PRÁCTICA 1 - AVANCE RÁPIDO

Estudiantes: Pasternak Kateryna y Yagual Mendez Iván Andres

Problema: 04 - boda

DISEÑO DEL ARGORITMO

Nuestro problema a resolver consiste en:

Tenemos un grupo de personas que sin sobrepasar el presupuesto desean comprar los modelos más caros de cada modelo.

Se debe elegir obligatoriamente un modelo por prenda.

El objetivo es maximizar el gasto total dentro del presupuesto y asegurarse de escoger un modelo para cada prenda. Para la resolución se debe hacer uso de un Algoritmo voraz que consiga encontrar los modelos más caros y que cuya suma sea menor o igual al presupuesto, para ello dentro de las posibles modelos obtiene el optimo local siguiendo un criterio intentando encontrar el optimo local, el cual no siempre será posible encontrarlo. Este algoritmo sacrifica la precisión a cambio de eficiencia, ya que no hace una búsqueda.

Diseño del algoritmo y justificación:

El algoritmo recibe como entrada un conjunto - lista de prendas (`garments`), donde cada prenda es representada por una lista de modelos, y cada modelo tiene un precio. Además, se proporcionan el presupuesto máximo disponible (`budget`) y el número de prendas a seleccionar (`num_garments`).

El objetivo del algoritmo es seleccionar un modelo específico para cada prenda. Para evitar tomar decisiones erróneas o sobrecargar el presupuesto demasiado pronto, inicializamos el proceso de selección de prendas basándonos en ciertos factores que asignan pesos a cada `garment`. Estos factores son los siguientes:

1. **Priorizar un rango de precios más pequeño:** para cada prenda, calculamos el precio máximo y el precio mínimo de los modelos disponibles. Priorizamos aquellas prendas cuyo rango de precios (la diferencia entre el modelo más caro y el más barato) sea más pequeño, ya que tienden a tener un número

más limitado de opciones, lo que significa que es más conveniente tratar estas prendas más pronto.

2. **Priorizar un menor número de modelos:** De manera similar, tratamos de priorizar las prendas que tienen menos modelos disponibles. Al hacerlo, nos aseguramos de no quedarnos sin opciones demasiado pronto, lo que podría limitar nuestras elecciones en etapas posteriores.

Proceso de selección:

1. **Selección de la prenda:** Se selecciona la prenda basada en el peso calculado previamente. Esta selección prioriza las prendas con un menor rango de precios y menor número de modelos.
2. **Cálculo del precio máximo posible por prenda:** Se calcula el precio máximo que se puede permitir para la prenda actual `max_price_per_garment`, basándose en el presupuesto restante y el número de prendas que aún deben seleccionarse. Este cálculo asegura que el algoritmo no se exceda en el presupuesto.
3. **Selección del modelo más adecuado:** Con la función `selection`, el algoritmo elige el precio más alto entre los modelos disponibles, siempre que no supere el precio máximo calculado. Si no se puede encontrar un modelo dentro de este rango, el algoritmo opta por el modelo más barato que no sobrepase el presupuesto restante.

Si en cualquier momento el presupuesto disponible se supera, el algoritmo finaliza y devuelve `None`, lo que indica que no es posible seleccionar los modelos dentro del presupuesto. Si todos los modelos han sido seleccionados correctamente, el algoritmo devuelve el gasto total realizado.

Funciones clave del algoritmo:

1. `greedy_selection`**:** Esta es la función principal del algoritmo. Recibe el presupuesto, la lista de prendas con sus precios y el número total de prendas a seleccionar. Utiliza la función `calculate_weights` para asignar pesos a cada prenda, y luego selecciona los modelos más adecuados mediante la función `selection`.
2. `selection`**:** Esta función se encarga de elegir el modelo más adecuado de una prenda, asegurándose de que el precio seleccionado no supere el

presupuesto restante. Si no es posible elegir un modelo dentro del rango de precios permitido, selecciona el modelo más barato posible que permita mantenerse dentro del presupuesto.

3. `calculate_weights`**:** Esta función calcula los pesos de las prendas, utilizando el rango de precios y el número de modelos disponibles. Los pesos son utilizados para ordenar las prendas, de modo que el algoritmo seleccione primero aquellas con menor variabilidad en los precios y con menos modelos disponibles.

Justificación de las decisiones de diseño:

- **Priorización de prendas:** El cálculo del peso, que combina el rango de precios y el número de modelos, permite que el algoritmo priorice las prendas con menos variabilidad en los precios y menos opciones, lo que facilita un selección más eficiente y menos arriesgada.
- **Selección del modelo adecuado:** La función `selection` asegura que el algoritmo no seleccione modelos excesivamente caros, lo que podría comprometer el presupuesto. La lógica de elegir el precio más alto dentro de lo posible maximiza el uso del presupuesto.
- **Eficiencia:** El algoritmo utiliza un enfoque voraz que garantiza la selección del modelo más adecuado en cada paso, maximizando el gasto sin exceder el presupuesto. Esto se logra manteniendo una selección en tiempo eficiente gracias a las comparaciones simples de los precios disponibles y el cálculo del peso.

