Universidad Nacional Autónoma de México.

Facultad de Estudios Superiores
"Aragón"

Ingeniería en Computación

Tarea 5
Problema Knapsack
Complejidad Algoritmica

Lara Martínez Christian Gael

1507

31 de Octubre del 2024

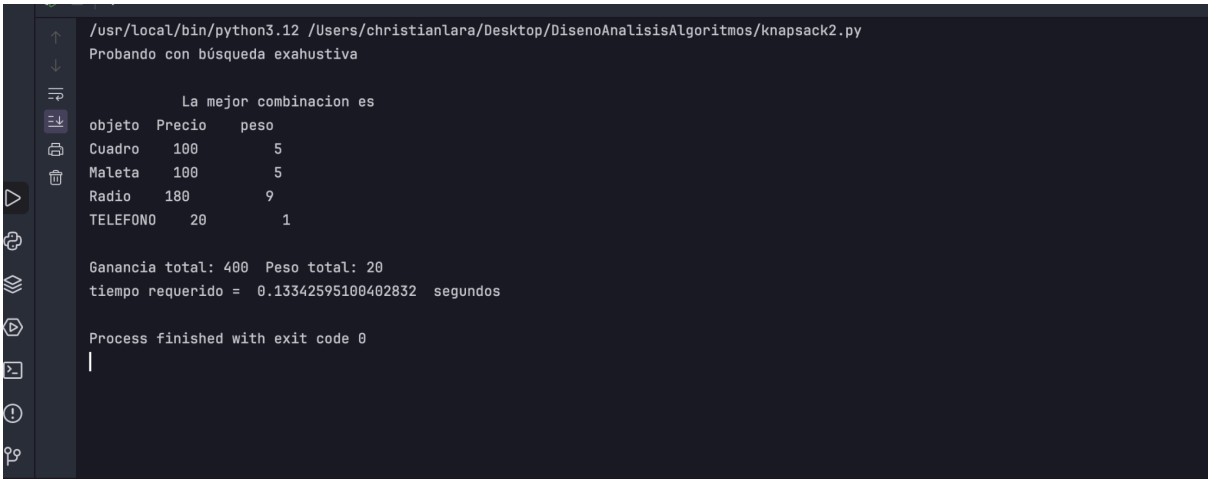
Diseño y Analísis de Algoritmos

Tarea 5, problema Knapsack

```python
# -*- coding: cp1252 -*-
import time
17 usages
class Cosas():
    nombre = ""
    precio = 0
    peso = 0
    llevado = 0
    valor = 0.0
##  El valor será obtenido dividiendo el precio entre el peso
    def __init__(self, nom, pre, pes, llev):
        self.nombre = nom
        self.precio = pre
        self.peso = pes
        self.llevado = llev
        self.valor = pre/pes

1 usage
def evalua(num, arregloCosas, restriccion):
    pos=0
    ganancia=0
    peso=0
    cad = bin(num)
    cad = cad[2:]
    cad = cad.rjust(len(arregloCosas))
    cad = cad.replace( __old: " ", __new: "0")
    for l in cad:
        if l == "1":
            ganancia += arregloCosas[pos].precio
            peso += arregloCosas[pos].peso
        pos+=1
    if peso > restriccion:
        return -1
    else:
        return ganancia

1 usage
def imprimeCombinacion(num):
    pos=0
    ganancia=0
    peso=0
    cad = bin(num)
    cad = cad[2:]
    cad = cad.rjust(len(arregloCosas))
    cad = cad.replace( __old: " ", __new: "0")
    print()
    print ("           La mejor combinacion es   ")
    print ("objeto  Precio    peso")
    for l in cad:
        if l == "1":
            texto =arregloCosas[pos].nombre + "    "
            texto = texto + str(arregloCosas[pos].precio)+"         "
            texto = texto + str(arregloCosas[pos].peso)
            print (texto)
            ganancia += arregloCosas[pos].precio
            peso += arregloCosas[pos].peso
```

```
        pos+=1
    print ()
    print ("Ganancia total: "+ str(ganancia)+ "  Peso total: "+str(peso))

arregloCosas=[]

arregloCosas.append(Cosas( nom: 'cuaderno', pre: 20, pes: 2, llev: 0))
arregloCosas.append(Cosas( nom: 'libro', pre: 10, pes: 1, llev: 0))
arregloCosas.append(Cosas( nom: 'Licuadora', pre: 100, pes: 10, llev: 0))
arregloCosas.append(Cosas( nom: 'Pintura', pre: 200, pes: 18, llev: 0))
