



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE COMPUTO

Proyecto final: líneas de producción

Los últimos:

**Torres Trejo Victor Federico
Sotelo Padrón Lara Leilani
Sánchez Flores Guillermo**

PROFESOR

Edgardo Adrián Franco Martínez

ASIGNATURA

Análisis y Diseño de Algoritmos

22 de diciembre de 2021

Índice general

1. Planteamiento del algoritmo y solución	3
1.1. Problema planteado	3
1.2. Solución en fuerza bruta	3
1.3. Solución en Programación Dinámica	4
1.4. Tecnologías web	5
2. Representación del algoritmo	7
2.1. Bocetos	7
2.1.1. Solución en fuerza bruta	7
2.1.2. Solución en Programación dinámica	7
3. Pruebas	9
4. Anexos	11
4.1. Código en fuerza bruta con C	11
4.2. Código en Programación Dinámica con C++	12
4.3. Código en JavaScript	13
4.3.1. Código para la Fuerza bruta	13
4.3.2. Código para la Programación dinámica	25

Capítulo 1

Planteamiento del algoritmo y solución

1.1. Problema planteado

Existe una fabrica de automóviles que tiene dos lineas de montaje, cada una con n estaciones.

Una estación se indica con los sub-índices i y j donde i es 1 o 2 que es la linea de montaje en la se encuentra dicha estación y j es el numero de la estación, cada tiempo se indica entonces por $a[i][j]$. Cada estación tiene un trabajo especifico, por lo que cada chasis de automóvil debe pasar por cada una de las n estaciones para salir de la fábrica. Las estaciones paralelas en cada linea realizan el mismo trabajo. Después de pasar por estación i y j , el chasis continua a la estación i ($j+1$) a menos que se decida transferirse a otra linea.

El continuar en la misma linea no tiene un costo adicional, pero transferirse de la linea i en la estación $(j - 1)$ a la otra linea en la estación j lleva un tiempo denotado por $t[i][j]$. Cada linea de montaje tiene su tiempo de entrada $e[i]$ y un tiempo de salida $x[i]$ que pueden ser distintas para las dos lineas.

1.2. Solución en fuerza bruta

La solución en bruta implica explorar todos las rutas posibles que pueda seguir el chasis de automóvil, al final tenemos 2^n resultados posibles y comparamos entre todos para ver con cual nos quedamos. El código se ve de la siguiente manera:

```
int solbruta(int a[2][n], int t[2][n - 1], int x[], int i, int camino,
            int linea) {
    printf("(%d, %d) camino: %d\n", linea, i, camino);
    if (i >= n)
        return camino + x[linea];
    int quedarse = 0, cambiarse = 0;
    // nos quedamos en la linea
    quedarse = solbruta(a, t, x, i + 1, camino + a[linea][i], linea);
    // nos cambiamos de linea
    cambiarse = solbruta(a, t, x, i + 1, camino + t[linea][i] + a[linea][i],
                        (linea == 0) ? 1 : 0);
```

```

    return min(quedarse, cambiarse);
}

```

Y se llama a la función de la siguiente manera:

```
printf("%d", min(solbruta(a, t, x, 0, e[0], 0), solbruta(a, t, x, 0, e[1], 1)));
```

El inconveniente de esta solución es que tenemos que explorar todas las rutas posibles, aun que no tiene sentido hacerlo se exploran hasta llegar al final del camino, esto tanto en computación como en la vida real es muy cansado e innecesario de realizar, es por ello que se emplea una solución usando Programación Dinámica.

1.3. Solución en Programación Dinámica

Viendo que la solución en fuerza bruta es muy tardada, tratamos de usar el concepto de "memoización" para no explorar todas las soluciones. Para usar esta técnica nos apoyamos en un dos arreglos de tamaño n , en cada casilla vamos guardando el mejor camino para ese punto de la estación desde la estación 1 hasta la n . Esto se hace obteniendo que camino nos sale mas barato, si seguir en la linea actual o desde la otra linea cambiarnos a la que estamos analizando, una vez obtenido el mínimo de estas dos, se guardan en la tabla y continuamos, en la siguiente iteración preguntamos si el camino acumulado para llegar a la estación $i - 1$ es mejor seguir o de la estación $(i - 1)$ de la otra linea es mejor cambiarnos a la actual, esto lo hacemos para todas las estaciones posibles para ambas lineas, de tal manera que en ambas casillas $(n - 1)$ se tiene el mejor camino posible para ambas lineas, al final se devuelve el camino mínimo entre estas dos casillas sumándole el tiempo de salida. Como podemos apreciar este algoritmo tardara n veces, y si nosotros comparamos una función gráficamente de 2^n vs n claramente es mejor la lineal, y es por ello que la solución en Programación dinámica es mas eficiente. Ademas que es parecido a lo que haría una persona en resolver este problema sin el uso de la computación. Comparar si es mejor quedarse o cambiarse e ir usando esas soluciones para armar una general, que es el concepto de la programación dinámica. Cabe destacar que esto es un enfoque top-down.

```

int lineasProduccion(int** a, int** t, int *e, int *x, int n) {
    int* linea1 = new int[n], *linea2 = new int [n], i;

    linea1[0] = e[0] + a[0][0];
    linea2[0] = e[1] + a[1][0];

    for (i = 1; i < n; ++i) {
        linea1[i] = min(linea1[i - 1] + a[0][i], linea2[i - 1] + t[1][i] + a[0][i]);
        linea2[i] = min(linea2[i - 1] + a[1][i], linea1[i - 1] + t[0][i] + a[1][i]);
    }
    return min(linea1[n - 1] + x[0], linea2[n - 1] + x[1]);
}

```

Ahora podemos ver que solo usamos la $(i - 1)$ casilla para armar la siguiente solución, por lo que visto desde la complejidad espacial que es el uso de memoria, las demás casillas pueden ser innecesarias después de 2 iteraciones, por ello usando 4 variables nosotros almacenamos ese mejor camino para la primer y segunda linea, posteriormente ese lo usamos para calcular el de la mejor linea de la siguiente estación y una vez que ya construimos esa solución, actualizamos los valores para volver a usarlo en la siguiente iteración. En esta solución vemos que solo usamos 4 variables en lugar de $2n$, por lo que este código es mas eficiente en cuestión de memoria.

```
int lineasProduccion(int** a, int** t, int *e, int *x, int n) {
    int first, second, up, down;
    first = e[0] + a[0][0];
    second = e[1] + a[1][0];
    for (int i = 1; i < n; i++) {
        up = std::min(first + a[0][i], second + t[1][i] + a[0][i]);
        down = std::min(second + a[1][i], first + t[0][i] + a[1][i]);
        first = up;
        second = down;
    }
    return std::min(first + x[0], second + x[1]);
}
```

1.4. Tecnologías web

Para poder representar la animación web no se hizo uso de ninguna librería de animación web, se hizo el uso de los conceptos de funciones asíncronas y **await** para resolver la animación. Esto debido a que necesitábamos que el carro se moviera a cierta "posición" en la pagina vista como un tablero. Por lo que transportar el carro era sencillo, solo es modificar cuestiones de propiedades de estilos de cada elemento, pero darle la sensación de animación es lo complicado, por ello es que donde entra await, esta función en JavaScript hace que el programa se espere hasta que la promesa responda y devuelva su resultado. Esta promesa devuelve su resultado una vez que la función **setTimeout** haya concluido, esta función espera una cantidad de segundos dados y ejecuta su porción de código una vez que se cumplió el tiempo establecidos en mili-segundos.

