*2017*

# Ejercicio 1: Producto máximo con suma fija.

Para resolver un problema con Programación Dinámica, necesitamos que éste tenga naturaleza n-etápica, que verifique el Principio de Optimalidad de Bellman. Además, se debe poder plantear una ecuación de recurrencia que represente la forma de ir logrando etapa por etapa la solución optimal, hasta encontrar la solución final.

## Descripción

Dado *C*, un número no negativo, ¿cuál es el mayor valor que podemos obtener multiplicando *n* números naturales que sumen *C*?

Más técnicamente, buscamos:

Maximizar 

 Sujeto a

Vamos a desarrollar las características necesarias para resolver el problema por Programación Dinámica:

## Naturaleza n-etápica

Sabemos que es un problema n-etápico, ya que tenemos:

*  como el máximo producto alcanzable.
*  como el valor de la primera subdivisión, es decir, del número que, al sumarse con las p partes restantes, resulte
*  como el valor de la suma de las partes restantes

## Verificación del POB

Vamos a verificar el POB mediante reducción al absurdo.

Sea enteros que suman y cuyo producto es máximo (solución óptima).

Se ha obtenido , entero óptimo, por lo que debe ser un subconjunto óptimo. De no ser óptimo:

* Sea una solución óptima, y por lo tanto, mejor que
* Entonces >
* Por lo tanto,  >

Esto es una contradicción, puesto que hemos partido de la hipótesis de que es solución óptima.

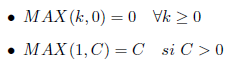
Por lo tanto, el POB se puede verificar y aplicar a este problema.

## Solución

La ecuación que describe el problema es:



### Vamos a demostrar que esto es cierto (y calcular la x máxima entre 1 y C) mediante inducción.

Primero, sabemos que:

(caso más pequeño, k=1)

#### Vamos a resolver, como caso base, para k=2:



Como sabemos que 



Vamos a sacar ese máximo derivando la función  e igualándola a 0.



Sustituyendo, , es decir,





#### Vamos a probar con k=3:



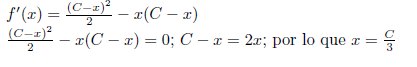


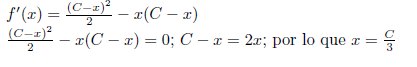
Como sabemos que



Por lo que 

Vamos a sacar ese máximo derivando la función  e igualándola a 0.







Sustituyendo,



Las soluciones son , ya que , y  e es bi-etápico, con soluciones

Los dos ejemplos anteriores nos llevan a pensar el máximo valor alcanzable del problema cuando k=i sería , con soluciones 

Es decir, planteamos la siguiente hipótesis:



Derivando la función entre corchetes e igualando a 0, comprobamos que el *x* máximo es

Para demostrar esto, vamos a usar inducción:

#### Suponiendo la anterior hipótesis como cierta, vamos a ver si se cumple para k=i+1:

De acuerdo con el POB, tenemos que:



Derivando e igualando a 0 la función entre corchetes, llegamos a que , siendo el máximo valor del producto,



Vemos que el resultado tiene sentido, por lo que la hipótesis se puede tomar como cierta.

Una vez demostrado por inducción, podemos llegar a la conclusión de que si tomamos como soluciones alcanzamos el valor máximo, el cual es

## Ecuación recurrente:

La ecuación que resolvería este problema, es:



Un ejemplo de uso: Producto máximo de tres números cuya suma resulte 9.

Anticipadamente, podemos saber el resultado con la fórmula :

Usando la ecuación recurrente:

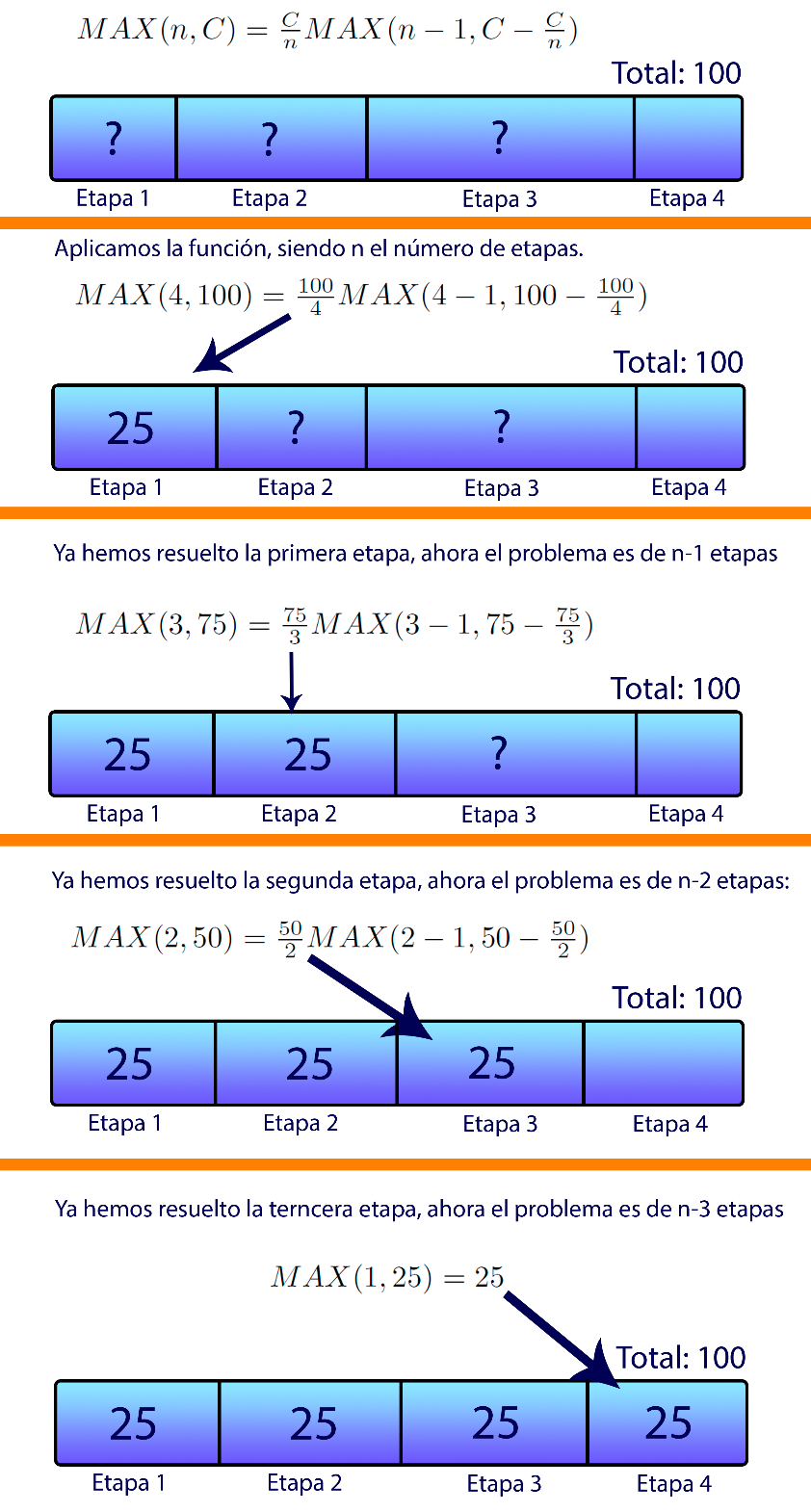


Como vemos, el resultado es correcto.

## Ejemplo visual

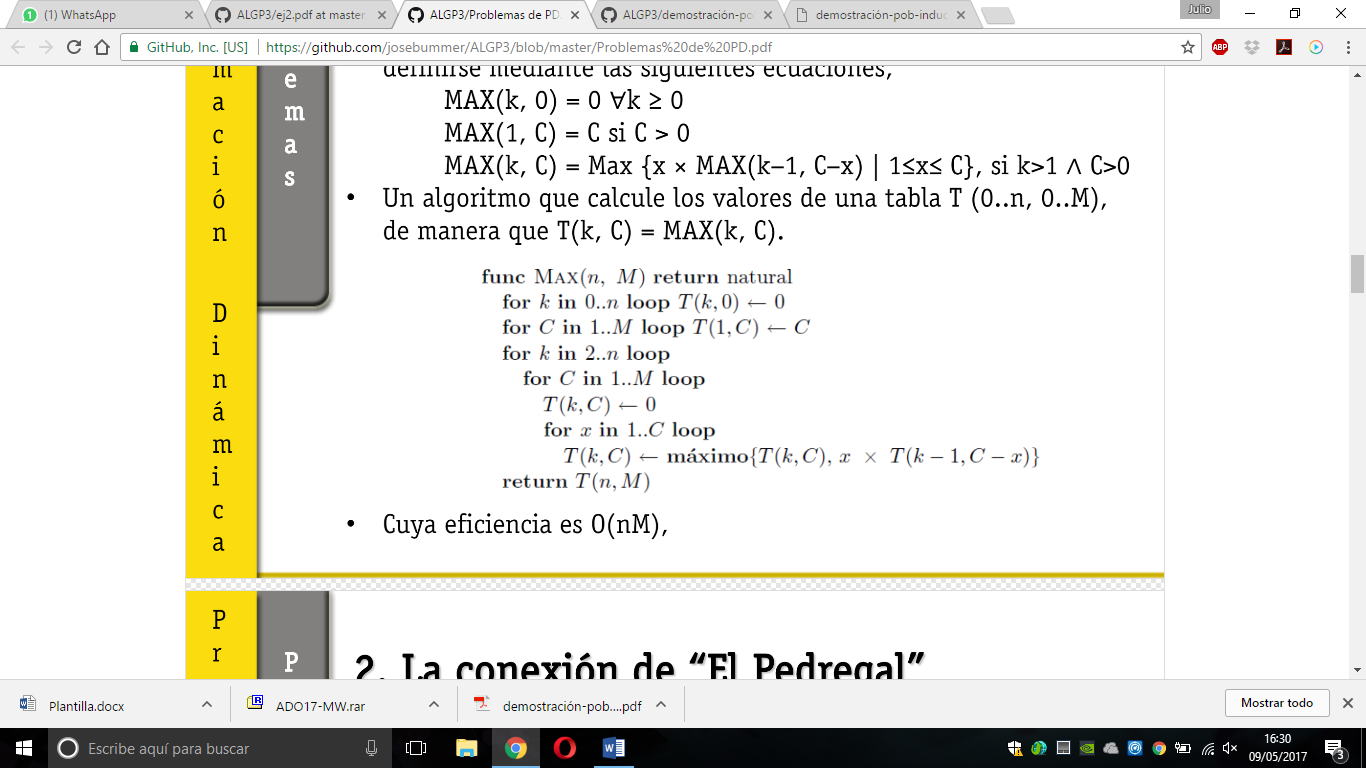
Vamos a comprobar visualmente cómo se resuelve el problema mediante PD para poder apreciar la naturaleza de “etapa a etapa”. En este ejemplo, buscaremos 4 números que sumen 100 y cuyo producto sea máximo. Como vemos, en cada recurrencia se resuelve una etapa, es decir, uno de los 4 números que son solución.

