Polo del Conocimiento

Pol. Con. (Edición núm. 70) Vol. 7, No 7
Julio 2022, pp. 1288-1302
ISSN: 2550 - 682X
DOI: 10.23857/pc.v7i7

Algoritmo de Bellman Ford para solucionar el problema de la ruta más corta entre nodos

Bellman Ford algorithm to solve the shortest path problem between nodes

Algoritmo de Bellman Ford para resolver o problema do caminho mais curto entre nós

Juan Carlos Yungán-Cazar [I]
jyungan@espoch.edu.ec
https://orcid.org/0000-0001-5682-0399

Edgar Gualberto Salazar-Álvarez [II]
edgar.salazar@espoch.edu.ec
https://orcid.org/0000-0003-0988-0641

Jhon Eduardo Villacrés-Sampedro [III]
jhon.villacres@espoch.edu.ec
https://orcid.org/0000-0002-8064-9680

Correspondencia: jyungan@espoch.edu.ec

Ciencias Tecnologías de la Información y la Comunicación
Artículo de Investigación

*** Recibido:** 23 de mayo de 2022 ***Aceptado:** 12 de junio de 2022 *** Publicado:** 19 de julio de 2022

I. Magíster en Interconectividad de Redes, Ingeniero en Sistemas Informáticos, Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador.
II. Magíster en Matemática Básica, Ingeniero en Sistemas, Escuela Superior Politécnica de Chimborazo. Riobamba, Ecuador.
III. Máster Universitario en Tecnología Educativa y Competencias Digitales, Magíster en Desarrollo de la Inteligencia y Educación, Ingeniero en Sistemas, Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador.

Resumen

Conocer la historia de la ruta más corta es algo impredecible, ya que el problema del camino más corto se ha venido presentando desde la antigüedad, por ejemplo: encontrar caminos cortos hacia la comida. Es por ello que se consideraba un problema elemental y fácil de resolver. Años más tarde investigadores desarrollaron métodos para dar una solución independiente a problemas como: rutas con menos tráfico, caminos en un laberinto, enrutamiento alternativo, llamadas a larga distancia, etc.). Su evolución a dar soluciones ha permitido desarrollar métodos matriciales para el camino más corto de longitud unitaria. El método Bellman-Ford permite encontrar la ruta más corta con un algoritmo más simple, maneja arcos negativos y su tiempo de ejecución es menor, lo que le convierte en el método a ser estudiado.

Palabras Clave: Bellman-Ford; Algoritmo; Ruta más corta.

Abstract

Knowing the history of the shortest route is somewhat unpredictable, since the problem of the shortest path has been present since ancient times, for example: finding short routes to food. That is why it was considered an elementary problem and easy to solve. Years later, researchers developed methods to provide an independent solution to problems such as: routes with less traffic, paths in a maze, alternative routing, long distance calls, etc.). Its evolution to provide solutions has allowed the development of matrix methods for the shortest path of unit length. The Bellman-Ford method allows finding the shortest path with a simpler algorithm, it handles negative arcs and its execution time is shorter, which makes it the method to be studied.

Keywords: Bellman-Ford; Algorithm; Shortest route.

Resumo

Conhecer a história do caminho mais curto é algo imprevisível, pois o problema do caminho mais curto está presente desde a antiguidade, por exemplo: encontrar caminhos curtos para alimentação. Por isso era considerado um problema elementar e de fácil solução. Anos depois, pesquisadores desenvolveram métodos para fornecer uma solução independente para problemas como: rotas com menos tráfego, caminhos em labirinto, roteamento alternativo, chamadas de longa distância etc.). Sua evolução para fornecer soluções permitiu o desenvolvimento de métodos matriciais para o caminho mais curto de comprimento unitário. O método de Bellman-Ford permite encontrar o

caminho mais curto com um algoritmo mais simples, trata arcos negativos e seu tempo de execução é menor, o que o torna o método a ser estudado.

Palavras-chave: Bellman-Ford; Algoritmo; Rota mais curta.

Introducción

Encontrar la ruta más corta, radica en hallar un camino entre un nodo origen y un nodo destino. Estos nodos se encuentran enlazados directa o indirectamente a través de arcos que tienen una ponderación, el cual puede ser entendido como costo, distancia, tiempo, etc. El algoritmo de Bellman Ford, soluciona el problema de la ruta más corta, empleando teoría de grafos y análisis y diseño de algoritmos.

