ALGORITMO DE APROXIMACIÓN

SUBSET SUM

Roy H.

DEFINICIÓN

CONCEPTOS

Sea $S = \{x_1, x_2, \ldots, x_k\}$ un conjunto de números naturales y un número objetivo *t*. Queremos determinar si el conjunto S contiene un subconjunto que suma t.

$$\left\{ \begin{array}{c} SUBSETSUM = \langle S, t \rangle \,\middle|\, S = \{x_1, x_2, \ldots, x_k\}, \\ y\ para\ algún\ \{y_1, y_2, \ldots, y_m\} \subseteq S, tal\ que \sum y_i = t \end{array} \right\}$$

EJEMPLO DE DEFINICIÓN

SAMPLE

Por ejemplo $\langle\{\mathbf{4}, 11, 16, \mathbf{21}, 27\}, 25\rangle \in SUBSETSUM$, porque 4 + 21 = 25.

CARACTERÍSTICAS

FEATURES

- SUBSET-SUM está en NP

- SUBSET-SUM está en NP – Completo

Algoritmo exacto para SUBSET-SUM.

ALGORITHM

Idea! $S = \{5,8,3\}$ y $t = 11$

$$P(S) = \{\phi, \{5\}, \{8\}, \{3\}, \{5,8\}, \{5,3\}, \{8,3\}, \{5,8,3\}\}$$

```
for sub_set in power_set:
    # print sub_set
    suma = 0
    for element in sub_set:
        # print element
        suma = suma + element
        if( suma == target ):
```

```
A:\Maestria\Complex\proyecto\src
λ python 1-powerset.py
S =  [5, 8, 3]
number_elements = 2**n =  8
P(S) =  [(), (5,), (8,), (3,), (5, 8), (5, 3), (8, 3), (5, 8, 3)]
target =  11
Encontrado =  (8, 3)
Tiempo de Ejecucion:  0.0061060576737
```

DEMO #1.

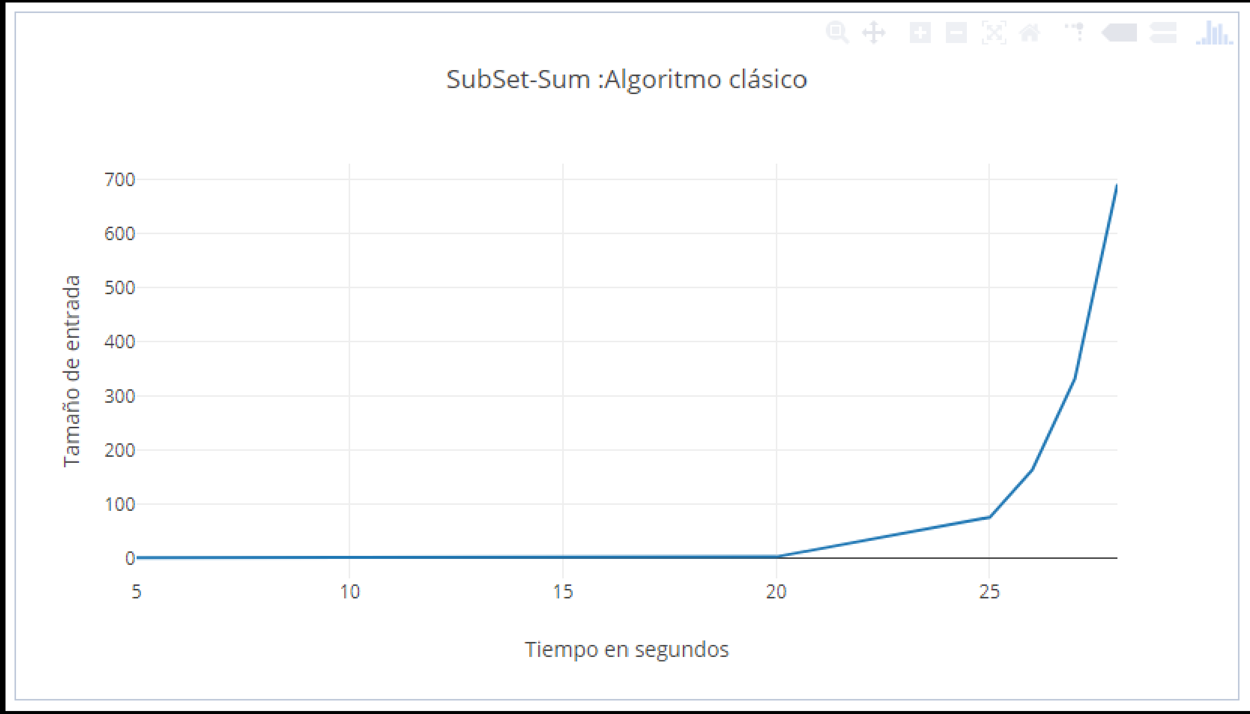
DEMO

Porque esto es un problema?