En resumen con el uso de await lo que logramos es que el código espere unos segundos para seguirse ejecutando, con ello movemos el carro lentamente y con la función sleep y el uso de await le damos pausa y un sentido de movimiento, ya que se mueve varias veces en una instrucción. Como en si el await solo funciona en funciones async, se tuvo que hacer la mayoría del código en funciones async para darle ese sentido de mono hilo, también para dar pausas entre movimientos usamos el await sleep. En si en esto consiste nuestra animación, modificaciones con el uso de DOM a los objetos del html y el uso de async/await para lograr esa percepción de movimiento.

Esta es la función sleep desarrollada:

```
function sleep(ms) {
```

```
    return new Promise(resolve => setTimeout(resolve, ms));  
}
```

Y la función que logra animar cierto coche:

```
async function animateCar(x1, y1, ncar) {  
    for (let i = 0; i < 20; i++) {  
        if(ncar == 1){  
            moveCar1(x1, y1);  
        }  
        else if(ncar == 2){  
            moveCar2(x1, y1);  
        }  
        else if(ncar == 3){  
            moveCar3(x1, y1);  
        }  
        else if(ncar == 4){  
            moveCar4(x1, y1);  
        }  
        else if(ncar == 5){  
            moveCar5(x1, y1);  
        }  
        else if(ncar == 6){  
            moveCar6(x1, y1);  
        }  
        else if(ncar == 7){  
            moveCar7(x1, y1);  
        }  
        else if(ncar == 8){  
            moveCar8(x1, y1);  
        }  
        await sleep(100);  
    }  
}
```

Capítulo 2

Representación del algoritmo

2.1. Bocetos

2.1.1. Solución en fuerza bruta

Para la solución bruta la animación fue sencilla de proponer o diseñar, es solo crear el dibujo de las líneas de producción con sus respectivas estaciones e ir haciendo mover todos los carros, para que al final se calcularan los 2^n resultados y pudiéramos ver cual era el optimo. Cada carro conforme visitaba los nodos se iba a actualizar su valor.

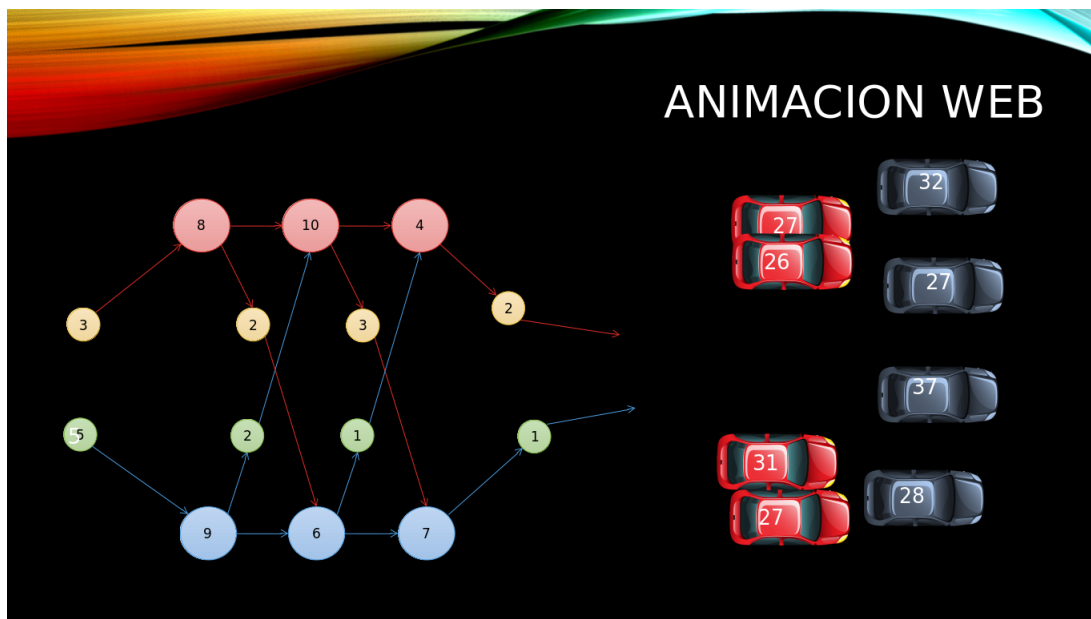


Figura 2.1: Boceto de la animación de la fuerza bruta

2.1.2. Solución en Programación dinámica

Para la solución en DP iba a ser similar a la bruta, porque al final si explora ciertas soluciones pero no todas, por lo que solo era cuestión de ver cual era el mejor camino para

así generar, eliminar y mover los carros, para esto había solo 4 casos:

1. ambos carros se quedaran en su linea, entonces avanzaban sin problema
2. el carro de la linea 1 se quedara ahí y para la linea 2 trajéramos el de la linea 1, que era duplicar el carro de la linea 1 y remover el actual de la 2
3. El carro de la linea 2 siguiera en su linea pero el de la linea 1 se tenía que traer desde la 2, por lo que eliminamos el carro actual de la 1 y duplicamos el de la 2 para llevar el duplicado a la 1.
4. Por ultimo era que para ambas lineas teniamos que cambiar de linea, por lo que se eliminaban y duplicaban ambos carros y se cambiaban de linea cada duplicado.

Una vez esto cuando salimos del ciclo for, llevamos los carros al final de la estación, y por ultimo eliminamos el que mas tiempo tardo.

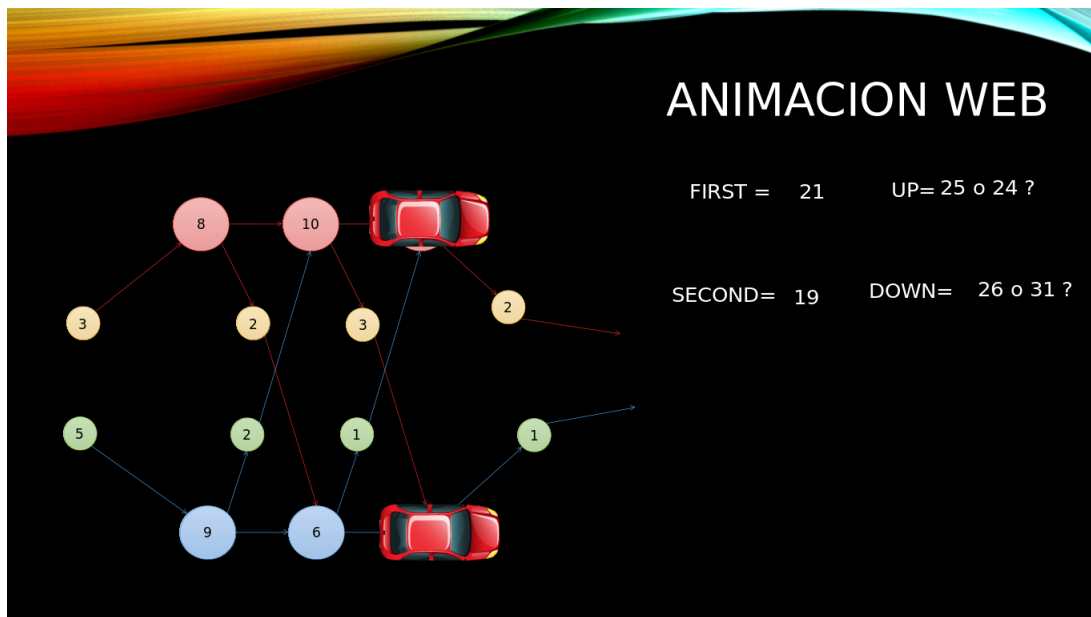


Figura 2.2: Boceto de la animación de Programación dinámica

Capítulo 3

Pruebas

Animación Programación Dinamica

Ingrese datos:

