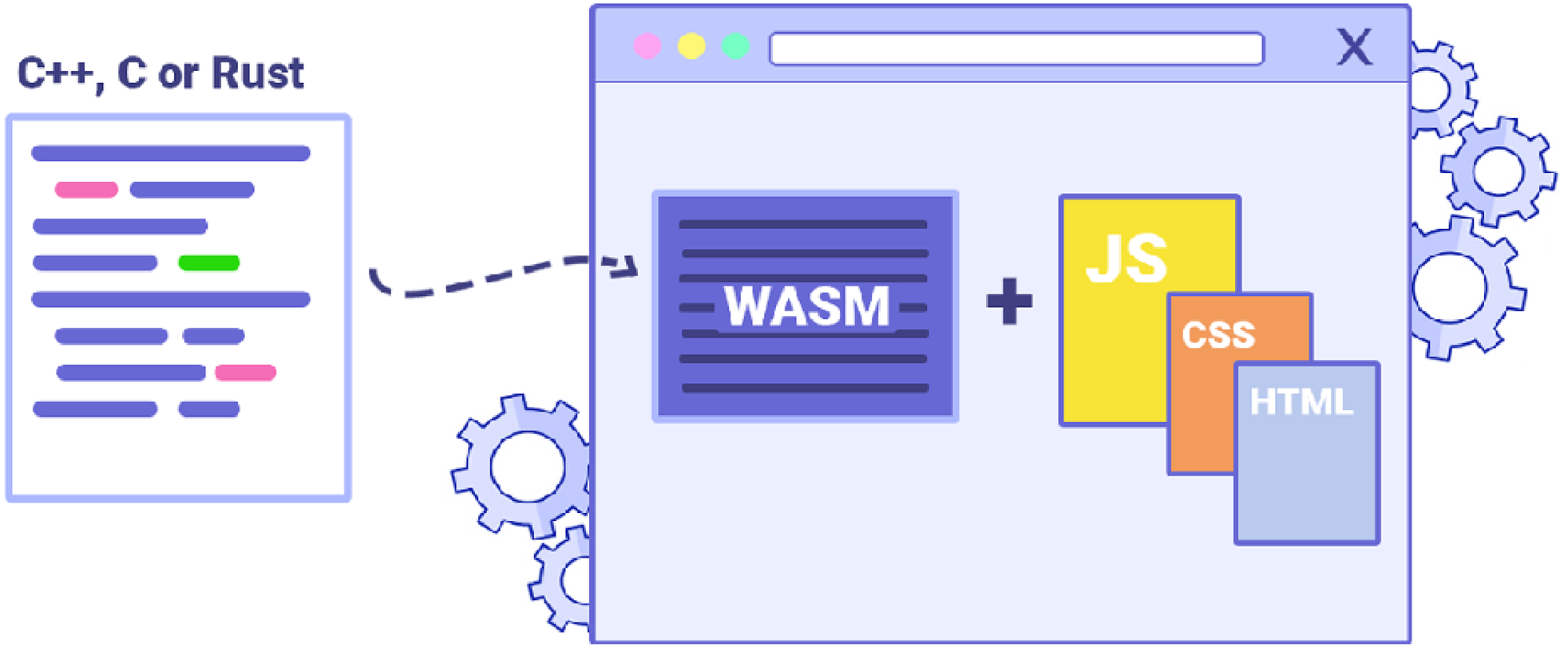
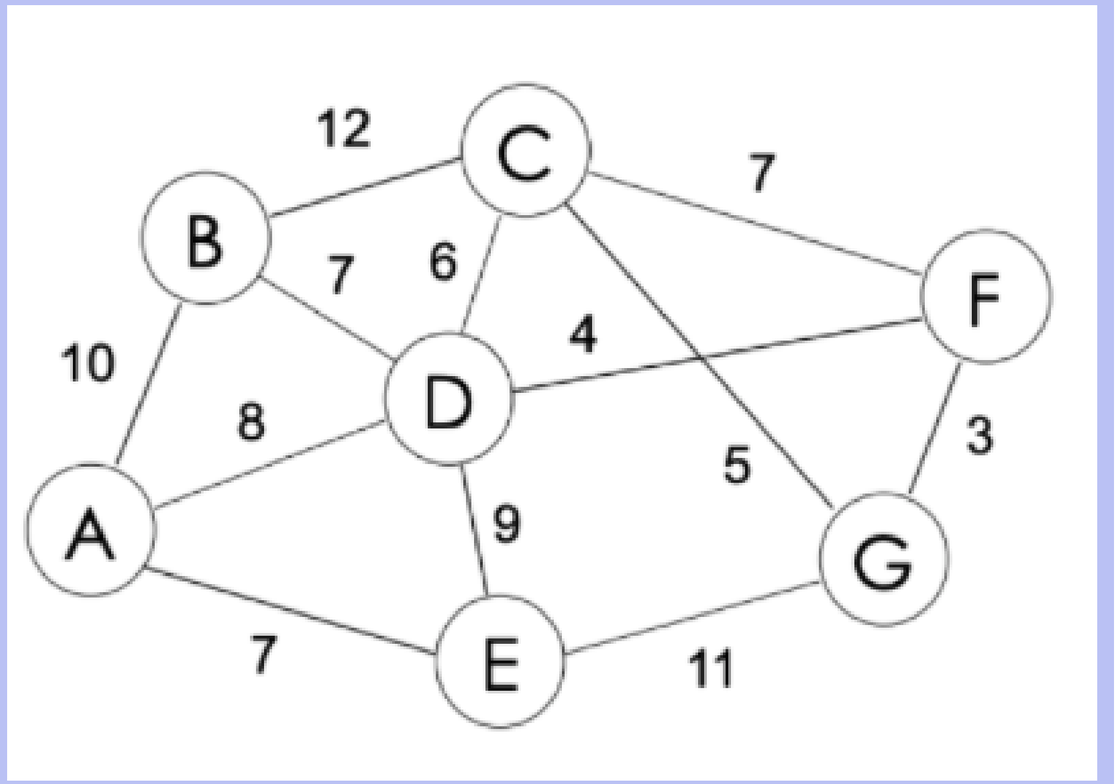