WHY

$P(S) = 2^n$, esto es: El # de operaciones se duplica con incrementar el valor de n en 1

Tamaño de entrada n	Conjunto Potencia 2^n	Algoritmo Clásico (segundos)
5	32	0.001871
10	1024	0.004472
20	1048576	1.895628
25	33554432	75.07551
26	67108864	162.88581
27	134217728	331.98794
28	268435456	691.35722
30	2^{30}	Intratable
40	2^{40}	Intratable
50	2^{50}	Intratable
100	2^{100}	Intratable
500	2^{500}	Intratable
1000	2^{1000}	Intratable
10000	2^{10000}	Intratable



SOLUCIÓN ALGORITMO DE APROXIMACIÓN

SOLVE

Idea Principal! Ataque al tamaño de la lista (Cormen, Leiserson, Rivest, & Stein, 2009), sea L la lista original y L’ la lista recortada.

$$\frac{y}{1+\delta} \leq z \ \leq y; \text{ parámetro de recorte } \delta \text{ tal que } 0 < \delta \ < 1$$

Entonces para cada elemento que fue removido de L, hay un elemento **z** que se aproxima a *y* que se mantiene en L’:

SOLUCIÓN ALGORITMO DE APROXIMACIÓN

SOLVE

Por ejemplo, si $\delta = 0.1$ y $L = \langle 10,11,12,15,20,21,22,23,24,29 \rangle$ y luego recortamos L obtenemos.

$$L' = \langle 10, 12, 15, 20\ , 23,29 \rangle$$

El elemento eliminado 11 está representado por 10, los elementos 21 y 22 están representados por 20 y el elemento 24 está representado por 23.

SOLUCIÓN ALGORITMO DE APROXIMACIÓN

SOLVE

El siguiente procedimiento recorta la lista $L = \langle y_1, y_2, \dots, y_m \rangle$ en un tiempo $O(m)$, dado L y δ, y suponiendo que L esta ordenado de forma creciente. La salida del procedimiento es una lista recortada y ordenada.

```
TRIM(L, δ)
let m be the length of L
L' = ⟨y_1⟩
last = y_1
for i = 2 to m
    if y_i > last · (1 + δ)        // y_i ≥ last because L is sorted
        append y_i onto the end of L'
        last = y_i
return L'
```

```
EXACT-SUBSET-SUM(S, t)
n = |S|
L_0 = ⟨0⟩
for i = 1 to n
    L_i = MERGE-LISTS(L_{i-1}, L_{i-1} + x_i)
    remove from L_i every element that is greater than t
return the largest element in L_n
```

SOLUCIÓN ALGORITMO DE APROXIMACIÓN

SOLVE

$S = \langle 104{,}102{,}201{,}101 \rangle$ con $t = 308$ y $\delta = 0.40$. El valor de recorte $\delta = \frac{\varepsilon}{8} = 0.05$ APPROX-SUBSET-SUM calcula como sigue los valores en las líneas indicadas:

```
S: [104, 102, 201, 101]; target: 308; epsilon: 0.4

L' [{'approx': 104, 'l_prima': [0, 104]}]

L' [{'approx': 104, 'l_prima': [0, 104]}, {'approx': 206, 'l_prima':
[0, 104, 102]}]

L' [{'approx': 104, 'l_prima': [0, 104]}, {'approx': 206, 'l_prima':
[0, 104, 102]}, {'approx': 305, 'l_prima': [0, 104, 201]}, {'approx':
407, 'l_prima': [0, 104, 102, 201]}]

L' [{'approx': 104, 'l_prima': [0, 104]}, {'approx': 205, 'l_prima':
[0, 104, 101]}, {'approx': 305, 'l_prima': [0, 104, 201]}, {'approx':
406, 'l_prima': [0, 104, 201, 101]}]

{'approx': 305, 'l_prima': [0, 104, 201]}
```

Calculo de aproximación para SUBSET-SUM.

