



Instituto Tecnológico de Costa Rica

Escuela de Computación

IC6600 - Investigación De Operaciones GR 60

Profesor: Jean Carlos Miranda Fajardo

**Proyecto Parte #7 - Árboles Binarios de
Búsqueda Óptimos**

Estudiantes:

Ricardo De Jesus Soto Araya - 2020035358

Brandon Guillermo Redondo Jimenez- 2019156567

Francisco González Madrigal - 2018107608

II Semestre - Sede de Limón

17 de noviembre de 2022

Índice

Manual de usuario	3
Instrucciones de compilación.	3
Instrucciones de instalación	3
Instrucciones de ejecución	3
Uso de la aplicación: Abrir módulo de Árbol Binarios de búsqueda óptimos	4
Descripción del problema	9

Manual de usuario

Instrucciones de compilación.

Requisitos previos:

- Tener instalado NodeJS. Node.js (nodejs.org)
- Tener instalado SweetAlert2 SweetAlert

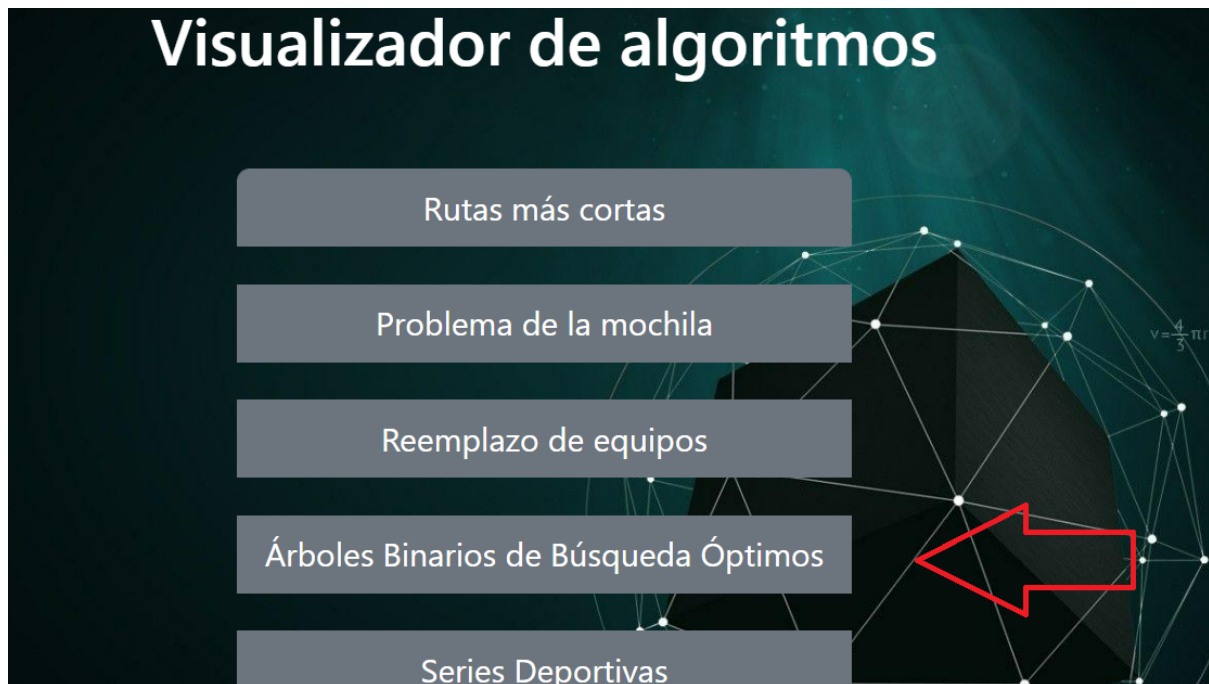
Instrucciones de instalación

- 1- Se descomprime el archivo .ZIP
- 2- Se abre una consola en la dirección de la carpeta descomprimida
- 3- Se escribe npm install para instalar los módulos necesarios de React (Se instalará sweetalert si no está previamente instalado.)

Instrucciones de ejecución

- 1- En consola desde la ubicación del proyecto -> npm start
- 2- Se procederá a mostrar la aplicación el navegador web predeterminado automáticamente.

Uso de la aplicación: Abrir módulo de Árbol Binarios de búsqueda óptimos



1- Se procede a seleccionar la opción de “Árbol Binarios de Búsqueda óptimos”.



