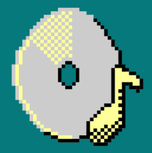


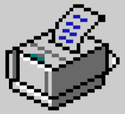



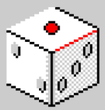

WASM

Grupo 5

Nicolás Barría
Sebastián Carrasco
Juan Vargas



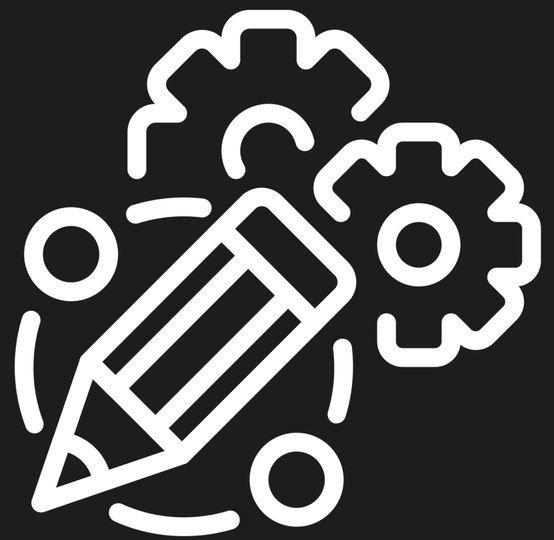




11:11 PM

Problema

N Tareas → **Asignar** → **M Clusters**



Solución

Random

Greedy

Demo

Random JS

```
1  const randomizeTaskAssignment = (times, n) => {
2    const clusters = Array.from({ length: n }, () => []);
3    for (let i = 0; i < times.length; i++) {
4      const cluster = clusters[Math.floor(Math.random() * n) % n];
5      cluster.push(times[i]);
6    }
7    return clusters;
8  };
```

Random C

```c
int y_x(int i, int j, int rows) { return i * rows + j; }
```

Random C

```c
int randomC(int n, int m, int *time_tasks, int *processors_assignments) {
  int *n_tasks_processors = calloc(m, sizeof(int));
  int random_processor;
  for (int i = 0; i < n; i++) {
    random_processor = rand() % m;
    processors_assignments[y_x(random_processor,
                               n_tasks_processors[random_processor], n)] =
        time_tasks[i];
    n_tasks_processors[random_processor]++;
  }
  free(n_tasks_processors);
  return 0;
}
```

Greedy JS

```
const greedyJS = (n, m, time_tasks) => {
  const processor_assignments = [];
  const time_processors_sum = [];
  for (let i = 0; i < m; i++) {
    processor_assignments.push([]);
    time_processors_sum.push(0);
  }
  time_tasks.sort((a, b) => b - a);
  let min_completion_time;
  let min_processor;
  for (let i = 0; i < n; i++) {
    min_completion_time = time_processors_sum[0];
    min_processor = 0;
    for (let j = 1; j < m; j++) {
      if (time_processors_sum[j] < min_completion_time) {
        min_completion_time = time_processors_sum[j];
        min_processor = j;
      }
    }
    time_processors_sum[min_processor] += time_tasks[i];
    processor_assignments[min_processor].push(time_tasks[i]);
  }
  return processor_assignments;
};
```

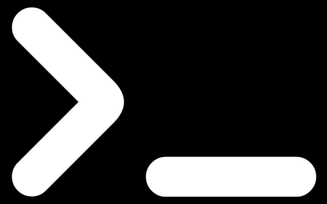
Greedy C

```c
int greedyC(int n, int m, int* time_tasks, int* processors_assignments) {
    int start_time = clock();
    int* time_processors_sum = calloc(m, sizeof(int));
    int* n_tasks_processors = calloc(m, sizeof(int));

    quicksort(time_tasks, 0, n - 1);
    for (int i = 0; i < n; i++) {
        int min_completion_time = time_processors_sum[0];
        int min_processor = 0;
        for (int j = 1; j < m; j++) {
            if (time_processors_sum[j] < min_completion_time) {
                min_completion_time = time_processors_sum[j];
                min_processor = j;
            }
        }
        time_processors_sum[min_processor] += time_tasks[i]; // assign task to the processor with earliest completion time
        processors_assignments[y_x(min_processor, n_tasks_processors[min_processor], n)] = time_tasks[i];
        n_tasks_processors[min_processor]++;
    }
    free(time_processors_sum);
    free(n_tasks_processors);

    return 0;
}
```

Conexión



```
emcc -
sEXPORTED_FUNCTIONS=_randomC,_greedyC,_malloc,_free
-sEXPORTED_RUNTIME_METHODS={cwrap,ccall} -O3 -o
algorithms.js algorithms.c
```

Llamando a C desde JS

```
1  Module.onRuntimeInitialized = () => {
2    greedyCButton.addEventListener('click', () => {
3      ...
4    });
5  }
```

