

Simulación paralela{

```
<Por="Daniel Gonzalez"/>
```

```
<Por="Carlos Rodriguez"/>
```

```
}
```

Problema asignado {

cree un programa de simulación paralela que pueda simular sistemas complejos, como el flujo de tráfico, la dinámica de la población o los patrones climáticos, automatas, distribuyendo la carga computacional entre múltiples procesadores o nodos.



}

Cambios al código {

Inclusión de la Biblioteca OpenMP:

- **NFA.c** incorpora la biblioteca **omp.h**, el código está optimizado para ejecutarse en múltiples núcleos de procesador simultáneamente, lo que puede mejorar significativamente la eficiencia y velocidad de ejecución del programa.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <stdbool.h>
5 #include <math.h>
6 #include <time.h>
7
8 #define MAX_SIZE 100 // Define a maximum size for the input
9
10 int row = 0;
11 // A structure to represent an adjacency list node
12 struct node|
13 {
14     int data;
15     struct node* next;
16     char edgetype;
17 }
18 }typedef node;
19
20 // Adds an edge to an adjacency list
21 node* push(node* first , char edgetype , int data)
22 {
23     node* new_node = (node*)malloc(sizeof(node));
24     new_node->edgetype = edgetype;
25     new_node->data = data;
26     new_node->next = NULL;
27     if (first==NULL)
28     {
29         first = new_node;
30         // printf("(%c, %d)\n", first->edgetype, first->data);
31         return new_node;
32     }
33 }
```

{}

Cambios al código {

Estructura del Código:

- La estructura básica de ambos códigos es, la definición de la estructura **node** y la función **push**.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <stdbool.h>
5 #include <math.h>
6 #include <time.h>
7
8 #define MAX_SIZE 100 // Define a maximum size for the input
9
10 int row = 0;
11 // A structure to represent an adjacency list node
12 struct node{
13     int data;
14     struct node* next;
15     char edgetype;
16 }
17
18 }typedef node;
19
20 // Adds an edge to an adjacency list
21 node* push(node* first , char edgetype , int data)
22 {
23     node* new_node = (node*)malloc(sizeof(node));
24     new_node->edgetype = edgetype;
25     new_node->data = data;
26     new_node->next = NULL;
27     if (first==NULL)
28     {
29         first = new_node;
30         // printf("(%c, %d)\n", first->edgetype, first->data);
31         return new_node;
32     }
33 }
```

{}

Cambios al código {

Otras mejoras:

- en **NFA.c** podrían incluir una gestión de memoria más eficiente, algoritmos optimizados, cambios en la estructura de datos que aumenten el rendimiento del programa.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <stdbool.h>
5 #include <math.h>
6 #include <time.h>
7
8 #define MAX_SIZE 100 // Define a maximum size for the input
9
10 int row = 0;
11 // A structure to represent an adjacency list node
12 struct node|
13 {
14     int data;
15     struct node* next;
16     char edgetype;
17 }
18 }typedef node;
19
20 // Adds an edge to an adjacency list
21 node* push(node* first , char edgetype , int data)
22 {
23     node* new_node = (node*)malloc(sizeof(node));
24     new_node->edgetype = edgetype;
25     new_node->data = data;
26     new_node->next = NULL;
27     if (first==NULL)
28     {
29         first = new_node;
30         // printf("(%c, %d)\n", first->edgetype, first->data);
31         return new_node;
32     }
33 }
```

{}

Ejecucion normal

PROBLEMS DEBUG CONSOLE PORTS TERMINAL OUTPUT

