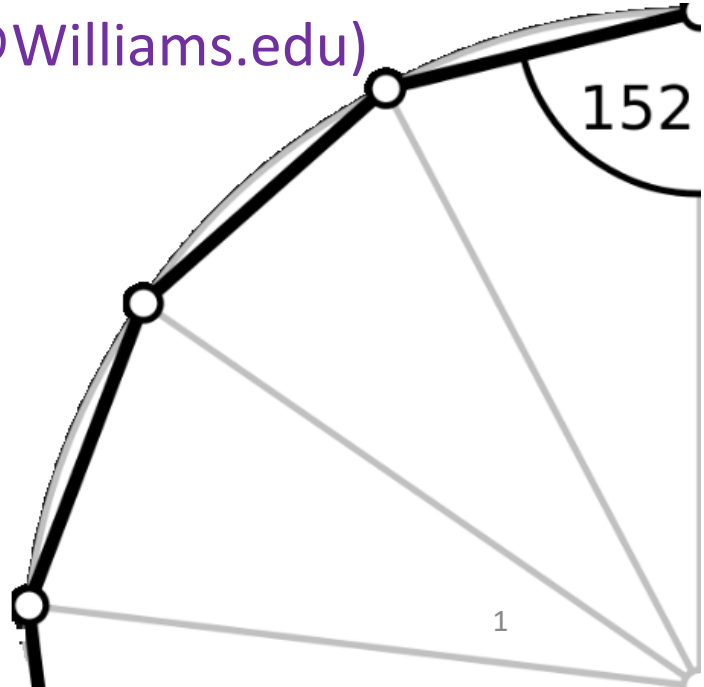


JOURNAL OF NUMBER THEORY
Teachers As Scholars
henry.bolter@gmail.com
These math outreach lectures are supported in part by the Journal of Number Theory and the Teachers as Scholars program; it is a pleasure to thank them for their support.

How to Attack Problems II: Legal 21

Steven Miller, Williams College (sjm1@Williams.edu)

