Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Lenguajes, Compiladores e Intérpretes
Módulo: Lenguajes

Teacher:
Marco Rivera Meneses

Tarea Funcional #1

Estudiantes:
Adrián Gómez Garro
Gretchell Ochoa Quintero
Marco Rivera Serrano

Grupo:
1

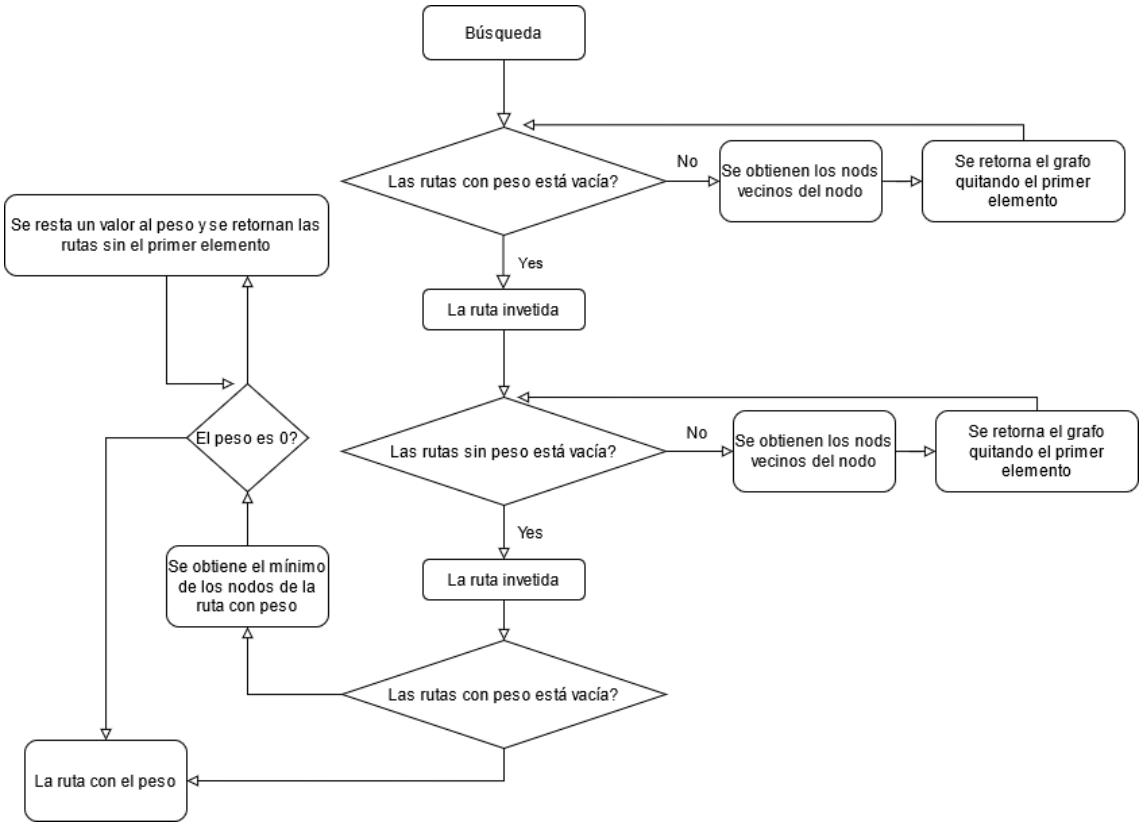
I Semestre

2021

Descripción de los algoritmos desarrollados

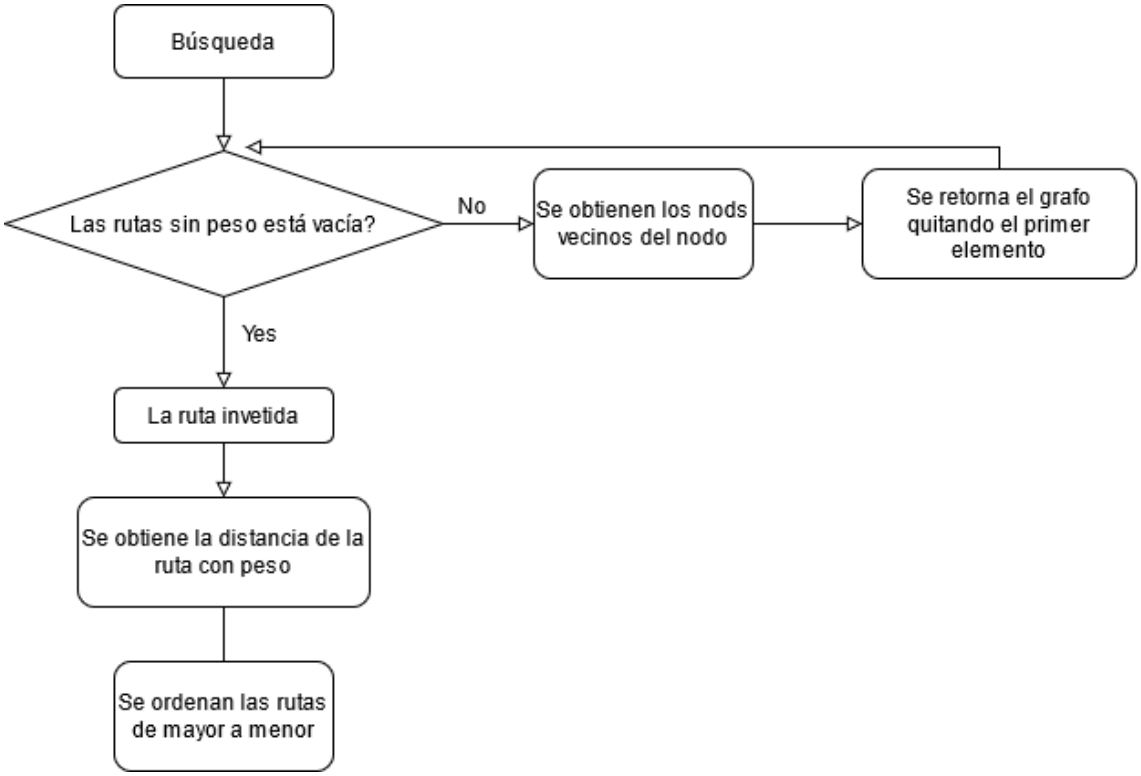
Algoritmo de búsqueda del camino más corto:

Para encontrar la ruta más corta, se utilizan dos listas, una con la ruta con peso y otra con la ruta sin peso. Luego, se consiguen los valores de menor cantidad hasta recorrer toda la ruta y se devuelven los nodos cuyo peso es el mínimo de todos sus vecinos.



Algoritmo de búsqueda de todos los caminos:

Para encontrar todos los caminos se realiza una búsqueda por anchura donde los pesos entre los nodos no importan, dado que se quieren conseguir los caminos sin importar la distancia entre ellos. Una vez se obtienen estos caminos, se retornan dentro de una lista.



Funciones implementadas:

- addNode: añade un nodo al grafo.
- addRoutes: añade una ruta entre nodos, haciéndolos vecinos.
