Méndez Pool Joan de Jesús / 160300102 IDeIO Miércoles 10/10/2018
Docente: Flores Granados / David Israel

Algoritmo Práctico de Mapas de Restricción

Halmiton Smith descubrió en 1970 que la enzima de restricción HindII que parte a las moléculas del ADN para cada secuencia GTGCAC ó GTTAAC, partiendo una molécula larga en un conjunto de fragmentos de restricción.

Anteriomente vimos la implementación de dos bioalgoritmos de fuerza bruta aplicados para resolver los cortes que produce la enzima de restricción los cuales se realizan en las moléculas del ADN, estos algoritmos fueron Brute_ForcePDP y Another_Brute_ ForcePDP, los cuales se encargaban de realizar todas las comparaciones en las combinatorias de $\binom{n}{k}$ elementos que pertenecen a un conjunto L, el cual dicho conjunto tiene las distancias entre los puntos.

Steven Skiena en 1990 describió un algoritmo de fuerza bruta diferente para el PDP que funciona bien durante la práctica. El primer paso es encuentrar la distancia más grande en L, por lo cual esto debe determinar los dos puntos más externos de X, y podemos eliminar esta distancia de L. Después se soluciona los dos puntos más externos, seleccionamos la mayor distancia restante en L, que habitualmente se llama δ. Uno de los puntos que generó δ debe ser uno de los dos puntos más externos, ya que δ es la mayor distancia restante; por lo tanto, tenemos una de dos opciones para colocar un punto: δ desde el punto más a la izquierda, o δ desde el punto más a la derecha. Se supone que se decide incluir el punto que es δ desde el punto más a la izquierda en el conjunto. Podemos calcular las distancias por pares entre esta nueva posición y todas las otras posiciones que se han elegido, y preguntar si estas distancias existen en el conjunto L. Si es el caso, se procede a eliminar esas distancias de L y se repite seleccionando la siguiente distancia más grande en L. Si estas distancias de pares no están en L, entonces elegimos la posición que es δ desde el punto más a la derecha y realizamos la misma consulta. Sin embargo, si estas distancias de pares tampoco están en L, se debe haber tomado una mala decisión en algún paso anterior y debemos retroceder unos pasos, revertir una decisión e intentar nuevamente. Si alguna vez llegamos al punto donde el conjunto L está vacío, entonces hemos encontrado una solución que es considerada válida.

El algoritmo aprovecha las propiedades de la recursividad para optimizar la solución y evitar usar fuerza bruta para el caso, de esta forma la solución del algoritmo es efectiva en menos operaciones en el algoritmo de Partial Digest.

El algoritmo es el siguiente:

```
PARTIALDIGEST(L)
 1  width ← Maximum element in L
 2  DELETE(width, L)
 3  X ← {0, width}
 4  PLACE(L, X)

PLACE(L, X)
 1  if L is empty
 2       output X
 3       return
 4  y ← Maximum element in L
 5  if Δ(y, X) ⊆ L
 6       Add y to X and remove lengths Δ(y, X) from L
 7       PLACE(L, X)
 8       Remove y from X and add lengths Δ(y, X) to L
 9  if Δ(width − y, X) ⊆ L
10       Add width − y to X and remove lengths Δ(width − y, X) from L
11       PLACE(L, X)
12       Remove width − y from X and add lengths Δ(width − y, X) to L
13  return
```

El tiempo de ejecución para el mejor de los casos es:

$$T(n)=T(n-1)+O(n)$$

El tiempo de ejecución para el peor de los casos es:

$$T(n)=2T(n-1)+O(n)$$

La función del tiempo de ejecución del algoritmo es:

$$T(n)=2T(n-1)+O(n)$$

UPPER BOUND:

Se propone una solución para : $O(n^2)$
Caso base: $(n-2)-T(2)-a$

$T(2)-a \leq c*n^2$
$T(2)-a \leq c*2^2$
$T(2)-a \leq c*4$
$\frac{a}{4} \leq c$

Caso $m \leq n$

$T(m) \leq c*m^2$
$T(n)-2T(n^2-1)+O(n)$
$T(n) \leq 2*c*(n^2-1)/4+O(n)$
$T(n) \leq \frac{(2c)}{4}*(n^2-1)+O(n)$
$T(n) \leq \frac{1}{2}c*n^2-1+O(n)$

Se cumple ya que n^2 es una función que va en crecimiento que se cumple cuando $c \leq 4$

LOWER BOUND:

Proponemos la solución para: $\Omega(n^2 lg(n))$
Caso base: $(n-2) \rightarrow T(2)-a$

$T(2)-a \geq c*n^2 \log(n)$
$T(2)-a \geq c*2^2 \log(2)$
$T(2)-a \geq c*4*1$
$\frac{a}{4} \geq c$

caso m < n

$T(m) \geq c*m^2 \log(m)$
$T(2) \geq 2T(n-1)+O(n) \geq 2[c*(n-1)^2 \log(n)]+O(n)^2$
$T(n) \geq \frac{2}{4}*c(n-1)^2 \log(n)-2c*((n^2)/4)\log(2)+O(n)^2$
$T(n) \geq \frac{1}{2}*cn^2-1\log(n)-\frac{1}{2}*cn^2 \log(2)+O(n)^2$
$T(n) \geq \frac{1}{2}*cn^2 \log(n)-\frac{1}{2}*cn^2 \log(2)+O(n)$
$T(n) \geq \frac{1}{2}*cn^2 \log(n)-\frac{1}{2}*cn^2+O(n)^2$

Se cumple ya que $n^2 lg(n)$ es una función que crece en menor proporción con respecto al peor de los casos.