Partition Problem

Francisco Javier Arocas Herrera
Óscar Hernández Díaz
Gabriel Melián Hernández
Adrián Epifanio Rodríguez Hernández

1. Introducción al Partition Problem

2. Variantes del Partition

- Suma de Subconjuntos
- Partition de números de múltiples vías.
- 3-Partition
- Partition product

3. Dificultad Computacional.

- Partimos de una instancia de suma de subconjuntos para decidir si hay un subconjunto S con suma T.
- Instancia de Partition: conjunto inicial más z1=suma(S) y z2 = 2T.
- Suma(S)+z1+z2 = 2*suma(S) + 2T. Suma objetivo de Partition = Suma(S)+T
- Si Suma(S’) = T → Suma(S’ U {z1}) = Suma(S) + T → S’ U {z1}, es solución a la instancia.
- NP-Hard.

4. Algoritmos Aproximados.

- Greedy Number Partitioning.
  - Sin ordenar → O(n), con relación de aproximación 3/2.
  - Ordenados → O(n log n), con relación de aproximación 7/6.
- Largest Differencing Method (Karmarkar-Karp Algorithm).
  - O(n log n), con relación de aproximación 7/6.
  - Funciona mejor en entornos de simulación.
- Multifit.
  - En el peor caso su relación de aproximación es 8/7.

5. Algoritmos Exactos

- Algoritmos que siempre encuentran la partición óptima.
  - Pueden utilizar un tiempo exponencial
- Estos algoritmos son:

1. Pseudo polynomial time number partitioning
2. Complete Greedy Algorithm (CGA)
3. Complete Karmarkar-Karp Algorithm (CKK)

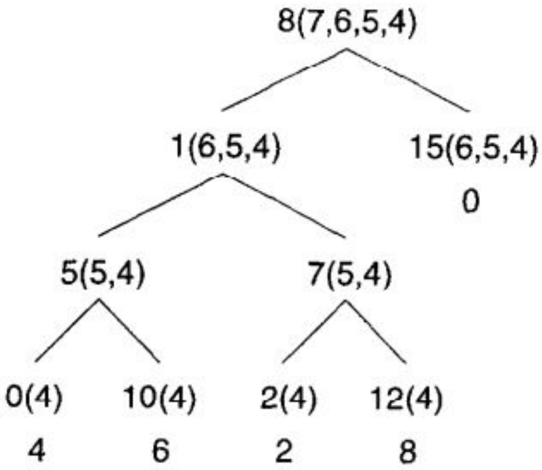
5.1 Pseudo polynomial time number partitioning

- Ocupa O(nm) de memoria
- M = número más grande en la entrada
- Se puede extender y utilizar para varias particiones, pero no es eficiente.

```
function can_be_partitioned_equally(S) is
    input: A list of integers S.
    output: True if S can be partitioned into two subsets that have equal sum.

    n ← |S|
    K ← sum(S)
    P ← empty boolean table of size (⌊K/2⌋ + 1) by (n + 1)

    initialize top row (P(0,x)) of P to True
    initialize leftmost column (P(x, 0)) of P, except for P(0, 0) to False

    for i from 1 to ⌊K/2⌋
        for j from 1 to n
            x = S[j-1]
            if (i-x) >= 0 then
                P(i, j) ← P(i, j-1) or P(i-x, j-1)
            else
                P(i, j) ← P(i, j-1)

   return P(⌊K/2⌋, n)
```

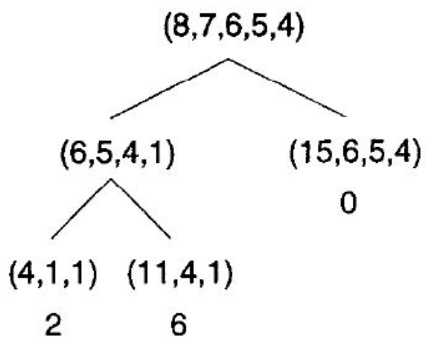
5.2 Complete Greedy Algorithm (CGA)

- Considera todas las particiones mediante la construcción de un árbol binario
- Requiere sólo O(n) espacio, pero puede tomar O(2^n) tiempo
- Se puede mejorar con una heurística



5.3 Complete Karmarkar-Karp algorithm (CKK)

- Es una extensión de el algoritmo complete greedy
- Considera todas las particiones mediante la construcción de un árbol binario
- El árbol es de menor tamaño que el de CGA. Es màs eficiente, y suele producir una solución mejor que el greedy.
- Require O(n) espacio, pero en el peor de los casos podría tomar O(2^n)



6. Ejemplo

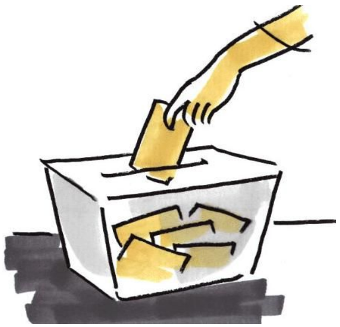
Dado el conjunto S = {3,1,1,2,2,1}

A	**S1 = {1,1,1,2}**	=	**S2 = {2,3}**.
B	**S1 = {2,1,1}**	≠	**S2 = {2,2,1,3}**
C	**S1 = {3,1,1}**	=	**S2 = {2,2,1}**

~~**S = {2,5}**~~

7. Aplicaciones

Manipulación en las elecciones



A **B** **C**

8. Formato Input

Nº de elementos
Elementos del conjunto

example.txt

```
10
15 5 10 1 9 20 10 4 4 2
```

9.Bibliografía

- Wikipedia: https://en.wikipedia.org/wiki/Partition_problem
- Definición: https://arxiv.org/ftp/cond-mat/papers/0310/0310317.pdf
- Github: https://github.com/AdrianEpi/CC-Partition

10. Contacto

Francisco Javier Arocas Herrera	*alu0100906813@ull.edu.es*
Óscar Hernández Díaz	*alu0101127163@ull.edu.es*
Gabriel Melián Hernández	*alu0100819786@ull.edu.es*
Adrián Epifanio Rodríguez Hernández	*alu0101158280@ull.edu.es*