Parcial de Programación 3

30 de noviembre de 2024

En recuadros con este formato aparecerán aclaraciones que cumplen una función explicativa pero que no eran requeridos como parte de la solución.

Ejercicio 1 (30 puntos)

En una fábrica tenemos que realizar n tareas de corte, $1, \ldots, n$, en ese orden, sobre materiales potencialmente diferentes, $m_1, \ldots, m_n$. Para esto contamos con una máquina cortadora, la cual lleva instalada una cuchilla que puede ser de tipo A o de tipo B. Dependiendo del material que cortemos, cada tipo de cuchilla sufre un desgaste diferente que provoca un cierto costo económico; sean $C_A(m)$ y $C_B(m)$ los costos generados por cortar el material m con cuchillas de tipo A y B, respectivamente. Al momento de iniciar cada tarea somos libres de instalar el tipo de cuchilla que nos convenga. Sin embargo, cambiar un tipo de cuchilla por otro implica detener la máquina por cierto tiempo, lo cual ocasiona un costo económico T. Nuestro problema consiste en elegir con qué tipo de cuchilla realizar cada una de las n tareas de modo de minimizar el costo total. Asumimos que al momento de comenzar a trabajar la máquina tiene instalada una cuchilla de tipo A.

(a) ¿Cuál es el costo de realizar la primera tarea con una cuchilla de tipo A? ¿Y con una cuchilla de tipo B? Recuerde que inicialmente la máquina tiene instalada una cuchilla de tipo A.

(b) Escriba una relación de recurrencia que permita resolver el problema mediante un algoritmo de programación dinámica. Especifique claramente los parámetros y la semántica de la función que defina. Explique la recurrencia.

(c) Escriba un algoritmo iterativo de programación dinámica que da como resultado el costo total óptimo.

(d) Escriba un algoritmo para obtener una solución óptima. La solución es una secuencia de largo n, en la que el $i-$ésimo elemento indica qué cuchilla, A o B, debe usarse para la tarea i. Puede usar estructuras de datos construidas en el algoritmo de la parte (c).

Solución:

En este ejercicio se espera:

(a) Dar las dos expresiones con la explicación de por qué son correctas. Por ejemplo, Considerando que la máquina inicialmente tiene instalada la cuchilla A y que para cada tarea existen dos opciones, utilizar la cuchilla A o la B, el costo de realizar el primer corte al material m_1 con la cuchilla A solo implica considerar el costo del desgaste $C_A(m_1)$, mientras que el costo de realizarlo con la cuchilla B implica considerar el costo del desgaste $C_B(m_1)$ y el costo de sustitución de cuchillas T.

(b) Diseñar una descomposición del problema en una estructura de subinstancias (subproblemas) de manera apropiada para resolver el problema mediante programación dinámica. La solución de cada instancia se debe poder obtener de manera eficiente mediante las soluciones de algunas de sus subinstancias. Para este problema la estructura de subinstancias debe tener dos dimensiones, una para el conjunto de tareas, y otra para la cuchilla. Entonces, se debe:

- Definir una función que calcula el costo mínimo de la subinstancia, por ejemplo $OPT(i, X)$, donde i representa un conjunto de tareas, y X un tipo de cuchilla. Debe definirse cuál es el rango de valores de i.
- Explicar claramente que significan i y X. Por ejemplo, i puede representar las primeras i tareas, las últimas i tareas, o las tareas que van desde la i hasta la n. El valor de X puede representar la cuchilla con la que se corta el material, o la que estaba instalada (la que se usó para la tarea anterior).
- Obtener el valor de $OPT(i, X)$ de los valores de OPT para subinstancias en las que el conjunto de tareas esté contenido de manera estricta en las de $OPT(i, X)$. Por ejemplo, si el parámetro i representa las tareas desde la 1 hasta la i, el cálculo de $OPT(i, X)$ se puede obtener de $OPT(i-1, X)$ y $OPT(i-1, \overline{X})$, donde $\overline{X}$ es la otra cuchilla.
- Incluir los casos base, aquellos en los que i representa solo una o ninguna tarea.
- Explicar por qué el planteo es correcto.

