Universidad Del Valle | Sede
Tuluá

# Fundamentos de programación funcional y concurrente

Ingeniería en Sistemas

Proyecto de Programación

Carlos Andrés Delgado Saavedra

Alejandro Sierra Betancourt 2259559

Juan Pablo Castaño Arango 2259487

Juan Manuel Ramírez Agudelo

2259482

15 diciembre 2023

**Análisis del problema:**

- **Análisis General del Problema de la Reconstrucción de Cadenas (PRC):**
  El problema de la reconstrucción de cadenas, abordado en este proyecto, surge en el ámbito de la biología, específicamente en la investigación del genoma humano. Los investigadores buscan identificar un patrón de nucleótidos que codifique la información necesaria para construir las proteínas que componen el cuerpo humano. Este patrón, conocido como el genoma humano, debe ser descrito mediante un proceso llamado secuenciación de la pieza de DNA. La tarea implica la reconstrucción de un patrón a partir de sus patrones conocidos presentes en la pieza de DNA.

- **Motivaciones y Desafíos:**

1. **Representación de Secuencias de DNA:**
   Las secuencias de DNA se representan como cadenas de caracteres sobre el alfabeto {A, C, T, G}. Los problemas asociados a la reconstrucción de cadenas se enuncian en términos de manipulación de cadenas de caracteres.

2. **Longitud Significativa de las Secuencias:**
   Las secuencias de DNA son extremadamente largas, siendo el genoma humano un ejemplo con aproximadamente 3 billones de caracteres. Esto presenta desafíos computacionales que requieren técnicas sofisticadas para su manipulación y análisis.

3. **Operación Básica para la Secuenciación:**
   La operación fundamental para la secuenciación es la pregunta ¿s es subcadena de $S$? (denotada como $¿\, s \preccurlyeq S?$). Esta operación, asumida como un experimento de laboratorio perfecto en este proyecto, proporciona respuestas precisas sobre la relación de subcadena entre dos secuencias.

4. **Oráculo para Experimentos de Laboratorio:**
   Se introduce la noción de un oráculo $\psi\delta$ que, en el contexto del proyecto, se supone capaz de responder de manera perfecta a la pregunta $\psi\delta(s) si\ s \preccurlyeq S$. Este oráculo simboliza la herramienta experimental de laboratorio que se utiliza para obtener información sobre la presencia o ausencia de subcadenas en la cadena buscada.

- **Objetivo del Proyecto:**

  El proyecto busca abordar, a modo de ejercicio, el desafío de reconstruir una secuencia de DNA a partir de experimentos con un oráculo que informa si una subsecuencia dada es o no parte de la cadena objetivo.

**Implementación de Soluciones Secuenciales**

En un principio para implementar cada una de las soluciones debemos tener en cuenta el qué debemos hacer en frente al oráculo, el oráculo es aquella forma que nos permite saber si una secuencia candidata pertenece a la secuencia principal de la cual no conocemos su estructura, pero si su tamaño, frente a esto se realizaran distintas preguntas al oráculo y se realizaran distintas acciones solo si son verdaderas.

Una vez conozcamos su funcionalidad podríamos implementar las siguientes soluciones:

- **Solución Ingenua:**

  La solución ingenua sería probar todas las combinaciones posibles de la secuencia que estamos buscando y verificar con el oráculo si es correcta. Si el oráculo dice que sí, entonces encontramos la cadena que buscamos. Sin embargo, esta aproximación no es la más eficiente en términos de tiempo. Esto se debe a que la operación que estamos realizando, conocida como producto cartesiano (todas las combinaciones posibles de 4 letras), se expresa matemáticamente como $4^n$ donde "n" es la longitud de la cadena que estamos buscando. En otras palabras, $4^n$ representa todas las secuencias candidatas posibles y, por lo tanto, puede llevar mucho tiempo si la cadena es larga.

- **Solución Mejorada:**

  Para la solución mejorada, nos basamos en una porción del seudocódigo, aunque alteramos la idea principal. Nuestra solución implica la generación progresiva de secuencias candidatas, construyendo letra por letra a partir del alfabeto. En este enfoque, se generan un total de $4^n$ secuencias candidatas, lo cual es más eficiente en comparación con la solución ingenua, aunque no completamente óptimo ya que aún conlleva una considerable cantidad de secuencias candidatas.