arregloCosas.append(Cosas( nom: 'STEREO', pre: 150, pes: 13, llev: 0))
arregloCosas.append(Cosas( nom: 'computadora', pre: 200, pes: 16, llev: 0))
arregloCosas.append(Cosas( nom: 'Microondas', pre: 200, pes: 15, llev: 0))
arregloCosas.append(Cosas( nom: 'sombrilla', pre: 40, pes: 3, llev: 0))
arregloCosas.append(Cosas( nom: 'Calculadora', pre: 103, pes: 6, llev: 0))
arregloCosas.append(Cosas( nom: 'Botas', pre: 89, pes: 5, llev: 0))
arregloCosas.append(Cosas( nom: 'Balon', pre: 54, pes: 3, llev: 0))
arregloCosas.append(Cosas( nom: 'JARRON', pre: 38, pes: 2, llev: 0))
arregloCosas.append(Cosas( nom: 'Cuadro', pre: 100, pes: 5, llev: 0))
arregloCosas.append(Cosas( nom: 'Maleta', pre: 100, pes: 5, llev: 0))
arregloCosas.append(Cosas( nom: 'Radio', pre: 180, pes: 9, llev: 0))
arregloCosas.append(Cosas( nom: 'Silla', pre: 240, pes: 12, llev: 0))
arregloCosas.append(Cosas( nom: 'TELEFONO', pre: 20, pes: 1, llev: 0))
##arregloCosas.append(Cosas('Xbox',369,15,0))
##arregloCosas.append(Cosas('florero',50,2,0))
#arregloCosas.append(Cosas('Luces',50,2,0))
##arregloCosas.append(Cosas('Estereo',340,13,0))
##arregloCosas.append(Cosas('Celular Nokia',28,1,0))
```

```
## Esta parte corresponde a la búsqueda exhaustiva
inicio = time.time()
print ("Probando con búsqueda exahustiva")
maximo = 0
mejorCombina = 0
for c in range(pow(2,len(arregloCosas))):
    obtenido = evalua (c, arregloCosas,  restriccion: 20)
    if obtenido > maximo:
        maximo = obtenido
        mejorCombina = c
imprimeCombinacion(mejorCombina)
fin = time.time()
tiempo = fin-inicio
print("tiempo requerido = ", tiempo, " segundos")
## Fin de la búsqueda exhaustiva
```



Clase Cosas: Define objetos con nombre, precio, peso y calcula su valor (precio/peso).

Función evalua: Calcula la ganancia total de una combinación binaria de objetos (num), sumando precios si no se excede el peso permitido. Devuelve -1 si excede el peso; si no, retorna la ganancia.

Función imprimeCombinacion: Muestra la combinación óptima de objetos con el total de ganancia y peso.

Búsqueda Exhaustiva: Recorre todas las combinaciones posibles, calcula la ganancia de cada una y guarda la mejor (con mayor ganancia sin exceder el peso). Finalmente, imprime la mejor combinación y el tiempo de ejecución.

