Búsqueda exhaustiva y programación dinámica

Bailin Huang - B53546 y Jose Andrés Mejías Rojas - B54275

***Resumen*—En este trabajo se analiza el comportamiento de la búsqueda exhaustiva y programación dinámica en un problema: en el distrito Catedral, mediante unas reglas en específico, se determina el camino más corto entre dos puntos arbitrarios. Se concluye que, en la teoría, el algoritmo de programación dinámica es el más eficiente a la hora de hacer los cálculos mencionados.**

I. Introducción

El propósito de este trabajo es determinar la eficiencia de los algoritmos de búsqueda exhaustiva y programación dinámica, en el caso de encontrar el camino más corto entre dos puntos en el distrito Catedral, San José, Costa Rica.[1] Se apoyó en la teoría mencionada en el libro de Cormen y colaboradores [1], para determinar elementos matemáticos necesarios. A partir de la implementación, se analizaron los resultados y así determinar cuál algoritmo es el más eficiente. Cabe destacar que solo se implementó el algoritmo de búsqueda exhaustiva, por lo que, en programación dinámica, se compararon con los planteamientos teóricos.[2]

II. Metodología

Para cumplir con lo planteado en la sección I, se realizaron los siguientes pasos:

II-A. Implementación

Los algoritmos se implementaron en el lenguaje de programación *C++*. La compilación del programa se hizo en *C++98* mediante el compilador *TDM-GCC-64*. Además de los algoritmos, se implementó un *main* para interactuar con el usuario y así poder realizar casos de prueba.

II-B. Datos

El grupo 2 de CI-1221 de la Universidad de Costa Rica recolectaron los datos necesarios en todo el distrito. El criterio para poner un puntaje a los caminos fue la cantidad de altos y reductores de velocidad que había entre intersección.[3] Con hojas de cálculo, se captaron todos los datos al programa, recorriendo todos los archivos.

II-C. Análisis de resultados

En análisis de los resultados se hicieron mediante casos de prueba arbitrarios, luego se compararon con lo que dice la teoría y así hacer una conclusión al respecto. Se calculó el tiempo de ejecución en cada caso para definir si hay un comportamiento similar.

[1]El criterio del camino más corto se menciona en la sección II-B.

[2]Los resultados de la implementación de la búsqueda exhaustiva con los aspectos matemáticos de la programación dinámica.

[3]Un camino se define como la unión entre dos intersecciones, sin pasar por una en medio, es decir, intersecciones aledañas.

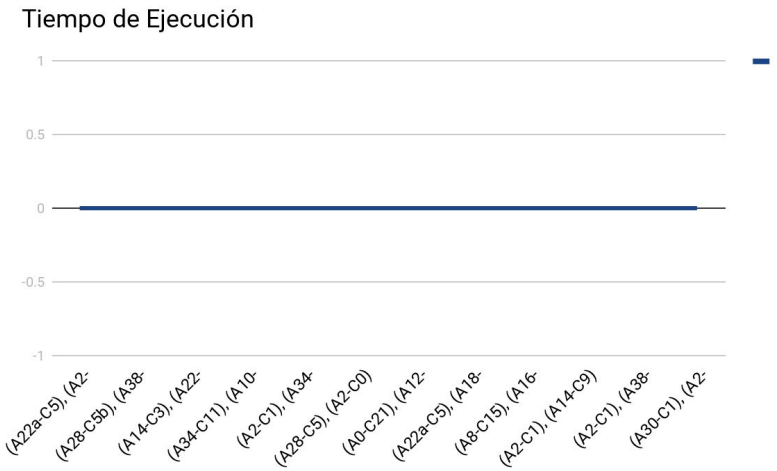

Figura 1. Gráfico compartivo de los tiempos de ejecución en los casos de prueba.

III. Resultados

Los tiempos de cada ejecución se puede ver en la figura 1, donde se puede ver que los tiempos no varía entre sí. En la sección III-A se detalla el planteamiento matemático para llevar a cabo el algoritmo de búsqueda exhaustiva. Los resultados son inesperados, porque se sabe que la búsqueda exahustiva tiende a ser ineficiente. El planteamiento matemático de la programación dinámica se puede ver en la sección III-B.

III-A. Búsqueda Exhaustiva

1. $\sigma =< \sigma_1, \sigma_2 \ldots \sigma_n >$, con$\sigma_i$ una dirección de los puntos cardinales, la dirección a la que va a ir.
2. $E = \{0, 1, 2\}$, con 0, 1, 2 representando la dirección a la que sigue. Ejemplo: 0 siendo sur, 1 siendo sureste, 2 siendo este.
3. $|E| = 3^n$.
4. $E' = F(\sigma_i) = -1$ o $\sum F(\sigma_i) > Solución\,Actual$.[4]

III-B. Programación dinámica

1. $F(i, j)$ =el camino más corto o con menos puntos, desde la posición (i, j) hasta (x, y), con x, y definidas previamente como el destino requerido.
2. $F(z, w)$ =el camino más corto, iniciando de la posición (z, w) hasta (x, y), con z, w definidas previamente como el origen.
3. $F(x, y) = 0$.
4. $F(i, j) = mín\{F(i, j) + B[i + 1, j], F(i, j + 1) + B[i + 1, j], F(i + 1, j + 1) + B[i+1, j+1]\}$. Tomando en cuenta

[4]Sea F una matriz de valores de altos para las direcciones.

que las sumas van a acorde a las reglas planteadas previamente.

IV. Conclusiones

Se puede concluir que los tiempos de crecimiento fueron los esperados dado que la matriz es de un tamaño pequeño. Esto produce el efecto de un tiempo de ejecución casi nulo para cada búsqueda exhaustiva, sin importar hacia qué lado del mapa es el destino, lo cual crea valores de 0 segundos por búsqueda. Además que, acorde a lo planteado en la programación dinámica y lo mencionado anteriormente, los tiempos puede que no varíen demasiado.

Referencias

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.