COMPUTACIÓN PARALELA

OPENMP – ENTREGA 1

Grupo 01

Jhon Steeven Cabanilla Alvarado

Miguel Chaveinte García

1. Comenta primero los problemas que tenía el código y cómo son las soluciones básicas necesarias.

```
/* Stop if searcher finds another trail */
int check;
check = accessMat( tainted, pos_row, pos_col );
accessMat( tainted, pos_row, pos_col ) = 1;
```

```
/* Stop if searcher finds another trail */
int check;
#pragma omp atomic capture
{
check = accessMat( trails, pos_row, pos_col );
accessMat( trails, pos_row, pos_col ) = search;
}
```

```
if ( check != 0 ) {
        searchers[ search ].follows = accessMat( trails, pos_row, pos_col );
        search_flag = 1;
}
else {
        /* Annotate the trail */
        accessMat( trails, pos_row, pos_col ) = search;

        /* Compute the height */
        accessMat( heights, pos_row, pos_col ) = get_height( pos_row, pos_col, rows, columns, x_min, x_max, y_min, y_max );
```

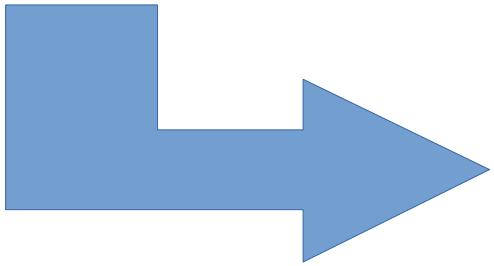
```
if ( check != -1 ) {
        #pragma omp atomic write
        accessMat( trails, pos_row, pos_col ) =check;
        search_flag = 1;
}
else {
        float local_max;

        /* Compute the height */
        local_max=accessMat( heights, pos_row, pos_col ) = get_height( pos_row, pos_col, rows, columns, x_min, x_max, y_min, y_max );
```

```
    /* 4. Compute searchers climbing trails */
    #pragma omp for private(search)
    for( search = 0; search < num_searchers; search++ ) {
            int search_flag = 0;
            while( ! search_flag ) {
                    search_flag = climbing_step( rows, columns, searchers, search, heights, trails, x_min, x_max, y_min, y_max );

ef DEBUG
def _OPENMP

his function is used only in sequential versions. Several threads exploring
t the same time can derive in mixed lines and confusing output of no value.

t_trails( rows, columns, trails );
t_heights( rows, columns, heights );
if
if
            }

    }
    #pragma omp for private(search)
    for( search = 0; search < num_searchers; search++ ) {
            searchers[ search ].follows = accessMat( trails, searchers[ search ].pos_row, searchers[ search ].pos_col );
    }
```

2. Comenta las cosas extras que se han conseguido explicando **por qué funcionan, o **de dónde surgió** la idea de probarlas.**

- Eliminar la matriz tainted.

```
/* 3.2. Terrain initialization */
for( i=0; i<rows; i++ ) {
        for( j=0; j<columns; j++ ) {
                accessMat( heights, i, j ) = INT_MIN;
                accessMat( trails, i, j ) = -1;
                accessMat( tainted, i, j ) = 0;
        }
}
```



```
/* 3.2. Terrain initialization */
#pragma omp parallel for
for( i=0; i<rows*columns; i++ ) {
        heights[i]= INT_MIN;
        trails[i]= -1;
        //tainted[i] = 0;
}
```

