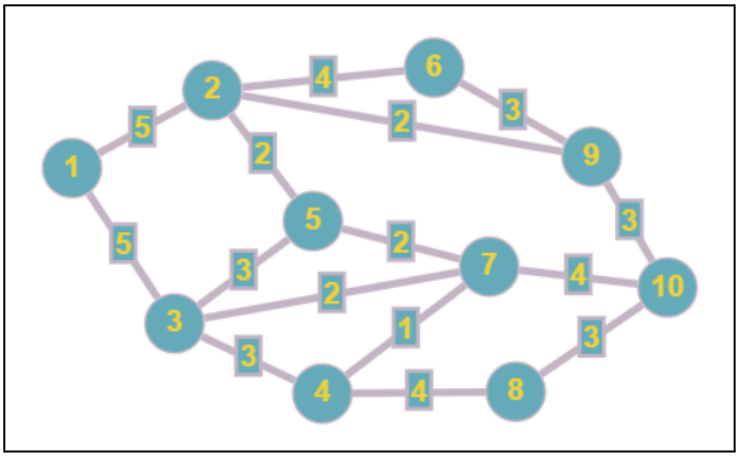
TRABAJO PRÁCTICO 2

PROBLEMA 1 - Programación Lineal



La red de telecomunicaciones de la Facultad se compone de varios nodos y enlaces, cada uno con una capacidad definida. En el diagrama presentamos el esquema general, donde las aristas representan la capacidad medida en MB. Se desea enviar un archivo de 10 MB desde el nodo 1 hasta el nodo 10. El protocolo de comunicación es TCP/IP, así que el archivo puede ser fragmentado en partes más pequeñas en cualquiera de los nodos de la red, y el protocolo mismo se encargará de reconstruirlo en el orden correcto al llegar al nodo 10.

Plantear y resolver un modelo de Programación Lineal que permita determinar cuál será la mejor forma de fragmentar y enviar el archivo utilizando la menor cantidad posible de enlaces.

Se pide:

1. Análisis:
    a. Supuestos. Indicar supuestos, condiciones o premisas bajo las cuales se plantea el modelo.
2. Diseño:
    a. Variables. Definir qué mide cada variable, e indicar si es continua, entera o binaria/indicativa. Indicar sus unidades.
    b. Constantes. Definir cuáles son las constantes utilizadas en el modelo y sus unidades.
3. Modelo de Programación Lineal. Definir y describir la función objetivo y las restricciones que tendrá el modelo. Todas deben ser lineales.
4. Solución:
    a. Desarrollar un programa que resuelva el modelo usando Python y Pulp o un software de programación lineal (Lindo, GLPK, CPLEX, etc.).
    b. Incluir todos los archivos necesarios para la ejecución
    c. Incluir un archivo con el resultado obtenido
5. Informe:
    a. Redactar un informe de la solución, indicando cómo se debe fragmentar y distribuir el archivo

PROBLEMA 2 - Redes de Flujo

Resolver el Problema 1 usando Redes de Flujo.

Se pide:

1. Análisis:
    a. Identificar supuestos, condiciones, limitaciones y/o premisas bajo los cuales funcionará el algoritmo desarrollado
2. Diseño:
    a. Explicación en prosa de cómo se adaptan los datos de entrada a una red de flujo, y de cómo se debe interpretar la salida del algoritmo de Ford-Fulkerson para resolver este problema. Se recomienda incluir diagramas de apoyo.
    b. Incluir un Pseudocódigo del algoritmo a desarrollar (asumir que ya se cuenta con un algoritmo que resuelve Redes de Flujo).
    c. Detallar las estructuras de datos utilizadas. Justificar su elección.
3. Seguimiento: Mostrar un ejemplo de seguimiento con un set de datos reducido
4. Complejidad: Realizar un análisis de la complejidad temporal a partir del pseudocódigo
5. Solución:
    a. Opción 1: resolver manualmente, indicando paso a paso cómo el algoritmo planteado encuentra los caminos de aumento y construye la red residual
    b. Opción 2: Desarrollar un programa que resuelva el modelo usando Python y una biblioteca de Redes de Flujo (propia o de terceros). Incluir todos los archivos necesarios para la ejecución. Incluir un archivo con el resultado obtenido.
6. Informe de Resultados:
    a. Redactar un informe de la solución, indicando cómo se debe fragmentar y distribuir el archivo
    b. Para resolver este problema, ¿es mejor utilizar Programación Lineal o Redes de Flujo? Justificar el criterio utilizado para comparar las dos técnicas.

PROBLEMA 3 - Algoritmos de Aproximación

Tenemos un conjunto de n objetos, donde el tamaño s_i del $i\text{-ésimo}$ objeto cumple que $0<s_i<1$. El objetivo es empaquetar todos los objetos en el mínimo número posible de recipientes de tamaño unitario: cada recipiente puede contener cualquier subconjunto de los objetos cuyo tamaño total no exceda 1. El problema así planteado es NP-Hard.

Se desea encontrar un algoritmo eficiente que encuentre una solución aproximada con una garantía de a lo sumo 2 veces el valor de la solución óptima.