Se ve claramente como sigue el POB, en el que en un problema de n etapas, encuentra la solución óptima para una etapa y se vuelve a resolver el problema para las n-1 etapas restantes.

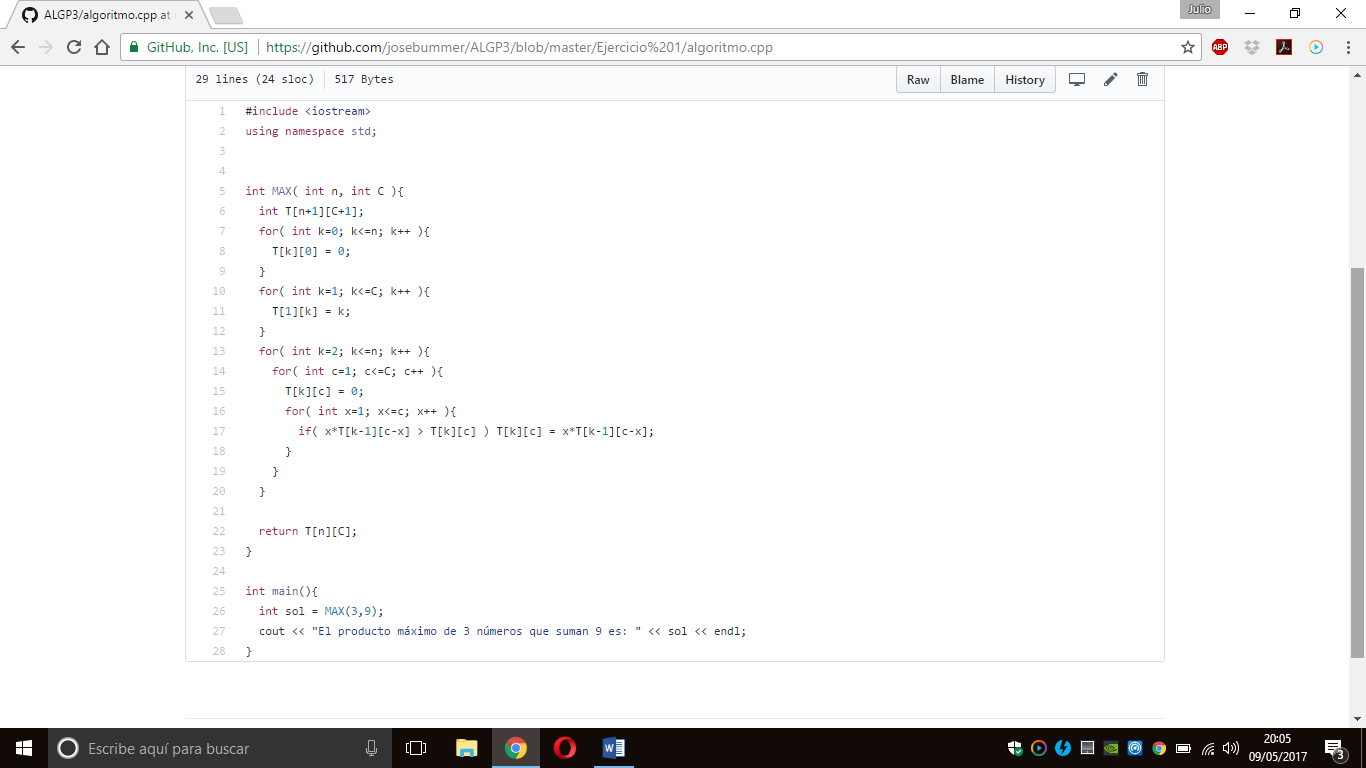


Como vemos, la solución es el producto de los cuatro números resultantes de resolver cada etapa: 25 x 25 x 25 x 25.

## Pseudo-código del algoritmo (eficiencia de O(nM)



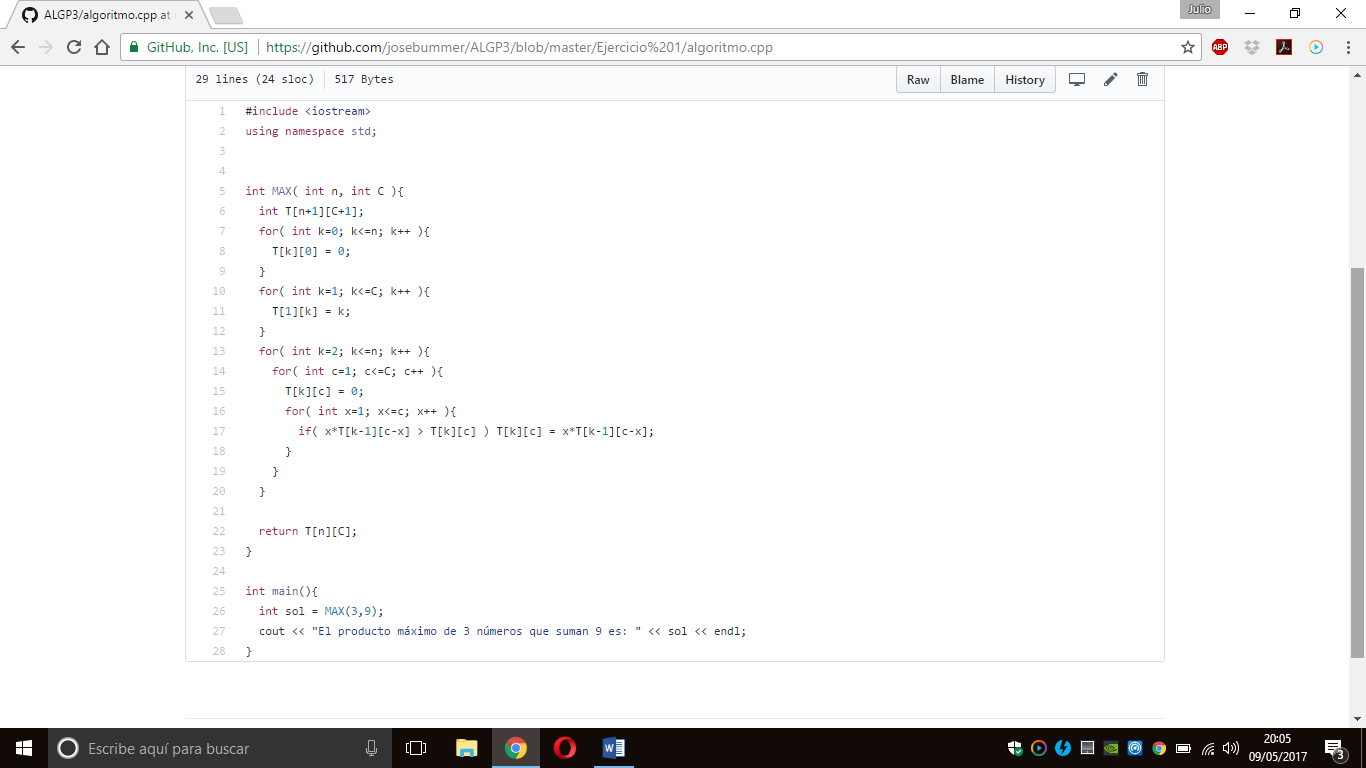
## Código implementado en C++



## Ejecución del algoritmo implementado

Ponemos a prueba el código anterior con los siguientes parámetros:

Máximo producto de 3 números que sumen 9.



El resultado de la ejecución es la siguiente.

https://raw.githubusercontent.com/josebummer/ALGP3/master/Ejercicio%201/Screenshot%20from%202017-05-09%2021-54-18.png?token=AWUMB50J9W1THVA_o2SWA-5LboncgPf7ks5ZG1-ywA%3D%3D

Como vemos, el resultado es correcto.