Es importante entender que un grafo es una dupla $G = (V, E)$ compuesta por un conjunto finito $V = V(G)$ de vértices, también llamados nodos y típicamente dibujados por círculos y un conjunto $E = E(G)$ que llamaremos conjunto de aristas. Mientras que, un algoritmo es una secuencia finita y ordenada de pasos para realizar una tarea en forma precisa.

Las redes de datos (grafos) utilizan protocolos para establecer comunicación entre diferentes equipos denominados host. Estos protocolos (algoritmos) están gobernados por reglas que determinan su comportamiento. El enrutamiento de mensajes entre equipos (nodos/routers) se lo hace a través de protocolos que garantizan la conectividad entre redes.

Desarrollo

La red (grafo) es un tipo común de estructura de datos que ofrece una visión holística y descendente para dar sentido a diversos sistemas interactivos, incluidos los sistemas sociales, los sistemas biológicos, los sistemas de tráfico, los sistemas de comunicación, etc., que se ven muy afectados por una pequeña porción de nodos influyentes, también llamados propagadores influyentes. Dichos nodos desempeñan un papel fundamental y pueden enriquecer significativamente nuestra comprensión de los sistemas mencionados. Por ejemplo, ser capaces de detectar eficaz y adecuadamente los nodos influyentes nos permite controlar la propagación de epidemias, diseñar un plan de marketing válido, evitar que la red eléctrica falle, predecir el flujo de tráfico futuro e identificar proteínas esenciales. (Liu et al., 2021)

Un grafo G está formado por dos conjuntos

Pol. Con. (Edición núm. 70) Vol. 7, No 7, Julio 2022, pp. 1288-1302, ISSN: 2550 - 682X

- V , un conjunto de vértices (también llamados nodos)
- E , un conjunto de aristas (cada arista está asociada a dos vértices).

En las situaciones de modelización, las aristas se utilizan para expresar algún tipo de relación entre los vértices. (Pruim, s. f.)

La teoría de grafos proporciona un lenguaje para hablar de las propiedades de las relaciones. Diseñar algoritmos de grafos realmente novedosos es una tarea muy difícil. La clave para utilizar eficazmente los algoritmos de grafos en las aplicaciones reside en modelar correctamente el problema para poder aprovechar los algoritmos existentes. Familiarizarse con muchos problemas algorítmicos de grafos diferentes es más importante que entender los detalles de los algoritmos de grafos particulares. (Skiena, 2008)

La ruta más corta

Si tenemos dos vértices en un grafo y múltiples caminos entre ellos, entonces hay un camino más corto en esa colección. Ese camino no es necesariamente único. El problema del camino más corto es el proceso de encontrar el camino más corto entre dos vértices de un grafo. Podemos considerarlo como la ruta más eficiente a través del grafo.

Otra forma de considerar el problema del camino más corto es recordar que un camino es una serie de relaciones derivadas. El camino más corto es la serie con la derivación más corta, o la relación más cercana. Dado que un grafo modela relaciones, a menudo nos interesan las relaciones más cercanas. (*Shortest Path Problem - an overview | ScienceDirect Topics*, s. f.)

El problema de la ruta más corta es uno de los problemas de optimización de redes más fundamentales. Este problema surge en la práctica y surge como un subproblema en muchos algoritmos de optimización de redes. (Cherkassky et al., 1996)

Algoritmos básicos

Los algoritmos más importantes para resolver los problemas de camino más corto son:

- La búsqueda de amplitud y la búsqueda de profundidad se refieren a diferentes órdenes de búsqueda; para la búsqueda de profundidad, se pueden encontrar casos en los que su implementación ingenua no encuentra una solución óptima, o no termina.
- El algoritmo de Dijkstra resuelve el problema del camino más corto de una sola fuente si todos los pesos de las aristas son mayores o iguales a cero. Sin empeorar la complejidad del

tiempo de ejecución, este algoritmo puede, de hecho, calcular los caminos más cortos desde un punto de partida s dado a todos los demás nodos.

- El algoritmo de Bellman-Ford también resuelve el problema de los caminos más cortos de una sola fuente, pero a diferencia del algoritmo de Dijkstra, los pesos de las aristas pueden ser negativos.
- El algoritmo de Floyd-Warshall resuelve el problema de las rutas más cortas de todos los pares.
- El algoritmo A* resuelve el problema del camino más corto de una sola fuente para costes de aristas no negativos. (*Shortest Path Problem - an overview | ScienceDirect Topics*, s. f.)

