



Primera Evaluación

Inteligencia Artificial 3CV15

Integrantes:

Alejandra Orozco Aguilar

Omar Daniel González Trejo

Árbol de Búsqueda



Consideraciones

- El total de nodos no puede ser mayor a 30
- El nodo meta debe ser especificado por el usuario después de haber creado el árbol
- La amplitud del árbol es igual al nivel que tenga el mayor número de nodos
- La profundidad es la ramificación o extensión de nodos más larga
- El número de nodos hijos siempre es igual al número de nodos totales menos uno
- El nodo raíz siempre es el nodo A
- Cada nivel es diferenciado por un color diferente

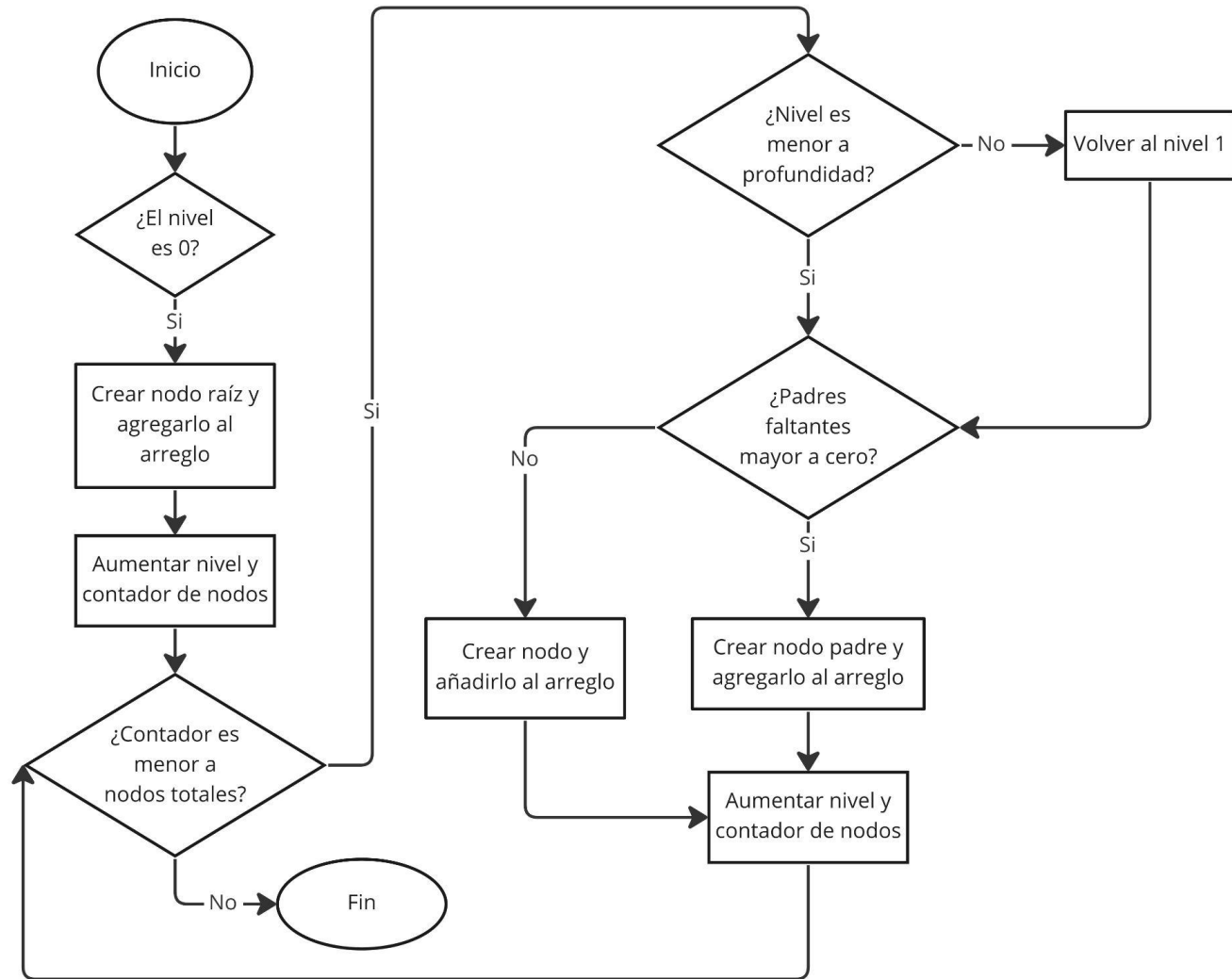


Nodos

Para la creación de los nodos se estableció una función prototipo llamada “Nodo” la cual contiene todas las características que tendrán cada uno de los nodos.

```
//Nodos plantilla para los nodos
function Nodo(idNodo,padre,x,y,nivel){
    this.idNodo = idNodo;
    this.padre = padre;
    this.hijos = [];
    this.x = x;
    this.y = y;
    this.nivel = nivel;
    this.color = colores[nivel];
    if(nivel == 0){
        this.raiz = true;
    }else{
        this.raiz = false;
    }
    this.meta = false;
    this.explorado = false;
}
```

Creación de los nodos.





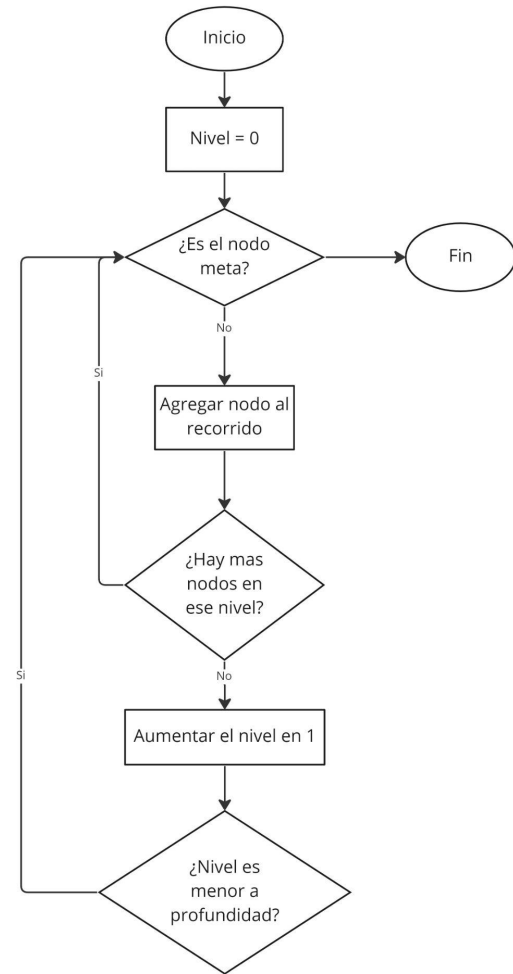
Código para la creación de nodos

El código de la derecha es la implementación del diagrama de flujo anterior.

