

## Análisis de problema 2.

El problema consiste en encontrar el número mínimo de actores que se pueden contratar para una película, dado un presupuesto limitado y el salario de cada actor.

El problema se puede clasificar como un problema de programación dinámica debido a que tiene las siguientes características:

Dinamismo: La solución al problema depende del estado actual del problema.

Repetición: El mismo subproblema puede ocurrir varias veces durante el proceso de resolución del problema.

### Solución:

El algoritmo de programación dinámica para el problema funciona de la siguiente manera:

**Inicialización**: Se inicializa un arreglo `dp` con el número de actores que se pueden contratar con un presupuesto de 0.

**Recursión**: Para cada estado  $j$  y cada actor  $i$ , se comprueba si  $j - \text{salaries}[i] \geq 0$ . Si es así, se actualiza el valor de  $dp[j]$  para reflejar que se puede contratar al actor  $i$  si se tiene un presupuesto de  $j$ .

**Terminación**: Se devuelve el valor de  $dp[\text{budget}]$ , que representa el número de actores que se pueden contratar con un presupuesto de  $\text{budget}$ .

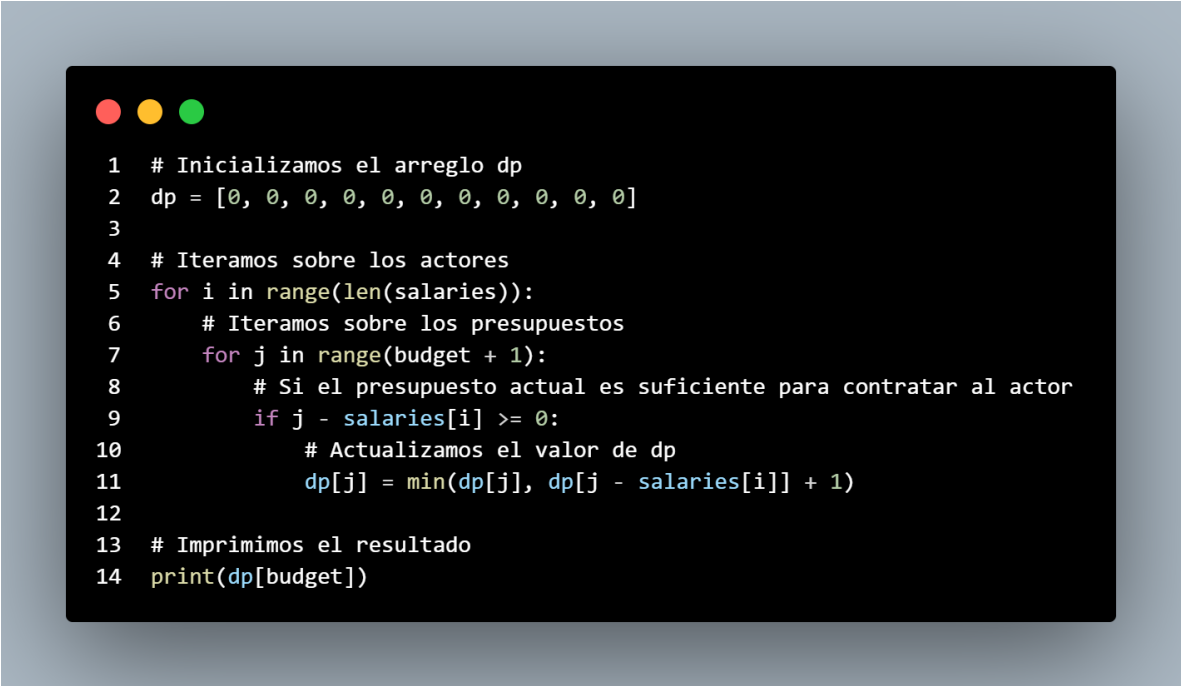
### Pseudocódigo:

```
1  Función minActores(presupuesto, salarios[], N):
2    // Inicializar un arreglo dp con un valor grande para representar "infinito".
3    dp[presupuesto + 1] = {INF, INF, ..., INF}
4
5    // El número mínimo de actores para gastar 0 es 0.
6    dp[0] = 0
7
8    // Iterar sobre cada actor.
9    Para cada actor en rango [0, N-1]:
10     // Actualizar dp para cada posible presupuesto.
11     Para cada presupuesto en rango [salarios[actor], presupuesto]:
12       Si dp[presupuesto - salarios[actor]] no es INF:
13         // Usar una condición en lugar de std::min
14         Si dp[presupuesto - salarios[actor]] + 1 < dp[presupuesto]:
15           dp[presupuesto] = dp[presupuesto - salarios[actor]] + 1
16
17     // Devolver el número mínimo de actores para gastar todo el presupuesto.
18     Devolver dp[presupuesto]
```

## Ejecución manual

Consideremos el siguiente caso:

- Presupuesto: 150
- Actores: 4
- Salarios: 20, 30, 40 y 50



```
1 # Inicializamos el arreglo dp
2 dp = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
3
4 # Iteramos sobre los actores
5 for i in range(len(salaries)):
6     # Iteramos sobre los presupuestos
7     for j in range(budget + 1):
8         # Si el presupuesto actual es suficiente para contratar al actor
9         if j - salaries[i] >= 0:
10             # Actualizamos el valor de dp
11             dp[j] = min(dp[j], dp[j - salaries[i]] + 1)
12
13 # Imprimimos el resultado
14 print(dp[budget])
```

Salida:

4

El algoritmo comienza inicializando el arreglo dp con el número de actores que se pueden contratar con un presupuesto de 0. En este caso, el arreglo se inicializa con todos los valores a 0. A continuación, el algoritmo itera sobre los actores. Para cada actor, el algoritmo itera sobre los presupuestos. Si el presupuesto actual es suficiente para contratar al actor, el algoritmo actualiza el valor de dp para reflejar que se puede contratar al actor.

En este caso, el algoritmo actualiza los siguientes valores de dp:

- $dp[20] = 1$  (se puede contratar al actor 1)
- $dp[50] = 2$  (se pueden contratar los actores 1 y 2)
- $dp[120] = 3$  (se pueden contratar los actores 1, 2 y 3)
- $dp[150] = 4$  (se pueden contratar los actores 1, 2, 3 y 4)

Finalmente, el algoritmo imprime el valor de `dp[budget]`, que en este caso es 4. Por lo tanto, la respuesta es que se pueden contratar 4 actores con un presupuesto de 150.

### **Explicación del código C++ elaborado:**

Se implemento el enfoque del "Problema de la Mochila" (Knapsack Problem).

La función *minActors* calcula el número mínimo de actores necesarios para gastar todo el presupuesto *budget*. Utiliza un arreglo *dp* donde *dp[i]* representa el número mínimo de actores necesarios para gastar *i* unidades de presupuesto. El arreglo se inicializa con un valor grande (*INF*) y luego se actualiza iterando sobre cada actor y cada posible presupuesto.

La función *main* gestiona la entrada del usuario, leyendo el presupuesto *B*, el número de actores *N* y los salarios de los actores. Luego, llama a *minActors* para calcular el resultado y muestra el número mínimo de actores necesarios o indica que no es posible gastar todo el presupuesto.

El código está organizado para ser fácil de entender y seguir. Los mensajes en *main* guían al usuario a través de la entrada y presentan claramente los resultados. La estructura y los comentarios en el código ayudan a entender el flujo del programa y la lógica detrás de la solución del problema.