Estaciones 1	<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="3"/>
Estaciones 2	<input type="text" value="4"/>	<input type="text" value="5"/>	<input type="text" value="6"/>
Entrada	<input type="text" value="7"/>	<input type="text" value="8"/>	
Salida	<input type="text" value="9"/>	<input type="text" value="2"/>	
Intercambio 1-2	<input type="text" value="1"/>	<input type="text" value="2"/>	
Intercambio 2-1	<input type="text" value="3"/>	<input type="text" value="4"/>	
Llamadas a minimo	<input type="text" value="5"/>		

Aplicar

SOLUCION

¿PORQUE ES DP?

DEMO

First: Second:

Up: Down:

Costo: 1

Costo: 2

Costo: 3

Costo: 7

Costo: 9

Costo: 8

Costo: 2

Costo: 4

Costo: 5

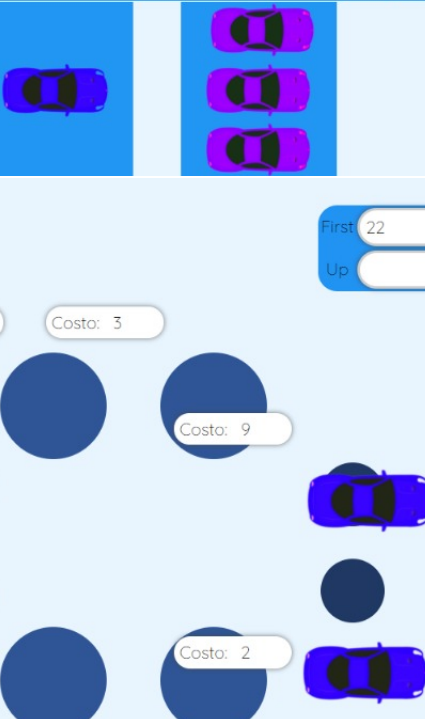
Costo: 6

1

2

3

4



Animación Fuerza Bruta

Ingrese datos:

Estaciones 1	<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="3"/>
Estaciones 2	<input type="text" value="4"/>	<input type="text" value="5"/>	<input type="text" value="6"/>
Entrada	<input type="text" value="7"/>	<input type="text" value="8"/>	
Salida	<input type="text" value="9"/>	<input type="text" value="2"/>	
Intercambio 1-2	<input type="text" value="1"/>	<input type="text" value="2"/>	
Intercambio 2-1	<input type="text" value="3"/>	<input type="text" value="4"/>	
Resultados	<input type="text" value="17"/>	<input type="text" value="21"/>	
Llamadas a minimo	<input type="text" value="15"/>		

[SOLUCION](#)
[¿PORQUE ES FUERZA BRUTA?](#)
[DEMO](#)

[Aplicar](#)

Carro Naranja

Carro Verde

Carro Azul

Carro Morado

Costo: 1 Costo: 2 Costo: 3

Costo: 7

1

2

Costo: 9

3

4

Costo: 8

5

6

Costo: 2

Costo: 4 Costo: 5 Costo: 6

Capítulo 4

Anexos

4.1. Código en fuerza bruta con C

```
#include <stdio.h>
#include <stdlib.h>
#define n 5

/**
 * Devuelve el minimo dados dos enteros dados
 * @param a primer numero a comparar
 * @param b el segundo numero a comparar
 * @return el menor numero de ambos
 */
int min(int a, int b) { return (a <= b) ? a : b; }

/**
 * Resuelve el problema de lineas de produccion explorando todos los caminos
 * posibles (quedarse en la linea actual o cambiarse) y devuelve el menor
 * de ambos caminos
 * @param a los tiempos que se tarda cada estacion en hacer un trabajo
 * @param t los tiempos que tarda cambiarse de una linea a otra
 * @param x los tiempos de salida de cada linea
 * @param i el numero de estacion que estamos explorando
 * @param camino el tiempo acumulado dado el camino tomado
 * @param linea la linea de produccion en la que estamos
 * @return el menor tiempo posible de todos los caminos posibles
 */
int solbruta(int a[2][n], int t[2][n - 1], int x[], int i, int camino,
             int linea) {
    printf("(%d, %d) camino: %d\n", linea, i, camino);
    if (i >= n)
        return camino + x[linea];
    int quedarse = 0, cambiarse = 0;
```

```

    // nos quedamos en la linea
    quedarse = solbruta(a, t, x, i + 1, camino + a[linea][i], linea);
    // nos cambiamos de linea
    cambiarse = solbruta(a, t, x, i + 1, camino + t[linea][i] + a[linea][i],
                        (linea == 0) ? 1 : 0);
    return min(quedarse, cambiarse);
}

int main() {

    int a[2][n] = {{8, 10, 4, 5, 9}, {9, 6, 7, 5, 6}}; // t de las estaciones
    int t[2][n - 1] = {{2, 3, 1, 3}, {2, 1, 2, 2}}; // cambio de estacion
    int e[2] = {3, 5}; // entrada a la linea
    int x[2] = {2, 1}; // salida de la linea
    // Que camino inicial es mejor?
    printf("%d", min(solbruta(a, t, x, 0, e[0], 0), solbruta(a, t, x, 0, e[1], 1)));
    return 0;
}

```

4.2. Código en Programación Dinámica con C++

```

/**
 * @brief Algoritmo que resuelve el problema de lineas de produccion
 * usando dp
 * @author Los ultimos
 */
#include <bits/stdc++.h>
/**
 * Resuelve el problema de lineas de produccion usando Dp
 *
 * @param a los tiempos que se tarda cada estacion en hacer un trabajo
 * @param t los tiempos que tarda cambiarse de una linea a otra
 * @param e los tiempos de entrada de cada linea
 * @param x los tiempos de salida de cada linea
 * @param n el tamaño de lineas a usar
 * @return el menor tiempo posible de ruta
 */
int lineasProduccion(int** a, int** t, int *e, int *x, int n) {
    int first, second, up, down;
    first = e[0] + a[0][0];
    second = e[1] + a[1][0];
    for (int i = 1; i < n; i++) {
        up = std::min(first + a[0][i], second + t[1][i] + a[0][i]);
        down = std::min(second + a[1][i], first + t[0][i] + a[1][i]);
    }
}

