Problem set 1 Task 3

only for problem a though | And that is not how many characters are in my name… or is it lol

A brief and detialed explanation of the 133% efficient search

By \*\*\*\*\*\*\*\*

2021

*Pre-note: Anytime you see ‘from H#’ it refers to the list of constraints listed below that I derived from the original constraints provided. None of these restrictions are a guess, they are all mathematical proofs that produce limits on what values have the ability to lead to an acceptable solution. Also I hope you find this documentation clear and insightful because I spent over 40 hours writing this and most the time was spent trying to write out constraints in a cohesive manner solely so others may potentially learn something new.*

My strategy was to simply reduce the nva as much as possible, and I thought to do this by limiting how many combinations of variables I needed to check. I started each problem with mathematical pre-analysis, try to eliminate any variables and find any additional constraints to limit the ranged of values that each variable needed to be tested with. I first noted that I only need to check values for D>22 (from H6). Then realized that if I looped over E and F first, then I could assign D the single value of E+F+21 (from H2) reducing complexity from O(n^6) to O(n^5). This made me realize how I’m important pre-analysis was. I found max values for E and F reducing nsa to 400k. Since I did not know a minimum value for E above 1, I decided to loop starting with the max value down to 1. Doing this reduced nsa to 80k! I used other constraints of limiting values and choosing order of which values to assign first to get a nsa below 7k then reductions slowed down. I managed to get below nva 500 by realizing that a did not need to be assigned a value at all if you have valid values for B, C, E and F. This is because of H1, if A = B+C+E+F then only need to check that B+C+E+F is < 51 because ‘A’ had a max value of 50 from domain. And thus checking B+C+E+F < 51 is the only requirement for passing constraint 1 and it can be done without ever assigning a value to ‘A’. A similar discover was made for D; any time that I needed the value of D, I would simply substitute D with (E+F+21) and now D never needed to be assigned a value because I assign E and F values first. This includes when checking if a solution follows constraints. For example you checking constraint is the same as . Then if it a solution print(‘D =’, (E+F+21)) and now you never to assign a value for D to a variable. This is also done later by substituting other variables such as by with just the value that they must hold. I also was able to ‘look ahead’ used if-statements to continue next iteration if I predicted that a solution would not be possible with current values before assigning any new ones. Notably with method ‘sumBC’ I implement rule H11 which, given E an F, gives the sum of B+C. I then check if sumBC follows constraints H14 and H15 by doing ((E+F+21)\*\*2-417)/E\*\*2-(E+F) %1==0 and ((E+F+21)\*\*2-417)/E\*\*2-(E+F) >(49-e-f). Or sumBC(e, f)%1==0 and sumBC(e, f)>(49-e-f) for better read ability (note that no new vars need to be assigned to do this.View main.py line 11). This lets me know if there is a possible solution for problem with only assigning E and F! Ok and that summarizes the strategies I applied throughout the project to find a solution for a nva = ...drum rolling… 4! I know that may not sound possible at first but hopefully after reading this it makes sense now why it is. Please PLEASE read over the source code and the documentation below on how I was able to derive the constraints that allow an algorithm to find a solution for 6 variables after assigning only 4. I spent way too long on this so hope you find the results as interesting as I did!

Problem a nva = 4

Problem b (without prob c constraints) nva = 28

Problem b (with prob c constraints) nva = 14

Problem c nva = 14

Remember that any solution for problem c must also be a valid solution for problem b and problem a. So because a solution that satisfied constraints from problem c took less nva then for problem b, when solving for problem b you just search for a solution for problem c and know that it is also a solution for b.

