

## Chocolate bar

1. Identify if this problem can be solved with dynamic programming and/or a greedy Algorithm

I solved the problem with DP


(a) If you say that can be solved with dynamic programming

i. Analyze the problem based on sub-problems.



ii. Identify where the overlap occurs.

El overlap ocurre cuando la porción de chocolate consumida en el paso anterior se superpone con la porción de chocolate disponible en el paso actual.

2 usages  Samuel Escalera

```
private static boolean isOverlap(int[][] dp, int i) {  
    return i - dp[i - 1][0] > dp[i - 1][1];  
}
```

Si la diferencia entre la cantidad de chocolate consumido en el paso anterior (**dp[i - 1][0]**) y la cantidad acumulada de chocolate consumido hasta el momento (**dp[i - 1][1]**) es mayor que la cantidad de chocolate disponible en el paso actual (**i**), entonces hay superposición y el algoritmo incrementa el contador **cnt**.

iii. Implement the code

The code has implemented in ChocolateBar.java

iv. What is the time complexity of your solution?

1. **initializeDp()**: Asigna valores iniciales a la matriz **dp**. Esto es una operación de complejidad  $O(1)$ .
2. **isOverlap**: Accede a elementos de la matriz **dp** =  $O(1)$ .
3. **updateDp()**: Accede y actualiza elementos de la matriz **dp** =  $O(1)$ .
4. **fillDp()**: Itera desde 2 hasta **n**, realizando operaciones  $O(1)$  en cada iteración. Por lo tanto, su complejidad es  $O(n)$ .

El tiempo de ejecución del algoritmo es  $O(n)$ , donde  $n$  es el tamaño de la barra de chocolate.