```

```

        first = up;
        second = down;
    }
    return std::min(first + x[0], second + x[1]);
}

int main() {
    int n, m;
    std::cin >> n;
    int** a = new int *[2];
    a[0] = new int [n];
    a[1] = new int [n];
    int** t = new int *[2];
    t[0] = new int [n];
    t[1] = new int [n];
    int* e = new int[2], * x = new int [2];
    for(int i = 0; i < n; i++)
        std::cin >> a[0][i];
    for(int i = 0; i < n; i++)
        std::cin >> a[1][i];
    for(int i = 1; i < n; i++)
        std::cin >> t[0][i];
    for(int i = 1; i < n; i++)
        std::cin >> t[1][i];
    std::cin >> e[0] >> e[1] >> x[0] >> x[1];
    //int a[][4] = {{4, 5, 3, 2}, {2, 10, 1, 4}};
    //int t[][4] = {{0, 7, 4, 5}, {0, 9, 2, 8}};
    // int e[] = {10, 12}, x[] = {18, 7};
    std::cout << lineasProduccion(a, t, e, x, n);
    return 0;
}

```

4.3. Código en JavaScript

4.3.1. Código para la Fuerza bruta

```

var car1 = document.getElementById('carro1');
var car2 = document.getElementById('carro2');
var car3 = document.getElementById('carro3');
var car4 = document.getElementById('carro4');
var car5 = document.getElementById('carro5');
var car6 = document.getElementById('carro6');
var car7 = document.getElementById('carro7');
var car8 = document.getElementById('carro8');