Método

La elección del método más adecuado y su correcta ejecución es lo que impulsa a realizar una buena investigación. Para el presente trabajo se ha considerado el método mixto (cualitativo y cuantitativo), ya que se está contemplado:

1. Se hará uso de las técnicas documentales y de experimento. La técnica documental para la recolección de información a través de publicaciones en revistas científicas. La técnica del experimento con el uso del programa para el análisis de grafos.
2. El Análisis y aplicación paso a paso del algoritmo de Bellman Ford se la realiza en una hoja de trabajo (Tabla 1)
3. Análisis y diseño de un programa escrito en Java que permita implementar el algoritmo de Bellman Ford.

Análisis

Algoritmo de Bellman-Ford

Este algoritmo busca la estructura del grafo y genera la mejor solución. Resuelve el problema del camino más corto de fuente única en el que los pesos de los bordes pueden ser negativos. El algoritmo de Bellman-Ford para las rutas más cortas es casi completamente intuitivo y devuelve un valor booleano que indica si hay o no un ciclo de peso negativo al que se puede acceder desde la fuente. Entonces, cuando hay un ciclo, el algoritmo indica que no existe una solución, pero cuando no la hay, el algoritmo produce los caminos más cortos y su peso. Además, la base es la

Pol. Con. (Edición núm. 70) Vol. 7, No 7, Julio 2022, pp. 1288-1302, ISSN: 2550 - 682X

siguiente: dado un gráfico con n vértices, la mayor cantidad de arcos que puede haber en un camino más corto es $n-1$. Esto se demuestra fácilmente por inducción. El camino más corto tiene más de $n-1$, entonces debe tener un circuito negativo, de manera similar, un gráfico con un circuito negativo tendrá un camino más corto de n o más arcos. Todas estas observaciones conducen al algoritmo Bellman-Ford. (Mukhlif & Saif, s. f.) (Figura 1)

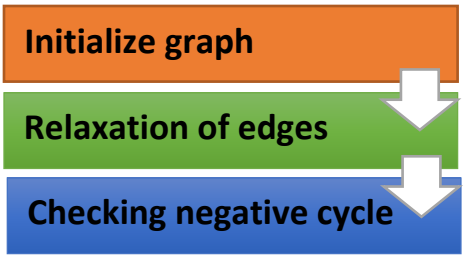

Figura 1: Proceso del algoritmo de Bellman Ford

Fuente: (Patel, 2014)

Algoritmo 1. Bellman Ford, Ruta mínima

Entrada: $G = (V, E)$: $grafo\ dirigido, c : E \rightarrow \mathbb{R}\ costo\ de\ cada\ arista, u\ vértice\ inicial.$

1. $L[u] \leftarrow 0$
2. $L[v] \leftarrow \infty, para\ todo\ v \neq u$
3. $para\ i = 1 \ldots |V| - 1$
4. $para\ cada\ arista\ (v, w)\ \epsilon\ E$
5. $si\ L[w] > L\ [v] + c(v, w)\ entonces$
6. $L[w] \leftarrow L\ [v] + c(v, w)$
7. fin
8. fin
9. fin

Salida: $G = L\ [v]\ distancia\ más\ corta\ de\ u\ al\ vertice\ v, \forall\ v\ \in V$

Pseudocódigo del algoritmo Bellman-Ford

1. Inicialice todos los vértices y establezca su distancia desde la fuente a un valor alto, como infinito, y establezca el valor de distancia para la fuente en 0 valores

2. Repita lo siguiente para el número de $(vértices - 1)$ tiempos tales como: para cada arista (V, E) que pertenece al gráfico dibujado si distancia$[V] > distancia[E] + peso(E, V)$ entonces la $distancia[v] = distancia[u] + peso\,(u, v)$, lo que significa que la distancia se actualiza.
3. Repita los siguientes pasos para cada borde (V, E) que pertenece al gráfico dibujado si $distancia[V] > distancia[E] + peso(E, V)$ entonces no existe ningún ciclo negativo - peso entre el origen y el destino.
4. Devuelve los valores de distancia para todos los nodos una vez completadas todas las iteraciones. (Mukhlif & Saif, s. f.)

