EVIDENCIA 3. PÓSTER ARGUMENTATIVO DE REFLEXIÓN

TC2038.651 - Análisis y diseño de algoritmos avanzados

Alumnos: Alejandro Díaz Villagómez - A01276769
Emiliano Saucedo Arriola - A01659258

Docente: Dr. Sergio Ruíz Loza

Noviembre 28, 2022

Introducción

A pesar de que cada día contamos con equipos de cómputo más potentes, hay problemas que requieren de una gran cantidad de recursos computacionales, por lo que uno como desarrollador debe ser capaz de optimizar algoritmos robustos para brindar soluciones que sean eficientes mediante programas que posean una complejidad de bajo costo. Por lo anterior, el análisis y diseño de algoritmos, resulta ser una actividad compleja, ya que es necesario comprender el problema, así como determinar los pasos a seguir para que, de esta manera seamos capaces de llegar a una solución que sea eficiente y de calidad.

Objetivos Generales

• Entender y aplicar algoritmos computacionales avanzados en situaciones de la vida real, con el fin de solucionarlas de manera efectiva.
˚• Crear soluciones eficientes (considerando tiempo y espacio máquina) para crear modelos de mayor valor que sean aplicables en situaciones más complejas.

Objetivos Particulares

• Reconocer patrones que nos permitan identificar si un archivo contiene algún tipo de malware (E1).
• Utilizar algorimtos de búsqueda eficientes para encontrar subsecuencias y/o repeticiones (E1).
• Establecer, con el apoyo de grafos, conexiones eficientes entre las colonias (E2).
• Examinar, con apoyo de la geometría computacional, el lugar más conveniente para ubicar una nueva sucursal (E2).

Actividad Integradora 1:

Transmisiones de datos comprometidos

Los algoritmos de procesamiento de cadenas son de gran interés debido a la gran variedad de aplicaciones que tienen y su utilidad para resolver problemáticas reales.
El objetivo principal de la evidencia está enfocado en el análisis de strings para determinar similitudes entre archivos grandes de texto y determinar si existe código malicioso en forma de palíndromo.

Competencias:
SICT0101 - SICT0401 - STC0101 - STC0102

01. Búsqueda de substrings (KMP)

Complejidad: O(m + n)

KMP (Knuth Morris Pratt).
Permite resolver problemas de coincidencia de cadenas al encontrar un patrón dentro de un string. Utiliza una tabla LPS (Longest Prefix-Suffix) para analizar la estructura y evitar la necesidad de analizar múltiples veces los caracteres de la cadena. Resulta más eficiente que utilizar un algoritmo ingenuo donde no recuerda ninguna información de las coincidencias anteriores.

02. Palíndromo más largo (Manacher)

Complejidad: O(n)

Algoritmo que permite la búsqueda del palíndromo más largo dentro de una cadena. Su eficiencia se debe a que su lógica se centra en la noción de que un palíndromo existe dentro de otro. Este algoritmo funciona cuando el string es impar, por lo que para ampliar su funcionamiento a casos donde la longitud sea par, se ajusta la cadena de entrada insertando caracteres especiales al inicio y fin, así como entre los caracteres de la cadena.

03. Subcadena común más larga (LCS)

Complejidad: O(m*n)

Con ayuda de este algoritmo es posible encontrar la subsecuencia más larga entre 2 cadenas de texto, donde los elementos no necesariamente son consecutivos pero sí aparecen en el mismo orden. Con este algoritmo fue posible encontrar los índices de la subsecuencia más larga en los archivos de transmisión. El enfoque utilizado fue programación dinámica.

Resultados

```
PARTE 1: Substring Search (KMP Algorithm)
        transmission1.txt && mcode1.txt --> true   |   index: 2241
        transmission1.txt && mcode2.txt --> true   |   index: 3370
        transmission1.txt && mcode3.txt --> true   |   index: 5061
        transmission2.txt && mcode1.txt --> false
        transmission2.txt && mcode2.txt --> false
        transmission2.txt && mcode3.txt --> false

PARTE 2: Longest Palindromic String (Manacher Algorithm)
        LPS transmission1.txt --> startIndex: 5061  |  lastIndex: 5106
        LPS transmission2.txt --> startIndex: 1995  |  lastIndex: 2003

PARTE 3: Longest Common Subsequence (Dynamic Programming)
        LCS transmission1.txt --> startIndex: 1  |  lastIndex: 5755
        LCS transmission2.txt --> startIndex: 1  |  lastIndex: 5551
```

Actividad Integradora 2:

Alta demanda para los Proveedores de Servicios de Internet (ISP)

Se desea incorporar una región geográfica al mundo tecnológico. Para ello, se consideran diferentes aspectos, como la manera de realizar un cableado de forma eficiente entre las ciudades, identificar la mejor ruta para que los trabajadores realicen sus operaciones dentro del espacio físico, calcular el flujo máximo de información que puede transmitirse y tomar decisiones con respecto a la proximidad de las centrales dado una contratación nueva.

Competencias:
SICT0101 - SICT0401 - STC0101 - STC0102

01. Cableado eficiente (Kruskal)

Complejidad: O(n^2 * m)

El método de Kruskal permite obtener un subgrafo conformado por las aristas de peso mínimo que conforman al grafo original. Debido a esto, Kruskal permite obtener cuáles serían las aristas (o conexiones) entre ciudades que poseen la menor distancia entre sí para llevar a cabo un cableado eficiente.