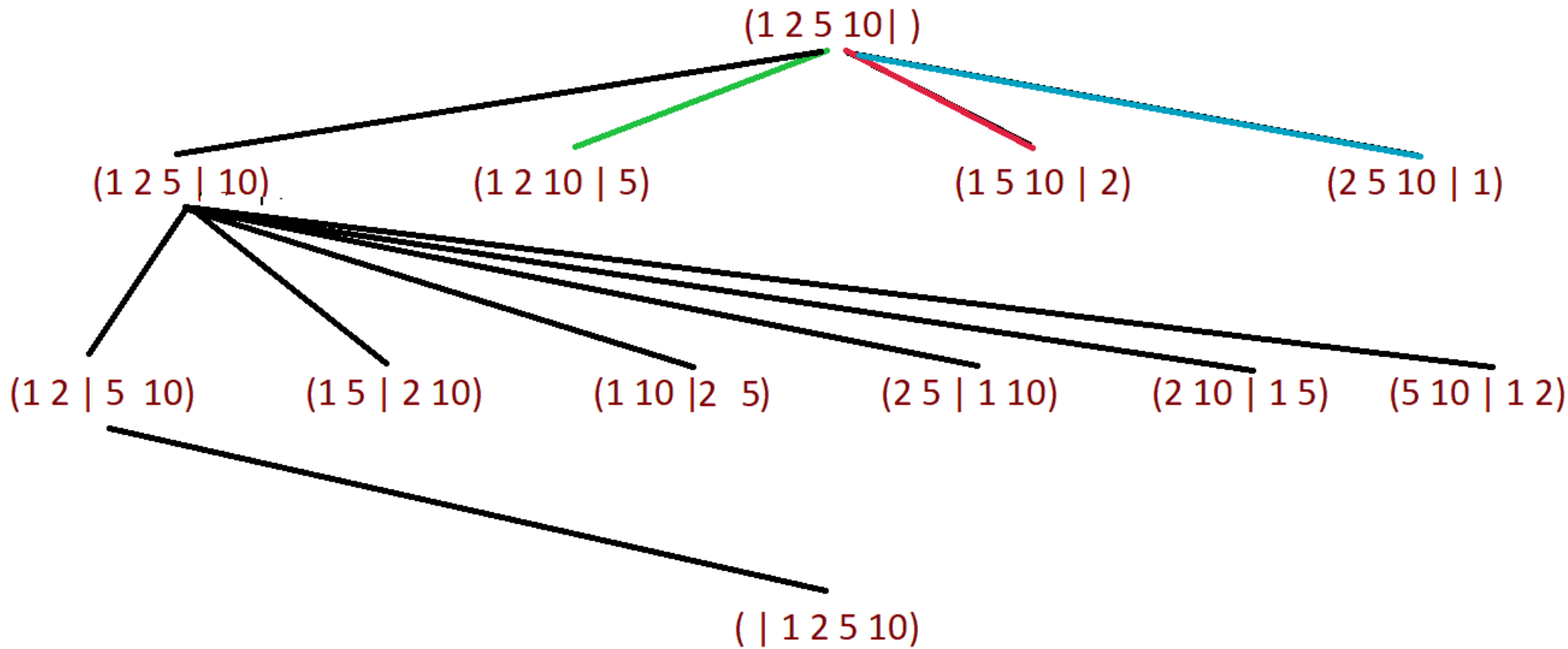


Last Time: *We'll Cross That Bridge* Problem!

- We have a rickety old bridge, it's nighttime and dark but we have one flashlight, and we are being chased by zombies (or coronavirus victims). They are 17.5 minutes from us.
- Only two people can cross the bridge at a time, and those crossing must have the flashlight. Thus two can go across with the flashlight and then one returns with the flashlight.
- We must get all people across before the attackers arrive. The four people are named 1, 2, 5 and 10; these names are how long it takes each of them to cross individually; if two go over together the time it takes is the larger of the two times. We cannot have three cross together or the bridge collapses.

Show it is possible to get the four people over in 17 minutes; if everyone is not over by 17.5 minutes the bridge is infected / destroyed and we lose….

Key Idea: Good way to list possibilities



New Problem: Legal 21

Young Saul, a budding mathematician and printer, is making himself a fake ID. He needs it to say he's 21. The problem is he's not using a computer, but rather he has some symbols he's bought from the store, and that's it. He has one 1, one 5, one 6, one 7, and an unlimited supply of + – * / (the operations addition, subtraction, multiplication and division). Using each number exactly once (but you can use any number of +, any number of -, …) how can he get 21 from 1, 5, 6, 7?

- Note: You can't do things like 15+6 = 21. You have to use the four operations as 'binary' operations: ((1+5)*6)+7.
- Note: We strongly oppose creating fake IDs…..

Legal 21: How to attack it?

Have the numbers 1, 5, 6, 7; can use any number of +, -, *, / and want to combine and get 21.

Can try some combinations to build some intuition. Take a moment and try -- did you get it, or were you at least close?

STOP! PAUSE THE VIDEO NOW TO THINK ABOUT THE QUESTION.

Legal 21: How to attack it?

Have the numbers 1, 5, 6, 7; can use any number of +, -, *, / and want to combine and get 21.

1+5+6+7 = 19

5*6-1-7 = 22

(7*6)/(5-1) = 21/2

Legal 21: How to attack it?

We need a way to go through all the possibilities and make sure we don't miss anything.

Note that +,-,* and / are all BINARY operations – they take two inputs and give one output.

Thus if we look at (6*7) / (5-1) we can view this as the following:

$b_{/}(b_{*}(6,7), b_{-}(5,1))$. This means we do a multiplication with two numbers, a subtraction with another two, and then divide the results.

Legal 21: How to attack it?

Thus if we look at (6*7) / (5-1) we can view this as the following:

$b_/(b_*(6,7), b_-(5,1))$. This means we do a multiplication with two numbers, a subtraction with another two, and then divide the results.

More generally, let b_i denote a binary operation. Then we can really view (6*7) / (5-1) as a specific realization of the structure

$$b_1(b_2(w,x), b_3(y,z))$$

where w, x, y, z are the numbers 1, 5, 6, 7 in some order, and the b_i's are different binary operations.

- How many ways can we order 1, 5, 6, 7 to substitute for w, x, y, z?
- How many ways can we choose binary operations for b_1, b_2, b_3?
- How many combinations are there to check for this structure?

