Algoritmos de caminos para peatones que reducen tanto el acoso callejero como la distancia

Samuel Aguilar Villada
saguilarv@eafit.edu.co

Angy Sánchez Moreno
ajsanchezm@eafit.edu.co

Resumen

El presente informe tiene por objetivo analizar y solucionar la búsqueda eficiente de rutas en una ciudad tan concurrida como lo es Medellín. Lo que se pretende es encontrar una estructura de datos que permita brindar varias opciones de rutas en las cuales se evaluará: la reducción en la distancia a recorrer y el acoso callejero que se pueda vivenciar.

El principal enfoque del proyecto es permitirle al usuario la mejor elección de ruta para llegar a su destino evaluando como se mencionó anteriormente dos características principales, con esto se quiere brindar un poco de confianza y tranquilidad a la hora de movilizarse por la ciudad debido a que se espera solucionar o menguar un problema que ha estado en auge en los últimos años.

Palabras clave

Ruta; mapeo; dirección; estructura de datos; movilidad; listas enlazadas; rehashing.

Palabras clave de la clasificación de la ACM

Data structures design and analysis, sorting and searching, design and analysis of algorithms, Operating systems.

1. Introducción

La evolución tecnológica a la que la sociedad ha estado expuesta en las últimas décadas ha brindado alternativas para hacer la movilidad en espacios sumamente concurridos un poco más práctica, este avance se ha logrado empleando diversos modelos de geolocalización que permiten generar rutas alternas evitando así trancones, accidentes automovilísticos, vías cerradas, entre otros problemas a los que se está expuesto en la vía, además de calcular el tiempo promedio en el desplazamiento. Sin embargo, un problema que aún no se ha abordado desde esta perspectiva – o era muy poco su grado de interés – es el acoso que se vive en las calles por las que los individuos se movilizan; así pues, esta deficiencia en la calidad de movilidad por la ciudad fue lo que dio el inicio a este proyecto.

2. Problema

El problema al cual nos enfrentamos es el aumento en el acoso callejero lo que

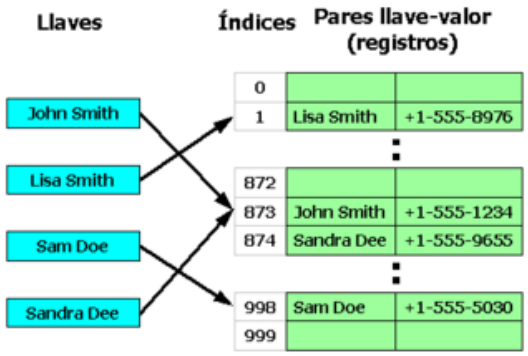
restringe la tranquilidad en el libre desplazamiento por ciertas áreas.

Para ello se hará uso de listas y archivos en los cuales se almacenarán los datos que brindarán información acerca de la mejor ruta, para la elección de la ruta más factible se tendrá en cuenta lo eficaz que sea el desplazamiento – menor tiempo posible – y la seguridad con la que se pueda transitar.

3. Trabajos relacionados

3.1 Borrados y Rehashing

Una tabla hash o mapa hash es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada usando el nombre, número de cuenta o id. De lo anterior se puede decir que la tabla hash almacena un conjunto de pares "(clave, valor)". La clave es única para cada elemento de la tabla y es el dato que se utiliza para buscar un determinado valor (acceder a la posición del elemento).



3.2 GoTo

Es una instrucción cuyo propósito es transferir el control a un punto determinado del código donde debe continuar la ejecución. Las ubicaciones a las que salta la instrucción generalmente se identifican mediante etiquetas . Goto es una instrucción de salto unidireccional.

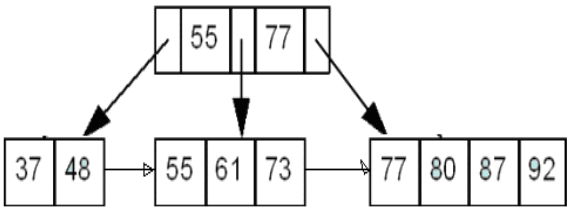
Python no incluye una palabra reservada goto, pero dada la flexibilidad del lenguaje, es posible implementar un operador de esta característica utilizando **Python-goto.**, opera con las palabras label, para definir una determinada porción del código, y goto, para realizar el salto.