Funcionamiento

Al igual que otros problemas de programación dinámica, el algoritmo calcula las rutas más cortas de forma ascendente. Primero calcula las distancias más cortas que tienen como máximo un borde en la ruta. Luego, calcula los caminos más cortos con 2 aristas como máximo, y así sucesivamente. Después de la i-ésima iteración del bucle exterior, se calculan los caminos más cortos con como máximo i aristas. Puede haber un máximo de |V| – 1 aristas en cualquier camino simple, por eso el ciclo externo ejecuta |v| - 1 vez. La idea es, asumiendo que no hay un ciclo de peso negativo, si hemos calculado las rutas más cortas con como máximo aristas i, entonces una iteración sobre todas las aristas garantiza dar la ruta más corta con aristas como máximo (i+1). («Bellman–Ford Algorithm | DP-23», 2012)

Este algoritmo tiene las ventajas de:

- Minimización de costes.
- Maximización del rendimiento.
- Permite dividir el tráfico.

Desventajas:

- En el protocolo de enrutamiento no se tiene en cuenta las ponderaciones.
- Respuesta lenta a los cambios en la topología de red. (Singh et al., 2018)

Diseño de clases: (Figura 2:**)**

Pol. Con. (Edición núm. 70) Vol. 7, No 7, Julio 2022, pp. 1288-1302, ISSN: 2550 - 682X

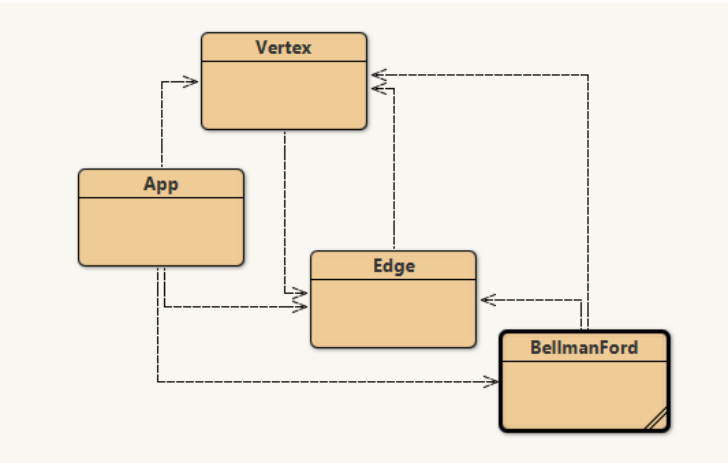

Figura 2: Diagrama de clases

Fuente: Juan C. Yungán-Cazar.

Implementación del programa

Clase Vertex: En esta clase se implementa los nodos. Está compuesta por los atributos nombre, estado de visitado, lista de aristas asociadas a los nodos, distancia mínima, y nodo antecesor. (Figura 3)

```
import java.util.ArrayList;
import java.util.List;
public class Vertex {
    private String name;
    private boolean visited;
    private List<Edge> edges;
    private double minDistance = Double.MAX_VALUE;
    private Vertex previousVertex;
    public Vertex(String name) {
    public void addAdge(Edge edge) {
    public boolean isVisited() {
    public void setVisited(boolean visited) {
    public List<Edge> getEdges() {
    public void setEdges(List<Edge> edges) {
    public double getMinDistance() {
    public void setMinDistance(double minDistance) {
    public Vertex getPreviousVertex() {
    public void setPreviousVertex(Vertex previousVertex) {
}
```

Figura 3: Implementación clase Vertex (Nodo)

Fuente: Juan C. Yungán-Cazar.

Clase Edge: En esta clase se implementa las aristas que conectan a los nodos. Está compuesta por los atributos ponderación o peso; referencia de nodo inicial y referencia de nodo final. (Figura 4)

```
public class Edge {
    private double weight;
    private Vertex startVertex;
    private Vertex targetVertex;
    public Edge(double weight, Vertex startVertex, Vertex targetVertex) {
    public double getWeight() {
    public void setWeight(double weight) {
    public Vertex getStartVertex() {
    public void setStartVertex(Vertex startVertex) {
    public Vertex getTargetVertex() {
    public void setTargetVertex(Vertex targetVertex) {
}
```

Figura 4: Implementación clase Edge (Arista)

Fuente: Juan C. Yungán-Cazar.