STOP! PAUSE THE VIDEO NOW TO THINK ABOUT THE QUESTION.

Legal 21: How to attack it?

Thus if we look at (6*7) / (5-1) we can view this as the following:

$b_/(b_*(6,7), b_-(5,1))$. This means we do a multiplication with two numbers, a subtraction with another two, and then divide the results.

More generally, let b_i denote a binary operation. Then we can really view (6*7) / (5-1) as a specific realization of the structure

$$b_1(b_2(w,x), b_3(y,z))$$

where w, x, y, z are the numbers 1, 5, 6, 7 in some order, and the b_i's are different binary operations.

- How many ways can we order 1, 5, 6, 7 to substitute for w, x, y, z? 4! = 4*3*2*1 = 24
- How many ways can we choose binary operations for b_1, b_2, b_3? 4 * 4 * 4 = 64
- How many combinations are there to check for this structure? 24 * 64 = 1536

Legal 21: How to attack it?

More generally, let b_i denote a binary operation. Then we can really view (6*7) / (5-1) as a specific realization of the structure

$$b_1(b_2(w,x), b_3(y,z))$$

We just showed there are 1536 ways to substitute for this!

That is a lot for us to do with pen and paper, but nothing for a computer!

Are there other structures we can have other than something like (6*7)/(5-1), i.e., $b_1(b_2(w,x), b_3(y,z))$? What are they?

STOP! PAUSE THE VIDEO NOW TO THINK ABOUT THE QUESTION.

Legal 21: Possible Structures

Here are all the possible structures

(1) $b_1(b_2(w,x), b_3(y,z))$. Example: (w+x) + (y+z).

(2) $b_1(b_2(b_3(w,x), y), z)$. Example: ((w+x)+y) + z.

(3) $b_1(z, b_2(y, b_3(w,x)))$. Example: z + (y + (w+x)).

(4) $b_1(b_2(y, b_3(w,x)), z)$. Example: (y + (w+x)) + z.

(5) $b_1(z, b_2(b_3(w,x), y))$. Example: z + ((w+x) + y).

Legal 21: Possible Structures

The hardest part is making sure you don't miss any sentence structures. For me, I found it very helpful to think about adding four numbers and all the different ways I could group it (hence the examples listed above).

We now loop through all 4!=24 ways of assigning 1,5,6,7 to w,x,y,z, and we loop through all ways of assigning binary operations (there are $4*4*4 = 4^3 = 64$ ways to do this).

Note that for some sentences, different assignments lead to the same output; if all the binary operations are addition then all 4! = 24 arrangements of the four numbers lead to the same output. It's faster to write simple code and execute it then to spend a lot of time telling the computer not to do certain calculations that give the same output as other cases. In programming, it's often good advice to not worry about being too clever unless you run into issues with how fast the code runs.

Thus, there are $5 * 4^3 * 4! = 7680$ candidates. While this is too large for most humans to do by hand, a computer can output the result of checking all of these almost instantaneously. The solution involves two divisions, which might explain why most people are unable to find it:

$\frac{6}{1-\frac{5}{7}}$. This gives us 21 from 42/2, and is of the sentence (3) type.

Related Problems

Same rules, try to get the target number using each input number once and only once; you can only use +, -, * and /.

- 21 from 1, 5, 6, 7
- 21 from 2, 3, 5, 7 (two different “sentence structures” work)
- 24 from 1, 5, 5, 5 (so you have exactly three 5’s and one 1).
- 24 from 5, 9, 13, 22
- 25 from 2, 4, 6, 8

Code

Of course, a great way to attack these problems is to write a computer program.