(c) Traducir la recurrencia en un algoritmo iterativo. Para eso se va a poblar una tabla bidimensional, en la que para cada par (tareas, cuchilla) haya una celda. Se debe:

- Empezar el algortimo con los casos base.
- Recorrer la tabla de modo que al evaluar la expresión con la que se calcula el valor de una celda, ya estén calculados los valores de las celdas involucradas en la expresión.
  Se debe calcular el valor de cada celda de la tabla, especificamente, para cada i se deben calcular dos valores, uno para cada cuchilla.
- Utilizar la expresión definida en la recurrencia para asignar el valor de la celda.
- Devolver el valor que representa el costo mínimo de realizar las n tareas asumiendo que inicialmente está instalada la cuchilla A. Este valor puede ser el de la última celda calculada, o una operación que involucre a algunas de ellas.

(d) Escribir un algoritmo que obtiene la solución del problema, esto es, una secuencia de largo n en la que en cada posición se indica la cuchilla que se debe usar en la tarea correspondiente para que el costo de todo el trabajo sea mínimo. Se debe:

- Recorrer la tabla construida en el punto anterior en el sentido opuesto del parámetro i, el que reprrsenta las tareas, a como fue construida. Es decir, se empieza por las celdas que representan las n tareas y se termina por las que representan solo una o ninguna tarea.
- En cada paso se procesa una celda que representa una tarea menos que en el paso anterior. Concretamente, para cada i se procesa solo una celda.
- La determinación de la cuchilla a usar debe ser consistente con la relación de recurrencia con la que se calculó el costo mínimo para la instancia representada por la celda.
- Se debe mantener cuál es la cuchilla elegida porque de eso depende si en el siguiente paso la celda que se procesa es (i, X) o $(i, \overline{X})$.

(a) Como inicialmente la máquina tiene instalada una cuchilla de tipo A, si se usa el tipo A para la primera tarea el costo es simplemente el del desgaste, que es $C_A(m_1)$. Si en cambio se usa el tipo B, además del costo de desgaste incurrimos el de intercambio de cuchillas, de modo que el costo en este caso es $C_B(m_1) + T$.

(b) Para un entero i, $1 \leq i \leq n$, y un tipo de cuchilla X, $X \in \{A, B\}$, definimos $OPT(i, X)$ como el costo óptimo para realizar las primeras i tareas terminando con una cuchilla de tipo X. A partir de la parte (a) concluimos que para $i = 1$ se cumplen las ecuaciones (2) y (3). Para cada paso i, $1 < i \leq n$, tenemos dos opciones: usamos la cuchilla que ya está instalada o la cambiamos. En cualquiera de los dos casos, si llamamos X al tipo de cuchilla que decidimos usar en el paso i, incurrimos un costo de desgaste $C_X(m_i)$; en el segundo caso incurrimos un costo adicional T por el cambio de cuchilla. Observamos que una solución óptima que usa una cuchilla de tipo X para la tarea i debe minimizar el costo incurrido en las primeras $i-1$ tareas y terminar usando la cuchilla de tipo X en el primer caso y del tipo opuesto en el segundo caso. Denotando $\bar{X}$ el tipo de cuchilla opuesto a X, vemos que la función OPT satisface (1).

$$OPT(i, X) = C_X(m_i) + \min\{OPT(i-1, X), OPT(i-1, \bar{X}) + T\}, \quad 1 < i \leq n, X \in \{A, B\} \tag{1}$$
$$OPT(1, A) = C_A(m_1), \tag{2}$$
$$OPT(1, B) = C_B(m_1) + T. \tag{3}$$

La siguiente recurrencia no resuelve el problema.

$$OPT(i) = OPT(i-1) + \begin{cases} \min\{C_A(m_i), T + C_B(m_i)\} & \text{si la tarea } i-1 \text{ se realizó con } A \\ \min\{C_B(m_i), T + C_A(m_i)\} & \text{si la tarea } i-1 \text{ se realizó con } B \end{cases}$$

La información de cuál fue la cuchilla utilizada no es parte de la recurrencia. Para que esa información esté disponible es que se necesita el segundo parámetro.