```
4
1 0 4 0 1 0 2 1 1 1 3
2 0 1 0 4
3 0 1 1 4
4 1 2 0 4 1 4
Insert the number of inputs to be evaluated:
10
00
11
000
001
011
100
110
111
0000
Program executed in: 0.000000 seconds
[daagonzalezb@guane09 final]$
```

Ejecucion en paralelo

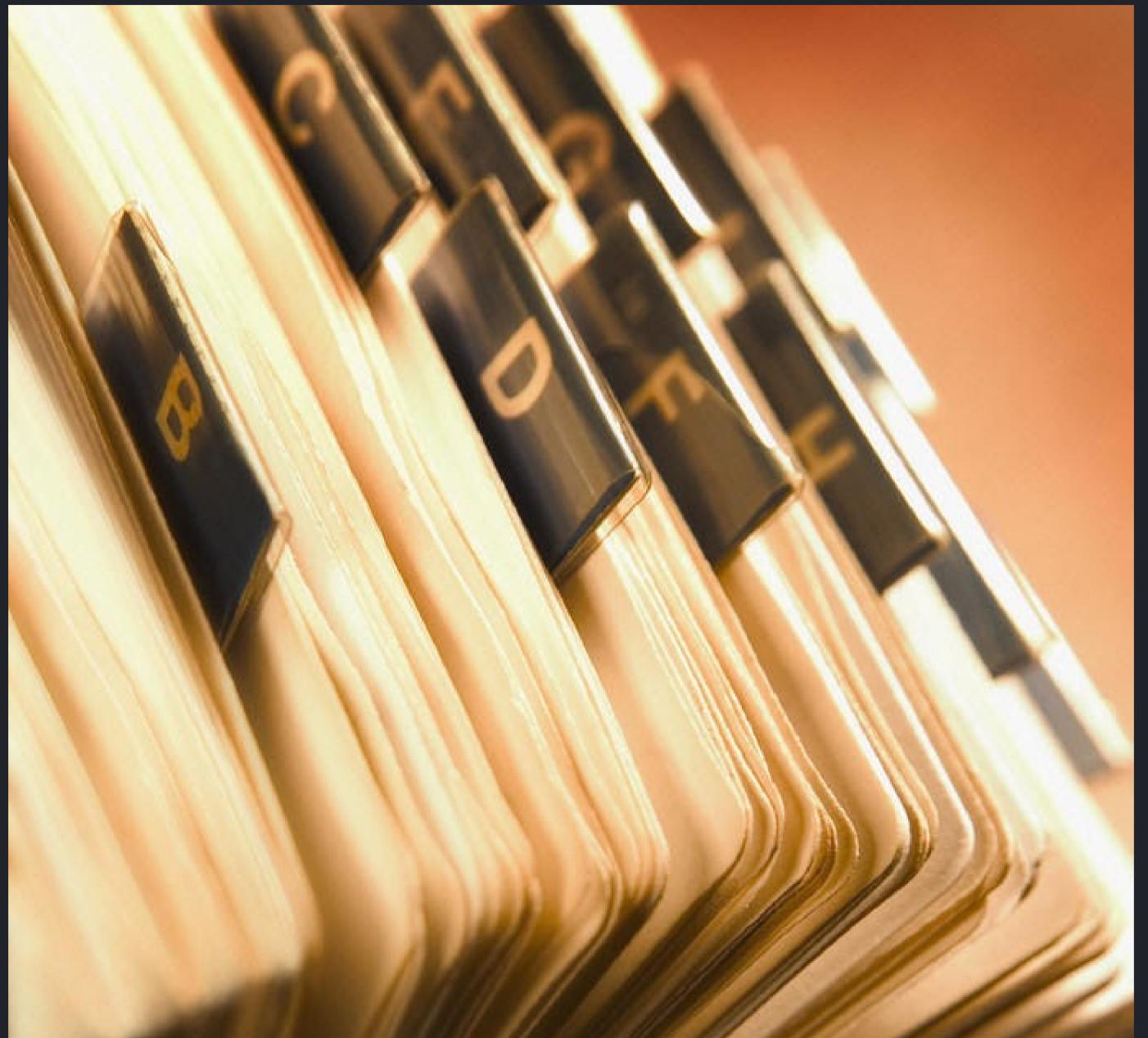
```
PROBLEMS DEBUG CONSOLE PORTS TERMINAL OUTPUT

4
1 0 4 0 1 0 2 1 1 1 3
2 0 1 0 4
3 0 1 1 4
4 1 2 0 4 1 4
Insert the number of inputs to be evaluated:
10
00
11
000
011
001
111
110
100
0010
0100
0001
1000
1100
1101
1011
1110
1111
Program executed in: 0.250000 seconds
[daagonzalezb@guane09 final]$
```

Archivo Readme {

¿Que Incluye?

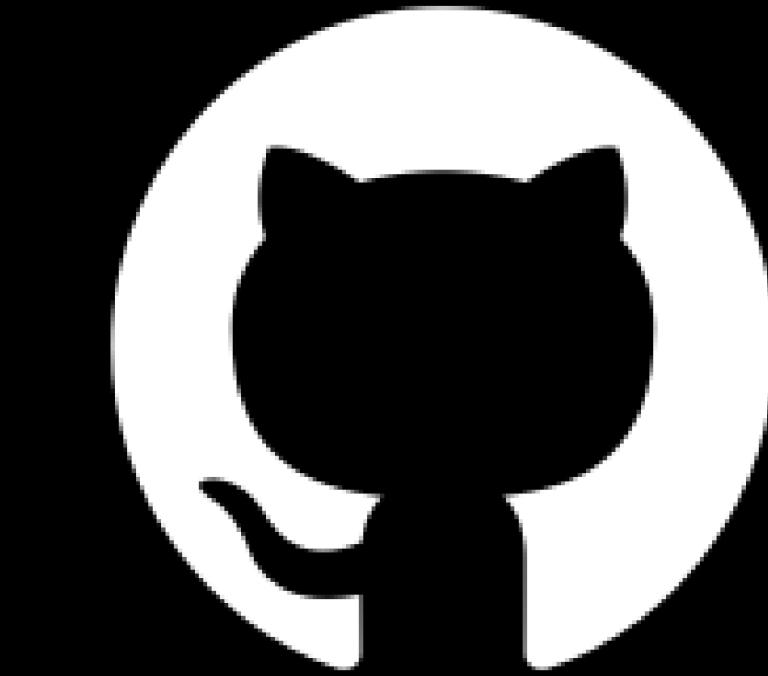
- Descripcion completa de “¿para que sirve el codigo?”.
- Caracteristicas principales y optimizacion del mismo.
- componentes principales de nuestro codigo.
- compilacion y detalles de compilacion.
- comando de uso.



}

Link al Repositorio {

[https://github.com/Carlos-
CodeBot/IntroPP2192655/](https://github.com/Carlos-CodeBot/IntroPP2192655/)



}

Gracias {

<Por="Daniel Gonzalez"/>

<Por="Carlos Rodriguez"/>

}