Problema del corte de vástagos

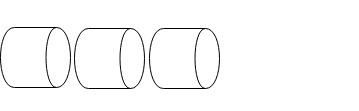
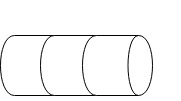
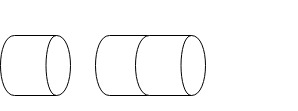
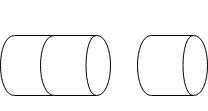
*Pablo Pastor Martín, Isaac Aimán Salas, Javier Ramos Fernández*

Introducción

El problema del ***corte de vástagos*** consiste en, dado un vástago de longitud y una colección de precios que contiene los precios de todas las piezas cuyo tamaño es menor que *n*, se quiere determinar los cortes del vástago de modo que el beneficio obtenido por la suma de los precios de cada una de las piezas resultantes de los cortes sea máximo. Un ejemplo de este problema sería el siguiente:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| longitud | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| precio | 2 | 6 | 9 | 10 | 13 | 15 | 18 | 20 |

En este cuadro tenemos, para diferentes longitudes de corte (, sus correspondientes precios . La tabla está ordenada por la longitud de corte, por lo que es la longitud de corte más pequeña que podemos vender. Por ejemplo, supongamos que la longitud del vástago es 3. Por lo tanto, las diferentes posibilidades de corte son las siguientes:



Podemos observar que , en el caso de que es 3, el beneficio máximo se obtiene sin hacer ningún corte. Si los valores de las longitudes de corte hubieran sido diferentes, el beneficio máximo no tendría por qué obtenerse sin realizar ningún corte. Propiedad de la subestructura óptima.

Si tenemos un vástago de longitud y realizamos un corte, podemos computar el precio que obtenemos de ese corte más el mejor precio de la longitud remanente del vástago. El objetivo es averiguar qué corte maximiza la suma de la primera pieza más el precio óptimo del resto del vástago. Este mejor precio de la longitud remanente del vástago es una versión más pequeña del mismo problema , de modo que una solución óptima del problema general utiliza soluciones óptimas de versiones más pequeñas del mismo problema.

Algoritmos

Fuerza bruta(recursivo)

El algoritmo de fuerza bruta enumera sistemáticamente todos los posibles casos para la solución del problema y elige, de entre todos esos casos, la mejor solución. Es decir, este algoritmo analiza todas la posibilidades de corte sobre un vástago y elige aquella con la que se consiga mayor beneficio.

Como podemos observar, hay maneras diferentes de cortar un vástago (para se pueden realizar 4 cortes, por ejemplo). Esto se puede ver suponiendo que en cada incremento de longitud tenemos la decisión binaria de si hacer o no un corte (obviamente el último incremento no se incluye ya que no produce nuevas piezas). De este modo el número de permutaciones de longitudes es igual al número de patrones binarios de cortes, de tal forma que hay posibilidades. Por lo tanto, para encontrar el valor óptimo, simplemente sumamos los precios de todas las piezas de cada permutación y seleccionamos el valor más alto.

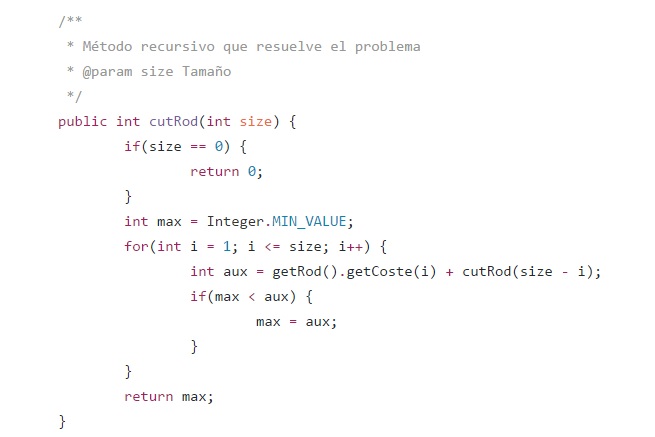
Antes de centrarnos en el algoritmo recursivo, vamos a formalizar el problema. Suponiendo que tenemos una pieza de longitud con un precio , si la solución óptima realiza un corte en el vástago en piezas de longitudes de tal modo que , entonces el beneficio de un vástago de longitud es . Por lo tanto, el valor óptimo se puede encontrar en términos de vástagos más pequeños observando que si realizamos un corte óptimo de longitud (y de este modo dando lugar a una pieza de longitud ), entonces ambas piezas *deben ser óptimas* (y estas piezas más pequeñas serán cortadas). De otro modo podríamos realizar un corte diferente que produjera un beneficio mayor, contradiciendo la suposición de que el primer corte fue óptimo.

A partir de esto, podemos escribir el beneficio óptimo de la siguiente manera:

El primer término () corresponde a no realizar ningún corte del vástago. Los otros términos corresponden a los beneficios máximos obtenidos tras hacer un corte inicial del vástago en dos piezas de longitudes y para y continuar haciendo cortes óptimos en esas piezas.

