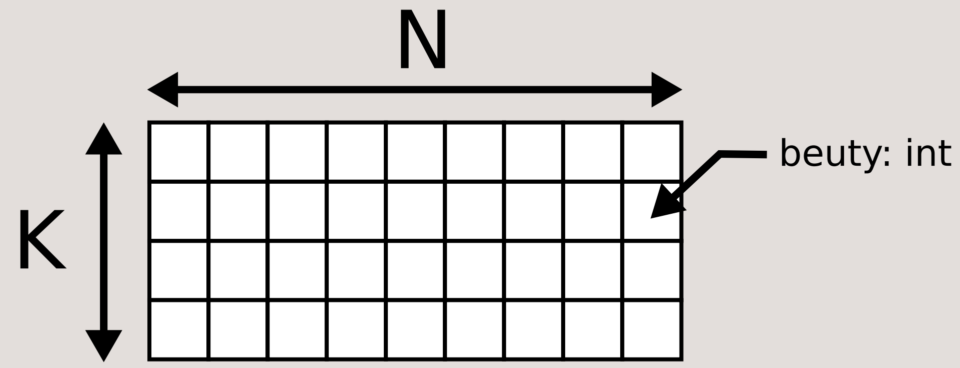
Kickstart round A 2020 Problema B: Plates



Definición

1<=i<=N
0<=j<=P

dp[i][j] = belleza máxima posible considerando los i primeros stacks, usando exactamente j platos

dp[i][j] = máxima beuty

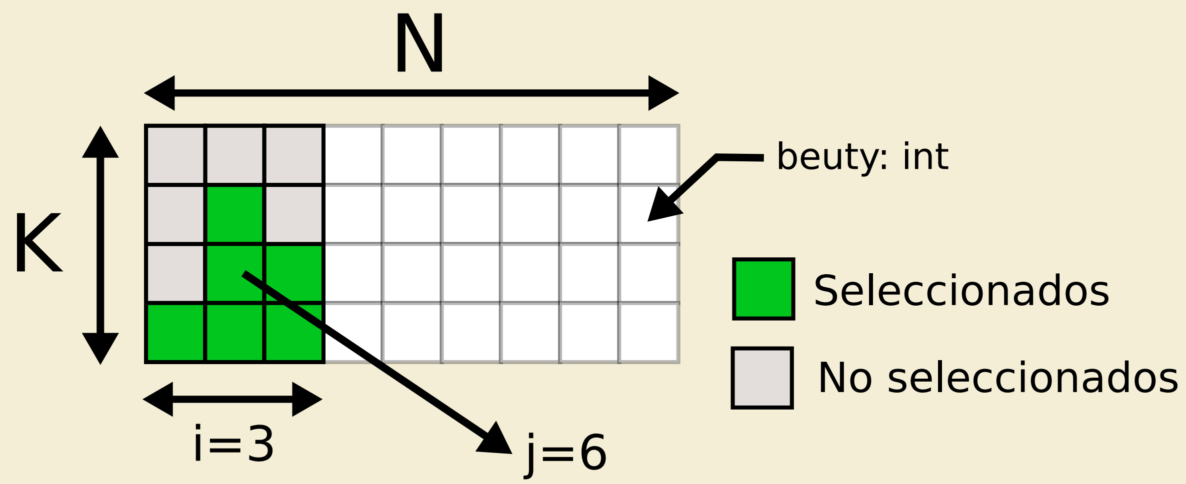
primeros i stacks

uso exactamente j platos

Detalles solo para gente que pregunta mucho

i=1 se usan los primeros j platos del primer stack
i=N se usan todos los stack
j=0 no se usan platos (beuty=0)

Ejemplo:



¿Por qué sirve?

Con este planteo posible de armar soluciones de i mayor usando soluciones con i menor.

¿y por qué es necesario j?

