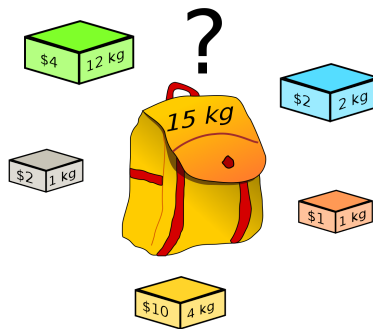




PROYECTO FINAL ALGORÍTMICA I

PROBLEMA DE LA MOCHILA



Integrantes:

- Dylan Chambi Frontanilla
- Samuel Matias Escobar Bejarano
- Fernando Albert Alvarado Lanza

Materia: Algoritmica 1

Docente: Paul Wilker Landaeta Flores

2021

1. Proceso de Instalación

Requisitos de Software:

- Sistema Operativo Windows, Mac o Linux
- IDE para Python 3.9.5
- IDL para Python 3.9.5
- Git Bash

Instalación Paso a Paso:

- Crearse cuenta en GitHub y ejecutar Git Bash
- Usar los comandos en Git Bash: (**git config --global user.name “nombre”**) para registrar el usuario y (**git config --global user.email “email”**) para el email
- Definir una ubicación donde se guardara los archivos del programa usando el comando (**cd “ruta de ubicación”**)
(reemplazar todo backslash por slash)
- Clonar el repositorio con el comando (**git clone**
<https://github.com/SamuelEscobar761/proyectoMochila.git>)
- Ejecutar el archivo “proyectoMochila.py” en uno de los entornos de desarrollo interactivo para Python.

2. Definición del Problema

Nuestro proyecto busca ayudar a cualquier tipo de almacén o persona que busca conseguir la mayor ganancia posible para su negocio en base a sus recursos: (presupuestos y espacio de almacén) para que de esta forma toda producción sea más eficiente y se minimicen las posibilidades de tener pérdidas para lo cual usaremos como base el algoritmo del problema de la mochila.

3. Explicación de algoritmo

Nuestro algoritmo requiere los siguientes datos proporcionados por el usuario:

- Cantidad de productos que se van a comparar
- Los nombres de dichos productos
- Espacio disponible en el almacén para cada producto
- Los costos de producción para cada producto
- La ganancia neta generada por el producto (Ganancia mensual, anual y diaria)

En base a estos requerimientos nuestro algoritmo no solo analiza la ganancia máxima por cada producto como si solo existiera uno sino que también analiza la posible multiplicidad de cada uno de los productos teniendo en cuenta un orden prioritario de las ganancias generadas por cada uno de estos, así mismo nuestro algoritmo toma en cuenta los límites proporcionados por el usuario de cada producto para calcular la ganancia máxima que se puede tener. Finalmente nuestro algoritmo hace una impresión de la cantidad de los diferentes productos que se deben producir así como su costo total y la ganancia máxima que se va a alcanzar con dicha producción.

Algoritmo de la mochila:







Para este proyecto hemos partido desde la base del algoritmo de la Mochila. La historia para este algoritmo parte de la situación en la que un mochilero quería llevar en su mochila la mayor suma de valor de los objetos que tenía en posesión sin sobrepasar un peso máximo. Para resolver este problema se creó el

algoritmo de la mochila, el cual simula múltiples mochilas que tienen como peso máximo de 1 hasta el peso máximo de la mochila original por cada objeto de valor que tiene el mochilero. Este algoritmo es de tipo Dynamic programming ya que, al poner el peso máximo de cada mochila simulada en relación a cada objeto de valor que tiene el mochilero se genera una matriz para la cual en cada línea se repite el valor total que podemos meter en la mochila en los índices de 0 a “peso del objeto de valor-1”. A partir de ahí recién se compara en cada casilla de la matriz qué pasaría con el valor total que cargamos, si metemos en la mochila un nuevo objeto de valor a lo que pasaría si mantenemos la mochila con los objetos anteriores al nuevo (el valor que se encuentra en la línea de arriba de la matriz). Finalmente, obtendremos la solución final en la última casilla de la matriz (en la última línea y la última columna).

Aplicación del algoritmo al proyecto:

Como ya lo mencionamos, el algoritmo de la mochila es la base de nuestro proyecto, por lo cual en lugar de considerar un valor de objeto, consideramos la ganancia neta que se obtiene al vender el producto, en lugar del peso del producto tomamos en cuenta el costo de producción de este producto y en lugar de un peso límite consideramos un presupuesto que se tiene para realizar una inversión. Así mismo, como tomamos en consideración el almacenamiento límite que se tiene para cada producto (m) analizamos cada línea de la matriz generada por el algoritmo de la mochila “ m ” veces ya que simulamos que tenemos una cantidad de 1 a “ m ” productos para producir. De esta forma llegamos a analizar toda combinación posible hasta llegar a los límites de almacenamiento destinados a cada producto.

Complejidad del algoritmo:

```
def contarGananciaLimite(PresupuestoActual):  
    for i in range(len(costProd)):   $n$   
        pesoActual = costProd[i]  
        valorActual = ganancias[i]  
        for x in range(cantidadMaxima[i]):   $c$   
            for j in range(PresupuestoActual, pesoActual-1, -1):   $p$   
                if cantObjetos[j-pesoActual]+ valorActual > cantObjetos[j]:   $2$   
                    for k in range(len(cantDiferentesPesos[j])):   $n$   
                        cantDiferentesPesos[j][costProd[k]] = cantDiferentesPesos[j-pesoActual][costProd[k]]   $2$   
                        cantDiferentesPesos[j][pesoActual] = cantDiferentesPesos[j][pesoActual]+1  $2$   
                        cantObjetos[j] = cantObjetos[j-pesoActual]+ valorActual  $3$ 
```

$$T(n) = ((2n + 7)cp + 2)n = 2n + 7ncp + 2n^2cp$$

$$O(n) = n^2cp$$

Donde:

n = El número de productos

p = El presupuesto

c = La cantidad máxima reservada en el almacen para cada producto

4. Conclusión

En conclusión nuestro proyecto cumple con la función de resolver nuestro problema no está demás decir que los datos que proporciona el usuario para que funcione el algoritmo son muy importantes para un mejor provecho del programa, tomando esto en cuenta el algoritmo si encuentra los datos para conseguir el mayor beneficio para una compania, empresa o emprendimiento de comercio de productos.

5.Bibliografía

-Libro: Guide to Competitive Programming Learning and Improving Algorithms
Through Contests Authors: Laaksonen, Antti
Capitulo 6.2.3