UNIVERSIDAD PRIVADA BOLIVIANA DIRECCIÓN DE PREGRADO

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

INVESTIGACIÓN

Complejidad Algorítmica DigitDP y SubsetSum

JUAN CLAUDIO CARRASCO TAPIA

LA PAZ – BOLIVIA

2022

Complejidad Subset Sum:

```
bool isSubsetSum(int set[], int n, int sum)
{
    bool subset[n + 1][sum + 1];
    for (int i = 0; i <= n; i++)
        subset[i][0] = true;
    for (int i = 1; i <= sum; i++)
        subset[0][i] = false;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= sum; j++) {
            if (j < set[i - 1])
                subset[i][j] = subset[i - 1][j];
            if (j >= set[i - 1])
                subset[i][j] = subset[i - 1][j]
                               || subset[i - 1][j - set[i - 1]];
        }
    }
```

El algoritmo recorre una matriz de 2 dimensiones, hacia abajo es del tamaño del arreglo inicial, y hacia un lado es del tamaño del valor de suma buscado.
Complejidad O(n*m) → (for i to n) y dentro (for j to m)

Complejidad Digit DP

```
int solve_dp(int pos, int mayor,int pares, int suma) {
    if(pos > number.size()) { // cuando la posicion exceda al numero dado
        return 0;
    }
    // Modificar de acuerdo al problema
    if(pos == number.size()) {
        // cout<<pares<<endl;
        if(pares >= 2) { // tiene 2 pares el numero
            return 1;
        }
        else {
            return 0;
        }
    }
    if(dp[pos][mayor][pares][suma] == -1) { // Pregunto si no lo he calculado
        int tope = 9;
        if(mayor == true) {  // el numero que voy a crear puede llegar a ser mayor
            tope = number[pos]-'0';  // solo podemos usar los numeros de 0 al tope -- '3'-'0'
        }
        dp[pos][mayor][pares][suma] = 0;
        for(int digito = 0; digito <= tope; digito++) {
            if(digito == tope ) {
                // cout<<pares<<" "<<digito<<" "<< (digito%2 == 0) <<endl;
                int total= suma +digito;
                int esPar = (total > 0) && (digito%2 == 1);
                dp[pos][mayor][pares][suma] += solve_dp(pos+1, true, pares + esPar, total );
            }
            else { // 0 1 2
                //cout<<pares<<" "<<digito<<" "<< (digito%2 == 0) <<endl;
                int total = suma + digito;
                int esPar = (total > 0) && (digito%2 == 1);
                dp[pos][mayor][pares][suma] += solve_dp(pos+1, false, pares + esPar, total);
            }
        }
    }
    return dp[pos][mayor][pares][suma];
}
```

Son 0 a 9 iteraciones para cada estado y se crean un total de pos*suma*tope(mayor) estados, por lo que

Complejidad O(n*suma*tope)