Pero además, esta recurrencia no es correcta. Si lo fuera, ya se podría construir la solución en esta recorrida, sin necesitar el algoritmo de la siguiente parte. Se está aplcando una estrategia *Greedy*. Aplicando esta recurrencia al contraejemplo que se muestra en la parte (d) este algoritmo obtendría $OPT(1) = 6$, porque si solo se realiza la primera tarea se debe elegir la cuchilla B, y $OPT(2) = 21$. Pero la solución es elegir la cuchilla A para ambas tareas.

Tampoco es correcta la siguiente solución que se basa en la última posición en que se cambia la cuchilla:

$$OPT(0) = 0,$$
$$OPT(i) = \min\left\{\sum_{j=1}^{i} C_A(m_j), \min_{1 \leq k \leq i}\left\{OPT(k-1) + T + \min\left\{\sum_{j=k}^{i} C_A(m_j), \sum_{j=k}^{i} C_B(m_j)\right\}\right\}\right\}.$$

El primer término corresponde al caso en que nunca se cambió la cuchilla. En el segundo término el último cambio se produce para realizar la tarea k; se suma el costo mínimo hasta la tarea $k-1$, el costo T del cambio, y la mejor de las dos opciones desde k hasta i manteniendo la misma cuchilla.

El error en esta solución es que para que sea correcto sumar T se debe saber que la tarea $k-1$ se realizó con una cuchilla distinta a la que se va a usar desde la tarea k, y esa información no está en la recurrencia.

La siguiente es otra solución incorrecta en la que en $OPT(i, X)$ la i representa las tareas desde la 1 hasta la i, igual que antes, pero X no necesariamente es la cuchilla usada en la tarea i, sino que es la que estaba instalada al momento de decidir cómo realizar la tarea i. Veamos solo una de las dos ecuaciones

$$OPT(i, A) = \text{mín}\{OPT(i-1, A) + C_A(m_i), OPT(i-1, B) + T + C_B(m_i)\}.$$

en donde se tiene en cuenta que la tarea i puede realizarse con la cuchilla A, pero también con la cuchilla B, y en el último caso se debe agregar el costo del cambio de cuchilla.

El error aquí es que para $OPT(i-1, X)$ no se está tomando el mismo significado que para $OPT(i, X)$, sino que se asume que X es la cuchilla con la que se realizó la tarea $i-1$ y no la cuchilla que estaba instalada.

En esta versión el parámetro de la cuchilla también representa la que está instalada.

$$\begin{aligned} OPT(i, A) = \text{mín}\{&OPT(i-1, A) + C_A(m_i), \\ &OPT(i-1, A) + T + C_B(m_i), \\ &OPT(i-1, B) + C_A(m_i), \\ &OPT(i-1, B) + T + C_B(m_i)\}, \end{aligned}$$

que es lo mismo que

$$OPT(i, A) = \text{mín}\{OPT(i-1, A), OPT(i-1, B)\} + \text{mín}\{C_A(m_i), T + C_B(m_i)\}.$$

Aquí correctamente se asume que para la tarea i está instalada la cuhilla A. El error es que no se sabe si ninguno de los valores $OPT(i-1, A)$ o $OPT(i-1, B)$ se obtienen usando la cuhilla A.