Si suponemos que no vamos a volver a cortar la primera pieza (ya que debe haber al menos una pieza en la solución óptima) y sólo (posiblemente) cortar la segunda pieza, podemos reescribir el beneficio de la *subestructura óptima* recursivamente como:

donde se repite el proceso para cada pieza posterior. De este modo se puede implementar esta aproximación utilizando una simple rutina recursiva.



/\*\*Meter árbol de recurrencia y análisis de complejidad\*\*/

*Programación dinámica*

El algoritmo recursivo que resuelve el problema de corte de vástagos se basa en resolver subproblemas más pequeños que se repiten varias veces a lo largo de su ejecución. Por lo tanto, estamos ante un algoritmo que divide un problema en subproblemas que se solapan y, así, este problema se puede resolver mediante el método de *programación dinámica*.

***Top-Down***

Si nos fijamos atentamente en el árbol de recurrencia del algoritmo recursivo, podemos observar que realiza más trabajo del necesario, ya que un mismo subproblema se resuelve varias veces. De esta forma, se puede mejorar el tiempo de ejecución del algoritmo si almacenamos las soluciones a los subproblemas en vez de volver a recalcular el mismo subproblema de forma reiterada. No obstante, este tiempo de ejecución mejora a costa de utilizar memoria adicional, por lo que debe haber un compromiso entre la memoria utilizada y el tiempo de ejecución del algoritmo.

Para implementar esta mejora del algoritmo, se puede utilizar la estrategia de procesamiento *top-down*, que consiste en resolver los subproblemas recordando las soluciones por si fueran necesarias nuevamente, combinando la recursión y la memorización. En el caso del problema del corte de vástagos se utiliza el método recursivo anterior, pero se modifica de tal forma que almacena el beneficio óptimo de las diferentes longitudes del vástago para no calcularlas de nuevo.

El algoritmo que implementa esta estrategia sigue un recorrido ***primero en profundo***, ya que se llega primero a los nodos hojas y luego se vá subiendo en el árbol.

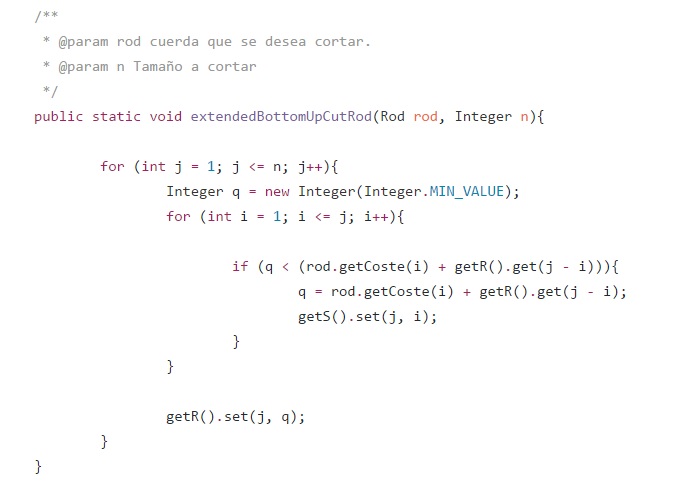
***Bottom-Up***

Otra forma de implementar el algoritmo que resuelve este problema con la programación dinámica es la estrategia *bottom-up*. Se resuelven los problemas empezando por las longitudes pequeñas y se guardan los beneficios óptimos de estas longitudes en un array (de tamaño ). Después, cuando se proceda a evaluar longitudes más grandes, simplemente buscamos estos valores para determinar el beneficio óptimo para la pieza de longitud más grande que las longitudes de las que ya se ha calculado el beneficio máximo.

Esta mejora del algoritmo hace un recorrido ***topológico inverso*** puesto que resuelven primero los nodos hijos (longitudes del vástago más pequeñas) y, a continuación, se van resolviendo los nodos predecesores (longitudes más grandes del vástago).

***Extended-Bottom-Up***

El algoritmo anterior se puede hacer más preciso si, además de guardar los beneficios máximos de las diferentes longitudes, se almacene los puntos donde se realicen esos cortes que hagan que el beneficio sea óptimo. Para ello, se utiliza un array adicional (de tamaño ) que guarde los cortes óptimos para cada tamaño del segmento. De esta manera, para saber los cortes que propician el beneficio máximo del vástago entero, se procede hacia atrás a través de los cortes, examinando y empezando en para ver donde se realiza cada corte subsecuente hasta que , lo que indica que cogemos la última pieza sin más cortes .



Para calcular el tiempo de ejecución del algoritmo utilizando la estrategia *bottom-up*, hay que tener en cuenta que se realizan dos bucles:

* Un bucle que va desde hasta para contemplar todas las longitudes posibles del vástago.
* Otro bucle que va desde hasta para contemplar todos los posibles cortes de un vástago de longitud .

Por tanto, el tiempo de ejecución del algoritmo es:

La última expresión es una progresión aritmética y, por tanto, utilizamos la siguiente fórmula:

Por tanto :

Como podemos observar, utilizando la estrategia *botttom-up* en el algoritmo ha permitido que el tiempo de ejecución del mismo haya pasado de ser *exponencial* a *polinomial*. Por lo tanto, esta estrategia permite que el problema de corte de vástagos sea tratable para tamaño de vástagos suficientemente grande.