7. Anexos

7.1. Anexo 1: Archivos y tutorial de uso

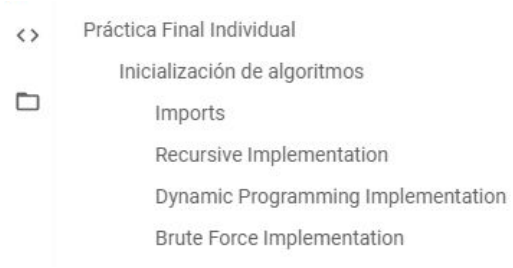
La entrega contiene los diferentes archivos:

- **execution_time**: carpeta en la que se encuentran los diferentes archivos .csv de los datos de la ejecución de los los tests realizados.
- **img**: aquí están las imágenes que se han usado en la memoria (excepto los gráficos).
- **Code.ipynb**: cuadernillo donde se implementan los algoritmos y se ejecutan las pruebas. Enlace del cuadernillo en Google Colab (solo accesible con un correo de la UC3M).
- **Experimental_vs_Analítico.xlsx**: archivo excel donde se encuentran las gráficas implementadas en la memoria con todos sus datos transformados.
- **memoria.pdf**: memoria explicando el problema, las implementaciones, costes y resultados.

7.1.1. Code.ipynb

7.1.1.1. Inicialización de los algoritmos

En esta parte se añaden los imports necesarios y las 3 diferentes implementaciones que usamos: recursiva, programación dinámica y fuerza bruta.



En la implementación de fuerza bruta se puede quitar unos comentarios que hay en el código para conseguir que el algoritmo te devuelva la tupla que maximiza los valores:

Descomentamos *best_tuple* del inicio de la función.

```
max_value = 0
# best_tuple = ()
```

Comentamos `max_value = max(current_value, max_value)`, `return max_value` y descomentamos el código comentado que aparece en la imagen:

```
  max_value = max(current_value, max_value)

  '''
  # Tuple return modify (you have to comment the previous line & uncomment 'best_tuple' inicialize)
  if (max_value < current_value):
    max_value = current_value
    best_tuple = comb_tuple

print("Combination: ")
print(best_tuple)
  '''

return max_value
```

7.1.1.2. Pruebas

- **Pruebas**
  - Test #0:
  - Resto de tests
    - Inicialización de variables globales
    - Test #1:
      - Ejecución de todos los algoritmos
        - Recursive implementation
        - Recursive implementation (worst case)
        - Dynamic programming implementation
        - Brute Force implementation
      - Post-procesado de datos
      - Gráficos
    - Test #2:
      - Ejecución de todos los algoritmos
        - Recursive implementation
        - Dynamic programming implementation
        - Brute Force implementation
      - Post-procesado de datos
      - Gráficos
    - Comparación de tests
    - Exportar datos
  - Test #3
    - Ejecución del algoritmo
    - Post-procesado
    - Gráfico
    - Exportar datos

El *Test #0* es una ejecución básica de las implementaciones donde se concentra todo.

El *resto de tests* tienen una estructura diferente. Al comienzo se inicializan unas variables que afectan a los tests *#1* y *#2*, que hacen más fácil el mantenimiento del código.

Después se ejecutan los diferentes algoritmos de los tests, se procesan (en función de la cantidad de tests que hemos hecho, y se saca unos gráficos orientativos acerca de la ejecución.

Después se sacan gráficos con todos los datos de los tests *#1* y *#2* para ver la diferencia entre ejecuciones.

Existe un apartado de *Exportar datos*, donde se pasan todos los datos a diferentes archivos .csv para su posterior análisis en una herramienta externa que nos permita más precisión.

El *Test #3* tiene los mismos apartados que los tests *#1* y *#2*.

Se ha decidido no pasar el cuadernillo a pdf debido a que no se ven las ejecuciones de los diferentes módulos.

7.1.1.3. Uso

El uso es muy sencillo, se conecta a Google Colab o se inicia un entorno Jupyter y se carga en función de los módulos que se quieran ejecutar y sus dependencias.