- createRoutes: crea la ruta entre dos nodos como una lista.
- solution?: valida si una ruta llega al nodo de llegada.
- weightedSolution?: valida si una ruta con peso llega al nodo de llagada.
- neighbours: retorna los vecinos del nodo si este existe.
- member?: valida si un elemento existe dentro de una ruta.
- memberWeight?: valida si un nodo con peso existe dentro de la lista.
- checkNeighbours: retorna los vecinos de una ruta dada.
- extend: Retorna el peso de cada vecino de una ruta dada.
- invert: revierte una ruta dada.
- searchAllRoutes: retorna todas las rutas de un nodo de inicio al de llegada.
- profNoWeight: búsqueda por anchura sin importar el peso.
- profWeight: búsqueda por anchura tomando en cuenta el peso.
- getDistance: retorna el peso de un nodo a otro.
- totalDistance: retorna el peso total de la ruta dada.
- minimum: retorna el valor mínimo de una lista.
- shorterRoute: retorna la ruta más corta de un nodo de inicio a uno de llegada.
- allRoutes: retorna todas las rutas.
- insert: implementación del algoritmo de ordenamiento insertion.
- sorter: Retorna una lista con las rutas ordenadas.

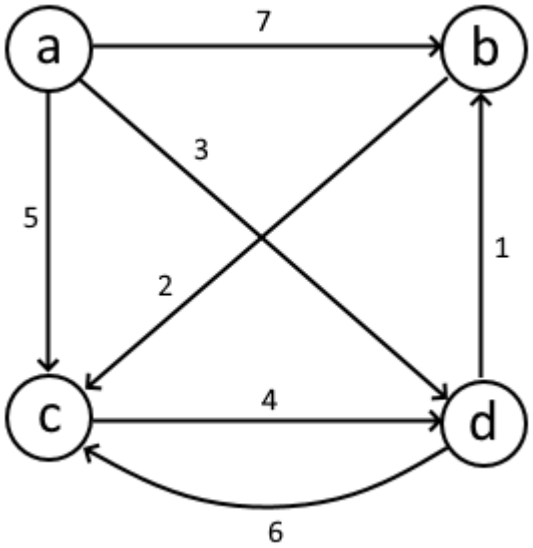
Estructura de datos:

