Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas

Producto Integrador de Aprendizaje

Manual de Usuario

Nombre de los integrantes:

Miguel Alejandro Rodriguez Rocha 1854339

Humberto Gerardo Peña Páez 1862464

Osvaldo Delgado Flores 1665567

Andrés Isaac Montes Bartolo 1854017

Luis Ángel Martínez Trejo 1941437

Jordan Eduardo Garcia Pecina 1719825

Maestro: Victor Manuel Sanchez Aguiñaga

Materia: Programación Lineal

Algoritmo 1 - Múltiples Mochilas

- Instalar

  Posicionarse en la carpeta del programa y en una consola de comandos ejecutar la siguiente línea (requisito tener instalado python)

  pip install pandas math

- Agregar datos

  Para agregar datos, hay que modificar el archivo de “data.csv” donde se encuentran dos tablas, una la cual hace referencia a los artículos y otra la cual hace referencia a las mochilas

  Ejemplo de entrada

	A	B	C	D	E	F
1	Nombre	Beneficio	Peso		Mochila	Capacidad
2	Matilda	100	150		Azul	300
3	Escopeta W-	200	280		Roja	240
4	Lightning Ha	150	160		Verde	320
5	Lanzallamas	270	320			
6	Lanzacohete	300	300			
7	M19	230	200			

  para insertar artículos se deberán agregar una fila con los datos del artículo de la siguiente manera

  | Nombre | Beneficio | Peso |

  Donde la columna nombre, es el nombre del artículo, beneficio es el valor que aporta el artículo, y el peso es la carga que ocupa.

  para insertar una mochila deberá llenar una fila con los datos de esta en la tabla de la izquierda de la siguiente forma

  | Mochila | Capacidad |

  donde la columna de mochila puede tener cualquier valor 1, 2, ‘Roja’, ‘Azul’ para identificarla, y la columna capacidad es el peso soportado por la mochila

- Ejecutar el algoritmo

  Para esto solo será necesario abrir algun IDE o programa que permita la ejecución de python y ejecutarlo, el programa leerá el archivo y automáticamente se imprimirá la solución, mostrando los artículos que entran en cada mochila

- Explicación del código

  Para realizar el algoritmo, se realiza primeramente la lectura de los archivos y se guardan en dos variables que guardan a la tabla de artículos y mochilas respectivamente, seguidamente para evitar problemas de lectura de filas de más se ejecutan dos ciclos que cuentan la cantidad de filas existentes de cada tabla y se recortan.

  Seguidamente se agregan dos columnas extras a los artículos, ‘X’ y ‘Ratio’, la primera guardará las mochilas en que se deben guardar, por lo tanto iniciara en 0’s, y la segunda guardará la razón de beneficio entre peso para cada artículo, luego esta tabla (articulos) será ordenada de manera descendente con respecto a la columna ‘Ratio’, esto para tener primero al artículo que más aporte, igualmente la tabla de mochilas será ordenada de manera descendente con respecto a la capacidad de cada una.

  aquí se agregan dos variables ‘peso_total’ y ‘beneficio_total’ iniciadas en cero usadas para almacenar el peso total, y beneficio total respectivamente, las cuales se usarán al imprimir los resultados.

  Luego para cada mochila se ejecuta el el algoritmo que llenaría una sola, este es:

  Se crea una variable ‘peso’ iniciada en cero la cual guarda el peso hasta el momento de los artículos seleccionados.

  Para cada artículo, si el artículo no ha sido asignado a una mochila previamente y si la suma del peso del artículo más la variable peso no sobrepasa la capacidad de la mochila, se inserta y en la tabla de artículos en la columna ‘X’ se asigna el nombre de la mochila y al peso se le suma el peso del artículo insertado, igualmente a las variables ‘peso_total’ y ‘beneficio_total’ se les suma el peso y beneficio del nuevo artículo.

Para finalizar se manda a imprimir los resultados de la siguiente forma:

Para cada mochila:

