Auxiliar 2: Sincronización de Threads

Profesor: Luis Mateu B.
Auxiliares: José Astorga, Pablo Jaramillo

22 de Marzo de 2023

Problema 1: Quicksort en N cores

La función de abajo es una implementación del algoritmo de *quicksort* para ordenar un arreglo de enteros.

```
void quicksort_seq(int a[], int i, int j){
    if (i < j){
        int h = particionar(a, i, j);
        quicksort_seq(a, i, h-1);
        quicksort_seq(a, h+1, j);
    }
}
```

Considere que usted tiene a su disposición la función *particionar*, la cual se encarga de seleccionar un elemento del arreglo como "pivote", dejando a su lado izquierdo los valores menores y a su lado derecho los valores mayores. La función retorna la posición final en la que se encuentra el "pivote".

Usted deberá paralelizar la función *quicksort* para una máquina de multi-core, siendo el encabezado de la función el siguiente:

```
void quicksort(int a[], int i, int j int n);
```

Donde n corresponde a la cantidad de cores.

Problema 2: Colecta

Se necesita crear un sistema para juntar exactamente una cantidad **X** de dinero:

(a) Definir el tipo de datos `Colecta`

(b) Programar la función `Colecta *nuevaColecta(double meta)` que crea y retorna una colecta para juntar **$meta.**

(c) Programar la función `double aportar(Colecta *c, double monto)`, que es invocada desde múltiples threads para contribuir con **$monto**. El valor de retorno de la función es el mínimo entre **monto** y lo que falta para llegar a la meta. **La función debe retornar una vez que la meta se cumpla**.