Benjamín Ayancán
Nicolás Benitez
Fernanda Durán

Web Assembly

Índice

Tarea



Implementar un programa que encontrara la mejor ruta para un vendedor dado que tiene que visitar a todos sus clientes.

La figura muestra la ubicación de los clientes y las distancias asociadas a cada camino disponible.

El problema consiste en encontrar la secuencia óptima.

.

Demo

Código

C

Función least

```c
int ary[MAX][MAX],completed[MAX],n,cost;

int least(int c, int** ary)
{
    int i,nc=999;
    int min=999,kmin;

    for(i=0;i < n;i++)
    {
        if((ary[c][i]!=0)&&(completed[i]==0))
            if(ary[c][i]+ary[i][c] < min)
            {
                min=ary[i][0]+ary[c][i];
                kmin=ary[c][i];
                nc=i;
            }
    }

    if(min!=999)
        cost+=kmin;

    return nc;
}
```

C

Función mincost

```c
void mincost(int city, int* memoria, int k,int** ary)
{
    int i, ncity;

    completed[city]=1;

    printf("%d--->",city+1);
    memoria[k] = city+1;
    ncity = least(city, ary);

    if(ncity==999)
    {
        ncity=0;
        printf("%d",ncity+1);
        cost+=ary[city][ncity];

        return;
    }

    mincost(ncity, memoria, k+1, ary);
    memoria[0] = cost;
}
```

C

Función dijkstra

```c
int dijkstra(int* memoria, int** ary, int dimension)
{
    cost = 0;
    n = dimension;
    int i;
    for(i=0;i < n;i++)
    {
        completed[i]=0;
    }

    printf("\n\nThe Path is:\n");
    mincost(0, memoria, 1,ary); //passing 0 because starting verte

    printf("\n\nMinimum cost is %d\n ",cost);


    return 0;
}
```

Javascript

Función makePtrOfArray

```
const graph = [
  [0, 4, 1, 3],
  [4, 0, 2, 1],
  [1, 2, 0, 5],
  [3, 1, 5, 0],
];
```

```
const makePtrOfArray = (myModule, N) => {
  const arrayPtr = myModule._calloc(N, 4);
  for (let i = 0; i < N; i++) {
    let rowsPtr = myModule._calloc(N, 4);
    myModule.setValue(arrayPtr + i * 4, rowsPtr, "i32");
    for (let j = 0; j < N; j++) {
      myModule.setValue(rowsPtr + j * 4, graph[i][j], "i32");
    }
  }
  return arrayPtr;
};
```

Javascript

Función getArrayFromPtr

```
const getArrayFromPtr = (myModule, ptr, N) => {
  let resultMatrix = [];
  for (let i = 0; i < N; i++) {
    resultMatrix[i] = myModule.getValue(ptr + i * 4, "i32");
  }
  return resultMatrix;
};
```

Javascript

Función addToTable

```
const addToTable = (resultArr, execTime) => {
  const table = document.getElementById("result-table");
  let row = table.insertRow(-1);
  let cell1 = row.insertCell(0);
  let cell2 = row.insertCell(1);
  let cell3 = row.insertCell(2);
  cell1.innerHTML = `${resultArr[0]}`;
  cell2.innerHTML = `${resultArr.slice(1).join("->")}`;
  cell3.innerHTML = `${execTime} ms`;
};
```

Javascript

Función cRunner

```
const cRunner = (Module) => {
  const arrayPtr = Module._calloc(10, 4);
  const G = makePtrOfArray(Module, 4);

  let startTime = window.performance.now();
  let result = Module._dijkstra(arrayPtr, G, 4);
  let endTime = window.performance.now();
  console.log("result", result);

  let matrix = getArrayFromPtr(Module, arrayPtr, 5);
  console.log("matrix", matrix);

  const resultTime = endTime - startTime;
  addToTable(matrix, resultTime);
  return [999, 999];
};
```

Javascript

```
const calcBtn = document.getElementById("calc-btn");

Module().then(function (mymod) {
  calcBtn.onclick = () => {
    let cResult,
      cTime = cRunner(mymod);
    addToTable(cResult, cTime);
  };
});
```

Dificultades

- Tuvimos muchos errores al momento de programar por lo que nos fuimos quedando atascados error tras error

- No logramos recibir el input desde HTML por lo que finalmente trabajamos con un grafo definido desde JS

Aprendizajes

- Logramos comprender la base de Web Assembly, programar y ejecutar desde C

Benjamín Ayancán
Nicolás Benitez
Fernanda Durán

Muchas gracias