Pablo Badani
Miguel Molina
Gabriel Neme
Richard Rojas

JAMBAO

Autores:
Pablo Badani
Miguel Molina
Gabriel Neme
Richard Rojas
Asesor:
Dr. Paul Wilker Landaeta Flores
Informe
Universidad Privada Boliviana
Algorítmica I
La Paz, Bolivia
15 de Mayo de 2021

Pablo Badani
Miguel Molina
Gabriel Neme
Richard Rojas

ÍNDICE:

Pablo Badani
Miguel Molina
Gabriel Neme
Richard Rojas

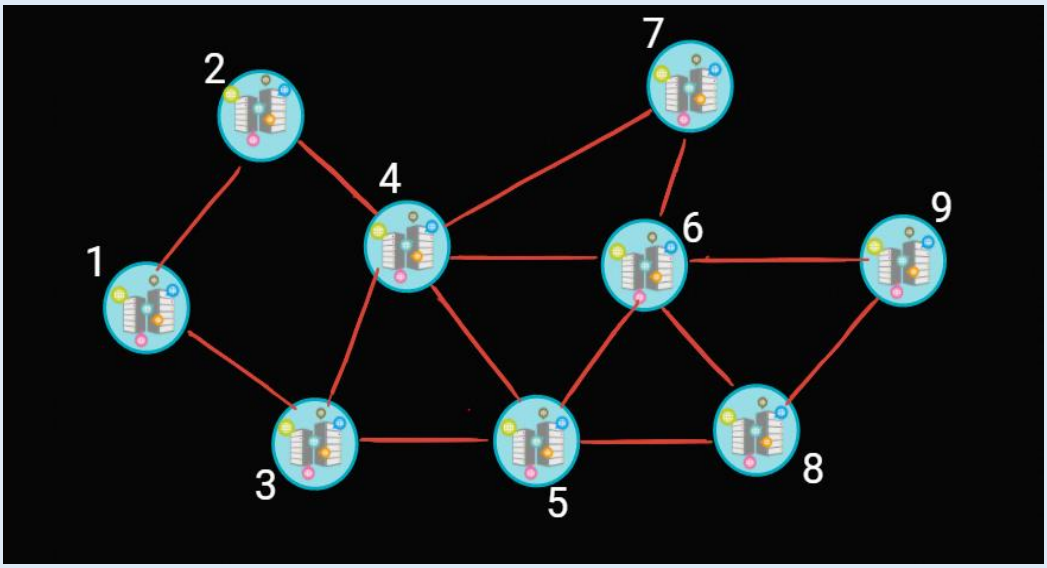
1. Proceso de instalación. -

Para empezar con el proyecto se procede a instalar un editor de código con el que debemos trabajar el programa, una buena opción es Visual Studio Code. Otra alternativa si es que el usuario no cuenta con los recursos necesarios para ejecutar los algoritmos, ya sea por falta de espacio en la memoria RAM o se tiene instalado un procesador no satisfactorio, es que utilice compilador online, este utiliza los recursos de la nube para correr los programas.
Crear una carpeta en donde guardaremos el código trabajado, se recomienda fabricarlo en un lugar fácil de recordar, ya que es muy común perder la ubicación de esta.
Después de realizar todos los pasos, procedemos a crear un repositorio en la página GitHub. Primero creamos una cuenta, o si ya se cuenta con una, ingresamos nuestro usuario; seguido a esto, en la pantalla de inicio nos dirigimos a nuestros repositorios en la barra de navegación, seleccione Create new (+) y luego elija "New repository". En la página Create a new repository (Crear un nuevo repositorio), haga lo siguiente: En el cuadro Repository name (Nombre del repositorio), indique el nombre apropiado para el repositorio. Seleccione Public. Si se selecciona la opción Public (Pública) predeterminada, cualquiera podrá ver este repositorio. Puede seleccionar la opción Private (Privado) para limitar quién puede ver el repositorio y confirmarlo (esto no es gratis).
Active la casilla de verificación "Initialize this repository with a README" (Inicializar este repositorio con un archivo README). Este archivo README.md permite que podamos escribir como una portada o carta de presentación de nuestra página en github, cada persona puede personalizarlo de la manera que cada uno prefiera agregando imágenes, dividiendo en carpetas los archivos que subamos, así para tener todo ordenado y clasificado, también podemos editar esto desde nuestro editor de texto favorito "Vscode". Elija "Create repository" crear repositorio.
"Algo interesante:"
Ahora con nuestro repositorio creado y personalizado tenemos que clonarlo para poder realizar el proceso de conectar nuestro Visual Studio Code con nuestro repositorio en Github, así podremos usar con diferentes comandos en la terminal para subir nuestros proyectos directamente desde visual studio code y lo guardamos en nuestro repositorio.