1ra-Prueba

```
arregloCosas.append(Cosas( nom: 'cuaderno', pre: 20, pes: 2, llev: 0))
arregloCosas.append(Cosas( nom: 'libro', pre: 10, pes: 1, llev: 0))
arregloCosas.append(Cosas( nom: 'Licuadora', pre: 100, pes: 10, llev: 0))
arregloCosas.append(Cosas( nom: 'Pintura', pre: 200, pes: 18, llev: 0))
arregloCosas.append(Cosas( nom: 'STEREO', pre: 150, pes: 13, llev: 0))
arregloCosas.append(Cosas( nom: 'computadora', pre: 200, pes: 16, llev: 0))
arregloCosas.append(Cosas( nom: 'Microondas', pre: 200, pes: 15, llev: 0))
arregloCosas.append(Cosas( nom: 'sombrilla', pre: 40, pes: 3, llev: 0))
arregloCosas.append(Cosas( nom: 'Calculadora', pre: 103, pes: 6, llev: 0))
arregloCosas.append(Cosas( nom: 'Botas', pre: 89, pes: 5, llev: 0))
arregloCosas.append(Cosas( nom: 'Balon', pre: 54, pes: 3, llev: 0))
arregloCosas.append(Cosas( nom: 'JARRON', pre: 38, pes: 2, llev: 0))
arregloCosas.append(Cosas( nom: 'Cuadro', pre: 100, pes: 5, llev: 0))
arregloCosas.append(Cosas( nom: 'Maleta', pre: 100, pes: 5, llev: 0))
arregloCosas.append(Cosas( nom: 'Radio', pre: 180, pes: 9, llev: 0))
arregloCosas.append(Cosas( nom: 'Silla', pre: 240, pes: 12, llev: 0))
arregloCosas.append(Cosas( nom: 'TELEFONO', pre: 20, pes: 1, llev: 0))
arregloCosas.append(Cosas( nom: 'Xbox', pre: 369, pes: 15, llev: 0))
##arregloCosas.append(Cosas('florero',50,2,0))
#arregloCosas.append(Cosas('Luces',50,2,0))
##arregloCosas.append(Cosas('Estereo',340,13,0))
##arregloCosas.append(Cosas('Celular Nokia',28,1,0))
##arregloCosas.append(Cosas('Guitarra',450,16,0))
##arregloCosas.append(Cosas('Wii',346,12,0))
##arregloCosas.append(Cosas('Sombrero',60,2,0))
##arregloCosas.append(Cosas('reloj',110,3,0))
##arregloCosas.append(Cosas('VAJILLA',160,4,0))
#arregloCosas.append(Cosas('Los relojes Blandos. Dalí',250,6,0))
##arregloCosas.append(Cosas('Calendario Maya',456,9,0))
##arregloCosas.append(Cosas('DVD',110,2,0))
#arregloCosas.append(Cosas('Busto Platón',300,10,0))
##arregloCosas.append(Cosas('Adorno',70,3,0))
```

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/k
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
Maleta    100         5
Xbox    369         15

Ganancia total: 469  Peso total: 20
tiempo requerido =  0.2796149253845215  segundos

Process finished with exit code 0
```

2da-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
JARRON    38         2
TELEFONO    20         1
Xbox    369         15
florero    50         2

Ganancia total: 477  Peso total: 20
tiempo requerido =  0.590811014175415  segundos

Process finished with exit code 0
```

3ra-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
TELEFONO    20         1
Xbox    369         15
florero    50         2
Luces    50         2

Ganancia total: 489  Peso total: 20
tiempo requerido =  1.222034215927124  segundos

Process finished with exit code 0
```

4ta-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
JARRON    38         2
TELEFONO    20         1
florero    50         2
Luces    50         2
Estereo    340         13

Ganancia total: 498  Peso total: 20
tiempo requerido =  2.60040283203125  segundos

Process finished with exit code 0
```

5ta-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
JARRON    38         2
florero    50         2
Luces    50         2
Estereo    340         13
Celular Nokia    28         1

Ganancia total: 506  Peso total: 20
tiempo requerido =  5.34655499458313  segundos

Process finished with exit code 0
```

6ta-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
florero    50         2
Luces    50         2
Guitarra    450         16

Ganancia total: 550  Peso total: 20
tiempo requerido =  11.033684968948364  segundos

Process finished with exit code 0
```

7ma-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
florero    50         2
Luces    50         2
Guitarra    450         16

Ganancia total: 550  Peso total: 20
tiempo requerido =  23.396296977996826  segundos

Process finished with exit code 0
```

8va-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
Luces    50         2
Guitarra    450         16
Sombrero    60         2

Ganancia total: 560  Peso total: 20
tiempo requerido =  47.29406476020813  segundos

Process finished with exit code 0
```

