DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Informe de Trabajo Práctico 1

Subset Sum Problem

...

Algoritmos y Estructuras de Datos III

Integrante	LU	Correo electrónico
Springhart, Gonzalo	308/17	glspringhart@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Introducción

En este informe vamos a comparar a eficiencia de distintos algoritmos utilizados para resolver un problema conocido como *Subset Sum Problem* (o Problema de suma de subconjuntos). El mismo consiste en lo siguiente, dado un conjunto S de n elementos, cada uno con un valor asociado v_i y un valor objetivo V, se quiere saber si existe un subconjunto de ítems de S que sumen exactamente el valor objetivo, y si existe dicho subconjunto, se quiere saber cuál es la mínima cardinalidad entre todos los subconjuntos posibles, en otras palabras, hay que decidir si existe $R \subseteq S$ tal que $\sum_{i \in R} vi = V$. Se asumen también que los valores de S son enteros no negativos (aunque el problema se puede resolver también sin necesidad de esta restricción).
El objetivo es ver cuál de los algoritmos es más eficiente al resolver el problema, se van a presentar 4 algoritmos que resuelven el problema, indicando como funcionan, justificando sus complejidades y comprobando a travéz de experimentos que estas complejidades son ciertas.

1. Algoritmos y justificación de complejidades

Se va a usar la siguiente notación:

- S es el conjunto, que tiene n elementos, cada uno con un valor asociado v_i con $i \in \{1, ..., n\}$
- V es el valor objetivo

1.1. Fuerza Bruta

El primer algoritmo presentado para resolver el problema es uno de *Fuerza Bruta*, básicamente el algoritmo genera todos los conjuntos posibles con los elementos de S y se fija cuál de ellos tiene elementos tales que su suma de exactamente V, mientras los calcula se va quedando con el que tiene menor cardinalidad.
El algoritmo es el siguiente:

```
procedure FUERZABRUTA(S, V)
    longMinima ← −1
    partes ← generarConjPartes(S)
    for all Conjunto c ∈ partes do
        sum ← sumarElem(c)
        if sum == V then
            if longMinima == −1 then
                longMinima ← |c|
            else
                longMinima ← min(longMinima, |c|)
            end if
        end if
    end for
     return longMinima
end procedure
```

Como se puede ver el algoritmo ejecuta un ciclo 2^n veces, dentro de cada ciclo se realiza la suma de sus elementos, esta suma se puede realizar en tiempo lineal, y las operaciones para calcular el mínimo después de hacer la suma son $O(1)$ Entonces la complejidad del algoritmo es $O(n * 2^n)$.

1.2. Backtracking

1.3. Programación Dinámica Top Down

1.4. Programación Dinámica Bottom Up