Report On Why ALLSUBSETS Computable Problem Is Outside Poly

Yiming Peng

Cover Page

Goal: I am trying to show ALLSUBSETS problem is computable but outside Poly with a proof. And given examples to be specific.

How it went: The computable problem I chose is called ALLSUBSETS, which is to find the possible subsets of a list of distinct integers in decimal notation separated by whitespace. And it does succeed in finding the possible subsets of a list of distinct integers. It will take a list of distinct integers in decimal notation separated by whitespace. For example: “1 2 3 4”. And the output is a list of all subsets that can be formed from the input. The elements of each set are separated by commas, and the sets are surrounded by braces. Whitespace may be included wherever convenient. From above: On input “1 2 3 4”, a solution is

{} {1} {2} {3} {4} {1,2} {1,3} {1,4} {2,3} {2,4}
{3,4} {1,2,3} {1,2,4} {1,3,4} {2,3,4} {1,2,3,4}

Thus, we successfully found the possible subset of the input list.

Example 2 input is “ ”, the output is {} which has 1 subset from 2^0=1

Example 3 input is “10, 20, 30 ” the output is {} {10} {20} {10, 20} {30} {10, 30} {20, 30} {10, 20, 30}

(1) describe the computational problem of my choice

The computational problem I chose is called ALLSUBSETS, which is to find all the possible subsets of a list of distinct integers in decimal notation separated by whitespace.

Input: A list of distinct integers in decimal notation separated by whitespace. For example: “1 3 5 7”.

Output: A list of all subsets that can be formed from the input. The elements of each set are separated by commas, and the sets are surrounded by braces. Whitespace may be included wherever convenient. From above: On input “1 3 5 7”, a solution is

{} {1} {3} {5} {7} {1,3} {1,5} {1,7} {3,5} {3,7}
{5,7} {1,3,5} {1,3,7} {1,5,7} {3,5,7} {1,3,5,7}

(2) Research in the textbook to present proof of why the problem is outside poly(textbook page 232 && 233)

To analyze allSubsets.py, we first know that a set of *k* elements has 2^*k* subsets. Hence, this program produces a list of at most 2^*n* subsets, each of size at most *n*. Based on this insight, it’s that allSubsets.py is in *O*(*n*2^*n*), proving that ALLSUBSETS is in Expo. That puts an asymptotic upper bound on the running time. To show that ALLSUBSETS is not in Poly by obtaining a lower bound. The length of the output is at least 2^*k*, since that is the number of subsets in the output. Thus it can be shown that it's always possible to choose $k \geq n/(3 \log10\ n)$. So the length of the output is at least 2^*n*/(3 log10 *n*), which grows faster than any polynomial. This proves that ALLSUBSETS is not in Poly. This also shows Poly is a proper subset of Expo. Thus, we can finally conclude that this ALLSUBSETS computational problem is outside poly.

Python Code:

```
def allSubsets(inString):

# the elements from which we can construct subsets

elems = inString.split()

# start with an empty list of subsets

theSubsets = [ ]

# add the empty set "[ ]" to the list of subsets theSubsets.append( [ ] )

# For each element of the input, append copies of all

# previously computed subsets, but with the additional new element # included.

for element in elems:

newSets = [ ]

for thisSet in theSubsets:

# create a new subset that includes the current element, # and append it to the list of subsets
newSets.append(thisSet + [element])

# Update the master list of subsets with all the newly created # subsets. This doubles the number of
subsets in theSubsets. theSubsets.extend(newSets)

return utils.formatSetOfSets(theSubsets)
```

(3) My understanding of why ALLSUBSETS is outside poly

Another way of doing this is:

Code Analysis(C++):

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> nums = {1, 3, 5, 7};
    vector<vector<int>> ans;
    for (int i = 0; i < (1 << (nums.size())); i++)
    {
        vector<int> temp;
        for (int j = 0; j < nums.size(); j++)
        {
            if (i & (1 << j))
            {
                temp.push_back(nums[j]);
            }
        }
        ans.push_back(temp);
    }
    for (auto x : ans)
    {
        if(x.size()==0)
        {
            cout << "{} ";
        }
        if(x.size()>0)
        cout<<"{";
        for (auto y : x)
        {
            if(y==x[x.size()-1])
            cout << y <<"}";
            else
            cout << y <<", ";
        }
        cout << "  ";
    }
    return 0;
}
```

Output : {} {1} {3} {1, 3} {5} {1, 5} {3, 5} {1, 3, 5} {7} {1, 7} {3, 7} {1, 3, 7} {5, 7} {1, 5, 7} {3, 5, 7} {1, 3, 5, 7}

We run two nested loops, one of range 2^n and the other of range n. so the final time complexity is O(n2^n). So this shows us that it is exponential, and the output requires at least O(2^n) amount of space, then any program needs at least O(2^n) to run as a lower bound because it always has 2^n(n is the number of elements in the input list) subsets of the input list, not only in my coding example.

To analyze allSubsets.c++, we first know that a set of *k* elements has 2^*k* subsets. Hence, this program produces a list of at most 2^*n* subsets, each of size at most *n*. Based on this insight, it's that allSubsets.c++ is in *O*(*n*2^*n*), proving that ALLSUBSETS is in Expo. That puts an asymptotic upper bound on the running time. To show that ALLSUBSETS is not in Poly by obtaining a lower bound. The length of the output is at least 2^*k* where k is the size of the input list, since that is the number of subsets in the output. Thus it can be shown that it at least needs O(2^n) amount of space for any program that can solve it as a lower bound, which grows faster than any polynomial. And because of this it always runs faster than poly, thus it's impossible that ALLSUBSETS computable problem is in poly in any situations.This proves that ALLSUBSETS is not in Poly. Or we can say it is superpolynomial because it is not in O(n^k) for any k>=1, which means it is faster than polynomial. Since the definition of poly is a class of computational problems that can be solved by a python program with running time O(n^k) for k>=0.This also shows Poly is a proper subset of Expo. Thus, we can eventually conclude that this ALLSUBSETS computable problem is outside poly.

Work Cited Page

Maccormick, John. *What Can Be Computed?* Princeton University Press, May 15, 2018.