```

```
var naranja = document.getElementById('naranja');
var azul = document.getElementById('azul');
var verde = document.getElementById('verde');
var morado = document.getElementById('morado');
var caminoo = document.getElementById('mejor');
var caminoo2 = document.getElementById('mejor2');
var maxi = document.getElementById('max');
var llamadas = 0;
var vecesQ = 0;
car1.style.position = 'relative';
car2.style.position = 'relative';
car3.style.position = 'relative';
car4.style.position = 'relative';
car5.style.position = 'relative';
car6.style.position = 'relative';
car7.style.position = 'relative';
car8.style.position = 'relative';
var camino_max = 10000000;
var camino_max2 = 10000000;
//Controlar el carro 1
var x_car1 = 0;
var y_car1 = 0;
//Controlar el carro 2
var x_car2 = 0;
var y_car2 = 0;
//Controlar el carro 3
var x_car3 = 0;
var y_car3 = 0;
//Controlar el carro 4
var x_car4 = 0;
var y_car4 = 0;
//Controlar el carro 5
var x_car5 = 0;
var y_car5 = 0;
//Controlar el carro 6
var x_car6 = 0;
var y_car6 = 0;
//Controlar el carro 7
var x_car7 = 0;
var y_car7 = 0;
//Controlar el carro 8
var x_car8 = 0;
var y_car8 = 0;
var a = [[0,0,0],[0,0,0]];
var t = [[0,0],[0,0]];
```

```
var e = [0,0];
var x = [0,0];

//Click de prueba
document.getElementById('demo').addEventListener('click', demoSolution);
document.getElementById('aplicar').addEventListener('click', aplicarSolution);
//car8.addEventListener('click', bruteSolution);

//Mover los carros
function moveCar1(x1, y1){
    x_car1 = x_car1 + x1;
    car1.style.left = x_car1 + "%";

    y_car1 = y_car1 + y1;
    car1.style.top = y_car1 + "%";
}
function moveCar2(x1, y1){
    x_car2 = x_car2 + x1;
    car2.style.left = x_car2 + "%";
    y_car2 = y_car2 + y1;
    car2.style.top = y_car2 + "%";
}
function moveCar3(x1, y1){
    x_car3 = x_car3 + x1;
    car3.style.left = x_car3 + "%";
    y_car3 = y_car3 + y1;
    car3.style.top = y_car3 + "%";
}
function moveCar4(x1, y1){
    x_car4 = x_car4 + x1;
    car4.style.left = x_car4 + "%";
    y_car4 = y_car4 + y1;
    car4.style.top = y_car4 + "%";
}
function moveCar5(x1, y1){
    x_car5 = x_car5 + x1;
    car5.style.left = x_car5 + "%";

    y_car5 = y_car5 + y1;
    car5.style.top = y_car5 + "%";
}
function moveCar6(x1, y1){
    x_car6 = x_car6 + x1;
    car6.style.left = x_car6 + "%";
    y_car6 = y_car6 + y1;
```

```
    car6.style.top = y_car6 + "%";
}
function moveCar7(x1, y1){
    x_car7 = x_car7 + x1;
    car7.style.left = x_car7 + "%";
    y_car7 = y_car7 + y1;
    car7.style.top = y_car7 + "%";
}
function moveCar8(x1, y1){
    x_car8 = x_car8 + x1;
    car8.style.left = x_car8 + "%";
    y_car8 = y_car8 + y1;
    car8.style.top = y_car8 + "%";
}

function abs(x){return (x < 0) ? -1*x : x;}

//Sleep para que se vea animado
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}

//Animar los carros
async function animateCar(x1, y1, ncar) {
    for (let i = 0; i < 20; i++) {
        if(ncar == 1){
            moveCar1(x1, y1);
        }
        else if(ncar == 2){
            moveCar2(x1, y1);
        }
        else if(ncar == 3){
            moveCar3(x1, y1);
        }
        else if(ncar == 4){
            moveCar4(x1, y1);
        }
        else if(ncar == 5){
            moveCar5(x1, y1);
        }
        else if(ncar == 6){
            moveCar6(x1, y1);
        }
        else if(ncar == 7){
            moveCar7(x1, y1);
        }
    }
}
```



```
        else if(ncar == 8){
            moveCar8(x1, y1);
        }
        await sleep(100);
    }
}
function bringCar(car){
    if(x_car1 == 0){
        return 1;
    }
    else if(x_car2 == 0){
        car2.style.left = getXcar(car) -100 + "%";
        car2.style.top = getYcar(car) + "%";
        x_car2 = getXcar(car)-100;
        y_car2 = getYcar(car);
        return 2;
    }
    else if(x_car3 == 0){
        car3.style.left = getXcar(car) -210+ "%";
        car3.style.top = getYcar(car) + "%";
        x_car3 = getXcar(car)-210;
        y_car3 = getYcar(car);
        return 3;
    }
    else if(x_car4 == 0){
        car4.style.left = getXcar(car) -210+ "%";
        car4.style.top = getYcar(car) + "%";
        x_car4 = getXcar(car)-190;
        y_car4 = getYcar(car);
        return 4;
    }
    else if(x_car5 == 0){
        car5.style.left = getXcar(car) + "%";
        car5.style.top = getYcar(car) + "%";
        x_car5 = getXcar(car);
        y_car5 = getYcar(car);
        return 5;
    }
    else if(x_car6 == 0){
        car6.style.left = getXcar(car) - 100 + "%";
        car6.style.top = getYcar(car) + "%";
        x_car6 = getXcar(car) - 80;
        y_car6 = getYcar(car);
        return 6;
    }
}
```

```
    else if(x_car7 == 0){
        car7.style.left = getXcar(car) -200 + "%";
        car7.style.top = getYcar(car) + "%";
        x_car7 = getXcar(car)-180;
        y_car7 = getYcar(car);
        return 7;
    }
    else if(x_car8 == 0){
        car8.style.left = getXcar(car)-200 + "%";
        car8.style.top = getYcar(car) + "%";
        x_car8 = getXcar(car)-200;
        y_car8 = getYcar(car);
        return 8;
    }
}
function returnToStart(car){
    if(car == 1){
        car1.style.left = 0 + "%";
        car1.style.top = 0 + "%";
    }
    else if(car == 2){
        car2.style.left = 0 + "%";
        car2.style.top = 0 + "%";
    }
    else if(car == 3){
        car3.style.left = 0 + "%";
        car3.style.top = 0 + "%";
    }
    else if(car == 4){
        car4.style.left = 0 + "%";
        car4.style.top = 0 + "%";
    }
    else if(car == 5){
        car5.style.left = 0 + "%";
        car5.style.top = 0 + "%";
    }
    else if(car == 6){
        car6.style.left = 0 + "%";
        car6.style.top = 0 + "%";
    }
    else if(car == 7){
        car7.style.left = 0 + "%";
        car7.style.top = 0 + "%";
    }
    else if(car == 8){
```

```
        car8.style.left = 0 + "%";
        car8.style.top = 0 + "%";
    }
}
function getXcar(car){
    if(car == 1){
        return x_car1;
    }
    if(car == 2){
        return x_car2;
    }
    if(car == 3){
        return x_car3;
    }
    if(car == 4){
        return x_car4;
    }
    if(car == 5){
        return x_car5;
    }
    if(car == 6){
        return x_car6;
    }
    if(car == 7){
        return x_car7;
    }
    if(car == 8){
        return x_car8;
    }
}
function getYcar(car){
    if(car == 1){
        return y_car1;
    }
    if(car == 2){
        return y_car2;
    }
    if(car == 3){
        return y_car3;
    }
    if(car == 4){
        return y_car4;
    }
    if(car == 5){
        return y_car5;
    }
}
```

```
    }
    if(car == 6){
        return y_car6;
    }
    if(car == 7){
        return y_car7;
    }
    if(car == 8){
        return y_car8;
    }
}
function printCamino(car, camino){
    if(car%4 == 1){
        naranja.value = ""+camino;
    }
    else if(car%4 == 2){
        azul.value = ""+camino;
    }
    else if(car%4 == 3){
        verde.value = ""+camino;
    }
    else if(car%4 == 0){
        morado.value = ""+camino;
    }
}
}
async function wait(i, car, linea, camino) {