2- Se mostrará la vista respectiva del inicio del algoritmo del “Árbol Binarios de Búsqueda óptimos”.

archivo de prueba:

Elegir archivo No se ha s...gún archivo

Crear dinámicamente:

Ingrese el número de llaves



Generar Algoritmo




3- Se procederá a ingresar 5 llaves en el algoritmo.

Ingrese el número de llaves

	Código	Peso
Llave 1	<input type="text"/>	<input type="text"/>
Llave 2	<input type="text"/>	<input type="text"/>
Llave 3	<input type="text"/>	<input type="text"/>
Llave 4	<input type="text"/>	<input type="text"/>
Llave 5	<input type="text"/>	<input type="text"/>

4- Se procederá a llenar información de cada nodo del árbol, en el caso del Código se procederá a ingresar las primeras 5 letras del abecedario.

Ingrese el número de llaves

5

	Codigo	Peso
Llave 1	A	2
Llave 2	B	3
Llave 3	C	4
Llave 4	E	1
Llave 5	F	7

Generar Algoritmo

5- Al ingresar los datos se verá una vista similar a la anteriormente mostrada.

	Codigo	Peso
Llave 1	A	2
Llave 2	B	3
Llave 3	C	4
Llave 4	E	1
Llave 5	F	7

Generar Algoritmo

6- Para mostrar el resultado del algoritmo se procede a generar algoritmos.

Creado dinámicamente:

Tabla R

	1	2	3	4	5	6
1	0	A	2	2	2	3
2		0	B	3	3	3
3			0	C	3	5
4				0	E	5
5					0	F
6						0

Tabla A

	1	2	3	4	5	6
1	0	2	7	15	18	33
2		0	3	10	12	27
3			0	4	6	18
4				0	1	9
5					0	7
6						0

Grabar archivo

7- Finalmente, se mostrarán las tablas R y A respectivamente.

Tabla A

	1	2	3	4	5	6
1	0	2	7	15	18	33
2		0	3	10	12	27
3			0	4	6	18
4				0	1	9
5					0	7
6						0

Grabar archivo

8- Si se desea descargar el archivo para utilizarlo en un futuro, se procede a seleccionar "Grabar archivo" para así descargar un archivo .txt

Arboles Binarios de Búsqueda Optimos

Seleccionar un archivo de prueba:

Elegir archivo No se ha seleccionado ningún archivo

Crear dinámicamente:

Ingrese el número de llaves

9- Para ingresar el archivo se procede a seleccionar el botón de "Elegir archivo" y se procede a elegir el archivo de .txt recientemente descargado.

Descripción del problema

Se requiere crear una nueva función dentro de la aplicación creada anteriormente en el proyecto la cual es la creación de un algoritmo capaz de calcular árboles binarios de búsqueda óptimos, por lo cual hay que programar en Javascript el código que servirá para solucionar el algoritmo anteriormente mencionado y así mostrarlo a su vez por múltiples componentes hechos en la librería de React.

Solución del problema

La solución realizada fue la utilización de múltiples componentes de React tanto funcionales como nativos de ES6, esto de forma que los nativos de ES6 son utilizados para realizar los cálculos del algoritmo, a su vez, los componentes funcionales fueron utilizados como componentes que serán reutilizables a lo largo de futuros cambios de la aplicación para así optimizar el tiempo de desarrollo y brindar más atención a la solución de los próximos algoritmos en lugar de programar como mostrarlos.

Con respecto al código utilizado para solucionar el algoritmo, lo que se procedió fue realizar un plan de exactamente qué procesos se realizan al elaborarlo a mano, después se procedió a realizar la codificación del mismo y finalmente la implementación del mismo en la parte visual de la aplicación para así llegar a la solución deseada y poderla mostrar al usuario por medio de los componentes anteriormente mencionados.

Análisis de resultados

Requerimiento	Estado	Observaciones
Ingresar número de cantidad de llaves/nodos	Completado	Funciona correctamente en cada una de sus funcionalidades.
Ingresar los códigos de cada llaves/nodos	Completado	Funciona correctamente en cada una de sus funcionalidades.
Ingresar los pesos de cada llaves/nodos	Completado	Funciona correctamente en cada una de sus funcionalidades.
Generar el algoritmo	Completado	Funciona correctamente en cada una de sus funcionalidades.
Mostrar tabla P	Completado	Funciona correctamente en cada una de sus funcionalidades.
Mostrar tabla A	Completado	Funciona correctamente en cada una de sus funcionalidades.
Guardar los datos de un problema anteriormente solucionado	Completado	Funciona correctamente en cada una de sus funcionalidades.
Cargar los datos de un problema anteriormente solucionado	Completado	Funciona correctamente en cada una de sus funcionalidades.