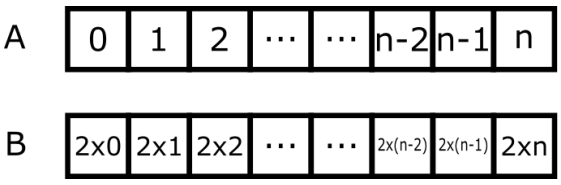
Uso de PThreads para paralelizar. Introducción al entorno OpenMP.

Una empresa de viajes para amantes de la entomología ha decidido agrupar las zonas que oferta según la similitud de especies de insectos que tienen. Para tal fin ha adquirido los conjuntos de especies presentes en cada región y han contratado a tu empresa para calcular una medida de similitud entre dichos conjuntos. Los conjuntos se representan como vectores en los que cada posición contiene el número identificativo de una cierta especie. El director de proyecto ha decidido usar la Distancia de Jaccard para realizar la medida de similitud. Además, como se espera usar el núcleo del software en el futuro para aplicar ese mismo cálculo en la identificación de plagio en grandes volúmenes de texto, se va a hacer uso de computación paralela. En la siguiente figura se resume el cálculo que se debe realizar:



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Deben entonces preparar un programa que **calcule en paralelo la Distancia de Jaccard** dados dos **conjuntos** de **enteros** representados **en vectores**, y que pueden tener **distinta longitud**. Como está trabajando en un prototipo no necesita acceder a datos reales, simplemente puede generar los conjuntos siguiendo el esquema de la figura. El programa recibirá por consola la **longitud de los dos vectores** y el **número de hilos** a desplegar cuando se ejecute. Finalmente, considerando que los usuarios finales no están acostumbrados a trabajar con programas en consola, es recomendable que se **filtren** mínimamente los parámetros recibidos para, por ejemplo, evitar longitudes negativas o un número mayor de hilos que de elementos por vector.

En este contexto, el plan de trabajo diseñado en la empresa define el desarrollo de tres versiones distintas del programa que trabajarán sobre los mismos datos:

1) Una versión **secuencial** a usar como referencia.
2) Una versión paralela basada en **PThreads sin** hacer uso de sincronización **MUTEX**.
3) Una versión paralela basada en **PThreads** haciendo uso de sincronización por **MUTEX**.

Por consola se mostrará tanto el **valor devuelto** como el **tiempo de ejecución**, en segundos, para cada versión propuesta.

Además de una descripción suficientemente detallada del programa elaborado, se pide complementar dicha información con el mostrado de un caso de ejecución con vectores de longitud 5 en el que se incluyan dichos vectores y el resultado devuelto en cada caso (desplegando desde 1 hasta 4 hilos para las versiones paralelas). Las tareas descritas hasta

ahora permiten hacerse una idea de que el programa desarrollado es funcional y se comporta según lo esperado. A partir de este punto, se pide realizar las siguientes actividades:

1) **Evaluar el rendimiento** de cada versión registrando los tiempos de ejecución, para las tres versiones anteriores, en cada uno de estos casos (y promediados tras 5 ejecuciones):
    a. Longitud de los vectores (número total de elementos de los dos vectores): (Concretar finalmente según la máquina disponible)
    b. Número de hilos: (Concretar finalmente según la máquina disponible)
2) Hacer una **gráfica de aceleración** ("*speedup*") conjunta con los datos recogidos anteriormente de las versiones paralelas con respecto a la secuencial. Además de mostrarse la evolución de las versiones paralelas, debe incluirse en la gráfica la aceleración ideal lineal.
3) **Comentar** los datos obtenidos. Resulta importante dar en primera instancia un tono general a los comentarios en relación a los beneficios logrados por la explotación del paralelismo en este problema. Los comentarios deben centrarse finalmente en las diferencias que se observen en el comportamiento de las dos versiones paralelas en un tono comparativo.

Finalmente, en el marco de este proyecto, el director de la empresa cree que la tecnología de la compañía está un poco anticuada y podría ser más productiva si sus empleados dominaran nuevas herramientas de programación paralela. Ha oído hablar de **OpenMP** y les pide **buscar** una breve definición, de no más de media cara de folio formato A4, del estándar de programación paralela en memoria compartida OpenMP. A partir la información buscada y las ideas generales aprendidas sobre OpenMP, se pide **comentar la siguiente función** que calcula el producto escalar de dos vectores:

```
double calcularProductoEscalarOMP(void){
    double productoEscalar = 0.0;
    #pragma omp parallel for reduction(+:productoEscalar) num_threads(numHilos)
    for(int i = 0; i<longitudVector; i++){
        productoEscalar += (vectorA[i]*vectorB[i]);
    }
    return productoEscalar;
}
```

Describa brevemente la estructura de la función y trate de comparar su nivel complejidad con respecto a la versión que ha visto en clase de teoría del cálculo del producto escalar: **¿qué cree que ofrecen entornos como OpenMP con respecto a PThreads? Incluya, junto a su informe, el código desarrollado en texto.** Adicionalmente, se podrán hacer **preguntas presenciales**.

Referencias de interés: [1] http://people.revoledu.com/kardi/tutorial/Similarity/Jaccard.html