    printCamino(car, camino);

    if(i >= 3){
        if(linea == 0){
            animateCar(5.5, 3.4, car);
            printCamino(car, camino + x[0]);
            meterCamino(camino + x[0]);
            return;
        }
        else{
            animateCar(5.5, -3.4, car);
            printCamino(car, camino + x[1]);
            meterCamino(camino + x[1]);
            return;
        }
    }

    }
    await sleep(5000);
    if(i < 2){
```

```
var nextCar = bringCar(car);
animateCar(6, 0, car);
}
wait(i+1, car, linea, (camino + a[linea][i]));
if(i < 2){
  await sleep(5000);
  if(linea == 0){
    animateCar(6, 9.5, nextCar);
  }
  else{
    animateCar(4.4, -9.5, nextCar);
  }
}
wait(i+1, nextCar, (linea == 0) ? 1 : 0, (camino + t[linea][i] + a[linea][i]));
var minimo = min(0,9);
console.log(minimo);
}
function min(a, b){
  console.log("Llamadas: "+ llamadas);
  llamadas = llamadas + 1;
  maxi.value = ""+ (llamadas);
  return (a <= b) ? a : b;
}
//solucion bruta
function meterCamino(camino){
  if(vecesQ == 0){
    if(camino < camino_max){
      camino_max = camino;
      camino0.value = ""+camino_max;
    }
  }
  else{
    if(camino < camino_max2){
      camino_max2 = camino;
      camino02.value = ""+camino_max2;
    }
  }
}
}
async function bruteSolution(){
  car1.style.left = "-50%";
  car1.style.top = "330%";
  x_car1 = -50;
  y_car1 = 330;
  await sleep(1000);
  animateCar(4.5, -3, 1);
```

```
wait(0, 1, 0, e[0]);

await sleep(35000);
naranja.value = "";
azul.value = "";
verde.value = "";
morado.value = "";
returnToStart(1);
returnToStart(2);
returnToStart(3);
returnToStart(4);
vecesQ = 1;
car5.style.left = "-50%";
car5.style.top = "350%";
x_car5 = -50;
y_car5 = 350;

await sleep(1000);
animateCar(4.5, 3, 5);
wait(0, 5, 1, e[1]);
var hola = min(0,1);
console.log(hola);
}
async function demoSolution(){
  var estacion1 = 8;
  var estacion2 = 10;
  var estacion3 = 4;
  var estacion4 = 9;
  var estacion5 = 6;
  var estacion6 = 7;
  var estacion7 = 3;
  var estacion8 = 5;
  var estacion9 = 2;
  var estacion10 = 1;
  var estacion11 = 2;
  var estacion12 = 3;
  var estacion13 = 2;
  var estacion14 = 1;
  var estacionobt1 = document.getElementById("estacion1obtenida");//a00
  var estacionobt2 = document.getElementById("estacion2obtenida");//a01
  var estacionobt3 = document.getElementById("estacion3obtenida");//a02
  var estacionobt4 = document.getElementById("estacion4obtenida");//a10
  var estacionobt5 = document.getElementById("estacion5obtenida");//a11
  var estacionobt6 = document.getElementById("estacion6obtenida");//a12
  var estacionobt7 = document.getElementById("estacion7obtenida");//e0
```

```
var estacionobt8 = document.getElementById("estacion8obtenida");//e1
var estacionobt9 = document.getElementById("estacion9obtenida");//s0
var estacionobt10 = document.getElementById("estacion10obtenida");//s1
var estacionobt11 = document.getElementById("estacion11obtenida");//t00
var estacionobt12 = document.getElementById("estacion12obtenida");//t01
var estacionobt13 = document.getElementById("estacion13obtenida");//t10
var estacionobt14 = document.getElementById("estacion14obtenida");//t11
estacionobt1.value = "Costo: " + " "+estacion1;
estacionobt2.value = "Costo: " + estacion2;
estacionobt3.value = "Costo: " + estacion3;
estacionobt4.value = "Costo: " + estacion4;
estacionobt5.value = "Costo: " + estacion5;
estacionobt6.value = "Costo: " + estacion6;
estacionobt7.value = "Costo: " + estacion7;
estacionobt8.value = "Costo: " + estacion8;
estacionobt9.value = "Costo: " + estacion9;
estacionobt10.value = "Costo: " + estacion10;
estacionobt11.value = ""+ estacion11;
estacionobt12.value = ""+ estacion12;
estacionobt13.value = ""+ estacion13;
estacionobt14.value = ""+ estacion14;
a[0][0] = estacion1;
a[0][1] = estacion2;
a[0][2] = estacion3;

a[1][0] = estacion4;
a[1][1] = estacion5;
a[1][2] = estacion6;

t[0][0] = estacion11;
t[0][1] = estacion12;

t[1][0] = estacion13;
t[1][1] = estacion14;

e[0] = estacion7;
e[1] = estacion8;

x[0] = estacion9;
x[1] = estacion10;
sleep(100);
bruteSolution();
}
async function aplicarSolution(){
    var estacion1 = document.getElementById("estacion1").value;
```

```
var estacion2 = document.getElementById('estacion2').value;
var estacion3 = document.getElementById('estacion3').value;
var estacion4 = document.getElementById('estacion4').value;
var estacion5 = document.getElementById('estacion5').value;
var estacion6 = document.getElementById('estacion6').value;
var estacion7 = document.getElementById('estacion7').value;
var estacion8 = document.getElementById('estacion8').value;
var estacion9 = document.getElementById('estacion9').value;
var estacion10 = document.getElementById('estacion10').value;
var estacion11 = document.getElementById('estacion11').value;
var estacion12 = document.getElementById('estacion12').value;
var estacion13 = document.getElementById('estacion13').value;
var estacion14 = document.getElementById('estacion14').value;
var estacionobt1 = document.getElementById("estacion1obtenida");//a00
var estacionobt2 = document.getElementById("estacion2obtenida");//a01
var estacionobt3 = document.getElementById("estacion3obtenida");//a02
var estacionobt4 = document.getElementById("estacion4obtenida");//a10
var estacionobt5 = document.getElementById("estacion5obtenida");//a11
var estacionobt6 = document.getElementById("estacion6obtenida");//a12
var estacionobt7 = document.getElementById("estacion7obtenida");//e0
var estacionobt8 = document.getElementById("estacion8obtenida");//e1
var estacionobt9 = document.getElementById("estacion9obtenida");//s0
var estacionobt10 = document.getElementById("estacion10obtenida");//s1
var estacionobt11 = document.getElementById("estacion11obtenida");//t00
var estacionobt12 = document.getElementById("estacion12obtenida");//t01
var estacionobt13 = document.getElementById("estacion13obtenida");//t10
var estacionobt14 = document.getElementById("estacion14obtenida");//t11
estacionobt1.value = "Costo: " + estacion1;
estacionobt2.value = "Costo: " + estacion2;
estacionobt3.value = "Costo: " + estacion3;
estacionobt4.value = "Costo: " + estacion4;
estacionobt5.value = "Costo: " + estacion5;
estacionobt6.value = "Costo: " + estacion6;
estacionobt7.value = "Costo: " + estacion7;
estacionobt8.value = "Costo: " + estacion8;
estacionobt9.value = "Costo: " + estacion9;
estacionobt10.value = "Costo: " + estacion10;
estacionobt11.value = "" + estacion11;
estacionobt12.value = "" + estacion12;
estacionobt13.value = "" + estacion13;
estacionobt14.value = "" + estacion14;
a[0][0] = parseInt(estacion1);
a[0][1] = parseInt(estacion2);
a[0][2] = parseInt(estacion3);
```



```
a[1][0] = parseInt(estacion4);
a[1][1] = parseInt(estacion5);
a[1][2] = parseInt(estacion6);

t[0][1] = parseInt(estacion11);
t[0][2] = parseInt(estacion12);

t[1][1] = parseInt(estacion13);
t[1][2] = parseInt(estacion14);

e[0] = parseInt(estacion7);
e[1] = parseInt(estacion8);

x[0] = parseInt(estacion9);
x[1] = parseInt(estacion10);
sleep(100);
bruteSolution();
}
```