SOLVE

El algoritmo de aproximación devuelve $z^{*} = 305$ como su respuesta para $\delta = 40\%$ la respuesta óptima 307 = 104 + 102 + 101; de hecho, tiene una aproximación para el objetivo de.

$$\frac{Valor\ \acute{o}ptimo * 100\%}{Valor\ obtenido} = \frac{305}{302} = \mathbf{99.016}\ \%$$

DEMO #2.

DEMO

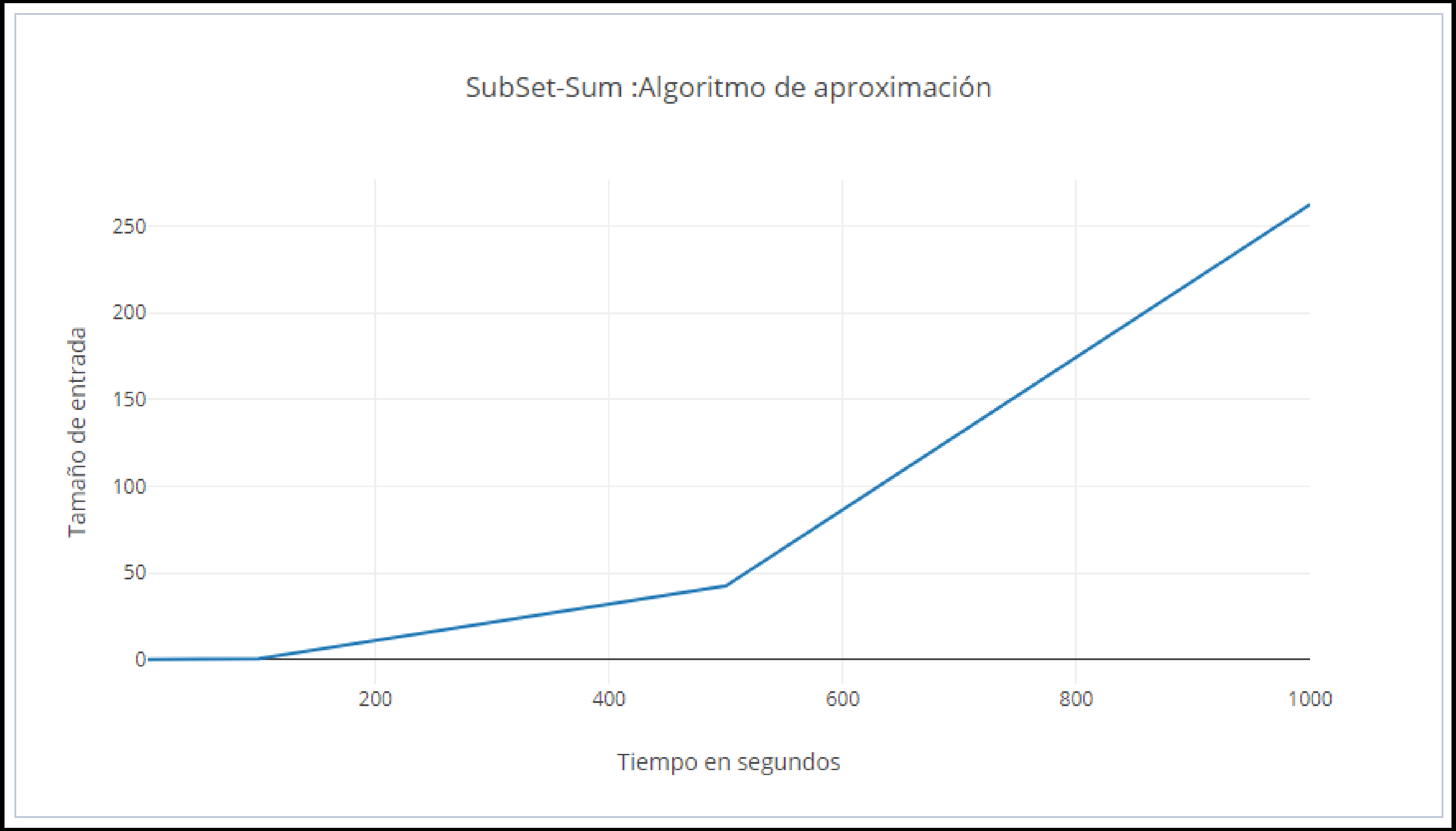
RESULTADOS ALGORITMO DE APROXIMACIÓN

SOLVE

Tamaño de entrada n	Conjunto Potencia 2^n	Algoritmo de Aproximación (segundos)	Algoritmo Clásico (segundos)	Diferencia
5	32	0.002146	0.001871	- 87.18%
10	1024	0.002518	0.004472	+177.60%
20	1048576	0.009648	1.895628	+196.47%
25	33554432	0.014034	75.07551	+5349.54%
26	67108864	0.015596	162.88581	+10444.07%
27	134217728	0.016705	331.98794	+19873.56%
28	268435456	0.017638	**691.35722**	+39197.03%
30	2^{30}	0.021640	Intratable	
40	2^{40}	0.034925	Intratable	
50	2^{50}	0.079997	Intratable	
100	2^{100}	0.481855	Intratable	
500	2^{500}	42.44653	Intratable	
1000	2^{1000}	**262.5024**	Intratable	
10000	2^{10000}	Intratable	Intratable	

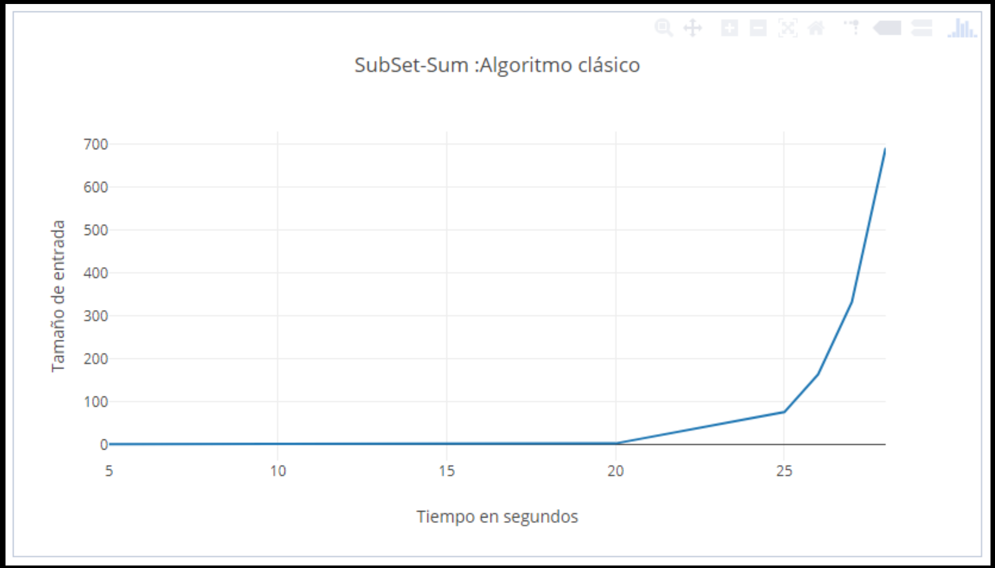
FUNCIÓN DE CRECIMIENTO – AA

PLOT



COMPARATIVA – FUNCIÓN DE CRECIMIENTO

PLOT



IN DEEP – MEMOIZATION RECURSIVO