02. Ruta a seguir para los repartidores (Held-Karp)

Complejidad: O(n^2*2^n)

Se busca marcar la ruta más barata para recorrer todas las colonias. Dicha ruta inicia en un nodo, recorre los demás (1 sola vez) y regresa al nodo de partida. Este problema es conocido como Travelling Salesman Person (TSP).
Held-Karp consiste en generar varios subconjuntos que representen los diferentes caminos. Nos apoyamos de programación dinámica. Se calculan los subconjuntos más pequeños pues son necesarios para los más grandes.

03. Flujo máximo (Ford-Fulkerson)

Complejidad: O(n^3*m)

Se desea conocer la cantidad máxima de información que puede fluir de un nodo “origen” a un nodo “sumidero”. El algoritmo plantea un grafo cuyas aristas poseen un flujo determinado, por lo que el flujo entrante debe ser el mismo que el saliente y no debe exceder a la capacidad máxima de la arista dada.
En la implementación realizada se hace un recorrido de tipo BFS (Breadth-First Search) que utiliza el método FIFO (First In, First Out) para encontrar el camino más corto de un grafo.

04. Búsqueda geométrica

Complejidad: O(n*log(n)+n)

Recibimos una lista de coordenadas en el formato (x, y), las cuales representan la posición geográfica de todas las centrales disponibles. Con base en las localizaciones recibidas, debemos identificar la central más cercana a una nueva sucursal. En aras de resolver esta pregunta, nos apoyaremos en la fórmula matemática para calcular la distancia entre 2 puntos:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Resultados

```
================================== PREGUNTA 1: CABLEADO
        Floyd-Warshall:
        Arco: (1, 2)            Distancia: 16   Km
        Arco: (1, 3)            Distancia: 34   Km
        Arco: (1, 4)            Distancia: 32   Km
        Arco: (2, 1)            Distancia: 16   Km
        Arco: (2, 3)            Distancia: 18   Km
        Arco: (2, 4)            Distancia: 21   Km
        Arco: (3, 1)            Distancia: 34   Km
        Arco: (3, 2)            Distancia: 18   Km
        Arco: (3, 4)            Distancia: 7    Km
        Arco: (4, 1)            Distancia: 32   Km
        Arco: (4, 2)            Distancia: 21   Km
        Arco: (4, 3)            Distancia: 7    Km

        Kruskal:
                MST:
                Arco: 3 -> 4: 7
                Arco: 1 -> 2: 16
                Arco: 2 -> 3: 18

================================== PREGUNTA 2: MEJOR RUTA
El camino más corto es:
        TSP: 1 --> 2 --> 4 --> 3 --> 1

================================== PREGUNTA 3: FLUJO MÁXIMO
        Flujo Máximo de Información: 78

================================== PREGUNTA 4: LISTA DE SUCURSALES
        Puntos ordenados de izquierda a derecha:
        1.- (200, 500)
        2.- (300, 100)
        3.- (301, 101)
        4.- (450, 150)
        5.- (520, 480)

        Distancia de la nueva contratación (301, 101) a la tienda (200, 500): 411.585Km
        Distancia de la nueva contratación (301, 101) a la tienda (300, 100): 1.41421Km
        Distancia de la nueva contratación (301, 101) a la tienda (450, 150): 156.85Km
        Distancia de la nueva contratación (301, 101) a la tienda (520, 480): 437.724Km

        La central más cercana a la nueva contratación es: (300, 100) con una distancia de 1.41421Km
```

Conclusión

A lo largo de este curso fue posible analizar, diseñar e implementar diferentes algoritmos que tienen aplicación en la vida real, no solo en el ámbito de las ciencias computacionales, sino también en otras áreas del conocimiento. El conocer estas herramientas computacionales nos da la oportunidad de implementar soluciones óptimas y eficientes que aprovechen de la mejor manera los recursos computacionales que se tienen al alcance para potenciar los resultados que se desean conseguir. Por lo anterior, decidir las estructuras y algoritmos es vital al momento de implementar alguna solución.

Referencias

Pratt KMP string matching algorithm KMP algorithm for pattern searching 9.1 Knuth-Morris- Bari, A. [UCZCFT11CWBi3MHNlGf019nw]. (2018, marzo 25).Youtube. https://www.youtube.com/watch?v=V5-7GzOfADQ . (2011, abril 3).
GeeksforGeeks. https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/
Loading. (s/f). Leetcode.com. Recuperado el 9 de septiembre de 2022, de https://leetcode.com/problems/longest-palindromic-substring/discuss/378722/Java-Manacher's-Algorithm-with-detailed-comments-beats-99.34!
Longest common subsequence. (s/f). Programiz.com. Recuperado el 9 de septiembre de 2022, de https://www.programiz.com/dsa/longest-common-subsequence
Time Complexity Infinity [UCf9ywn38ypqEIC6McbwH9sQ]. (2018, diciembre 17). LeetCode 5 - Longest Palindromic Subtring. Youtube. https://www.youtube.com/watch?v=SV1ZaKCozS4
GeeksforGeeks.(2022b, octubre 16).Kruskal's Minimum Spanning Tree Algorithm | Greedy Algo-2. https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/
GeeksforGeeks. (2022, noviembre 11). Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming). https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/
GeeksforGeeks. (2022, agosto 31). Dijsktra's algorithm. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
GeeksforGeeks. (2022c, agosto 31). Floyd Warshall Algorithm | DP-16. https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/
GeeksforGeeks. (2022a, junio 21). Ford-Fulkerson Algorithm for Maximum Flow Problem. https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/
GeeksforGeeks. (2020, 4 junio). Proof that traveling salesman problem is NP Hard. https://www.geeksforgeeks.org/proof-that-traveling-salesman-problem-is-np-hard/