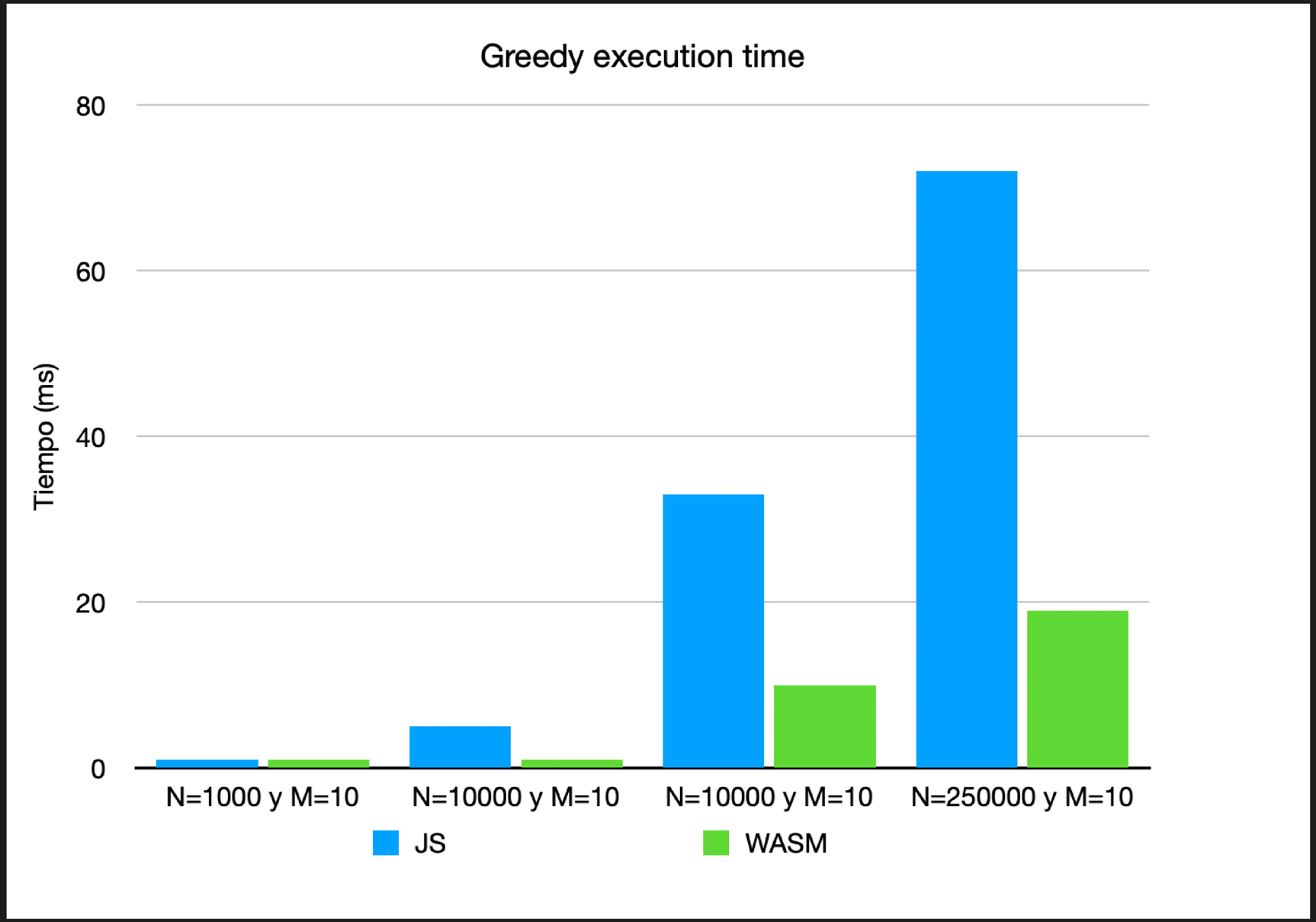
Llamando a C desde JS

```
1   const { m, times } = getParams();
2   const timesArray = Uint32Array.from(times);
3   const timePtr = Module._malloc(timesArray.byteLength);
4   Module.HEAPU32.set(timesArray, timePtr >> 2);
5
6   const clusterArray = Uint32Array.from({ length: m * timesArray.length });
7   const clusterPtr = Module._malloc(clusterArray.byteLength);
8   Module.HEAPU32.set(clusterArray, clusterPtr >> 2);
9
10  const updatedClusterArray = Module.HEAPU32.subarray(
11    clusterPtr >> 2,
12    (clusterPtr >> 2) + m * timesArray.length,
13    );
```