Las coordenadas se calculan con base al nivel y el número de nodos en este, se le asigna un id único conforme se van creando y el padre se asigna tomando en consideración si ya se cumplió con el número de padres solicitado por el usuario.

```
118 //Crear todos los Nodos y los agrega a un arreglo
119 function crearNodos(){
120     let x;
121     let y;
122     let nRaiz
123     //Nodo Raiz (nivel 0)
124     if(nivel == 0){
125         x = (canvas.width-200)/2;
126         y = 50;
127         nRaiz = new Nodo(ids[cont],null,x,y,nivel);
128         cont ++;
129         nivel ++;
130         nodos.push(nRaiz);
131     }
132     while (cont < numNodos) {
133         while (nivel < (profundidad)) {
134             x = (distribucionX()*(nodosPorNivel(nivel)))+100;
135             y = (distribucionY()*(nivel+.5));
136             let nodo;
137             if(nivel == 1){
138                 nodo = new Nodo(ids[cont],nRaiz,x,y,nivel);
139                 nRaiz.hijos.push(nodo);
140             }else{
141                 let index = asignarPadre();
142                 if(index != -1){
143                     nivel = nodos[index].nivel+1;
144                     x = (distribucionX()*(nodosPorNivel(nivel)))+100;
145                     y = (distribucionY()*(nivel+.5));
146                     nodo = new Nodo(ids[cont],nodos[index],x,y,nodos[index].nivel+1);
147                     nodos[index].hijos.push(nodo);
```