  Es importante señalar que, a medida que se generan las secuencias candidatas, aquellas que no cumplen con la condición establecida por el oráculo son eliminadas de la lista de todas las posibles secuencias candidatas. Este proceso de filtrado contribuye a mejorar la eficiencia de la solución, ya que descartamos aquellas secuencias que no son viables de manera progresiva, reduciendo así la carga de trabajo en comparación con la aproximación ingenua.

- **Solución Turbo:**

  La solución turbo, en su implementación, considera la generación de las $4^n$ posibles combinaciones, pero su enfoque difiere significativamente del método ingenuo. En lugar de generar todas las combinaciones de manera indiscriminada, la solución turbo construye la secuencia de forma iterativa, letra por letra, utilizando únicamente las letras esenciales del alfabeto. Este enfoque más selectivo contribuye a una generación más eficiente y específica de secuencias. Posteriormente, se aplica un filtro de manera secuencial, evaluando cada secuencia generada mediante una consulta al oráculo. Aquellas secuencias que pertenecen al oráculo se mantienen, mientras que las que no cumplen con esta condición, junto con todas sus posibles combinaciones, son eliminadas. Este enfoque optimizado reduce significativamente la carga computacional al descartar rápidamente las secuencias que no son relevantes según el criterio del oráculo.

- **Solución Turbo Mejorada:**

  La versión mejorada del método de reconstrucción de cadenas turbo se basa en construir cada posible cadena, letra por letra, pero con la novedad de aplicar un filtro específico a cada candidato de la cadena principal. Para entender mejor cómo funciona esta función, podemos imaginarla como un proceso de verificación continua: se revisa si la secuencia candidata encaja con la secuencia principal, es decir, si el oráculo la considera válida. Si el oráculo da el visto bueno, el proceso sigue generando letras una a una hasta completar la secuencia principal. En cada paso de generación de una nueva letra para cada candidato, se elimina aquel candidato que no cumple con la condición del oráculo.

**Implementación de Soluciones Paralelas**

- **Solución Ingenua Paralela:**

  En la solución ingenua paralela, se optó por asignar un hilo a cada cadena generada a partir del producto cartesiano. Aunque esta aproximación paralela facilita la distribución de tareas, podría enfrentar desafíos de eficiencia cuando se manejan grandes conjuntos de cadenas, ya que la asignación de hilos de manera directa puede llevar a una gestión subóptima de recursos y una posible saturación de los mismos.

- **Solución Mejorada Paralela:**

  La solución mejorada paralela destaca al asignar hilos a casos recursivos en los que se generan cada posible candidata para la cadena. Este enfoque permite optimizar la generación

de cadenas candidatas de manera más eficiente al utilizar hilos de manera más selectiva y adaptativa a la estructura del problema, mejorando así la gestión de recursos y la eficiencia del proceso.

- **Solución Turbo Paralela:**
  En la solución turbo paralela, se implementa un enfoque dinámico al generar un hilo para cada posible candidato, incrementando así el número de hilos. Este enfoque puede mejorar la velocidad de búsqueda al aprovechar la paralelización en la generación de múltiples candidatos simultáneamente.

- **Solución Turbo Mejorada Paralela:**
  La solución turbo mejorada paralela destaca al crear un hilo que concatena cada candidato con la nueva letra del alfabeto. Esta estrategia fusiona eficientemente la generación y el filtrado de secuencias, optimizando el proceso al utilizar un hilo de manera más específica. La concatenación selectiva puede reducir la complejidad y mejorar la eficiencia en comparación con enfoques más ingenuos.

**Manejo del paradigma funcional en el código**

- La función generarCadenas encapsula la recursión, generando combinaciones de cadenas mediante un enfoque inteligente. Utilizando la recursión como una herramienta para construir gradualmente las posibles combinaciones de caracteres. Además, se hace uso de expresiones for.
- El reconocimiento de patrones, se puede encontrar en la función buscarCadena. Al utilizar coincidencias de patrones, esta función explora secuencialmente las cadenas restantes hasta encontrar una que cumpla con el oráculo proporcionado.
- La función reconstruirCadenaMejorado se puede evidenciar el reconocimiento de patrones y filtrar resultados de una manera más eficiente. Aquí, la función se beneficia de funciones de alto orden, como flatMap.
- En la función reconstruirCadenaTurboPar se hace uso del paralelismo mediante la expresión task, dividiendo las tareas en subprocesos paralelos para mejorar el rendimiento. La comparación de tiempos secuenciales y paralelos mediante la función compararSecuencialParalela proporciona una evaluación cuantitativa del impacto del paralelismo en el rendimiento.