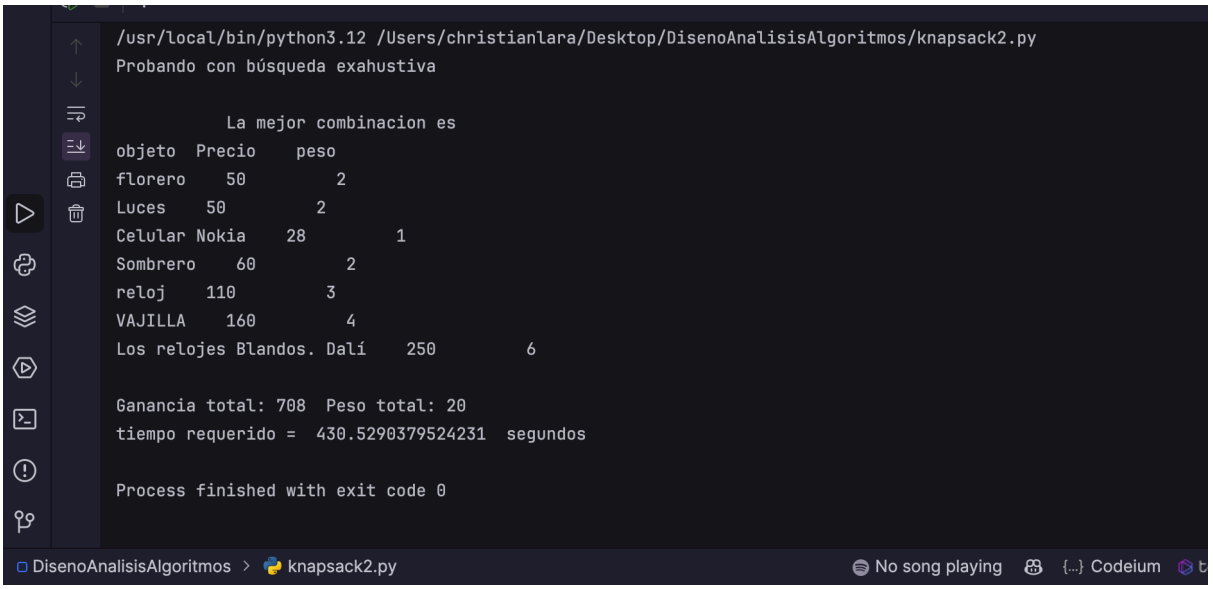
9na-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
Luces    50         2
Celular Nokia    28         1
Wii    346         12
Sombrero    60         2
reloj    110         3

Ganancia total: 594  Peso total: 20
tiempo requerido =  99.68637108802795  segundos

Process finished with exit code 0
```

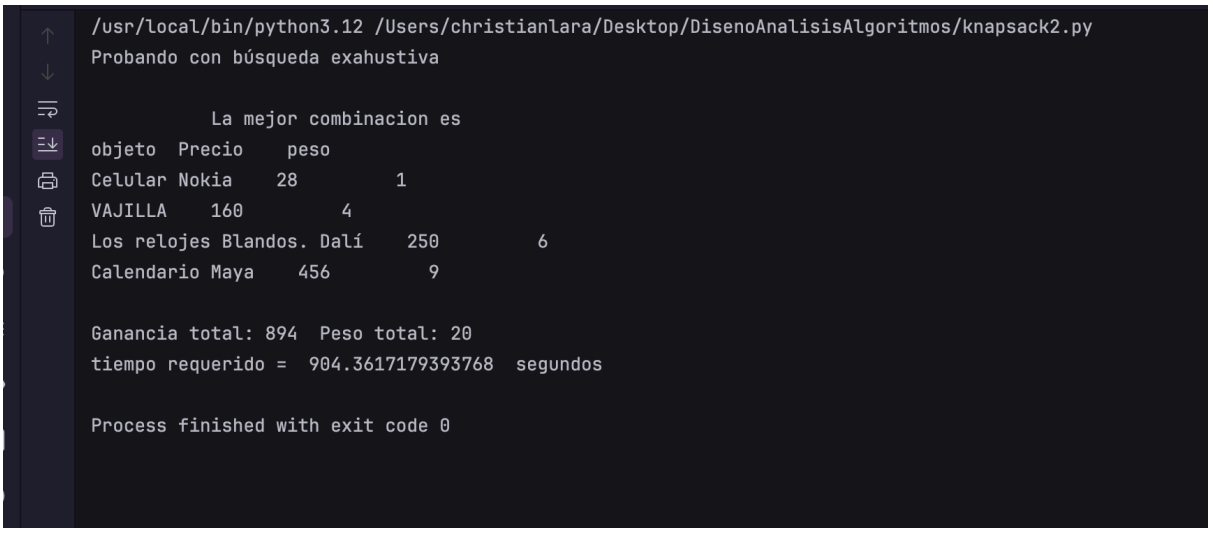
10ma-Prueba

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/DisenoAnalisisAlgoritmos/knapsack2.py
Probando con búsqueda exahustiva

          La mejor combinacion es
objeto  Precio    peso
Celular Nokia    28         1
Wii    346         12
reloj    110         3
VAJILLA    160         4

Ganancia total: 644  Peso total: 20
tiempo requerido =  205.05418801307678  segundos

Process finished with exit code 0
```