Búsqueda por amplitud





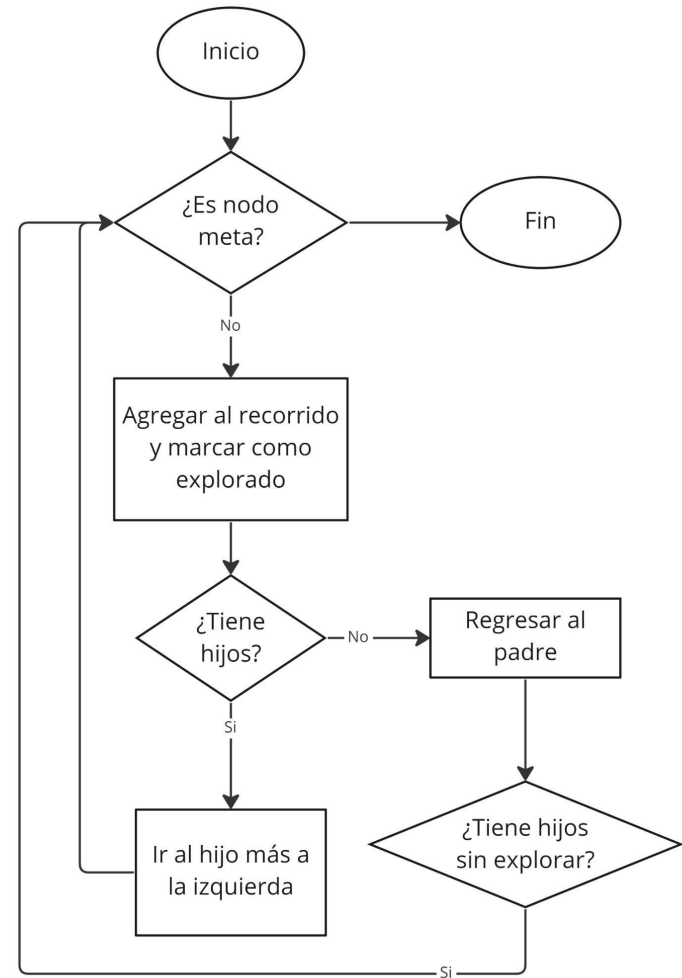
Código de la búsqueda por amplitud

Para la búsqueda por amplitud se van recorriendo todos los nodos de un mismo nivel en busca del nodo meta.

Si no se encuentra en el nivel actual busca en el nivel siguiente y así sucesivamente hasta llegar a la meta.

```
//Amplitud
case "1":
    for (let i = 0; i < profundidad; i++) {
        for (let j = 0; j < nodos.length; j++) {
            if (nodos[j].nivel == i) {
                if (nodos[j].idNodo == meta) {
                    recorrido.push(nodos[j].idNodo);
                    nodos[j].meta=true;
                    flag = true;
                    break;
                }else{
                    recorrido.push(nodos[j].idNodo);
                }
            }
        }
        if(flag == true){
            break;
        }
    }
    imprimirRecorrido(recorrido);
    break;
```


Búsqueda por profundidad





Código de la búsqueda por profundidad

Para este método hacemos uso de la variable “explorado” que se encuentra en todos los nodos,

Si el nodo ya fue explorado no se agrega al recorrido y se revisa entonces su hijo más a la izquierda, si el nodo no tiene hijos o todo ya fueron explorados vuelve a su padre y repite el proceso hasta encontrar la meta

```
//Profundidad
case "2":
    recorrido.push(nodos[0].idNodo);
    nodos[0].explorado = true;
    if(nodos[0].hijos.length != 0){
        for(let i = 0; i < nodos[0].hijos.length; i++){
            if(nodos[0].hijos[i].idNodo != meta){
                if (nodos[0].hijos[i].explorado == false) {
                    recorrido.push(nodos[0].hijos[i].idNodo);
                    nodos[0].hijos[i].explorado = true;
                    explorar(nodos[0].hijos[i]);
                }
            }else{
                recorrido.push(nodos[0].hijos[i].idNodo);
                imprimirRecorrido(recorrido);
                break;
            }
        }
    }
    break;
```

Tabla de Percepciones de la creación del árbol

Secuencia de percepciones	Factores para tomar la decisión	Pantalla	Acción
No se ha llenado algún valor del formulario	Entradas del formulario	“Por favor llena todos los campos”	Mostrar alerta
Tiene demasiados nodos	Amplitud Profundidad	“Tu número de nodos es muy grande para esa amplitud y profundidad”	Mostrar alerta
Tiene muy pocos nodos	Amplitud Profundidad	“El número total de nodos es muy pequeño”	Mostrar alerta
Tiene demasiados nodos padres	Nodos totales Amplitud Profundidad	“El número de nodos padres es muy grande”	Mostrar alerta