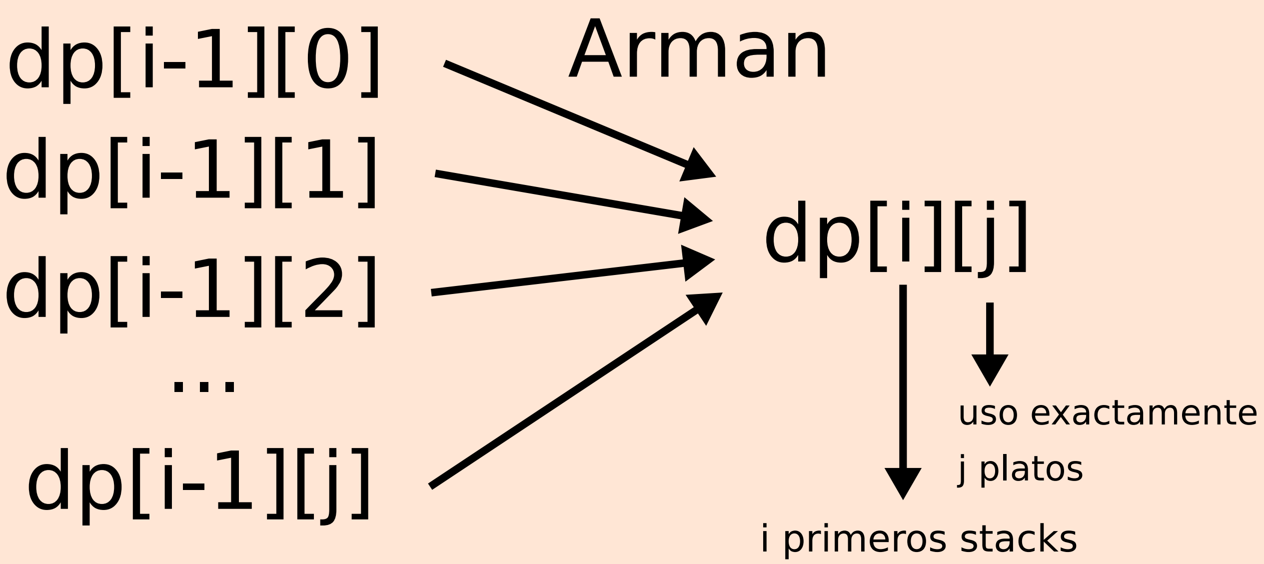
Sin j no es posible armar soluciones de i mayor usando soluciones de i menor

¿Más precisamente?

dp[i][j] para cualquier 1<=i<=N y cualquier 0<=j<=P puede ser

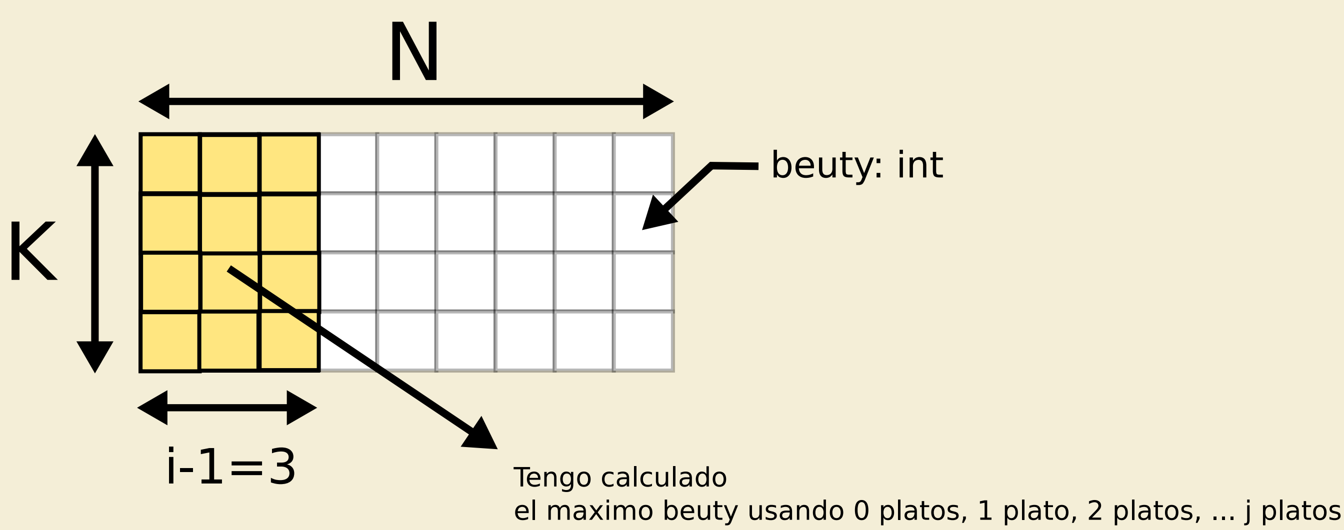
calculado si se conoce dp[i-1][0], dp[i-1][1], dp[i-1][2] ... dp[i-1][j]

¿Qué?



¿Por que?

Supongamos que ya tengo calculado
los dp[i-1][0], dp[i-1][1] ... dp[i-1][j]