Clase BellmanFord: En esta clase se impelenta el algorirmo de Bellman Ford. Está compuesta por los atributos lista de nodos (vertexList) y lista de aristas (edgexList). (Figura 5)

```
import java.util.List;
public class BellmanFord {
    private List<Vertex> vertexList;
    private List<Edge> edgeList;
    public BellmanFord(List<Edge> edgeList, List<Vertex> vertexList) {
    public void shotestPath(Vertex sourceVertex, Vertex targetVertex) {
    private boolean hasCycle(Edge edge) {
}
```

Figura 5: Implementación clase BellmanFord

Fuente: Juan C. Yungán-Cazar.

Escenario de aplicación

Se diseña un grafo dirigido con 9 nodos que representan a diferentes ciudades del país y las aristas representan a las conexiones viales entre ellas con su respectiva distancia. (Figura 6)

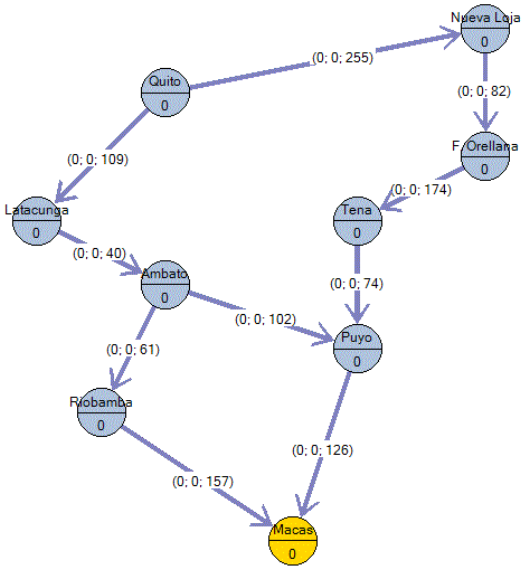

Figura 6: Escenario propuesto

Fuente: Juan C. Yungán-Cazar.

Clase App: En esta clase se implementa el escenario de estudio. (Figura 7)

```
import java.util.ArrayList;
import java.util.List;
public class App {
    public static void main(String[] args) {
        List<Vertex> vertexList = new ArrayList<>();
        vertexList.add(new Vertex("Quito"));
        vertexList.add(new Vertex("Latacunga"));
        vertexList.add(new Vertex("Ambato"));
        vertexList.add(new Vertex("Riobamba"));
        vertexList.add(new Vertex("Nueva_Loja"));
        vertexList.add(new Vertex("F_Orellana"));
        vertexList.add(new Vertex("Tena"));
        vertexList.add(new Vertex("Puyo"));
        vertexList.add(new Vertex("Macas"));
        List<Edge> edgeList = new ArrayList<>();
        edgeList.add(new Edge(109, vertexList.get(0), vertexList.get(1)));
        edgeList.add(new Edge(255, vertexList.get(0), vertexList.get(4)));
        edgeList.add(new Edge(40, vertexList.get(1), vertexList.get(2)));
        edgeList.add(new Edge(61, vertexList.get(2), vertexList.get(3)));
        edgeList.add(new Edge(102, vertexList.get(2), vertexList.get(6)));
        edgeList.add(new Edge(157, vertexList.get(3), vertexList.get(8)));
        edgeList.add(new Edge(82, vertexList.get(4), vertexList.get(5)));
        edgeList.add(new Edge(174, vertexList.get(5), vertexList.get(6)));
        edgeList.add(new Edge(74, vertexList.get(6), vertexList.get(7)));
        edgeList.add(new Edge(126, vertexList.get(7), vertexList.get(8)));
        BellmanFord algorithm = new BellmanFord(edgeList, vertexList);
        algorithm.shotestPath(vertexList.get(0), vertexList.get(8));
    }
}
```

Figura 7: Implementación clase App (Escenario de pruebas)

Fuente: Juan C. Yungán-Cazar.