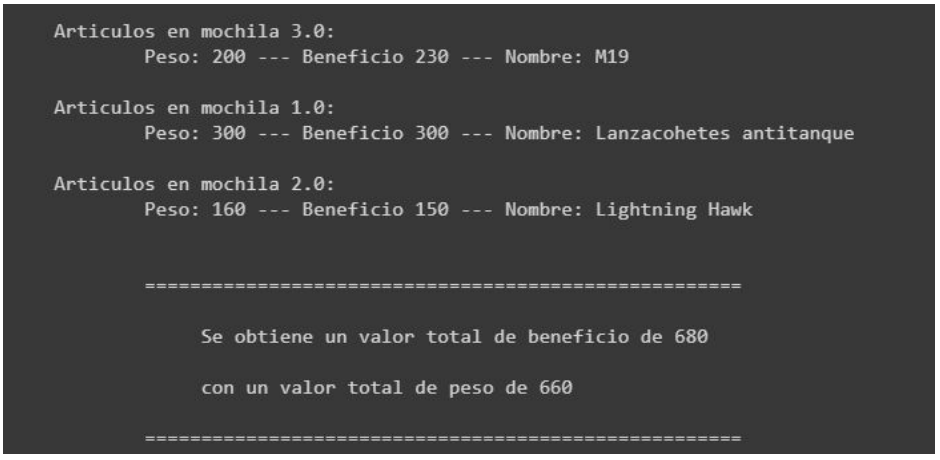
Mochila {nombre de la mochila}:

> Para cada artículo:
>
> Peso: {Peso del artículo} - Beneficio {Beneficio del artículo} - Nombre: {Nombre del artículo}

Se obtiene un valor total de beneficio de {beneficio_total}

con un valor total de peso de {peso_total}

Ejemplo de salida



El resultado obtenido sería la mejor forma de llevar los artículos en las distintas mochilas maximizando el beneficio y minimizando el peso total de carga total

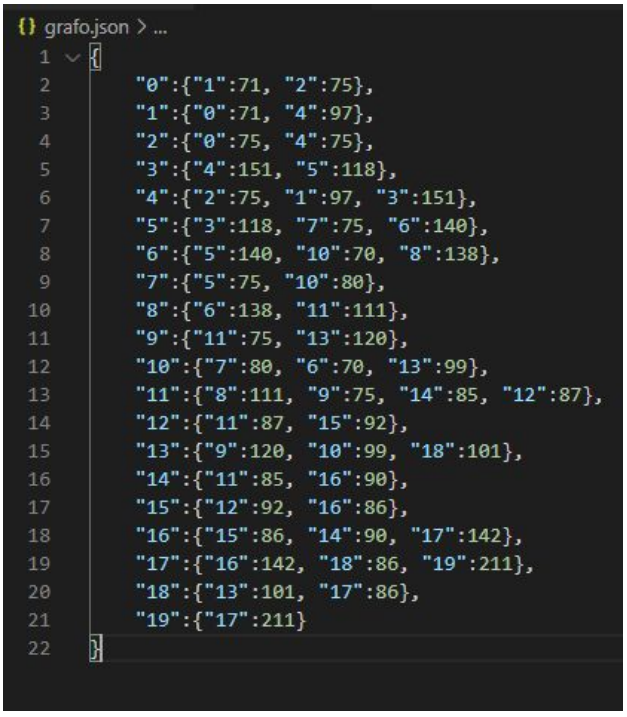
Algoritmo 2 - Agente Viajero

Introducción al algoritmo:
Para resolver el problema del agente viajero había distintos algoritmos para su solución, en este caso utilizamos como base el algoritmo *DFS*(búsqueda en profundidad) el cual nos permite recorrer todos los nodos de un grafo sin regresar al anterior, adaptando dicho algoritmo para guiarse por el vecino con el menor valor.

El algoritmo consiste en escoger algún vértice que será nuestro *inicio*, otro vértice que será nuestro *destino*, al igual tendremos una lista para los vértices *visitados*, otra lista *stack* que nos servirá para colocar los nodos adyacentes al vértice que estamos visitando y por último una lista que nos permite tener registrado el *camino* que nos lleva hacía el *destino*.

Lo que se hará básicamente será inicializar con el nodo que colocamos en *inicio* nuestro *stack* para después pasarlo a *visitados* y así ir recorriendo cada uno de los nodos, el programa terminará cuando el *stack* ya no tenga más nodos en el tope y acabaremos con el camino más corto en nuestra lista *camino*.

Introducción al usuario:
El programa trabaja mediante dos archivos, **“agenteviajerodfs.py”** y **“grafo.json”**. Por practicidad estaremos trabajando con un archivo JSON, dentro de este indicaremos la forma de nuestro grafo.

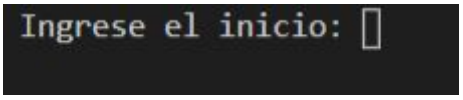


Siendo los caracteres del “0” al “19” nuestros vértices y vamos a añadir nuestros nodos adyacentes a nuestro vértice junto con un valor para indicar su distancia. Ejemplo:

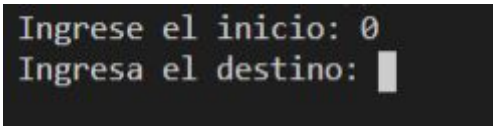
“0” es nuestro vertice 0.
{“1”:71**, “2”:**75**}** nuestro vértice 0 es adyacente a el nodo **1** y hay una distancia de 71 hasta ese nodo. Al igual que el nodo **2**, es adyacente al vértice 0 y existe una distancia de 75.