(c) El algoritmo para calcular el costo total mínimo se presenta en la figura 1.

```
1 OPT[1, A] ← C_A(m_1)
2 OPT[1, B] ← C_B(m_1) + T
3 para i = 2 a n
4 |   OPT(i, A) ← C_A(m_i) + mín{OPT(i − 1, A), OPT(i − 1, B) + T}
5 |   OPT(i, B) ← C_B(m_i) + mín{OPT(i − 1, B), OPT(i − 1, A) + T}
6 Devolver mín{OPT(n, A), OPT(n, B)}
```

Figura 1: Algoritmo para calcular el costo total mínimo. Como producto secundario calcula $OPT(i, X)$, para $1 \leq i \leq n$, $X \in \{A, B\}$.

(d) El algoritmo de la figura 2 construye una solución óptima en un arreglo de tamaño n, SOL, donde el contenido de cada posición i, $1 \leq i \leq n$, indica el tipo de cuchilla que se usa para la tarea i. El algoritmo utiliza el arreglo OPT producido por el algoritmo de la parte anterior.

```
1 si OPT(n, A) < OPT(n, B) entonces X ← A en otro caso X ← B
2 para i = n downto 2
3 |   SOL[i] ← X
4 |   si OPT(i − 1, X̄) + T < OPT(i − 1, X) entonces X ← X̄
5 SOL[1] ← X
6 Devolver SOL
```

Figura 2: Algoritmo para construir una solución óptima.

Es incorrecto construir la solución comparando en cada paso los valores de $OPT[i, A]$ y $OPT[i, B]$;

```
1 para i = n downto 1
2 |  si OPT(i, A) < OPT(i, B) entonces
3 |  |  SOL[i] ← A
4 |  en otro caso
5 |  |  SOL[i] ← B
6 Devolver SOL
```

Se debe notar que el resultado sería el mismo si la recorrida se hiciera en orden ascendente de i, el mismo que el de la parte (c). Siendo así, esta solución podría haberse construido en esa parte anterior, como en las estrategias *Greedy*.

Veamos el siguiente contraejemplo, en el que $T = 5$.

i	1	2
$C_A(m_i)$	10	10
$C_B(m_i)$	1	100

El cálculo de la tabla de valores óptimos da el siguiente resultado.

i	1	2
OPT_A	10	20
OPT_B	6	106

La solución incorrecta obtenida de esta manera es $[B, A]$, cuyo costo es 21, debido a los dos cambios, más $C_B(1) + C_A(2)$.

Pero la solución correcta es $[A, A]$ con costo 20. En cada paso se debe registrar cuál es la cuchilla correcta, y determinar, con base en la relación de recurrencia, de que manera se llegó a esa decisión.

Veamos una solución alternativa en la que el subproblema $i-$ésimo consiste en determinar el costo mínimo para realizar desde la tarea i hasta la tarea n. Tal como en la versión anterior, en la estructura de subproblemas hay que tener en cuenta la cuchilla, que en esta versión va a ser la cuchilla que estaba instalada al empezar a realizar la tarea (o sea, la que se usó en la tarea anterior, o A, si la tarea es la 1). Para presentar otra notación, en lugar de tratar la cuchilla como un segundo parámetro de la recurrencia, mantendremos dos relaciones de recurrencia $OPT_A(i)$ y $OPT_B(i)$. Entonces, para $X \in \{A, B\}$, $1 \leq i \leq n$, $OPT_X(i)$ es el costo mínimo para realizar desde la tarea i hasta la tarea n, ambas incluidas, si al empezar la tarea i está instalada la cuchilla X (que se puede cambiar).

(a) Igual a la versión anterior.

(b) Si al empezar la tarea i está instalada la cuchilla A las opciones son

- mantener esa cuchilla, procesar el material m_i con un costo $C_A(m_i)$ y se entrega la máquina para realizar la tarea $i + 1$ teniendo instalada la cuchilla A, o
- cambiar de cuchilla con un costo T, procesar el material m_i con un costo $C_B(m_i)$ y se entrega la máquina para realizar la tarea $i + 1$ teniendo instalada la cuchilla B.

Si está instalada la cuchilla B los argumentos son análogos.

Como caso base podemos agregar una tarea $n + 1$ con costo nulo.