11va-Prueba

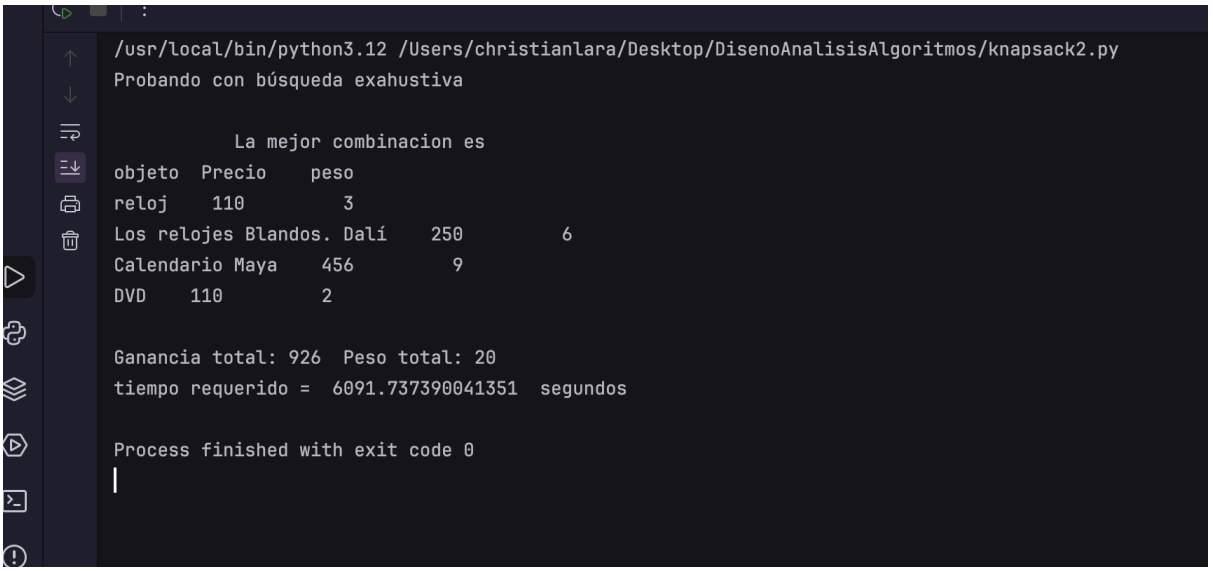


12va-Prueba

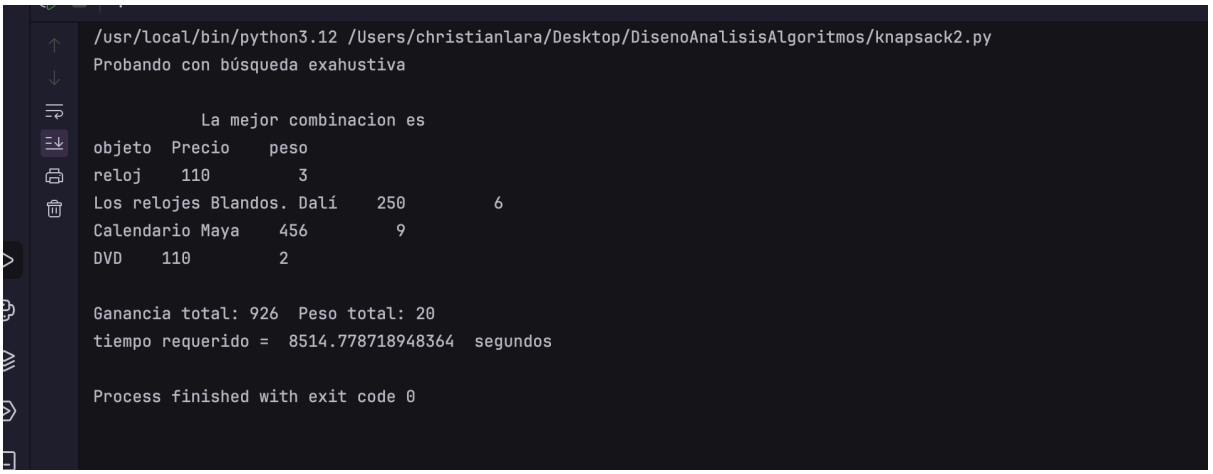


13va-Prueba

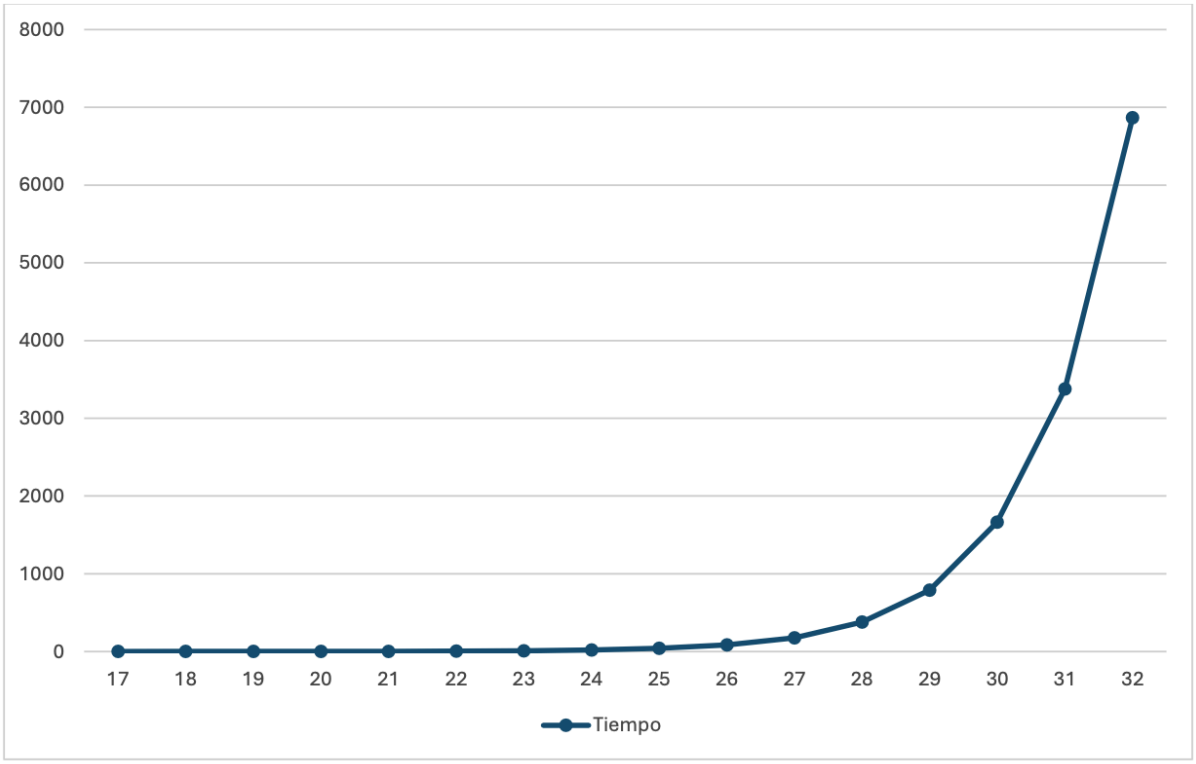
14va-prueba

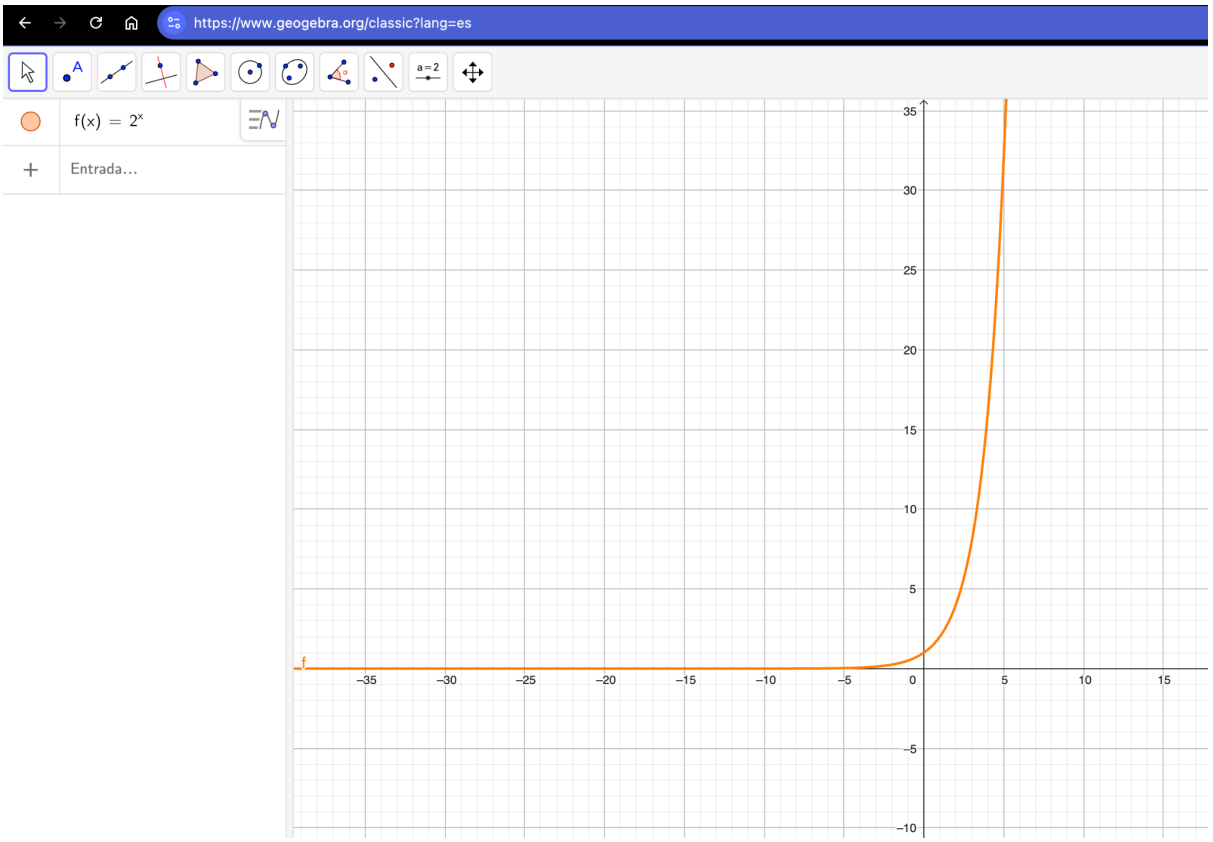


15va - Prueba



Gráficas::


8000
7000
6000
5000
4000
3000
2000
1000
0
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Tiempo


https://www.geogebra.org/classic?lang=es
f(x) = 2^x
Entrada...
35
30
25
20
15
10
5
-5
-10
-35 -30 -25 -20 -15 -10 -5 0 5 10 15

Búsqueda exhaustiva:

El código recorre todas las posibles combinaciones de los elementos. Para cada combinación, usa evalua para calcular la ganancia y el peso. Si la combinación es válida y tiene mayor ganancia que la anterior mejor, se actualiza la mejor combinación.

Complejidad Algorítmica

Complejidad temporal: La complejidad de esta solución es O(2n)O(2^n)O(2n), donde nnn es el número de elementos en arregloCosas. Este crecimiento exponencial se debe a que revisa todas las combinaciones posibles de elementos.

Complejidad espacial: El uso de espacio es O(n) O(n) O(n) debido al almacenamiento temporal de cada combinación y de la mejor combinación.

Reflexión sobre el Tiempo de Compilación

El tiempo de compilación es insignificante en comparación con el tiempo de ejecución debido a la cantidad de combinaciones a evaluar.

Para entradas con más elementos, el tiempo de ejecución crecerá exponencialmente, lo que hace que este enfoque sea ineficiente.