WARNING: MY CODE IS ON THE NEXT SLIDE – I WILL GO THROUGH IT VERY FAST SO UNLESS YOU PAUSE THE VIDEO YOU SHOULD NOT SEE IT.

```
FindCombination[a_, b_, c_, d_, x_] := Module[{},
   (*enter four numbers and what you want the combination to be*)list = {a, b, c, d};
   (*g defines the binary operation on two of our numbers.different values of u lead to different choices among+-*/,
   noting that we cannot divide by zero.If we would have had a divide by 0 we instead output 9480928501.32414312,
   as it is very unlikely that would lead to a correct solution,but of course if it does we can just check*)
   list = {a, b, c, d};
   g[u_, v_, m_] := Switch[m, 1, u + v, 2, u - v, 3, u * v, 4, If[v == 0, 9480928501.32414312, 1.0 u / v]];
   (*our print command,then go thru the 5 sentence types*)
   printop[m_] := Switch[m, 1, "+", 2, "-", 3, "*", 4, "/"];
   For[n = 1, n ≤ Length[Permutations[list]], n++,
    permlist = Permutations[list][[n]];
    x1 = permlist[[1]];
    x2 = permlist[[2]];
    x3 = permlist[[3]];
    x4 = permlist[[4]];
    For[i1 = 1, i1 ≤ 4, i1++,
     For[i2 = 1, i2 ≤ 4, i2++,
      For[i3 = 1, i3 ≤ 4, i3++,
       {
        foundsoln = 0;
        If[g[g[x1, x2, i1], g[x3, x4, i2], i3] == x, foundsoln = 1];
        If[g[g[x1, g[x2, x3, i1], i2], x4, i3] == x, foundsoln = 2];
        If[g[g[g[x1, x2, i1], x3, i2], x4, i3] == x, foundsoln = 3];
        If[g[x1, g[x2, g[x3, x4, i1], i2], i3] == x, foundsoln = 4];
        If[g[x1, g[g[x2, x3, i1], x4, i2], i3] == x, foundsoln = 5];
        If[foundsoln > 0,
         Switch[foundsoln, 1,
           Print["(", x1, printop[i1], x2, ")", printop[i3], "(", x3, printop[i2], x4, ")"],
           2,
           Print["(", x1, printop[i2], "(", x2, printop[i1], x3, "))", printop[i3], x4],
           3,
           Print["((", x1, printop[i1], x2, ")", printop[i2], x3, ")", printop[i3], x4],
           4,
           Print[x1, printop[i3], "(", x2, printop[i2], "(", x3, printop[i1], x4, "))"],
           5,
           Print[x1, printop[i3], "((", x2, printop[i1], x3, ")", printop[i2], x4]
          ]; (* end of switch *)
        ]; (* end of if *)
       } (* end of i3 loop *)
      ]]];
```

Another challenge….

- You have one of each digit from 1 to 9: thus you have 1, 2, 3, 4, 5, 6, 7, 8 and 9.
- You need to use each number exactly once
- You have to have three fractions; the numerators are one digit and the denominators are two digits.
- The sum of the three fractions is 1 – how can this be done?

Example: Could try $\frac{9}{24} + \frac{8}{13} + \frac{5}{67} = \frac{36341}{71288}$, which is not 1.

How can you find the solution?

How to attack the sum of fractions problem?

How many possibilities are there? At most

How to attack the sum of fractions problem?

How many possibilities are there? At most 9! = 9*8*7*6*5*4*3*2*1 or 362,880. a computer can do this easily! Don't even need to write efficiently. The code below (in Mathematica) takes about 4 seconds on my laptop to find the answer.

WARNING: MY CODE IS ON THE NEXT SLIDE – I WILL GO THROUGH IT VERY FAST SO UNLESS YOU PAUSE THE VIDEO YOU SHOULD NOT SEE IT.

How to attack the sum of fractions problem?

How many possibilities are there? At most 9! = 9*8*7*6*5*4*3*2*1 or 362,880. a computer can do this easily! Don't even need to write efficiently. The code below (in Mathematica) takes about 4 seconds on my laptop to find the answer.

```
list = {1,2,3,4,5,6,7,8,9};
plist = Permutations[list];
For[n = 1, n <= 9!, n++,
  {
   temp = plist[[n]];
   sum = Sum[temp[[1+i*3]] / (10 temp[[2+i*3]] + temp[[3+i*3]]),   {i,0,2}];
   If[sum == 1, Print[temp]];
   }]; (* end of n loop *)
```

Summary

This is our second lecture on how to attack problems.

The key takeaway is we want to find a good way to methodically go through all the possibilities.

The challenge is to be exhaustive – we must make sure we don’t miss anything!

If there aren’t too many cases we can do it by hand, else it is programming time.