Se pide:

1. Análisis:
    a. Identificar supuestos, condiciones, limitaciones y/o premisas bajo los cuales funcionará el algoritmo desarrollado
    b. Analizar el tamaño del espacio de soluciones factibles del problema
2. Diseño:
    a. Incluir un Pseudocódigo
    b. Mostrar cómo se cumple con la garantía solicitada (se pueden usar citas y referencias)
    c. Detallar las estructuras de datos utilizadas. Justificar su elección.
3. Seguimiento: Mostrar un ejemplo de seguimiento con un set de datos reducido
4. Complejidad: Realizar un análisis de la complejidad temporal a partir del pseudocódigo. Comparar con el tamaño del espacio de soluciones factibles.
5. Sets de datos: diseñar sets de datos apropiados.
    a. Se pueden generar utilizando una función random con una semilla fija, para permitir la reproducibilidad de los resultados, o ser generados externamente e incluidos como archivos que lee el programa.
    b. Cada set de datos debe ser incluido en la entrega, junto con el resultado obtenido en cada caso.
    c. El programa entregado debe generar los sets de datos en tiempo de ejecución o leerlos desde archivos incluidos en la entrega.
6. Tiempos de Ejecución: medir los tiempos de ejecución de cada set de datos y presentarlos en un gráfico.
7. Informe de Resultados:
    a. Redactar un informe de resultados comparando los tiempos de ejecución con la complejidad temporal.
    b. El gráfico comparativo de tiempos debe incluir tanto la curva con los valores medidos como la curva correspondiente a la complejidad temporal determinada.

PROBLEMA 4 - Algoritmos Randomizados

En criptografía, una Prueba de Conocimiento Cero (Zero-Knowledge Proof) es un conjunto de técnicas en el que una parte (el probador) puede convencer a otra parte (el verificador) de que una afirmación dada es verdadera, sin revelar al verificador ninguna información adicional sobre la afirmación (Aad, 2023). El siguiente es un problema que puede ser resuelto mediante una Prueba de Conocimiento Cero:

> *Victor es daltónico y no puede distinguir el rojo y el verde. Peggy tiene dos esferas idénticas excepto en el color: una es roja y la otra es verde. Victor no cree que las dos esferas sean diferentes. Peggy quiere probar a Victor que las dos esferas se pueden distinguir entre sí por el color, sin mostrar cuál es la verde y cuál es la roja.*

Implementar un algoritmo randomizado mediante el cual Peggy pueda demostrar a Victor que puede distinguir las dos esferas usando una Prueba de Conocimiento Cero.

Se pide:

1. Análisis:
    a. Identificar supuestos, condiciones, limitaciones y/o premisas bajo los cuales funcionará el algoritmo desarrollado
    b. Analizar y justificar el cumplimiento de las tres propiedades de las Pruebas de Conocimiento Cero: Completitud, Solidez y Conocimiento Cero
    c. Cuántas veces debería repetirse el algoritmo para que Victor tenga una certeza de al menos el 90% de que Peggy realmente sabe distinguir las dos esferas?
2. Diseño:
    a. Incluir un Pseudocódigo
    b. Detallar las estructuras de datos utilizadas. Justificar su elección.
2. Seguimiento: Mostrar un ejemplo de seguimiento con un set de datos reducido.
3. Complejidad: Realizar un análisis de la complejidad temporal a partir del pseudocódigo.
4. Sets de datos:
    a. Establecer cantidades de repeticiones que permitan ver cómo cambia el grado de certeza a medida que aumenta dicha cantidad
    b. El programa entregado debe generar las repeticiones en tiempo de ejecución o leerlas desde archivos incluidos en la entrega.
5. Grado de Certeza: analizar las repeticiones realizadas y determinar cuál es el grado de certeza alcanzado según la cantidad de repeticiones.
6. Informe de Resultados:
    a. Realizar un gráfico donde se pueda observar la evolución del grado de certeza en función de la cantidad de repeticiones. Sugerencia: usar un gráfico de dispersión.
    b. Redactar un informe de resultados comparando la cantidad de repeticiones con el grado de certeza alcanzado.
    c. ¿Qué aplicaciones prácticas tienen las técnicas de Zero-Knowledge Proof? Describir, citar y referenciar dos aplicaciones relevantes en la actualidad.

Condiciones Generales de Entrega

- El trabajo debe ser entregado en un archivo zip conteniendo:
    - Documento con carátula, índice y numeración de páginas. La carátula debe incluir nombre y padrón de los integrantes del grupo. Debe presentarse en formato PDF.
    - Archivos con los fuentes de los algoritmos desarrollados.
    - Archivo README indicando el lenguaje de programación utilizado, versión mínima y bibliotecas requeridas, e instrucciones para ejecutar.
    - Archivos con los sets de datos utilizados
    - Archivos con resultados obtenidos para cada set de datos
- IMPORTANTE: el tamaño de los sets de datos debe ser tal que permita obtener suficiente información como para graficar los tiempos de ejecución y verificar la complejidad temporal.
- Aquí no toleramos el plagio: Por ello, se pueden incluir citas en el texto del informe siguiendo el modelo propuesto por Rivas (2022) y luego incorporar el listado completo en un anexo al final, usando normas APA 7ma edición.

Referencias

- Aad, I. (2023). *Zero-Knowledge Proof*. En: Mulder, V., Mermoud, A., Lenders, V., Tellenbach, B. (editores) *Trends in Data Protection and Encryption Technologies* . Springer, Cham. https://doi.org/10.1007/978-3-031-33386-6_6
- Rivas, A. (2022). *Cómo hacer una lista de referencias con Normas APA*. Guía Normas APA. https://normasapa.in/como-hacer-la-lista-de-referencias/