Tabla de Percepciones de la creación del árbol

Secuencia de percepciones	Factores para tomar la decisión	Pantalla	Acción
Tiene muy pocos nodos padres	Nodos totales Amplitud Profundidad	“El número de nodos padres es muy pequeño”	Mostrar alerta
Hay mas de 30 nodos	Nodos totales	“Este programa acepta a lo mucho 30 nodos ”	Mostrar alerta
Click en el botón generar árbol	Botón	Mostrar árbol	Dibujar árbol
Click en el botón limpiar	Botón	Limpiar formulario y canvas	Reiniciar pantalla
Click en buscar	Botón	Aparece el recorrido	Mostrar recorrido

Estacionamiento



Consideraciones

- El número de cajones debe de ser mayor a 0 y menor o igual a doce
- Cuando se agoten los lugares muestra una alerta con un mensaje
- El id del cajón debe ser mayor a 0 y menor al número de lugares disponibles
- Si se intenta sacar el auto de un cajón donde no hay auto, se muestra una alerta

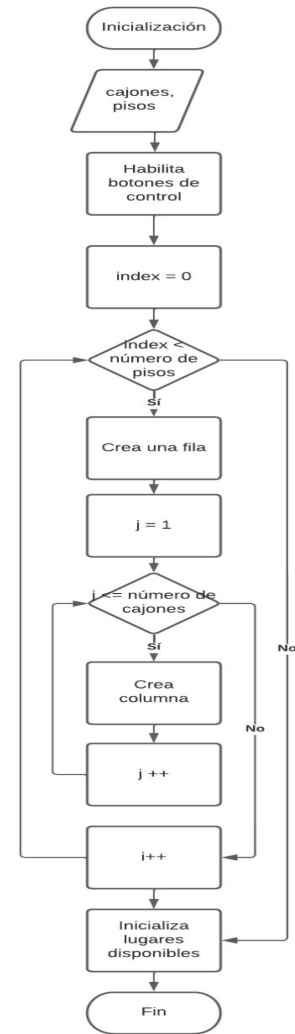



Inicialización

- I. Obtén número de cajones y pisos
- II. Habilitar botones de control
- III. Por cada número de pisos se crea una fila y por cada fila se agregan tantos cajones como número de cajones
- IV. Inicial los lugares disponibles

```
let row, col;
//Crea y dibuja el estacionamiento.
for (let i = 0; i < noPisos; i++) {
  row = document.createElement("div");
  row.className = "row";
  estacionamiento.appendChild(row);
  for (let j = 1; j <= noCajones; j++) {
    col = document.createElement("div");
    col.className = "col bg-success m-2 p-2 text-center fs-3 fw-bold text-white";
    col.innerText = j + i * noCajones;
    row.appendChild(col);
    lugares.push(col);
  }
}
//Crea limite para los autos que se pueden sacar
idAuto.setAttribute("max", lugares.length);
//Inicia los lugares disponibles
lugaresDisponibles = lugares.length;
mensajeLugares.innerText = `Hay ${lugaresDisponibles} lugares disponibles`;
```