Una vez entendido cómo leemos los grafos podemos salvar nuestro archivo y ejecutar nuestro programa **“agenteviajerodfs.py”**.

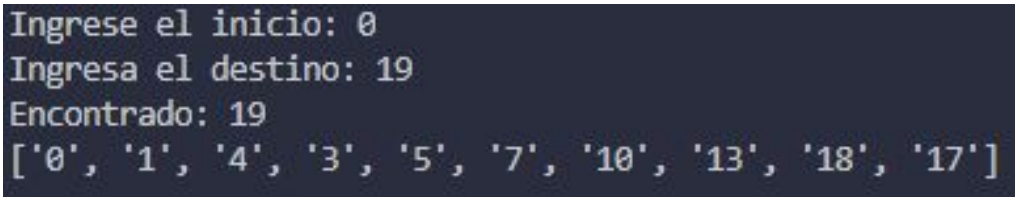
Al ejecutar nuestro programa se nos aparecerá un mensaje por consola para ingresar el nodo desde donde empezaremos(en nuestro caso 0).



Ahora ingresamos al nodo destino(en nuestro caso 19).



Al introducir nuestros datos nos darán cómo resultado el camino más corto hacía el 19.



Si no se obtiene un resultado, no existe un camino óptimo o compatible con el algoritmo. De igual manera los caminos obtenidos pueden no ser el óptimo ya que son basados en el vecino con el menor valor.

Algoritmo 3 - Compañeros de Habitación

Carga de datos

Para cargar las instancias a comprobar con el algoritmo, hay que modificar el archivo adjunto llamado *preferences.csv* creando en él una tabla de dimensiones ***n*** filas y ***n-1*** columnas. La columna donde se le asigna la preferencia de la persona es el index de la tabla, este no se agrega, se da por hecho dependiendo el número de la fila.
En este problema suponemos que la tabla de preferencias está correctamente escrita, es decir, en cada fila independiente no se repite ningún dato.
Otra manera de agregar los datos a la instancia, es abrir el archivo *preferences.csv* con bloc de notas e ingresar los datos separados por una coma.

Importante

- Solo se deben ingresar **números** a la tabla, siendo estos ***n*** números diferentes.
- El número de datos debe ser número **par** y el índice debe iniciar con el número 0.
- El programa puede presentar un error y no dar ningún resultado ya que, en base a esas preferencias, no hay una solución estable.

Algunos ejemplos de instancias aceptadas para este algoritmo:

2,1,5,3,4	3,1,4,2,5	2,3,1,5,4
5,4,3,0,2	2,4,3,0,5	5,4,3,0,2
0,3,4,1,5	0,3,4,1,5	1,3,4,0,5
4,1,2,5,0	0,1,2,5,4	4,1,2,5,0
2,0,1,3,5	2,0,1,3,5	2,0,1,3,5
4,0,2,3,1	3,0,2,4,1	4,0,2,3,1

Funcionamiento

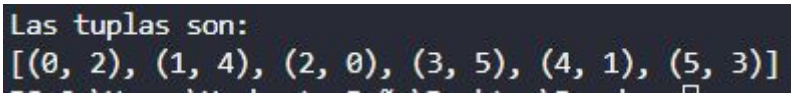
Es importante, antes de compilar y ejecutar el programa, abrir consola, posicionarte en la carpeta donde se encuentra el archivo y escribir el siguiente comando:

pip install pandas

Esto, para instalar el módulo necesario para que el programa pueda ejecutarse sin algún error a la hora de leer el csv.

Interpretación de resultados

Una vez compilado y ejecutado el archivo **‘compañeros.py**’, tomará los datos de preferences.csv y nos mostrará el mejor acomodo de las habitaciones en esta forma:



Son las tuplas, que de acuerdo al algoritmo se definen como la mejor opción para cada persona.

Archivos

Se agregan los distintos archivos:

- compañeros.py. Archivo a compilar y ejecutar, en este se encuentra toda la lógica del algoritmo.
- preferences.csv. Archivo donde se cargan las instancias a comprobar en el programa.
- ejemplos.txt. Archivo donde cargamos ejemplos de instancias que resultaron exitosas.