Resultados

Tabla 1: Resultados del análisis y aplicación paso a paso del algoritmo

	Quito	Latacunga	Ambato	Riobamba	Macas	Nueva Loja	F. Orellana	Puyo	Tena
Quito	0	109	149	210	367	255	337	251	511
Latacunga	Infinito	0	40	101	258	Infinito	Infinito	142	Infinito
Ambato	Infinito	Infinito	0	61	218	Infinito	Infinito	102	Infinito
Riobamba	Infinito	Infinito	Infinito	0	157	Infinito	Infinito	Infinito	Infinito
Macas	Infinito	Infinito	Infinito	Infinito	0	Infinito	Infinito	Infinito	Infinito
Nueva Loja	Infinito	Infinito	Infinito	Infinito	456	0	82	330	256
F. Orellana	Infinito	Infinito	Infinito	Infinito	374	Infinito	0	248	174
Puyo	Infinito	Infinito	Infinito	Infinito	126	Infinito	Infinito	0	Infinito
Tena	Infinito	Infinito	Infinito	Infinito	200	Infinito	Infinito	74	0

Tabla 2: Resultados de rutas mínimas calculadas por el programa

ORIGEN	DESTINO	COSTE	RUTA
Quito	Latacunga	109	Latacunga = Quito, Latacunga
Quito	Ambato	149	Ambato = Quito, Latacunga, Ambato
Quito	Riobamba	210	Riobamba = Quito, Latacunga, Ambato, Riobamba

Pol. Con. (Edición núm. 70) Vol. 7, No 7, Julio 2022, pp. 1288-1302, ISSN: 2550 - 682X

Quito	Macas	367	Macas = Quito, Latacunga, Ambato, Riobamba, Macas
Quito	Nueva Loja	255	Nueva Loja = Quito, Nueva Loja
Quito	F.Orellana	337	F. Orellana = Quito, Nueva Loja, F. Orellana
Quito	Puyo	251	Puyo = Quito, Latacunga, Ambato, Puyo
Quito	Tena	511	Tena = Quito, Nueva Loja, F. Orellana, Tena

ORIGEN	DESTINO	COSTE	RUTA
Latacunga	Ambato	40	Ambato = Latacunga, Ambato
Latacunga	Riobamba	101	Riobamba = Latacunga, Ambato, Riobamba
Latacunga	Macas	258	Latacunga, Ambato, Riobamba, Macas
Latacunga	Puyo	142	Puyo = Latacunga, Ambato, Puyo

ORIGEN	DESTINO	COSTE	RUTA
Ambato	Riobamba	61	Riobamba = Ambato, Riobamba
Ambato	Macas	218	Macas = Ambato, Riobamba, Macas
Ambato	Puyo	102	Puyo = Ambato, Puyo

ORIGEN	DESTINO	COSTE	RUTA
Riobamba	Macas	157	Macas = Riobamba, Macas

ORIGEN	DESTINO	COSTE	RUTA
Nueva Loja	Macas	456	Macas = Nueva Loja, F. Orellana, Tena, Puyo, Macas
Nueva Loja	F. Orellana	82	F. Orellana = Nueva Loja, F. Orellana
Nueva Loja	Puyo	330	Puyo = Nueva Loja, F. Orellana, Tena, Puyo
Nueva Loja	Tena	256	Tena = Nueva Loja, F. Orellana, Tena

Pol. Con. (Edición núm. 70) Vol. 7, No 7, Julio 2022, pp. 1288-1302, ISSN: 2550 - 682X

ORIGEN	DESTINO	COSTE	RUTA
F.Orellana	Macas	374	Macas = F. Orellana, Tena, Puyo, Macas
F.Orellana	Puyo	248	Puyo = F. Orellana, Tena, Puyo
F.Orellana	Tena	174	Tena = F. Orellana, Tena

ORIGEN	DESTINO	COSTE	RUTA
Tena	Macas	200	Macas = Tena, Puyo, Macas
Tena	Puyo	74	Puyo = Tena, Puyo

ORIGEN	DESTINO	COSTE	RUTA
Puyo	Macas	126	Macas = Puyo, Macas

El camino más corto calculado por el algoritmo tomando como nodo origen Quito hasta el nodo destino Macas es el que visita los nodos: Latacunga con una ponderación de 109 Km, Ambato (40 Km) con una ponderación acumulada de 149 Km, Riobamba (61 Km) con una ponderación acumulada de 210 Km; hasta llegar al nodo destino Macas (157 Km) con una ponderación total de 367 Km.

El siguiente camino más corto calculado por el algoritmo tomando como nodo origen Quito hasta el nodo destino Macas es el que visita los nodos: Latacunga con una ponderación de 109 Km, Ambato (40 Km) con una ponderación acumulada de 149 Km, Puyo (102 Km) con una ponderación acumulada de 251 Km; hasta llegar al nodo destino Macas (126 Km) con una ponderación total de 377 Km.