|  |  |  |
| --- | --- | --- |
| Rule | Constraints | Explanation |
| H1 |  | From C1 |
| H2 |  | From C2 |
| H3 |  | From C3 |
| H4 |  | From C4 |
| H5 |  | From Domain. Every variable has a min value of 1, and a max value of 50. |
| H6 |  | From H2 and H5. Because E and F both have min values of 1, the min value for D is 1+1+21. Thus D >=23 i.e. D > 22 |
| H7 |  | From H2 and H5. Because D has max value of 50 D=E+F+21  50 E+F + 21  29 E+F  30 > E+F |
| H8 |  | From H3 |
| H9 |  | From H8.  The largest possible value for E will be the max value for D divided by the min value for A. Look at E1 for a simple explanation on why this is true (I say “simple” but it took me 4 hours to come up with it lol). |
| H10 |  | From H3  Just moved things so A is alone on one side |
| H11 |  | From substituting A in H10 from H1  This gives a proof for what (B+C) must equal for C3 to be satisfied, given E and F (Note D can be substituted with (E+F+21) |
| H12 |  | From H1  Also implies…  then |
| H13 |  | From H7 |
| H14 | = integer | From H11  Because B and C are integers, the sum of B+C must also be an integer. Thus the result of this equation must also be an integer.  I.e **Given E and F, if the result is a decimal then** the sum of (B+C) would have to be a decimal for C3 to be satisfied (shown in H11). We don’t need to know the specific value of B and C, we already to know that they are both integers, and the sum of two integers must also be an integer. Therefore the equation must result in an integer value (greater than 2 as well) or **there will be no possible combinations of B and C that are both integers and equal that result.** |
| H15 |  | From H11 and H1  Because B+C+E+F = A, and A<51  B+C+E+F <51  B+C < 51 - (E+F) |
| H16 |  | From H11 |
|  |  |  |
|  | Constraints from 3b |  |
| C5 |  |  |
| C6 |  |  |
| C7 |  |  |
| C8 |  |  |
| C9 |  |  |
| H19 |  | From C9 |
| H20 |  | From C7  J > 9 because lower values would require taking a root of a negative number in  Run function h20() in proofs.py for more  G < 13 because if J=50 (the max value for J), the square root of 4\*J-39 is 12.69. Which means G can never be larger than 12.  G must be in [1, 5, 7, 9, 11] exclusively because these are the only values for G that result in a whole number for J in the top equation. Thus we get constraints  On the possible values for G and the only possible value for J given G |
| H21 |  | From C7 and with |
| H22 |  | From from H19 and H21 |
| H23 |  | From C8 and G=11 from H21  H < F because if you take the cube root of a negative number, it becomes complex. And I is not allowed to be a complex number. |
|  |  |  |
| H24 |  | From C9:  From H22 replace G with 11:  After expanding, it should be obvious that we can limit the possible values for C because of the domain F.  C > 1 because C=1 would yield F=100, which is not allowed.  C < 5 because C = 5 yields a decimal, and any larger value for C will yield negative values for F; which are not allowed.  This leads to the new constraints of C and F. |
| H25 |  | From H23 and G=11 from H21 |
| H26 |  | From H25  Since we know H > 0 and H < 20. Since H < F and F 20.  We can say and  The min for I can now be written as      F needs to be it’s min (from H24) to find min for I      Same is done to find max of I, but using the max val of F which is 20 (from H24)            I < 14 |
|  |  |  |
|  | Constraints from 3c |  |
| C10 |  |  |
| C11 |  |  |
| C12 |  |  |
| C13 |  |  |
| C14 |  |  |
| C15 |  |  |
| H30 |  | From C10  Let M be max then Thus K=10.3 at most  C14 also proves K<11 because solving for M+O with K=11 is  M+O=121-10 which means that M+O would have equal 111 which is not allowed because M and O have to be 50 or less each. |
| H31 |  | From C10  If K is less then 4, it will result in M being negative which is not allowed.  Odd values of K will result in M being a decimal which is not allowed. |
| H32 |  | From C14 |
| H33 |  | From C12  This also suggest |
|  |  | I came up with a theory that L must be less than 17 from this. Though it is only a theory and not a proof so I chose not to use it |

Further Explanations

|  |  |  |
| --- | --- | --- |
| E1 |  | To understand why the max value for E is found by substituting D for its max value, and substituting A for its min value, let’s image a simpler constraint  Which is the same as  If we want to find the max value for **this** E (given the same domain and constraints C1, C2, and C3 still exist) it should be easy to see that if D and A can be any int {1, 2, …, 50} that E would be the largest when D is its max possible value (50) and A is its min possible value (1)  i.e.  [= **50**  Opposed to  [= **0.02** |
| E1.1 |  | The max value for D is 50 since there are no constraints on its limit other than the domain. The min value for A is 31 because when D = 50…  D = E+F+21  E+F=D-21  E+F=50-21  (E+F)=29  A = B+C+(E+F)  A = B+C+(29)  Let B=1 and C=1 since we are still trying to get minimum possible value for A when D=50  A=1+1+29  A=31 |
| E1.2 |  | From computing left side of < from E1.1 |
| E1.3 |  | Since E must be an integer, the maximum value for E is 8 |