Entonces, tenemos la siguiente relación de recurrencia, con $X \in \{A, B\}$.

$$\begin{aligned} OPT_X(n+1) &= 0, \\ OPT_X(i) &= \min\{C_X(m_i) + OPT_X(i+1), T + C_{\overline{X}}(m_i) + OPT_{\overline{X}}(i+1)\}, 1 \leq i \leq n. \end{aligned}$$

Teniendo en cuenta que al empezar la tarea 1 está instalada la cuchilla A, no hace falta calcular $OPT_B(1)$.

(c)

1 $OPT_A[n+1] \leftarrow 0, OPT_B[n+1] \leftarrow 0$
2 **para** i **desde** n **decrementando hasta** 1
3 **para** $X \in \{A, B\}$
4 $OPT_X[i] = \text{mín}\{C_X(m_i) + OPT_X[i+1], T + C_{\overline{X}}(m_i) + OPT_{\overline{X}}[i+1]\}$
5 Devolver $OPT_A[1]\}$.

(d)

1 $X \leftarrow A$
2 **para** i **desde** 1 **hasta** n
3 **si** $OPT_X(i) < C_X(m_i) + OPT_X[i+1]$ **entonces** $X \leftarrow \overline{X}$
4 $SOL[i] \leftarrow X$
5 Devolver SOL

Ejercicio 2 (30 puntos)

Consideramos un conjunto de cláusulas $\mathcal{C} = C_1, C_2, \ldots, C_k$, donde cada cláusula es la disyunción de exactamente 4 términos sobre 4 variables booleanas distintas,

$$C_i = t_{i,1} \vee t_{i,2} \vee t_{i,3} \vee t_{i,4}\,.$$

Denotamos con $X = \{x_1, x_2, , \ldots, x_n\}$ al conjuntos de variables que participan en las cláusulas de $\mathcal{C}$. Definimos el problema de decisión *4-SAT* de la siguiente manera: ¿Existe una asignación de verdad para las variables de X tal que todas las cláusulas de $\mathcal{C}$ se satisfacen?

Una instancia del problema se representa mediante dos matrices de dimensiones $k \times 4$, denotadas V y N. Para cada i, $1 \leq i \leq k$, y cada j, $1 \leq j \leq 4$, $V_{i,j}$ es un índice en el conjunto $\{1, 2, \ldots, n\}$ que identifica a la variable que participa en el término $t_{i,j}$, y $N_{i,j}$ es un valor booleano que es igual a **true** si el término $t_{i,j}$ es la negación de una variable. Por ejemplo, para

$$C_i = x_5 \vee \bar{x}_2 \vee x_8 \vee x_4\,,$$

la fila i de V es $(5, 2, 8, 4)$ y la fila i de N es (**false, true, false, false**).

Se pide:

Demuestre que *4-SAT* es NP-Completo.

Solución:

(a)

De la respuesta se espera lo siguiente:

- Una estrategia clara y detallada para demostrar que 4-SAT es NP-completo, que debe incluir:
  1. Mostrar que 4-SAT pertenece a NP.
  2. Probar que $\forall Y \in NP, Y \leq_p 4-SAT$.
- Para demostrar que el problema pertenece a NP, se espera:
  - La definición de un certificado apropiado para el problema.
  - La definición de un algoritmo de certificación adecuado, con una descripción de sus entradas.
  - La aplicación precisa de la definición de NP, incluyendo:
    - Mostrar que el certificador eficiente propuesto opera en tiempo polinomial respecto a la suma de los largos de sus entradas.
    - Mostrar que el certificado propuesto para una instancia SÍ tiene un largo polinomial en el tamaño de la instancia.
    - Demostrar que una instancia es SÍ si y solo si existe un certificado que hace que el certificador responda SÍ.
- Para concluir que 4-SAT es NP-completo, se espera:
  - Seleccionar un problema conocido que sea NP-completo.
  - Describir una transformación en tiempo polinomial desde el problema conocido hacia 4-SAT. En particular, realizar la reducción en sentido contrario se considera un error grave.
  - Justificar que la transformación se realiza en tiempo polinomial.
  - Demostrar que la transformación garantiza que las respuestas de los dos problemas son equivalentes, es decir, demostrar que una instancia del problema conocido es SÍ si y solo si su transformación en una instancia de 4-SAT es SÍ.
  - Concluir que el problema es NP-completo basándose en la propiedad de transitividad de la reducción polinómica.