**Uso de técnicas de paralelización en el código**

1. **Paralelización de tareas:**

    - En la función reconstuirCadenaIngenuoPar, se emplea la expresión task para generar las cadenas en paralelo.
    - En la función reconstruirCadenaMejoradoPar, se utilizan tareas (task) para realizar operaciones recursivas de manera paralela.
    - La función reconstruirCadenaTurboPar implementa una estrategia similar al dividir las tareas en subprocesos paralelos mediante la creación de tareas con task.
    - En la función reconstruirCadenaTurboMejoradaPar, se utiliza la paralelización de tareas mediante task para generar y evaluar candidatos en paralelo.

2. **Paralelización de datos:**

    - La función compararSecuencialParalela permite comparar los tiempos de ejecución secuenciales y paralelos, brindándonos evaluación cuantitativa del impacto del paralelismo en el rendimiento.
    - La paralelización de datos se puede evidenciar en varias funciones, como reconstruirCadenaTurboPar y reconstruirCadenaTurboMejoradaPar, donde se dividen las tareas de manera que diferentes subconjuntos de datos se procesen en paralelo.

**¿Qué impacto tuvieron en el desempeño del programa?**

El impacto de cada una de las funciones en el programa se evaluó con respecto a la eficiencia temporal y el uso de recursos computacionales, en el contexto de la reconstrucción de cadenas. Si bien todas las funciones lograron cumplir satisfactoriamente con el objetivo principal en términos de cadenas de tamaño moderado, es crucial señalar que la eficacia de estas soluciones se vio reducida al aplicar paralelización. Aunque las funciones lograron demostrar su viabilidad para secuencias de dimensiones más modestas, la extensión a tamaños de secuencia considerables resultó impracticable debido a los prolongados tiempos de ejecución observados. En consecuencia, se concluye que, a pesar de la eficacia en situaciones limitadas, la aplicación de la paralelización no constituye una mejora sustancial para la resolución del problema.

En términos de manejos de las técnicas de paralelización concluimos que no es la estrategia adecuada para este caso debido a que cada una de estas funciones generan demasiadas tareas, y estas mismas conllevan a demasiados procesos los cuales son demasiados tardíos, haciendo que el problema se extienda a más de lo debido.

**Análisis de los resultados**

- **Solución ingenua:**

| Caracteres | reconstuirCadenaIngenuo | reconstuirCadenaIngenuoPar | aceleración |
|---|---|---|---|
| 1 | 0.0019553 | 0.0063908 | 0.305955436 |
| 2 | 0.0006460 | 0.0012667 | 0.509986579 |
| 3 | 0.0015348 | 0.0013574 | 1.130691027 |
| 4 | 0.0012642 | 0.0011563 | 1.093314884 |
| 5 | 0.0130501 | 0.0035264 | 3.700686252 |
| 6 | 0.0055568 | 0.0040288 | 1.379269261 |
| 7 | 0.0105262 | 0.007585 | 1.387765326 |
| 8 | 0.0314455 | 0.0142229 | 2.210906355 |
| 9 | 0.1807824 | 0.1625379 | 1.112247667 |
| 10 | 0.9608236 | 0.9563801 | 1.004646165 |
| 11 | 4.3524028 | 3.1004656 | 1.403790063 |
| 12 | 8.2957184 | 16.6270991 | 0.498927585 |

- **Solución Mejorada:**

| Caracteres | reconstuirCadenaMejorado | reconstruirCadenaMejoradoPar | aceleración |
|---|---|---|---|
| 1 | 0.0037912 | 0.0095021 | 0.398985487 |
| 2 | 6.76E-04 | 0.0084407 | 0.080028908 |
| 3 | 0.0130204 | 0.0022683 | 5.740157827 |
| 4 | 0.0074661 | 0.0028137 | 2.653481181 |
| 5 | 0.0113447 | 0.0143266 | 0.791862689 |
| 6 | 0.0084908 | 0.0242662 | 0.349902333 |
| 7 | 0.0264082 | 0.0410365 | 0.643529541 |
| 8 | 0.0903061 | 0.1179745 | 0.765471352 |
| 9 | 0.2064924 | 0.1143935 | 1.805106059 |
| 10 | 0.6100858 | 0.4487761 | 1.359443607 |
| 11 | 2.3980774 | 2.1574778 | 1.111518923 |
| 12 | 10.4993717 | 7.895833 | 1.329735786 |
| 13 | 47.2545055 | 22.099993 | 2.138213596 |