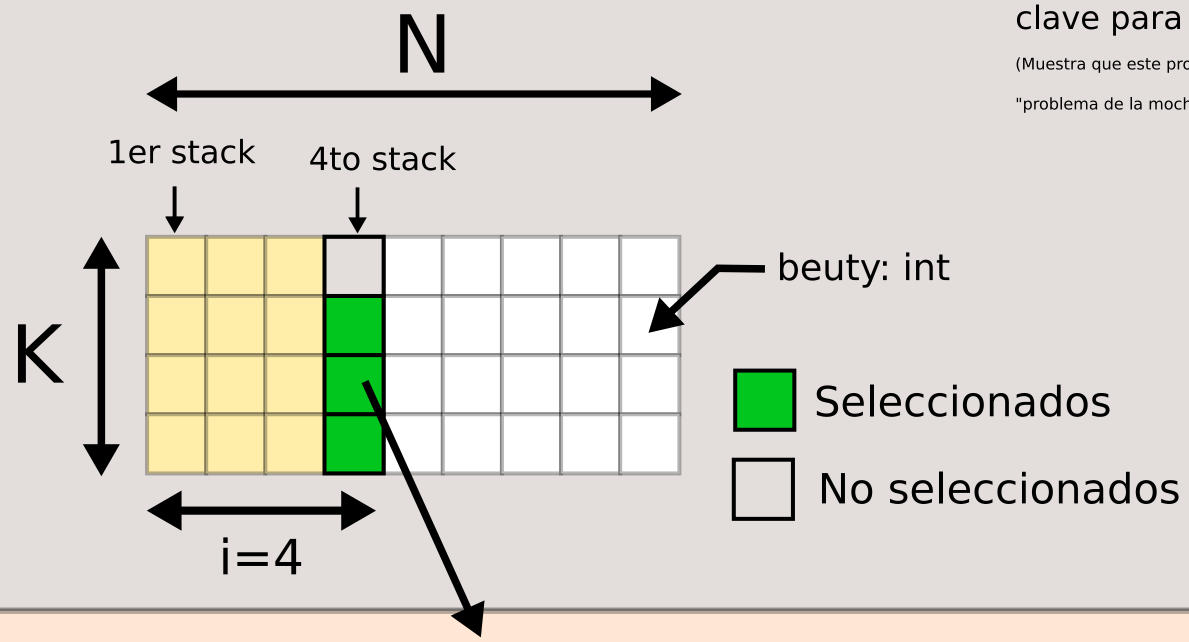
(i=4, i-1=3 para poder ilustrar más facil)

¿Comó cálculo entonces la solución con los i=4 primeros stacks y j platos?

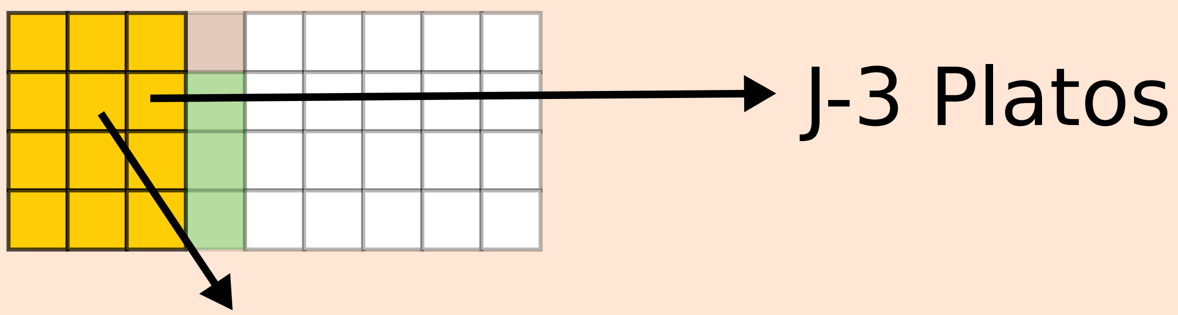
Observación: Considerar que en el cuarto stack sea cual sea la mejor elección voy a utilizar un número de platos que puede ir de 0 a K

Entender esta frase es clave para el problema.c

(Muestra que este problema es equivalente al "problema de la mochila")



Por ejemplo si utilizo 3 platos del 4to stack....



Lo razonable sería utilizar j-3 platos en los 3 stacks anteriores

Que significa, en total el beuty=dp[i-1][j-3]+beuty de los 3 platos agregados

¡Hagamos fuerza bruta con todos los posibles valores de platos a tomar del 4to stack!

Para cualquier alternativa, incluyendo la de mayor beuty, alguna cantidad de platos 0<=platos<=K voy a tomar

Obviamente de todas las variantes probadas, la de mayor

Escribiendo los cálculos...

beuty chance 0 = dp[i-1][j-0] + beuty 0 platos 4to stack
beuty chance 1 = dp[i-1][j-1] + beuty 1 plato 4to stack
beuty chance 2 = dp[i-1][j-2] + beuty 2 platos 4to stack
beuty chance 3 = dp[i-1][j-3] + beuty 3 platos 4to stack
beuty chance 4 = dp[i-1][j-4] + beuty 4 platos 4to stack

dp[i][j]=máx(beuty chance 0,
beuty chance 1,
beuty chance 2,
beuty chance 3,
beuty chance 4)

En forma génerica ...

dp[i][j]=máx(dp[i-1][j] + beuty 0 platos stack i,
dp[i-1][j-1] + beuty 1 plato stack i ,
dp[i-1][j-2] + beuty 2 platos stack i,
...
dp[i-1][j-K] + beuty K platos stack i)

Conclusión:

La solución tiene 3 for (uno adentro del otro)

Uno para i

Uno para j

Otro para cuantos platos uso en el stack número i

Máx beuty global=dp[N][K]

Ariel Nowik
anowik@itba.edu.ar