4.3.2. Código para la Programación dinámica

```
var car1 = document.getElementById('carro1');
var car2 = document.getElementById('carro2');
var car3 = document.getElementById('carro3');
var car4 = document.getElementById('carro4');
var car5 = document.getElementById('carro5');
var car6 = document.getElementById('carro6');
var car7 = document.getElementById('carro7');
var car8 = document.getElementById('carro8');
var firstt = document.getElementById('first');
var secondd = document.getElementById('second');
var upp = document.getElementById('up');
var downn = document.getElementById('down');
var maxi = document.getElementById('max');
var llamadas = 0;
car1.style.position = 'relative';
car2.style.position = 'relative';
car3.style.position = 'relative';
car4.style.position = 'relative';
car5.style.position = 'relative';
car6.style.position = 'relative';
car7.style.position = 'relative';
car8.style.position = 'relative';
var a = [[0,0,0],[0,0,0]];
var t = [[0,0,0],[0,0,0]];
```

```
var e = [0,0];
var x = [0,0];

//Controlar el carro 1
var x_car1 = 0;
var y_car1 = 0;
//Controlar el carro 2
var x_car2 = 0;
var y_car2 = 0;
//Controlar el carro 3
var x_car3 = 0;
var y_car3 = 0;
//Controlar el carro 4
var x_car4 = 0;
var y_car4 = 0;
//Controlar el carro 5
var x_car5 = 0;
var y_car5 = 0;
//Controlar el carro 6
var x_car6 = 0;
var y_car6 = 0;
//Controlar el carro 7
var x_car7 = 0;
var y_car7 = 0;
//Controlar el carro 8
var x_car8 = 0;
var y_car8 = 0;

//Click de prueba
document.getElementById('demo').addEventListener('click', demoSolution);
document.getElementById('aplicar').addEventListener('click', displaySolution);
//car8.addEventListener('click', bruteSolution);

//Mover los carros
function moveCar1(x1, y1){
    x_car1 = x_car1 + x1;
    car1.style.left = x_car1 + "%";

    y_car1 = y_car1 + y1;
    car1.style.top = y_car1 + "%";
}
function moveCar2(x1, y1){
    x_car2 = x_car2 + x1;
    car2.style.left = x_car2 + "%";
```

```
        y_car2 = y_car2 + y1;
        car2.style.top = y_car2 + "%";
    }
    function moveCar3(x1, y1){
        x_car3 = x_car3 + x1;
        car3.style.left = x_car3 + "%";
        y_car3 = y_car3 + y1;
        car3.style.top = y_car3 + "%";
    }
    function moveCar4(x1, y1){
        x_car4 = x_car4 + x1;
        car4.style.left = x_car4 + "%";
        y_car4 = y_car4 + y1;
        car4.style.top = y_car4 + "%";
    }
    function moveCar5(x1, y1){
        x_car5 = x_car5 + x1;
        car5.style.left = x_car5 + "%";

        y_car5 = y_car5 + y1;
        car5.style.top = y_car5 + "%";
    }
    function moveCar6(x1, y1){
        x_car6 = x_car6 + x1;
        car6.style.left = x_car6 + "%";
        y_car6 = y_car6 + y1;
        car6.style.top = y_car6 + "%";
    }
    function moveCar7(x1, y1){
        x_car7 = x_car7 + x1;
        car7.style.left = x_car7 + "%";
        y_car7 = y_car7 + y1;
        car7.style.top = y_car7 + "%";
    }
    function moveCar8(x1, y1){
        x_car8 = x_car8 + x1;
        car8.style.left = x_car8 + "%";
        y_car8 = y_car8 + y1;
        car8.style.top = y_car8 + "%";
    }

    function returnToStart(car){
        if(car == 1){
            car1.style.left = 0 + "%";
            car1.style.top = 0 + "%";
```

```
    }
    else if(car == 2){
        car2.style.left = 0 + "%";
        car2.style.top = 0 + "%";
    }
    else if(car == 3){
        car3.style.left = 0 + "%";
        car3.style.top = 0 + "%";
    }
    else if(car == 4){
        car4.style.left = 0 + "%";
        car4.style.top = 0 + "%";
    }
    else if(car == 5){
        car5.style.left = 0 + "%";
        car5.style.top = 0 + "%";
    }
    else if(car == 6){
        car6.style.left = 0 + "%";
        car6.style.top = 0 + "%";
    }
    else if(car == 7){
        car7.style.left = 0 + "%";
        car7.style.top = 0 + "%";
    }
    else if(car == 8){
        car8.style.left = 0 + "%";
        car8.style.top = 0 + "%";
    }
}

function abs(x){return (x < 0) ? -1*x : x;}

//Sleep para que se vea animado
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}

//Animar los carros
async function animateCar(x1, y1, ncar) {
    for (let i = 0; i < 20; i++) {
        if(ncar == 1){
            moveCar1(x1, y1);
        }
        else if(ncar == 2){
            moveCar2(x1, y1);
        }
    }
}
```

```
        else if(ncar == 3){
            moveCar3(x1, y1);
        }
        else if(ncar == 4){
            moveCar4(x1, y1);
        }
        else if(ncar == 5){
            moveCar5(x1, y1);
        }
        else if(ncar == 6){
            moveCar6(x1, y1);
        }
        else if(ncar == 7){
            moveCar7(x1, y1);
        }
        else if(ncar == 8){
            moveCar8(x1, y1);
        }
        await sleep(100);
    }
}

function bringCar(x, y){
    if(x_car2 == 0){
        y = (y-25);
        car2.style.left = x + "%";
        car2.style.top = y + "%";
        x_car2 = x;
        y_car2 = y;
        return 2;
    }
    else if(x_car3 == 0){
        y = (y-25);
        car3.style.left = x + "%";
        car3.style.top = y + "%";
        x_car3 = x;
        y_car3 = y;
        return 3;
    }
    else if(x_car4 == 0){
        car4.style.left = x + "%";
        car4.style.top = y + "%";
        x_car4 = x;
        y_car4 = y;
        return 4;
    }
}
```

```
    else if(x_car6 == 0){
        car6.style.left = x + "%";
        car6.style.top = y + "%";
        x_car6 = x;
        y_car6 = y;
        return 6;
    }
    else if(x_car7 == 0){
        car7.style.left = x + "%";
        car7.style.top = y + "%";
        x_car7 = x;
        y_car7 = y;
        return 7;
    }
    else if(x_car8 == 0){
        car8.style.left = x + "%";
        car8.style.top = y + "%";
        x_car8 = x;
        y_car8 = y;
        return 8;
    }
}
function getXcar(car){
    if(car == 1){
        return x_car1;
    }
    if(car == 2){
        return x_car2;
    }
    if(car == 3){
        return x_car3;
    }
    if(car == 4){
        return x_car4;
    }
    if(car == 5){
        return x_car5;
    }
    if(car == 6){
        return x_car6;
    }
    if(car == 7){
        return x_car7;
    }
    if(car == 8){
```

```
        return x_car8;
    }
}

function getYcar(car){
    if(car == 1){
        return y_car1;
    }
    if(car == 2){
        return y_car2;
    }
    if(car == 3){
        return y_car3;
    }
    if(car == 4){
        return y_car4;
    }
    if(car == 5){
        return y_car5;
    }
    if(car == 6){
        return y_car6;
    }
    if(car == 7){
        return y_car7;
    }
    if(car == 8){
        return y_car8;
    }
}

async function wait() {
    car1.style.left = "-100%";
    car1.style.top = "212%";
    car5.style.left = "-100%";
    car5.style.top = "252%";

    var first, second, up, down;
    var currentCar1 = 1;
    var currentCar2 = 5;
    var toDelete;
    first = e[0] + a [0][0];
    animateCar(1, -0.2, 1);
    second = e[1] + a[1][0];
    animateCar(1, 0.2, 5);
```

```
console.log(x_car1 + " " + x_car5);
for(let i = 1; i < 3; i++){
  await sleep(3000);
  up = min(first + a[0][i], second + t[1][i] + a[0][i]);
  down = min(second + a[1][i], first + t[0][i] + a[1][i]);
  if(up == first + a[0][i] && down == second + a[1][i])
  {
    animateCar(1, 0, currentCar1);
    animateCar(1, 0, currentCar2);
  }
  else if(up == second + t[1][i] + a[0][i] && down == second + a[1][i])
  {
    toDelete = currentCar1;
    currentCar1 = bringCar(currentCar2);
    returnToStart(toDelete);
    animateCar(1, -2.2, currentCar1);
    animateCar(1, 0, currentCar2);
  }
  else if(up == first + a[0][i] && down == first + t[0][i] + a[1][i]){
    var aux = currentCar1;
    var toDelete2 = currentCar2;
    currentCar1 = bringCar(currentCar2);
    currentCar2 = bringCar(aux);
    returnToStart(aux);
    returnToStart(toDelete2);
    animateCar(1, -2.2, currentCar1);
    animateCar(1, 2.2, currentCar2);
  }
  first = up;
  second = down;
}
await sleep(3000);
animateCar(1, 0.4, currentCar1);

animateCar(1, -0.4, currentCar2);
await sleep(3000);
}
```



```
function min(a, b){
    llamadas = llamadas + 1;
    maxi.value = ""+ (llamadas);
    return (a <= b) ? a : b;
}
//solucion bruta

async function bruteSolution(){
    car1.style.left = "-100%";
    car1.style.top = "213%";
    car5.style.left = "-227%";
    car5.style.top = "247%";
    x_car1 = -100;
    y_car1 = 213;
    x_car5 = -227;
    y_car5 = 247;
    var first, second, up, down;
    var currentCar1 = 1;
    var currentCar2 = 5;
    var toBringx, toBringy;
    var str1, str2, str3, str4, str5;
    first = e[0] + a [0][0];
    animateCar(3.7, -1.5, 1);
    second = e[1] + a[1][0];
    animateCar(3.7, 1.5, 5);
    firstt.value = ""+first;
    secondd.value = ""+second;
    for(let i = 1; i < 3; i++){
        await sleep(4000);
        str1 = first+a[0][i];
        str2 = ""+str1
        str3 = second+t[1][i]+a[0][i];
        str4 = ""+str3;
        upp.value = str2 +" o " +str4;
        str1 = second+ a[1][i];
        str2 = ""+ str2;
        str3 = first+t[0][i]+ a[1][i];
        str4 = ""+str3
        downn.value =str2+" o " +str4;
        up = min(first + a[0][i], second + t[1][i] + a[0][i]);
        down = min(second + a[1][i], first + t[0][i] + a[1][i]);

        if(up == first + a[0][i] && down == second + a[1][i]){
```

```

        animateCar(4.4, 0, currentCar1);
        animateCar(4.4, 0, currentCar2);
    }
    else if(up == second + t[1][i] + a[0][i] && down == second + a[1][i]){
        toBringx = getXcar(currentCar2);
        toBringy = getYcar(currentCar2);
        returnToStart(currentCar1);
        animateCar(4.4, 0, currentCar2);
        currentCar1 = bringCar(toBringx, toBringy);
        animateCar(4.4, -5, currentCar1);
    }
    else if(up == first + a[0][i] && down == first + t[0][i] + a[1][i]){
        toBringx = getXcar(currentCar1);
        toBringy = getYcar(currentCar1);
        returnToStart(currentCar2);
        animateCar(4.4, 0, currentCar1);
        currentCar2 = bringCar(toBringx, toBringy);
        animateCar(4.4, 5, currentCar2);
    }
    else if(up == first + a[0][i] && down == first + t[0][i] + a[1][i]){
        toBringx = getXcar(currentCar1);
        toBringy = getYcar(currentCar1);
        var aux = currentCar1;
        var toDelete2 = currentCar2;
        currentCar1 = bringCar(toBringx, toBringy);
        toBringx = getXcar(aux);
        toBringy = getYcar(aux);
        currentCar2 = bringCar(toBringx, toBringy);
        returnToStart(aux);
        returnToStart(toDelete2);
        animateCar(4.4, -5, currentCar1);
        animateCar(4.4, 5, currentCar2);
    }
    first = up;
    second = down;
    firstt.value = ""+first;
    secondd.value = ""+second;
}
await sleep(3000);
animateCar(4.4, 1.6, currentCar1);

animateCar(4.4, -1.6, currentCar2);
await sleep(3000);
first = first +x[0];

