cpt_s 350

Final Take Home Exam

11641327 Yu-Chieh Wang

2020/4/30

1. The values of k and m.

   Here, k represents the number of variables, and m represents the number of constraints. Therefore, we got k=2 and m=4 in the picture (1).

2. The size of the input.

   In unsigned int, there are only 32 bits(1 byte) which is able to represent the biggest number: 4294967295. In the graph (2), assume each size of variable and parameter is 1 byte. Therefore, the totally size will be k+mk+m bytes.

3. Why the algorithm cannot solve the LD problem?

   Enumerating all possible integers is not an algorithm; specially, the number of each integer is unbounded. To check the value of a variable by enumeration, it might run 2^{32} times from 0 to the largest value of a nonnegative integer. Therefore, the worst case of getting the values of all variables will be

   $$O(2^{32} \times 2^{32} \times \ldots \times 2^{32}) = O(2^{32+32+\ldots+32}) = O(2^{32k}).$$

   When the value of k is huge, the algorithm might run forever.

4. Implement functions.

   (algorithm2.py)

```
/usr/local/bin/python3.7 /Users/angel/PycharmProjects/Algorithm/FinalExam/algorithm2.py
c1= 3 , c2= -2 ,c3= -1 ,c= 3
c1= 6 , c2= -4 ,c3= 1 ,c= 3
get a path
[[0, 1, -1, 2, 0, 1, -1, 2, 0, 1, 1], [-1, 2, 0, 3, -1, 2, 0, 3, 0, 0, 0]]
x1= 0 ,x2= 1 ,x3= 1

Process finished with exit code 0
```

```
/usr/local/bin/python3.7 /Users/angel/PycharmProjects/Algorithm/FinalExam/algorithm2.py
c1= 3 , c2= -2 ,c3= 1 ,c= 5
c1= 6 , c2= -4 ,c3= 2 ,c= 9
there is no solution

Process finished with exit code 0
```

5. The worst case time complexity for the algorithm in question 4.

   Given: $C_1X_1 + C2X_2 + C3X3 + C = 0$

   First, assume we have a $\hat{C} = max\{\,|C_1|, |C_2|, |C_3|, |C|\,\}$, the worst case input size N in the algorithm is $O(log_2\hat{C})$, so we get $\hat{C} = O(2^{O(N)})$.

   Then, consider the size of input in M, we get the worst case time complexity:

   $(2C_{max} + 1) \cdot (K_C + 1) = O(N \cdot \hat{C}) = O(N \cdot 2^{O(N)})$ which is exponential time.

6. Sketch how to generalize the algorithm to general LD instance Q in graph (2).

   The original equation is: $C_1X_1 + C2X_2 + C3X3 + C = 0$

   As we know, C is broken down into $b_1b_2b_3 \ldots b_n$ which are used in the algorithm to help check a path from a node to a node. If we want to solve inequalities instead of equations, enumerating the number of C might be a good way. For example,

   $C_1X_1 + C2X_2 + C3X3 + C > 0$

   So, we can assume the original C as $C + 1, C + 2, \ldots C + n$.

   Here, we have limited the size of C which is a nonnegative integer, so it is impossible to enumerate C to infinity.

7. Why the generalized algorithm runs in exponential time?

   Because we use binary to calculate those variables. Therefore, when the input size is

larger, the running time of the program is multiplied by the exponential time.

8. An application that the generalized algorithm in question 6 above can be used.

This algorithm is too useless for equations with only two variables X_1 and X_2. There is a simpler algorithm on the Internet that uses the least common factor of two coefficients C_1 and C_2 to find the smallest solution. Also, other solutions can be calculated step by step using this smallest solution.

However, this algorithm is not suitable for calculating equations with more than three variables, especially, solving inequalities. Although more variables will run the calculation time exponentially, but for solving this difficult problem, this is already the fastest and most effective algorithm. Take a common application as an example: lucky draw. Generally speaking, when a company make preparations of a luck draw, it will consider the gold fee for the purchase of prizes, the number and price of prizes, and the number of winners. Here, variables $X_1, X_2, X_3, \ldots, X_n$ represent our n kinds of prizes, $C_1, C_2, C_3, \ldots, C_n$ represent how much does each prize cost, and last number C represents how much did the company prepare for this event.

For example, The company's budget is only 1,000 US dollars. It costs 500 US dollars to buy a TV, 500 US dollars for a computer and 250 US dollars for a mobile phone. Assuming that the company requires 1,000 yuan to be used up, we list the following equation:

$$-500X_1 - 500X_2 - 250X_3 + 1000 = 0$$

(What we should pay attention to here is that because the last number is restricted to non-negative integers, the previous coefficients must be added with a negative sign to be true.) According to this equation, we can get the following solution:

Solution	X_1	X_2	X_3
1	0	0	4
2	0	2	0
3	2	0	0
4	1	1	0
5	1	0	2
6	0	1	2

However, the ability of this algorithm is more than that, we can add more conditions to it. For instance, the company requires exactly three prizes. Then, we get our second equation as follow:

$$X_1 + X_2 + X_3 = 3$$

So, we will get two solutions {(1, 0, 2), (0, 1, 2)}. Or we can write the equation as an inequality, and the solution will get three or more prizes:

$$X_1 + X_2 + X_3 \geq 3$$

Then, we will get three solutions {(0, 0, 4), (1, 0, 2), (0, 1, 2)}

In addition, we also found a way to help companies save costs. For example, the company hopes to spend less than 1,000, and there is no requirement for the type and price of the prize, but there must be three prizes. At this time we can list the equation:

$$-500X_1 - 500X_2 - 250X_3 + 1000 > 0$$

$$X_1 + X_2 + X_3 = 3$$

Therefore, we get the only solution {(0, 0, 3)}, which means the prizes are three identical phones, and it helps the company save 250 dollars.

As a result, the above is just a simple example. Luck draw in real life must be prepared with many prizes($X_1, X_2, X_3, \ldots, X_n$). It is very time-consuming to rely on manual labor alone, not to mention want to control the number of certain prizes or reduce costs. However, the algorithm can accept all requirements and list all the results for selection. Although the previous topic mentioned that the algorithm is calculated in exponential time, compared with manual calculation, this algorithm saves a lot of time and cost. The only drawback is that this algorithm can only be applied to complete entities. In other words, it cannot be used for objects whose calculated variables are decimal points, such as half a cake, weighed vegetables and meat. Nevertheless, the calculation of non-integers is much more complicated than we thought, even if it is really calculated, it may not necessarily be used in the real world. For example, I have $ 20. I can buy 1.6666 pounds of beef and 3.567 pounds of chicken at the same time. This makes the simple operation more complicated. Therefore, in order to do this kind of operation with floating point numbers, in addition to finding out the algorithm, we must also consider its practicability. As far as the current results are concerned, this algorithm is very practical and very valuable.