VALIDACIÓN DEL ALGORITMO

El algoritmo ha sido validado utilizando archivos de entrada que contienen diversas combinaciones de presupuestos y precios de prendas. En este proceso, se evalúa la precisión del cálculo y la calidad de la optimización alcanzada.

La función de validación compara las soluciones generadas por el algoritmo con las soluciones esperadas contenidas en un archivo de salida, devolviendo dos porcentajes clave:

1. **Porcentaje de corrección:** Indica la precisión con la que el algoritmo ha calculado los modelos seleccionados, verificando que coincidan con las soluciones esperadas. Si todos los modelos seleccionados por el algoritmo

coinciden con las soluciones correctas proporcionadas, se alcanza un 100% de corrección.

2. **Porcentaje de optimización:** Evalúa la calidad de las soluciones obtenidas, comparando la suma total de los modelos seleccionados con las soluciones ideales esperadas. Este porcentaje refleja cuán cerca se encuentra el resultado de la solución óptima en términos de gasto total, sin exceder el presupuesto disponible. En nuestros casos de prueba, hemos obtenido un porcentaje de optimización cercano al 98%, lo que indica una aproximación muy alta a las soluciones ideales.

Utilizando los ficheros de entrada y salida proporcionados, hemos obtenido un porcentaje de corrección del **100%** y un porcentaje de optimización cercano al **98%**, lo cual demuestra la precisión y efectividad del algoritmo en la resolución del problema.

ESTUDIO TEORICO Y EXPERIMENTAL

Estimación teórica de la complejidad temporal del algoritmo:

Definición de variables:

- Definimos n como el número de prendas y m como el número promedio de modelos por prenda. El número total de modelos de todas las prendas es $N = n \times m$.

Complejidad de cada función:

- `calculate_weights`: Esta función recorre cada prenda y calcula los valores `max` y `min` en listas de m modelos. Dado que la operación de encontrar el máximo o el mínimo en cada sublista de modelos tiene una complejidad de $O(m)$, la función `calculate_weights` tiene una complejidad total de $O(n \times m)$ para procesar todas las prendas.
- `select_best_weighted_garment`: Esta función encuentra el mayor peso entre n prendas, lo cual tiene una complejidad de $O(n)$. Como el algoritmo selecciona una prenda en cada iteración y esta operación se repite n veces, la complejidad de `select_best_weighted_garment` en el contexto del bucle principal se convierte en $O(n^2)$.

- `selection`: Esta función procesa una lista de precios de m modelos en cada iteración del bucle principal. Dado que `selection` se ejecuta n veces en el bucle principal, su complejidad es $O(n \times m)$.

Complejidad total:

- Sumando las complejidades de todas las funciones dentro del contexto del bucle principal, obtenemos: $O(n \times m) + O(n^2) + O(n \times m) = O(n^2 + n \times m)$

Simplificación en términos de N:

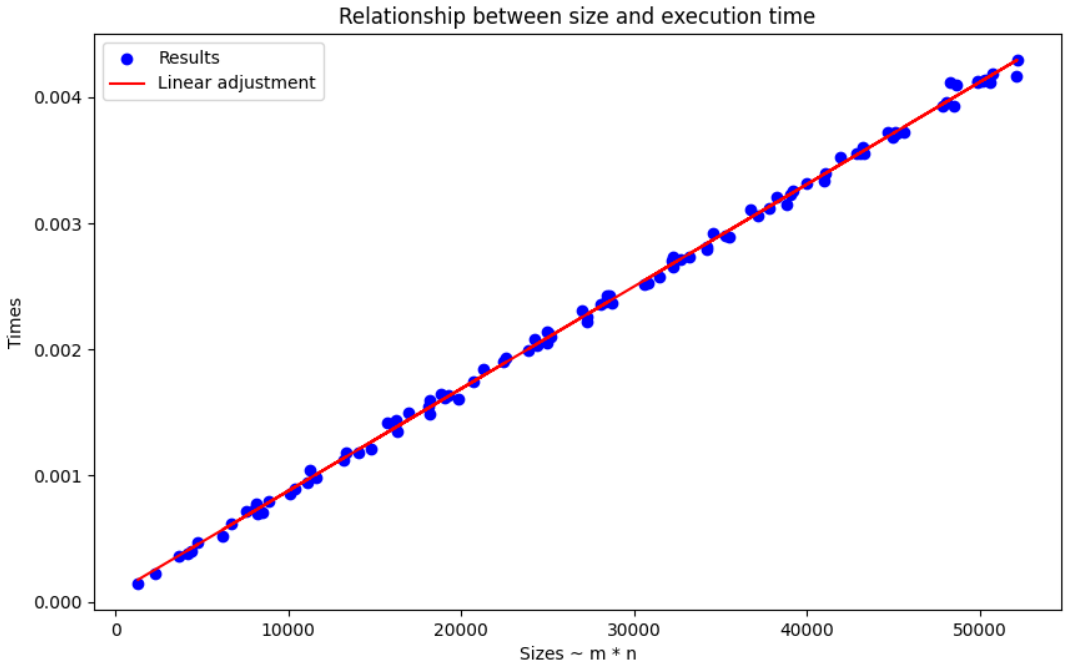
- Como definimos anteriormente, $N = n \times m$, que representa el tamaño total del problema. Así, la complejidad puede expresarse en función de N como: $O\left(\frac{n}{m} \cdot N + N\right)$
- Además, si consideramos que el número de prendas n no supera el número de modelos m, podemos simplificar aún más. Dado que $n \leq m$, tenemos: $\frac{n}{m} \leq 1$
- Bajo esta condición, la complejidad puede reducirse a $O(N)$ en general, donde N es el tamaño total del problema