```

```
second = second+x[1];
firstt.value = ""+first;
secondd.value = ""+ second;
upp.value = "";
downn.value = "";
var hola = min(first, second);
console.log(hola);
}
//var a = [[8,10,4], [9,6,7]];

//var t = [[0,2,3], [0,2,1]];
//var e = [3,5], x = [2,1];
async function demoSolution(){
    var estacion1 = 8;
    var estacion2 = 10;
    var estacion3 = 4;
    var estacion4 = 9;
    var estacion5 = 6;
    var estacion6 = 7;
    var estacion7 = 3;
    var estacion8 = 5;
    var estacion9 = 2;
    var estacion10 = 1;
    var estacion11 = 2;
    var estacion12 = 3;
    var estacion13 = 2;
    var estacion14 = 1;
    var estacionobt1 = document.getElementById("estacion1obtenida");//a00
    var estacionobt2 = document.getElementById("estacion2obtenida");//a01
    var estacionobt3 = document.getElementById("estacion3obtenida");//a02
    var estacionobt4 = document.getElementById("estacion4obtenida");//a10
    var estacionobt5 = document.getElementById("estacion5obtenida");//a11
    var estacionobt6 = document.getElementById("estacion6obtenida");//a12
    var estacionobt7 = document.getElementById("estacion7obtenida");//e0
    var estacionobt8 = document.getElementById("estacion8obtenida");//e1
    var estacionobt9 = document.getElementById("estacion9obtenida");//s0
    var estacionobt10 = document.getElementById("estacion10obtenida");//s1
    var estacionobt11 = document.getElementById("estacion11obtenida");//t00
    var estacionobt12 = document.getElementById("estacion12obtenida");//t01
    var estacionobt13 = document.getElementById("estacion13obtenida");//t10
    var estacionobt14 = document.getElementById("estacion14obtenida");//t11
    estacionobt1.value = "Costo:  "+ estacion1;
    estacionobt2.value = "Costo:  "+ estacion2;
    estacionobt3.value = "Costo:  "+ estacion3;
    estacionobt4.value = "Costo:  "+ estacion4;
```

```
    estacionobt5.value = "Costo: " + estacion5;
    estacionobt6.value = "Costo: " + estacion6;
    estacionobt7.value = "Costo: " + estacion7;
    estacionobt8.value = "Costo: " + estacion8;
    estacionobt9.value = "Costo: " + estacion9;
    estacionobt10.value = "Costo: " + estacion10;
    estacionobt11.value = "" + estacion11;
    estacionobt12.value = "" + estacion12;
    estacionobt13.value = "" + estacion13;
    estacionobt14.value = "" + estacion14;
    a[0][0] = estacion1;
    a[0][1] = estacion2;
    a[0][2] = estacion3;

    a[1][0] = estacion4;
    a[1][1] = estacion5;
    a[1][2] = estacion6;

    t[0][1] = estacion11;
    t[0][2] = estacion12;

    t[1][1] = estacion13;
    t[1][2] = estacion14;

    e[0] = estacion7;
    e[1] = estacion8;

    x[0] = estacion9;
    x[1] = estacion10;
    sleep(100);
    bruteSolution();
}
async function displaySolution() {
    var estacion1 = document.getElementById("estacion1").value;
    var estacion2 = document.getElementById('estacion2').value;
    var estacion3 = document.getElementById('estacion3').value;
    var estacion4 = document.getElementById('estacion4').value;
    var estacion5 = document.getElementById('estacion5').value;
    var estacion6 = document.getElementById('estacion6').value;
    var estacion7 = document.getElementById('estacion7').value;
    var estacion8 = document.getElementById('estacion8').value;
    var estacion9 = document.getElementById('estacion9').value;
    var estacion10 = document.getElementById('estacion10').value;
    var estacion11 = document.getElementById('estacion11').value;
    var estacion12 = document.getElementById('estacion12').value;
```

```
var estacion13 = document.getElementById('estacion13').value;
var estacion14 = document.getElementById('estacion14').value;
var estacionobt1 = document.getElementById("estacion1obtenida");//a00
var estacionobt2 = document.getElementById("estacion2obtenida");//a01
var estacionobt3 = document.getElementById("estacion3obtenida");//a02
var estacionobt4 = document.getElementById("estacion4obtenida");//a10
var estacionobt5 = document.getElementById("estacion5obtenida");//a11
var estacionobt6 = document.getElementById("estacion6obtenida");//a12
var estacionobt7 = document.getElementById("estacion7obtenida");//e0
var estacionobt8 = document.getElementById("estacion8obtenida");//e1
var estacionobt9 = document.getElementById("estacion9obtenida");//s0
var estacionobt10 = document.getElementById("estacion10obtenida");//s1
var estacionobt11 = document.getElementById("estacion11obtenida");//t00
var estacionobt12 = document.getElementById("estacion12obtenida");//t01
var estacionobt13 = document.getElementById("estacion13obtenida");//t10
var estacionobt14 = document.getElementById("estacion14obtenida");//t11
estacionobt1.value = "Costo:  "+ " "+estacion1;
estacionobt2.value = "Costo:  "+ estacion2;
estacionobt3.value = "Costo:  "+ estacion3;
estacionobt4.value = "Costo:  "+ estacion4;
estacionobt5.value = "Costo:  "+ estacion5;
estacionobt6.value = "Costo:  "+ estacion6;
estacionobt7.value = "Costo:  "+ estacion7;
estacionobt8.value = "Costo:  "+ estacion8;
estacionobt9.value = "Costo:  "+ estacion9;
estacionobt10.value = "Costo:  "+ estacion10;
estacionobt11.value = ""+ estacion11;
estacionobt12.value = ""+ estacion12;
estacionobt13.value = ""+ estacion13;
estacionobt14.value = ""+ estacion14;
a[0][0] = parseInt(estacion1);
a[0][1] = parseInt(estacion2);
a[0][2] = parseInt(estacion3);

a[1][0] = parseInt(estacion4);
a[1][1] = parseInt(estacion5);
a[1][2] = parseInt(estacion6);

t[0][1] = parseInt(estacion11);
t[0][2] = parseInt(estacion12);

t[1][1] = parseInt(estacion13);
t[1][2] = parseInt(estacion14);

e[0] = parseInt(estacion7);
```

```
e[1] = parseInt(estacion8);  
  
x[0] = parseInt(estacion9);  
x[1] = parseInt(estacion10);  
sleep(100);  
bruteSolution();  
}
```