Una instancia de *4-SAT* está dada por un par de matrices, $I_{4S} = (N, V)$, como se especifica en la letra.

Definimos un certificado para *4-SAT* como una tira de n valores booleanos, $c = c_1, c_2, \ldots, c_n$, donde cada c_h determina el valor en una asignación de verdad para la variable x_h.

Un algoritmo certificador recibe como entradas una instancia, I_{4S}, y un certificado, c, y evalúa las k cláusulas con los valores $x_h = c_h$, $1 \leq h \leq n$. Si todas las cláusulas se satisfacen devuelve 1 y en caso contrario devuelve 0.

1. El tiempo de ejecución es $O(k)$, ya que cada cláusula implica evaluar 4 términos, que requiere tiempo $O(1)$. Este tiempo de ejecución es lineal en el tamaño de N, que es $4k$, por lo cual, con más razón, es polinomial en la suma de los tamaños de I_{4S} más el de c.
2. Además, como todas las variables aparecen en al menos alguna cláusula, tenemos $n \leq 4k$, por lo cual el tamaño de c es menor o igual que el de N. En consecuencia, el tamaño de c es polinomial en el tamaño de la representación de I_{4S}.
3. (Directo) Si la instancia I_{4S} es SÍ, existe una asignación de verdad de las variables X que satisface las k cláusulas y por tanto, existe un certificado c que hace que el certificador devuelva 1.
4. (Recíproco) Si existe un certificado que hace que el certificador devuelva 1, en ese caso la verificación de todas las cláusulas fue verdadera, por lo que existe una asignación de verdad para las variables X que satisface todas las cláusulas, es decir, la instancia I_{4S} es SÍ.
5. Por los dos ítems anteriores, queda demostrada la correctitud del algoritmo certificador.

A partir de los puntos anteriores concluimos que existe un certificador eficiente para *4-SAT*, por lo cual *4-SAT* pertenece a la clase $\mathcal{NP}$.

Mostramos a continuación que *3-SAT* $\leq_P$ *4-SAT*. Una instancia de *3-SAT* con k' cláusulas, $\mathcal{C}'$, está dada por un par de matrices, $I_{3S} = (N', V')$, con una representación análoga a la de I_{4S}, pero de dimensiones $k' \times 3$. Denotamos con $X' = \{x_1, x_2, , \ldots, x_{n'}\}$ al conjuntos de variables que participan en las cláusulas de $\mathcal{C}'$.

Reducimos I_{3S} a I_{4S} de la siguiente manera.

1. El conjunto de variables sobre los que definimos las cláusulas es $X = X' \cup \{u, v, w, z\}$.
2. Definimos $k = k' + 8$.
3. Para cada i, $1 \leq i \leq k'$, definimos

$$C_i = t'_{i,1} \vee t'_{i,2} \vee t'_{i,3} \vee z\,, \qquad 1 \leq i \leq k'\,, \tag{4}$$

   donde $t'_{i,j}$, $1 \leq j \leq 3$, son los términos de la cláusula de C'_i de I_{3S}.
4. Para cada $k' < i \leq k' + 8$, definimos 8 cláusulas distintas de la firma

$$C_i = t_{i,1} \vee t_{i,2} \vee t_{i,3} \vee \bar{z}\,, \qquad k' < i \leq k' + 8\,, \tag{5}$$

   donde los términos $t_{i,1}, t_{i,2}, t_{i,3}$ están definidos sobre las variables u, v, w, respectivamente, y las 8 cláusulas cubren las 8 posibles combinaciones donde cada variable ocurre negada o no negada.