Conclusión: La complejidad temporal teórica del algoritmo, en términos de su escala con respecto al tamaño del problema, se aproxima a $O(N)$, donde Nes el número total de modelos.

Estudio experimental del tiempo de ejecución del algoritmo:

1. Utilizando la función `generar_fichero_entrada_progresiva` del archivo `create_files.py`, generamos un archivo de prueba llamado *"casos_grandes.in"*, el cual contiene 100 casos con una variedad en el número de prendas entre 5 y 50, y un rango de modelos entre 100 y 1000.
2. Con la función `time_test`, que emplea la biblioteca `time`, calculamos el tiempo de ejecución en segundos para los 100 casos de prueba en el archivo generado. Luego, con `plot_results`, utilizamos la biblioteca `numpy` para realizar un ajuste lineal sobre los datos y `matplotlib` para graficar la dispersión de los tiempos frente a los tamaños de los problemas (es decir, el número total de

modelos en cada caso). La gráfica muestra una relación lineal, lo cual coincide con nuestra estimación teórica.



1. Al considerar tanto la estimación teórica como los resultados experimentales, podemos concluir que el algoritmo es eficiente, dado que su crecimiento en complejidad es de orden lineal.

CONCLUSIÓN

A lo largo de la actividad, se ha desarrollado un algoritmo de selección voraz para resolver el problema de optimización con restricciones de presupuesto. El proceso ha sido fascinante, ya que no solo permitió aplicar conceptos fundamentales de algoritmos y estructuras de datos, sino también analizar su comportamiento en términos de complejidad teórica y realizar una validación experimental.

El estudio teórico nos ha proporcionado una estimación clara de la complejidad temporal del algoritmo, y los resultados experimentales han corroborado esa teoría al mostrar un comportamiento lineal en función del tamaño del problema. Esto demuestra que el algoritmo es eficiente y adecuado para los casos planteados. Sin embargo, algunas discrepancias menores, como la gestión de

memoria o el uso de bibliotecas externas, pueden introducir pequeñas variaciones en los tiempos observados.

A nivel personal, ha sido una experiencia enriquecedora en la que hemos podido poner en práctica nuestras habilidades de programación, análisis y validación de algoritmos. El trabajo en equipo ha sido clave para resolver los desafíos del proyecto, y la colaboración constante nos permitió optimizar el enfoque de solución y alcanzar un resultado satisfactorio.

En cuanto al desarrollo del proyecto, también fue crucial realizar un análisis detallado de las funciones que componen el algoritmo y su relación con el tiempo de ejecución, lo cual es fundamental en problemas de optimización. Además, la comparación entre la estimación teórica y los resultados experimentales refuerza la comprensión de la eficiencia del algoritmo.

Estimación de tiempo en horas:

Estudiante 1: Kateryna

- Desarrollo del algoritmo y pruebas iniciales: 4 horas
- Estudio teórico y análisis de complejidad: 1 hora
- Validación experimental y ajustes de gráficos: 2 horas
- Documentación: 1 hora

  Total: 8 horas

Estudiante 2: Iván

- Mejora del algoritmo: 3 horas
- Diseño de los tests y estudio experimental: 3 horas
- Redacción de conclusiones: 1 hora
- Optimización y pruebas adicionales del algoritmo: 2 horas

  Total: 9 horas