- Inicialización del buscadores.

```
/* 3.3. Searchers initialization */
int search;
for( search = 0; search < num_searchers; search++ ) {
        searchers[ search ].id = search;
        searchers[ search ].pos_row = (int)( rows * erand48( random_seq ) );
        searchers[ search ].pos_col = (int)( columns * erand48( random_seq ) );
        searchers[ search ].steps = 0;
        searchers[ search ].follows = -1;
        total_steps[ search ] = 0;
}
```

```c
/* 3.3. Searchers initialization */
int search;

for( search = 0; search < num_searchers; search++ ) {
        searchers[ search ].pos_row = (int)( rows * erand48( random_seq ) );
        searchers[ search ].pos_col = (int)( columns * erand48( random_seq ) );
}

#pragma omp parallel
{

#pragma omp for
for( search = 0; search < num_searchers; search++ ) {
        searchers[ search ].id = search;
        searchers[ search ].steps = 0;
        searchers[ search ].follows = -1;
        total_steps[ search ] = 0;
}
```

```
/* Compute the height */
local_max=accessMat( heights, pos_row, pos_col ) = get_height( pos_row, pos_col, rows, columns, x_min, x_max, y_min, y_max );

int climbing_direction = 0;
float altura;
if ( pos_row > 0 ) {
        /* Compute the height in the neighbor if needed */
        altura = accessMat( heights, pos_row-1, pos_col );
        if ( altura == INT_MIN )
                altura=accessMat( heights, pos_row-1, pos_col ) = get_height( pos_row-1, pos_col, rows, columns, x_min, x_max, y_min, y_max );

        /* Annotate the travelling direction if higher */
        if ( altura > local_max ) {
                climbing_direction = 1;
                local_max = altura;
        }
}
if ( pos_row < rows-1 ) {
        /* Compute the height in the neighbor if needed */
        altura=accessMat( heights, pos_row+1, pos_col );
        if ( altura == INT_MIN )
                altura=accessMat( heights, pos_row+1, pos_col ) = get_height( pos_row+1, pos_col, rows, columns, x_min, x_max, y_min, y_max );

        /* Annotate the travelling direction if higher */
        if ( altura > local_max ) {
                climbing_direction = 2;
                local_max = altura;
        }
```

PARALELIZACIÓN

```
    /* 4. Compute searchers climbing trails */
    #pragma omp for private(search)
    for( search = 0; search < num_searchers; search++ ) {
            int search_flag = 0;
            while( ! search_flag ) {
                    search_flag = climbing_step( rows, columns, searchers, search, heights, trails, x_min, x_max, y_min, y_max );

ef DEBUG
def _OPENMP

his function is used only in sequential versions. Several threads exploring
t the same time can derive in mixed lines and confusing output of no value.

t_trails( rows, columns, trails );
t_heights( rows, columns, heights );
if
if
            }

    }
    #pragma omp for private(search)
    for( search = 0; search < num_searchers; search++ ) {
            searchers[ search ].follows = accessMat( trails, searchers[ search ].pos_row, searchers[ search ].pos_col );
    }
```

```
/* 5. Compute the leading follower of each searcher */
#pragma omp for private(search)
for( search = 0; search < num_searchers; search++ ) {
        int search_flag = 0;
        int parent = search;
        int follows_to = searchers[ parent ].follows;

        while( ! search_flag ) {
                if ( follows_to == parent ) search_flag = 1;
                else {
                        parent = follows_to;
                        follows_to = searchers[ parent ].follows;
                }
        }
        searchers[ search ].follows = follows_to;
}
```

REDUCCIONES

```
/* 7. Compute statistical data */
#pragma omp for private(search) reduction(max:max_accum_steps,max_height) reduction(+:num_local_max)
for( search = 0; search < num_searchers; search++ ) {
        /* Maximum of accumulated trail steps to a local maximum */
        if ( max_accum_steps < total_steps[ search ] )
                max_accum_steps = total_steps[ search ];

        /* If this searcher found a maximum, check the maximum value */
        if ( searchers[ search ].follows == search ) {
                num_local_max++;
                int pos_row = searchers[ search ].pos_row;
                int pos_col = searchers[ search ].pos_col;
                int altura=accessMat( heights, pos_row, pos_col );
                if ( max_height <  altura)
                        max_height = altura;
        }
}

#pragma omp for reduction(+:total_tainted)
for( i=0; i<rows*columns; i++ ) {
        if ( trails[i] !=-1)
                total_tainted++;
}

} //Fin región paralela
```

3. Comenta otras cosas que se han probado y no han funcionado. Si tenéis alguna idea de por qué, explicadlo.

- Calcular las alturas fuera de la función *climbing_step*

- Utilizar *sections* para los diferentes if de la función *climbing_step*

- Cláusula *schedule*

4. Comenta como te has organizado con tu compañero para trabajar durante el concurso

5. Comenta fuentes, páginas web u otros recursos que has consultado durante el desarrollo de la práctica/concurso

- Apuntes de la asignatura

- https://stackoverflow.com/

- https://bisqwit.iki.fi/story/howto/openmp/

- https://www.openmp.org/