Inicialización



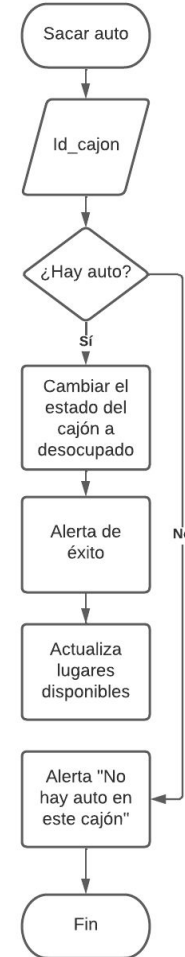


Sacar auto

- I. Obtén ID del cajón
- II. Verifica si hay un auto en dicho cajón
- III. Si no hay auto, genera una alerta
- IV. Si hay auto, cambia el estado del cajón a desocupado e incrementa la cantidad de lugares disponibles

```
let alerta;  
e.preventDefault();  
/**  
 * Id del auto que se desea sacar  
 * @type {Number}  
 */  
const noAuto = document.querySelector('[name="no_auto"]').value;  
//Si en el lugar no hay ningún auto, generas una alerta y la muestras por 1 segundo.  
if (lugares[noAuto - 1].classList.contains("bg-success")) {  
    alerta = creaAlerta(`No hay ningun auto en el lugar ${noAuto}`, "alert-danger");  
    estacionamiento.insertAdjacentElement("afterend", alerta);  
    setTimeout(() => {  
        alerta.remove();  
    }, 1000);  
} else { //Cambia el estado del cajón a desocupado e incrementa la cantidad de lugares disponibles  
    lugares[noAuto - 1].classList.remove("bg-danger");  
    lugares[noAuto - 1].classList.add("bg-success");  
    lugaresDisponibles++;  
    mensajeLugares.innerHTML = `Hay ${lugaresDisponibles} lugares disponibles`;  
}
```

Sacar auto





Estacionar

- I. Busca si hay lugares disponibles
- II. Si hay lugar, estaciona el auto, marca el lugar como ocupado y manda alerta de éxito
- III. Actualiza los lugares disponibles
- IV. Verifica si el auto se pudo estacionar, sino manda alerta de error

```
let alerta;  
let estacionado = false;  
//Busca un lugar disponible y estaciona el auto si es posible  
for (let i = lugares.length - 1; i >= 0; i--) {  
  if (lugares[i].classList.contains("bg-success") && lugaresDisponibles > 0) {  
    //Cambia el estado a ocupado  
    lugares[i].classList.remove("bg-success");  
    lugares[i].classList.add("bg-danger");  
    //Muestra alerta por 1 segundo  
    alerta = creaAlerta("Auto estacionado exitosamente", "alert-success");  
    estacionamiento.insertAdjacentElement("afterend", alerta);  
    setTimeout(() => {  
      alerta.remove();  
    }, 1000);  
    estacionado = true;  
    lugaresDisponibles--;  
    mensajeLugares.innerHTML = `Hay ${lugaresDisponibles} lugares disponibles`;  
    break;  
  }  
}  
//Verifica si el auto se pudo estacionar en algún lugar disponible, sino alerta al usuario.  
if (!estacionado) {  
  alerta = creaAlerta("No hay más lugares disponibles", "alert-danger");  
  estacionamiento.insertAdjacentElement("afterend", alerta);  
  setTimeout(() => {  
    alerta.remove();  
  }, 1000);  
}
```

Estacionar



Tabla de percepciones del estacionamiento

Secuencia de percepciones	Factores para tomar la decisión	Pantalla	Acción
El número de cajones está fuera de rango	Número de cajones	"El valor debe de ser inferior o igual a 12"	Mostrar alerta
Se estaciona el auto en algún cajón disponible	Cajón disponible	"Auto estacionado exitosamente"	Mostrar alerta
Estacionarse cuando no hay más lugares disponibles	Lugares disponibles	"No hay más lugares disponibles"	Mostrar alerta
Se intenta sacar un auto de un cajón vacío	Estado del cajón	"No hay ningún auto en el lugar #id"	Mostrar alerta
Se saca un auto con éxito	Estado del cajón	"Auto sacado con éxito"	Mostrar alerta



Instrucciones para árbol

1. Ingresa la amplitud, profundidad, número total de nodos y número total de nodos padre
2. Da click en generar árbol
3. Si se muestra una alerta, cambiar los valores de los campos hasta que sean valores válidos para la generación del árbol
4. Una vez generado el árbol, ingresa el identificador de un nodo en mayúsculas para poder hacer los recorridos
5. Selecciona un recorrido
6. Da click en buscar y espera el resultado
7. Si se quiere hacer otra búsqueda, se puede cambiar el nodo meta y a su vez seleccionar otro tipo de búsqueda



Instrucciones para el estacionamiento

1. Ingresa número de pisos y número de cajones. El programa no se ejecutará si se ingresan valores inválidos.
2. Una vez ingresados los valores da click en iniciar para poder ver el estacionamiento
3. Para estacionar un auto, solo da click en estacionar y se mostrará un mensaje indicando el estado del auto en el cajón. Si ya no hay lugares disponibles, se mostrará una alerta que indica por lo que se tiene que sacar algún auto ya estacionado.
4. Para sacar un auto solo ingresa el identificador del cajón y da click en sacar auto
5. Finalmente para detener el programa solo da click en detener y se deshabilitarán los botones. Para habilitarlos de nuevo es necesario ingresar nuevos datos o los mismos, y dar clic en Inicio.