INSTITUTO TECNOLÓGICO DE CIUDAD MADERO.



UNIDAD 2

PRACTICA 4

ALUMNA: CAVAZOS ARGOT ANA VICTORIA

N° CONTROL: 15071292

PROFESOR: DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN

MATERIA: PROGRAMACIÓN PARALELA

FECHA DE ENTREGA: 15 DE OCTUBRE 2018

Índice:

[Ejercicio 1: 3](#_Toc524903542)

[Introducción: 3](#_Toc524903543)

[Marco teórico: 3](#_Toc524903544)

[Reverse and Add: 3](#_Toc524903545)

[Metodología: 3](#_Toc524903546)

[Preguntas: 4](#_Toc524903547)

[Conclusiones: 5](#_Toc524903548)

Ejercicio:

Introducción:

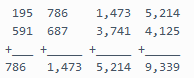
Resuelve el siguiente ejercicio:

Marco teórico:

Reverse and Add:

La función reverse and add empieza con un número, se invierte el número y se le suma al original. Si la suma no es un palíndromo (es decir, que no sean el mismo número en cada extremo) se repite el procedimiento hasta que se haga palíndromo.

Por ejemplo, si empezamos con 195 como el numero inicial, obtenemos 9339 como palíndromo resultante después de 4 adiciones:



Este método encuentra un palíndromo en pocos pasos para casi todos los enteros. Pero hay interesantes excepciones. 196 es el primer número que no produce palíndromo.

Para este método se debe recibir un número y retornar el palíndromo resultante (si es que existe) junto con las iteraciones/adiciones que tomaron encontrarlo.

Se asume que todos los números usados para probar el método terminaran con una cantidad de iteraciones menor a 1000 (adiciones) y el palíndromo resultante no deberá de ser más grande que 4,294,967,295.

## Metodología:

**Valor:** 265

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 265 |  | 827 |  | 1555 |  | 7106 |  | 13123 |
| + | 562 | + | 728 | + | 5551 | + | 6017 | + | 32131 |
|  | 827 |  | 1555 |  | 7106 |  | 13123 |  | **45254** |

Se realizaron 5 adiciones para obtener 45254 que es un palindromo.

Experimentación y resultados:

Información sobre el equipo:

**Modelo**: Dell OptiPlex 7010

**Procesador**: Intel(R) Core(TM) i5-3550 CPU @ 3.30GHz

**Memoria RAM**: 4.00 GB

**Tipo de sistema**: Sistema operativo de 64 bits

**Sistema operativo utilizado**: Windows 7 Ultimate Service Pack 1

Tabla de resultados:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Experimento | SEED | Cantidad de valores | Ultimo valor | Iteraciones | Palíndromo | Tiempo secuencial  (segundos) | Hilos | Tiempo paralelo OMP  (segundos) |
| 1 | 20 | 100,000 | 21570 | 7 | 5280825 | **10.873** | **2** | 14.984 |
| **4** | 14.934 |
| **8** | 15.793 |
| **16** | 15.575 |
| 2 | 20 | 200,000 | 11001 | 1 | 21012 | **22.386** | 2 | 25.959 |
| 4 | 25.896 |
| 8 | 26.02 |
| 16 | 25.864 |
| 3 | 20 | 300,000 | 4492 | 16 | 1561182975 | **34.515** | 2 | 39.443 |
| 4 | 38.968 |
| 8 | 38.859 |
| 16 | 38.656 |
| 4 | 20 | 400,000 | 18789 | 12 | 695686596 | **45.946** | 2 | 51.443 |
| 4 | 51.526 |
| 8 | 52.01 |
| 16 | 52.15 |
| 5 | 20 | 500,000 | 18410 | 1 | 19891 | **56.718** | 2 | 64.802 |
| 4 | 64.896 |
| 8 | 65.580 |
| 16 | 64.724 |

Preguntas:

**¿El problema puede ser paralelizado?**

El programa requiere calcular N veces los palíndromos de distintos números por lo que podría dividirse el trabajo con la paralelizacion.

**¿Cómo sería particionado el problema?**

Un hilo podría encargase de calcular el palíndromo de cada valor que se le asigne mientras otros trabajan en lo mismo.

**¿Se requiere comunicación?**

Si el caso es dividir la carga de datos entre hilos los procesos son independientes por lo que no se requiere comunicación.

**¿Hay dependencia de datos?**

No, cada dato es independiente.

**¿Se requiere sincronización?**

Si el caso es dividir la carga de datos, no es necesaria la sincronización.

**¿Será el equilibrio de carga una preocupación?**

Si, ya que si nos dan una gran cantidad de números la repartición de datos podría mejorar el tiempo de ejecución.

Conclusiones:

Este ejercicio corto demuestra cómo es posible generar números palíndromos por medio de un digito cualquiera y la suma de su inverso. Esto calculado de manera manual resulta fácil, sin embargo, al momento de codificar se presenta la dificultad de invertir un digitito.

Para invertir un digito se optó por obtener cada uno de sus dígitos mediante el residuo de 10 y almacenarlo en un arreglo que nos permita leer de manera más fácil sus dígitos.

Este problema, aunque es pequeño, a diferencia de los ejercicios anteriores, llego a tardar más en su tiempo de ejecución. Sin embargo, la paralelización de este problema no permite mejorar el tiempo de ejecución.

En este caso se intentó paralizar el método recursivo sin embargo el tiempo de ejecución empeoraba. Al final se paralelizo el ciclo que manda a llamar el cálculo del numero revertido, esto no mejoro el tiempo de manera notable, pero almenos no aumento tanto que al paralelizar el método recursivo.