Pablo Badani
Miguel Molina
Gabriel Neme
Richard Rojas

2. Definición del Proyecto. -

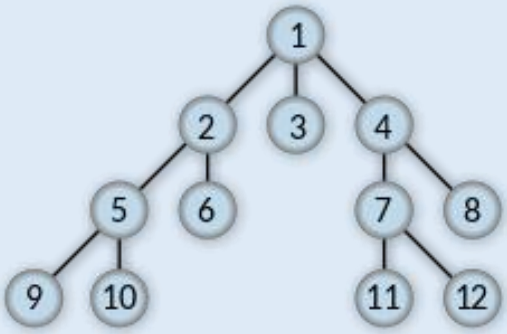
Para nuestro proyecto escogimos el algoritmo BFS (en inglés es Breadth First Search, Búsqueda en Anchura), como temática, formulamos una idea y tomamos un ejemplo en la vida real que es una red de servidores conectados entre sí, ya sea definido o no. La idea surge de preguntarnos de que manera están conectados dichos servidores y como llegan a recorrerse entre sí o llegar a "visitarse" entre sí.
El algoritmo implementado va ayudar a que las personas conozcan un tiempo estimado de llegada de un servidor a otro, verificando si hay una conexión entre el punto inicial y el punto final. Nuestro programa construye una pre visualización del grafo a partir de sus entradas, ayudando a que sea más sencillo visualizar las conexiones entre servidores.
La aplicación de este algoritmo tendrá una variedad de opciones con el que el usuario pueda interactuar. Ya sea que muestre el número de servidores por el que pasó la información, el recorrido que realizó los datos mandados, cual es el único camino para llegar de un nodo a otro o el árbol de recorrido que tendrá toda esta red de servidores. La misión de nuestra propuesta, es facilitar el diagnostico de tiempo tomando un paquete de datos para ir de un punto a otro, así ayudar a que haya menos retrasos al momento de compartir información.
Aquí tenemos un ejemplo gráfico de nuestro proyecto:



3. Explicación del Algoritmo:

El algoritmo BFS (breadth-first search), encuentra los caminos más cortos desde un vértice de origen dado a todos los demás vértices, en términos del número de aristas en los caminos. Es una forma de encontrar todos los vértices alcanzables de un grafo partiendo de un vértice origen dado. Como en el algoritmo de búsqueda en profundidad, BFS recorre una componente conexa de un grafo y define un árbol de expansión.
Podemos ver que todo parte de un nodo inicial que será la raíz del árbol que se forma, luego ve los adyacentes a ese nodo y los agrega en una cola "stack", como la prioridad de una cola es FIFO (primero en entrar es el primero en salir), los siguientes nodos a evaluar serán los adyacentes previamente insertados. una cosa bastante importante es el hecho de que no se

pueden visitar 2 veces el mismo nodo o estado. ya que sino podríamos terminar en un ciclo interminable o simplemente no hallar el punto deseado en el menor número de pasos.



4. Proceso de configuración. –

a) Requisitos de software

Para la ejecución del programa utilizamos distintos tipos de software, ya sea en una computadora Core i7 con 12 GB de Ram, 1 tb de disco sólido y 250 de SSD;
una similar con 8 GB de RAM, o una PC Core i5 de quinta generación, RAM de 8 GB y un tb HDD. Pero incluso en el ordenador con características
muy bajas se logra correr el algoritmo, esta cuenta con 4 GB de RAM, 1 tb de memoria HDD y un procesador RYZEN 3.
Mas con los últimos avances en tecnología, servidores, conexiones a internet más sencillas, se puede programar en línea.
utilizando los recursos de la nube, lo cual ayuda mucho a computadoras con componentes no tan poderosos para el área de la programación.

b) Instrucciones paso a paso para ejecutar el programa

Como ya creamos nuestro repositorio en GitHub, seguimos los siguientes pasos:
- Ingresamos al apartado del código.
- Copiamos el algoritmo al programa que utilizamos para escribir nuestros programas.
- Se tiene casos de ejemplo, y también se puede ingresar su propio caso.
- Al ejecutar solo se debe ver la pantalla de muestra con la resolución del problema propuesto.

c) Explicación del algoritmo

El algoritmo bfs es un algoritmo de búsqueda utilizada en el hallazgo de elementos de un grafo, se toma un punto o nodo de partida
y uno para finalizar, seguido a esto comenzamos viendo cuales son los siguientes nodos conectados a él, que los llamaremos "vecinos".
Una vez encontrados todos, agarramos a uno de los conectados y buscamos de la misma manera sus nodos vecinos. Pero cada vez que vistamos un
punto lo marcamos como visitado y ya no lo tomamos en cuenta al realizar este proceso de nuevo.
Como código se utilizará una cola y un vector "visitado" de tipo booleano. Donde se utilizara el primer arreglo para ir guardando los nodos que vayamos a visitar
para luego sacarlos cada vez que encontremos a todos sus vecinos. Mientras que el segundo arreglo tiene como objetivo evitar la repetición de puntos que ya fueron utilizados.

Pablo Badani
Miguel Molina
Gabriel Neme
Richard Rojas

Una vez terminado esto, tenemos todo el camino para encontrar el camino de un nodo inicial a un punto final.

5. Conclusión. -

Para concluir podemos decir que el algoritmo BFS, aunque es una herramienta útil, no existen muchas aplicaciones en las que se utiliza de manera eficiente. Nuestro proyecto si lo requirió, más en nuestra investigación nos fue difícil hallar una aplicación que no sea la resolución de un laberinto, o encontrar el camino en un cierto mapa. Por otra parte, el hecho de buscar y razonar más las posibles aplicaciones nos ayudó a entender cada vez más su funcionamiento y así poder aplicarlo a alguna situación de nuestra vida, cumpliendo así el objetivo de realizar este proyecto.