3.3 Árbol B+

Es un tipo de estructura de datos de árbol, representa una colección de datos ordenados de manera que se permite una

inserción y borrado eficientes de elementos. Es un índice, multinivel, dinámico, con un límite máximo y mínimo en el número de claves por nodo. En un árbol B+, toda la información se guarda en las hojas. Los nodos internos solo contienen claves y punteros. Todas las hojas se encuentran en el mismo nivel, que corresponde al más bajo. Los nodos hoja se encuentran unidos entre sí como una lista enlazada para permitir principalmente recuperación en rango mediante búsqueda secuencial. Su principal característica es que todas las claves se encuentran en las hojas. Los árboles B+ ocupan algo más de espacio que los árboles B, pues existe duplicidad en algunas claves.

El valor principal de un árbol B+ está en el almacenamiento de datos para una recuperación eficiente en un contexto de almacenamiento orientado a bloques , en particular, sistemas de archivos . Esto se debe principalmente a que, a diferencia de los árboles de búsqueda binarios , los árboles B+ tienen un abanico muy alto (número de punteros a nodos secundarios en un nodo), lo que reduce el número de operaciones de E/S (conjunto de acciones necesarias para la transferencia de un conjunto de datos) necesarias para encontrar un elemento en el árbol.



4. Estructura de datos

4.1 Diseño de la estructura de datos

Dado que una estructura de datos es una forma particular de organizar datos en una computadora para que puedan ser utilizados de manera eficiente, se convirtió en el modelo perfecto que se adapta a las necesidades de este proyecto, es así como al hacer uso de esta estructura de datos se logra satisfacer todos los objetivos que tiene el proyecto de manera correcta; permitiéndonos:

- Analizar el archivo de direcciones registradas.
- Dividir por secciones dichas direcciones en 7 categorías.
- Obtener individualmente cada información de las categorías para los subprocesos.
- Comparar dos direcciones tomando el atributo de coordenadas.
- Imprimir por pantalla direcciones guardadas.

- Situarse en el archivo de direcciones para verificar si posee la dirección en el mapa guardado.

4.2 Operaciones de la estructura de datos

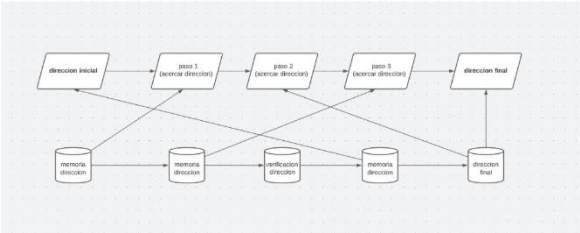


Tabla 1

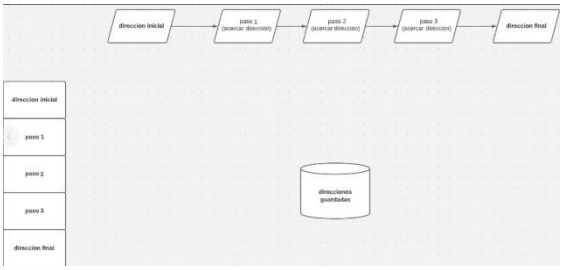


Tabla 2

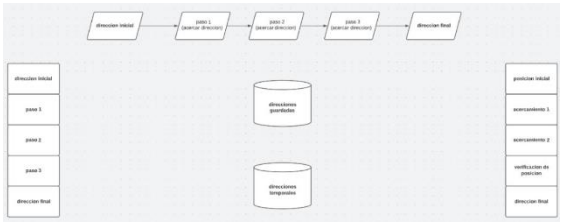


Tabla 3

Tabla 1, 2 y 3: A partir del mapa la estructura de datos recorre el archivo de coordenadas agregando direcciones según la información suministrada

4.3 Análisis de complejidad

metodo	complejidad
vaciar	O(1)
AgregarDireccion	O(1)
AgregarPosicion	O(n*m)
BuscarDireccion	O(n*2)
NuevaPosicion	O(n*m)

Tabla 4: Tabla para reportar la complejidad del algoritmo

4.4 Memoria

conjunto de datos	1	2	3	4
conjunto de memoria	4823400B	6432412B	7954371B	9674321B

Tabla 5: Consumo de memoria en la estructura de datos

Referencias

1. https://en.wikipedia.org/wiki/Goto
2. https://ccia.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb_B3.htm
3. https://recursospython.com/guias-y-manuales/como-utilizar-goto/
4. https://ccia.ugr.es/~jfv/ed1/tedi/cdrom/docs/tablash.html