Esta reducción requiere tiempo lineal en el tamaño de I_{3S}, ya que consisten en:

1. Construir V agregando una columna a V' con el índice de la variable z,
2. construir N agregando una columna a N' con entradas en **false**,
3. Agregar 8 filas a V y N para representar las cláusulas de (5).

Si la instancia I_{3S} es SÍ, existe una asignación de verdad para X' que satisface todas sus cláusulas. Tomando esta misma asignación para X, extendida con z = **false** y valores arbitrarios para u, v, w, vemos que todas las cláusulas de I_{4S} también se satisfacen y por lo tanto es SÍ.

Si I_{4S} es SÍ, existe una asignación de verdad para X que satisface todas sus cláusulas. En particular la satisfacción de las cláusulas de (5) implica que z = **false**. Por lo tanto, en todas las cláusulas C_i de (4) se satisface alguno de los términos $t'_{i,j}$, $1 \leq j \leq 3$, lo cual implica que esta asignación, restringida a las variables de X', satisface las cláusulas de I_{3S}, que por lo tanto es SÍ.

Otra reducción.

1. El conjunto de variables sobre los que definimos las cláusulas es $X = X' \cup \{x_{n+1}\}$.
2. Definimos $k = 2k'$.
3. Para cada i, $1 \leq i \leq k'$, definimos

$$C_i = t'_{i,1} \vee t'_{i,2} \vee t'_{i,3} \vee x_{n+1}\,, \qquad 1 \leq i \leq k' \tag{6}$$
$$C_{i+k'} = t'_{i,1} \vee t'_{i,2} \vee t'_{i,3} \vee \overline{x_{n+1}}\,, \qquad 1 \leq i \leq k' \tag{7}$$
$$\tag{8}$$

   donde $t'_{i,j}$, $1 \leq j \leq 3$, son los términos de la cláusula de C'_i de I_{3S}.

Esta reducción requiere tiempo lineal en el tamaño de I_{3S}, ya que consisten en:

1. Construir V creando dos filas iguales a cada fila de V' a las que se agrega una entrada con el valor $n+1$,
2. construir N creando dos filas iguales a cada fila de N', a una de las cuales se agraga una entradas en **false**, y a la otra una entrada en **true**.

Si la instancia I_{3S} es SÍ, existe una asignación de verdad para X' que satisface todas sus cláusulas. Tomando esta misma asignación para X, extendida con un valor arbitrario a x_{n+1} veamos que todas las cláusulas de I_{4S} también se satisfacen y por lo tanto es SÍ. Cada par de cláusulas C_i y $C_{i+k'}$ de I_{4S} proviene de una cláusula C'_i de I_{3S}. En la asignación para X' hay un término t'_{ij} que es evaluado **true**. En la asignación propuesta para X el término t_{ij} es evaluado **true** en las cláusulas C_i y $C_{i+k'}$ de I_{4S} lo que hace que ambas cláusulas se satisfagan.

Si la instancia I_{4S} es SÍ, existe una asignación de verdad para X que satisface todas sus cláusulas. Veamos que esa misma asignación restringida a X' satisface todas las cláusulas de I_{3S}, por lo que es SÍ. Cada cláusula C'_i de I_{3S} generó las cláusulas C_i y $C_{i+k'}$ de I_{4S}. Por construcción, en cada asignación para X, es evaluado **false** o bien cada término $t_{i,4}$, o bien cada término $t_{i+k',4}$. Sin pérdida de generalidad supongamos que cada término $t_{i,4}$ es evaluado **false**. Entonces, como la cláusula C_i se satisface existe un término $t_{i,j}$, $1 \leq j \leq 3$ que es evaluado **true**. En la asignación restingida propuesta para X' el término $t'_{i,j}$ es evaluado **true**, por lo que C'_i se satisface.