El camino más largo calculado por el algoritmo tomando como nodo origen Quito hasta el nodo destino Macas es el que visita los nodos: Nueva Loja con una ponderación de 255 Km, Francisco de Orellana (82 Km) con una ponderación acumulada de 337 Km, Tena (174 Km) con una ponderación acumulada 511 Km, Puyo (74 Km) con una ponderación acumulada de 585 Km; hasta llegar al nodo destino Macas (126 Km) con una ponderación total de 711 Km.

Conclusiones

Un grafo es una representación gráfica de nodos (ciudades, routers) conectados por arcos (carreteras, enlaces seriales)

Pol. Con. (Edición núm. 70) Vol. 7, No 7, Julio 2022, pp. 1288-1302, ISSN: 2550 - 682X

La ruta mínima o camino más corto simplemente es la distancia entre nodos (origen y destino) con menor distancia.

El problema del camino más corto es recordar que un camino es una serie de relaciones derivadas. El camino más corto es la serie con la derivación más corta o la relación más cercana. Dado que un gráfico está modelando relaciones, a menudo nos interesan las relaciones más cercanas.

El algoritmo de Bellman Ford plantea una solución al problema de la ruta más corta, pero no es el único, por ejemplo, para resolver este problema también existe el algoritmo de Dijkstra.

El tiempo de ejecución del algoritmo de Bellman Ford es bajo (microsegundos) lo que le convierte en un método muy eficiente para un pequeño número de nodos.

Encontrar la ruta más corta, por lo general es un problema frecuente en temas de logística y transporte, las redes de comunicación de datos; entre las más importantes.

Áreas del conocimiento involucrada: Teoría de grafos, Análisis y diseño de algoritmos, Estructura de datos, Algebra lineal, Logística y transporte, Redes de computadoras, etc.

Referencias

1. Bellman–Ford Algorithm | DP-23. (2012, diciembre 1). GeeksforGeeks. https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/
2. Cherkassky, B. V., Goldberg, A. V., & Radzik, T. (1996). Shortest paths algorithms: Theory and experimental evaluation. Mathematical Programming, 73(2), 129-174. https://doi.org/10.1007/BF02592101
3. Liu, Y., Wei, X., Chen, W., Hu, L., & He, Z. (2021). A graph-traversal approach to identify influential nodes in a network. Patterns, 2(9), 100321. https://doi.org/10.1016/j.patter.2021.100321
4. Moreno, E. (2012). Grafos: Fundamentos y Algoritmos. Santiago de Chile, Chile: Editorial ebooks Patagonia - J.C. Sáez Editor.
5. Recuperado de https://elibro.net/es/ereader/espoch/68438?page=82.
6. Mukhlif, F., & Saif, A. (s. f.). Comparative Study On Bellman-Ford And Dijkstra Algorithms. 6.

7. Patel, V. (2014). A survey paper of Bellman-ford algorithm and Dijkstra algorithm for finding shortest path in GIS application. 4(1), 3.

8. Pérez Aguila, R. (2013). Una introducción a las matemáticas discretas y teoría de grafos. El Cid Editor.

9. Recuperado de https://elibro.net/es/lc/espoch/titulos/36562

10. Pruim, R. (s. f.). 1 Introduction to Graphs | Graphs. Recuperado 27 de junio de 2022, de https://rpruim.github.io/m252/S19/from-class/graphs/introduction-to-graphs.html#graphs

11. Reid, A. y Lorenz, J. (2018). Introducción al enrutamiento y la conmutación en la empresa: guía de estudio de CCNA Discovery. Pearson Educación.

12. Recuperado de: https://elibro.net/es/lc/espoch/titulos/53862

13. Rodríguez Villalobos, A. (2017). Grafos: software para la construcción, edición y análisis de grafos. Bubok Publishing S.L.

14. Recuperado de https://elibro.net/es/lc/espoch/titulos/55604

15. Shortest Path Problem—An overview | ScienceDirect Topics. (s. f.). Recuperado 27 de junio de 2022, de https://www.sciencedirect.com/topics/computer-science/shortest-path-problem

16. Singh, J. B., Tripathi, R. C., & Research, D. (2018). Investigation of Bellman–Ford Algorithm, Dijkstra's Algorithm for suitability of SPP. 6(1), 4.

17. Skiena, S. S. (2008). The Algorithm Design Manual. Springer London. https://doi.org/10.1007/978-1-84800-070-4