ADVANCED

Tamaño de entrada n	Conjunto Potencia 2^n	AA (Cormen)	Algoritmo Clásico (segundos)	Memoization
28	**268435456**	**0.017638**	**691.35722**	***0.02686***
30	2^{30}	0.021640	Intratable	*0.024980*
40	2^{40}	0.034925	Intratable	*0.045571*
50	2^{50}	0.079997	Intratable	*0.982519*
100	2^{100}	0.481855	Intratable	***8.926779***
500	2^{500}	42.44653	Intratable	*MemoryError*
1000	2^{1000}	**262.5024**	Intratable	*MemoryError*
10000	2^{10000}	Intratable	Intratable	*MemoryError*

DEMO #3.

DEMO

IN DEEP – AA (Koiliaris & Xu)

ADVANCED

Koiliaris, Konstantinos; Xu Chao - A Faster Pseudopolynomial Time Algorithm for Subset Sum 2015 arxiv.

Tamaño de entrada n	AA (Cormen)	Algoritmo Clásico (segundos)	Memoization	Koiliaris
28	0.017638	**691.35722**	0.02686	
30	0.021640	Intratable	0.024980	
40	0.034925	Intratable	0.045571	
50	0.079997	Intratable	0.982519	*0.000219*
100	0.481855	Intratable	8.926779	*0.000421*
500	42.44653	Intratable	MemoryError	*10.95765*
1000	**262.5024**	**Intratable**	**MemoryError**	**182.7596**
5000	Intratable	Intratable	MemoryError	Intratable
10000	Intratable	Intratable	MemoryError	Intratable

DEMO #4.

DEMO

End!

ADVANCED

→