Dentro del programa, un grafo tiene la siguiente estructura:

```
'(("a" (("b" 7) ("c" 5) ("d" 3))) ("b" (("c" 2))) ("c" (("d" 4)))
("d" (("c" 6) ("b" 1))))
```

Donde la estructura general es la siguiente ‘((id_nodo1 ((vecino1 peso) (vecino2 peso))))

Y la representación gráfica del ejemplo anterior es la siguiente:



Problemas sin solución:

Al haber una limitada documentación, no se logró realizar una interfaz deseada.

Si el grafo contiene muchas rutas o nodos presentes, la eficiencia del programa podría verse reducida según el hardware en que se esté corriendo, por lo que se pide mantener un máximo de 20 nodos en el grafo.

Problemas encontrados:

Para obtener las rutas de mayor a menor, se obtuvo por agregarlas según su peso al resultado final.

Para poder obtener todas las rutas, se decidió realizar una implementación del algoritmo de búsqueda por anchura con un acercamiento de paradigma funcional para no utilizar funciones primitivas fuera de este.

Para obtener la ruta más corta, se utilizó el peso de menor tamaño según el nodo, es decir un algoritmo que opte por el menor costo (greedy algorithm).

Plan de actividades:

- Desarrollo de la estructura de datos y búsqueda de información de implementación de grafos dentro del entorno de Racket, asignado a Marco Rivera con fecha límite para el 20 de mayo.
- Desarrollo de una ventana principal que reciba inputs del usuario, asignado a Gretchell Ochoa con fecha límite para el 20 de mayo.
- Desarrollo del algoritmo de búsqueda asignado a Marco Rivera con fecha límite para el 25 de mayo.
- Desarrollo del dibujo del grafo, Adrián Gómez con fecha límite para el 26 de mayo.
- Implementación de todas las partes involucradas, asignado a Adrián Gómez, Gretchell Ochoa, Marco Rivera con fecha límite para el 28 de mayo.

Conclusiones

Al estar en un nuevo entorno de programación como lo es el funcional, es muy complicado para los estudiantes tener una comprensión adecuada del lenguaje y sus funcionalidades. Además, debido a las situaciones actuales, se dificulta mucho la resolución de dudas.

Con respecto a la elaboración de un algoritmo complejo, su implementación puede llegar a ser un desafío para aquellos que no están acostumbrados. Además, se requiere un conocimiento sobre las funciones de car, cdr y sus respectivas extensiones como caar, cddr y combinaciones.

Sobre la interfaz gráfica, presentó un gran desafío dado que la documentación es bastante limitada y debido al tiempo dado, no se puede experimentar mucho en ese ámbito, resultado en errores, interfaces incompletas, inexistentes o que no cumplen con la funcionalidad pedida.

Recomendaciones

Realizar la división de las tareas lo más pronto posible para evitar cualquier atraso o falta de tiempo en las resoluciones de estas.

Realizar una exhaustiva búsqueda sobre las librerías para el manejo y cumplimiento de las interfaces gráficas.

Bibliografía

Algorithms that Backtrack. (26 de 09 de 2003). Obtenido de Section 28: http://htdp.org/2003-09-26/Book/curriculum-Z-H-35.html#node_chap_28

Collen Lewis. (10 de febrero del 2013). Racket Programming: Graph representations: nodes, kids, leaf?, grandkids [Archivo de video]. YouTube. https://www.youtube.com/watch?v=UHh3HDh6Qtw

Collen Lewis. (26 de enero del 2013). Racket Programming: Ways of representing and tracing recursive calls [Archivo de video]. YouTube. https://www.youtube.com/watch?v=v6eR2N-CYwE

Guzmán, J. E. (2005). Introducción a la programación con Scheme. En J. E. Guzmán, *Introducción a la programación con Scheme* (págs. 376-388). Cartago: Editorial Tecnológica de Costa Rica.

Bitácora

18/05/21. Primera reunión, se realizó la distribución de tareas y definición de problemas en general.

20/05/21. Se presentaron las estructuras de datos para el grafo y una ventana inicial.

23/05/21. Se presentan los algoritmos de búsqueda y funciones extra para el funcionamiento correcto.

25/05/21. Se hacen pruebas de conexión entre la interfaz y la lógica