CS 401 Programming Assignment I Report

# Participants

Ivan Madrid-

Sai Nadkarni- 672756678

# Source Code

Github: https://github.com/sai-n20/subsetsum-DP

# Number of distinct subsets

For distinct subsets we created a table where we pass in the target value. We marked and check for base case values of 0 or 1 elements. Then we iterate from i=1 to the size of the vector elems, within that iteration we also iterate from x=1 to the target value. We then check the following include and exclude cases:

if (elems.at(i).x <= x) {

includingCurrentValue = distinctSubsets[i - 1][x - elems.at(i).x];

}

excludingCurrentValue = distinctSubsets[i - 1][x];

distinctSubsets[i][x] = includingCurrentValue + excludingCurrentValue;

when the function is returned, it will return the last value within the table which holds the value of the total subsets that have a sum of the target sum.

# Size of smallest subset

For the smallest size we implemented a function where we pass in the target sum value and the size of the elems vector. First, we check the condition where to solve the trivial incase target matches the input amount. Then we iterate from i=0 to the size of the elems vector; we also have an inner loop that iterates from x=1 to the target value. We then create an include variable and an exclude variable to keep track of the values. We then have two if statements, one that checks if elems[i].x <= x which will include the current value. The other if statement, i >0 is for the excluding the current value. We then check which is the minimum value and return that value.

for (int i = 0; i < num; i++) {

if (elems.at(i).x == tgt) { return 1; }

minCount[i][0] = 0;

}

for (int i = 0; i < num; i++) {

for (int x = 1; x <= tgt; x++) {

unsigned int includingCurrentValue = 429496729;

unsigned int excludingCurrentValue = 429496729;

if (elems[i].x <= x) {

includingCurrentValue = 1 + minCount[i][x - elems[i].x];

}

if (i > 0) {

excludingCurrentValue = minCount[i - 1][x];

}

minCount[i][x] = std::min(includingCurrentValue, excludingCurrentValue);

}

}

return minCount[num - 1][tgt];

# Number of distinct minimum size subsets

First we created a table that will fill up with Booleans if there was a subset to be acheieved where we iterate of the size of the elems vector and then inner iterate from the value of the target +1 from there we do the following value setup to the table:

dp[i][j] = (elems[i].x <= j) ? dp[i-1][j] ||

dp[i-1][j-elems[i].x]

: dp[i - 1][j];

We then call printSubsetsRec() that will pass in a vector and then check to see if the vector is no larger than the smallest subset

if(p.size() > min)

if value I== 0 and the target/sum is not 0 and if the value it achieveable we then push and print each value in the new vector p

if (i == 0 && sum != 0 && dp[0][sum])

{

p.push\_back(elems[i].x);

display(p);

return;

}

If the target/sum becomes 0 we display the vector

// If sum becomes 0

if (i == 0 && sum == 0)

{

display(p);

return;

}

If the given target/sum can be achieved from ignoring the current value then we recursively call the function again.

if (dp[i-1][sum])

{

// Create a new vector to store path

vector<int> b = p;

printSubsetsRec( i-1, sum, b, min);

}

If the taget/sum can be achieved with the current value we then call the function recursively with the current value

if (sum >= elems[i].x && dp[i-1][sum-elems[i].x])

{

p.push\_back(elems[i].x);

printSubsetsRec( i-1, sum-elems[i].x, p, min);

}

# Lexicographic first minimum size subset

# Data on various runs

**Scenario: Electoral college tie scenario, T = 269**

Text

Description automatically generated

**Scenario: Republican victory among purple states**

**Scenario: Democratic victory among purple states**