BUSQUEDA DE RUTAS EFICIENTES DE REPARTO

PRESENTADO POR:

MIGUEL ANGEL SALAMANCA

JUAN CAMILO BAZURTO ARIAS

PRESENTADO A:

SEBASTIAN CAMILO MARTINEZ REYES

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

ALGORITMOS Y ESTRUCTURAS DE DATOS

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2021 – 1

1. **Contexto**

Suponga que usted es una persona que tiene que entregar un pedido de un punto 2, usted estando en el punto 1 debería buscar la ruta mas eficiente con el fin de optimizar su trabajo. Sin embargo, seria una tarea muy tediosa ver el mapa de su ubicación y buscar la ruta mas eficiente o preguntar a gente conocida si la conoce. Con la teoría de grafos y la tecnología que tenemos hoy en día, podemos resolver este problema.

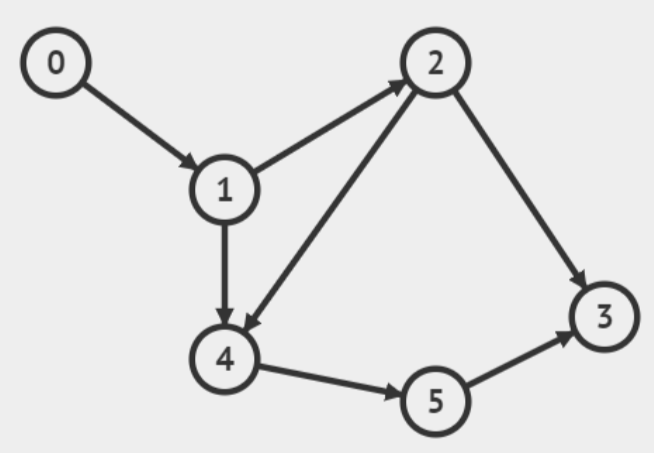
1. **Requisitos**
   1. Especificación
      1. Entrada

Se genera una ruta con sus respectivos nodos y conexiones, y se agregaran pedidos para esta ruta, pueden ser uno o más.

* + 1. Salida

Se mostrará la ruta más eficiente por BFS, si hay mas de un pedido se mostraran las etiquetas de como debe ser el orden de los pedidos por Topological-Sort, el cual utiliza un algoritmo muy similar a DFS.

1. **Diseño**
   1. Estrategia
      1. Descripción general
         1. Grafos.



Como podemos ver en el grafo anterior, utilizaremos la teoría de grafos para representar los puntos y rutas que puede tomar un repartidor, por ejemplo, el repartidor esta en el punto cero y necesita ir al punto cinco, nuestra labor es mostrarle la ruta más eficiente para llegar el punto cero al punto cinco, cuando los posibles puntos uno, dos, tres y cuatro.

* + - 1. BFS.

Si deseamos encontrar la ruta mas corta desde el punto cero al punto 3 tendremos que saber cuales son las distancias entre los nodos y a partir de ahí, tomar la de menor valor. Todo esto nos lo permite el algoritmo Breadth-First Search (BFS), que nos retorna todas las rutas posibles para llegar a cualquier nodo distinto al que nos encontramos.

Sabiendo esto, el usuario solo deberá brindar su ubicación y el destino al que desea llegar, el algoritmo BFS calculará todas las rutas posibles y se elegirá la más corta, mostrándosela al usuario.

* + - 1. DFS.

Para ordenar las entregas de los pedidos que tiene un repartidor, hay que llevar un orden de entrega que haga la tarea mas eficiente, con el algoritmo DFS y el ordenamiento Topological Sort, podemos brindarle al usuario etiquetar cada lugar que va a representar el orden en que los pedidos van a ser entregados.

* + - 1. Conjuntos Disyuntos.

En el caso que a un repartidor tenga varios pedidos en tiempos de entrega distintos, la ruta va a tener que ser modificada para poder entregar estos en un tiempo optimo, con la ayuda de conjuntos disyuntos y su algoritmo unión.find vamos a poder agregar nuevos pedidos a una ruta ya creada.

* 1. Casos de Prueba

Como casos de prueba se tomaron los siguientes casos:

* Nodos:

V = [0, 1, 2, 3, 4, 5]

* Conexiones:

E = [(0, 1), (0, 2), (1, 2), (2, 3), (4, 3), (4, 5)]

Los resultados de estos casos fueron los siguientes:

* Ruta más eficiente:

0

0

1

0->1

2

0->2

3

0->2->3

4

4

5

5

* Orden de pedidos:

4, 5, 0, 1 ,2 ,3

1. **Análisis**
   1. Temporal
      1. BFS

La complejidad de este algortimos es O(V + E)

* + 1. Topological-Sort

La complejidad de este algoritmo es O(V + E)

* + 1. Nuevo Pedido

La complejidad de este algoritmo es O(V)