Llamando a C desde JS

```
1  const initialTime = performance.now();
2  Module.ccall(
3    'greedyC',
4    'int',
5    ['number', 'number', 'number', 'number'],
6    [timesArray.length, m, timePtr, clusterPtr],
7  );
8  const endTime = performance.now();
9  greedyCTime.innerHTML = `${endTime - initialTime} ms`;
10 const updatedCluster = Array.from(updatedClusterArray);
11 const matrix = arrayToMatrix(updatedCluster, m, times.length);
12 Module._free(timePtr);
13 Module._free(clusterPtr);
```

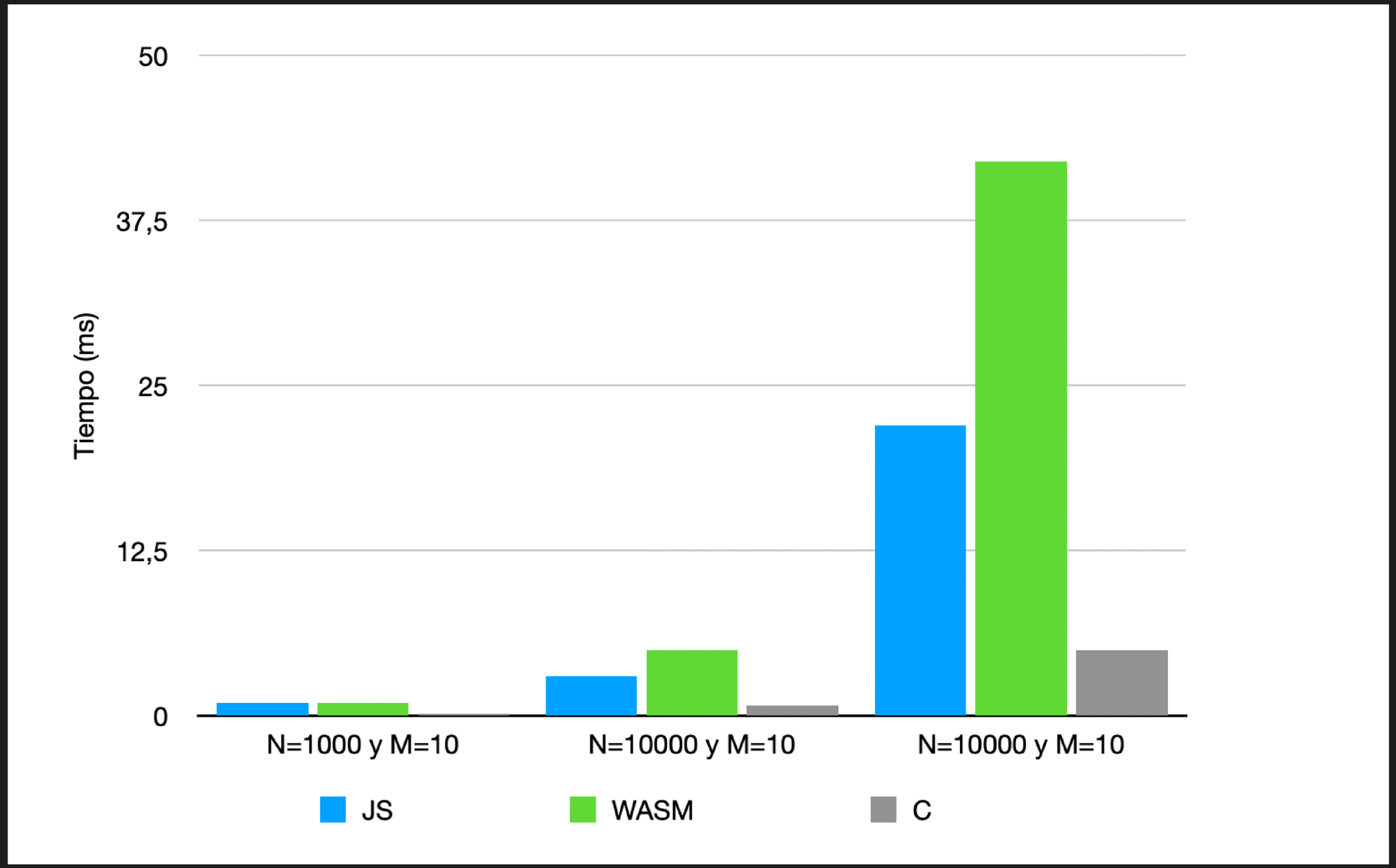
Comparación Greedy



Problemas en implementación

- Límites de memoria de emscripten.
- Liberación de memoria desde js.
- qsort de stdlib de C.

Problemas en implementación





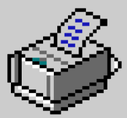


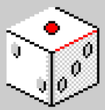


WASM

Grupo 5

Nicolás Barría
Sebastián Carrasco
Juan Vargas



11:11 PM