- **Solución Turbo:**

| Caracteres | reconstruirCadenaTurbo | reconstruirCadenaTurboPar | aceleración |
|---|---|---|---|
| 1 | 0.0016668 | 0.0007909 | 2.1074725 |
| 2 | 0.0026776 | 0.0071326 | 0.375403079 |
| 3 | 0.0013325 | 0.0067844 | 0.196406462 |
| 4 | 0.0024359 | 0.002783 | 0.875278476 |
| 5 | 0.0064441 | 0.007251 | 0.888718797 |
| 6 | 0.0169546 | 0.0138037 | 1.228264886 |
| 7 | 0.0076439 | 0.0141587 | 0.539873011 |
| 8 | 0.039627 | 0.1035537 | 0.38267102 |
| 9 | 0.2857338 | 0.3288599 | 0.868861786 |
| 10 | 0.9909723 | 1.3022953 | 0.760942852 |
| 11 | 10.1511709 | 8.1353219 | 1.247789703 |

- **Solución Turbo Mejorada:**

| Caracteres | reconstruirCadenaTurboMejorada | reconstruirCadenaTurboMejoradaPar | aceleración |
|---|---|---|---|
| 2 | 0.0045062 | 0.013156 | 0.342520523 |
| 4 | 0.0012179 | 0.0013195 | 0.923001137 |
| 8 | 0.0026761 | 0.0022042 | 1.21409128 |
| 16 | 0.0077216 | 0.0068132 | 1.133329419 |
| 32 | 0.008163 | 0.0111098 | 0.734756701 |
| 64 | 0.0947142 | 0.0463815 | 2.042068497 |
| 128 | 0.2812563 | 0.2868334 | 0.980556309 |
| 256 | 1.9754965 | 1.9603578 | 1.007722417 |
| 512 | 15.6840971 | 15.4778469 | 1.01332551 |
| 1024 | 129.6691165 | 142.3223759 | 0.911094378 |

- **Comparación entre la función Ingenua secuencial e Ingenua paralelizada:**

  Como se puede observar en la tabla de resultados de la solución ingenua es mejor usar su versión paralela, dado que en la mayoría de “Caracteres” se tiene un menor tiempo de ejecución.

- **Comparación entre la función Mejorada secuencial y Mejorada paralelizada:**

  Con los resultados observados en las tablas anteriores, se puede decir que es mejor utilizar su versión paralela dado que a partir de 9 caracteres su tiempo de ejecución es menor, pero también se puede usar su versión secuencial si se tiene 8 o menos caracteres dado que en estos tiempos de ejecución es menor.

- **Comparación entre la función Turbo secuencial y Turbo Paralelizada:**

  En los resultados presentes en la tabla de comparación de estas dos funciones se puede observar que es mejor usar su versión secuencial reconstruirCadenaTurbo dado que en la mayoría de caracteres se tiene que su tiempo de ejecución es menor.

- **Comparación entra la función Turbo mejorada secuencial y Turbo mejorada paralelizada:**

  En este caso, se observa que la aceleración es variable y no sigue una tendencia clara.

  Lo cual nos dice que, para algunos casos, como “Caracteres” igual a 4, 8, y 16, la versión paralela muestra una aceleración positiva, indicando mejoras en el rendimiento. Sin embargo, para “Caracteres” igual a 1024, la versión paralela muestra una aceleración inferior a 1, indicando un rendimiento más lento que la versión secuencial en este caso específico.

- **Conclusión a partir del grado de la aceleración:**

  Respecto al grado de la aceleración se puede concluir que depende de los tiempos que se obtuvieron en cada uno de los tamaños de “Caracteres” correspondientes de las funciones secuenciales y paralelas, entre más alto sea el tiempo de la versión secuencial y entre menor sea el tiempo de la versión paralela, el cociente (aceleración) será mejor.

**Conclusión:**

Se presentan soluciones secuenciales y paralelas, destacando la complejidad computacional del problema. Aunque se implementan estrategias de paralelización, los resultados muestran que, en términos generales, la eficiencia de las soluciones no mejora significativamente al aplicar paralelización. La elección entre soluciones secuenciales y paralelas depende del tamaño de la cadena, y la aceleración varía según el enfoque. Se concluye que, para este problema específico, la paralelización no constituye una mejora sustancial en la eficiencia temporal.