Propuesta de Proyecto: Optimización del Estudio Académico mediante Programación Dinámica

Durante la vida universitaria, es común enfrentar semanas cargadas de exámenes en distintas materias, lo que genera un serio reto al momento de organizar el tiempo de estudio. Muchas veces, la distribución del tiempo se decide por intuición: priorizar la materia más difícil, la más cercana o la que genera más estrés. Sin embargo, este enfoque puede resultar ineficiente y no siempre lleva a un mejor rendimiento.

Este proyecto propone una solución basada en programación dinámica (PD) para distribuir de forma óptima las horas de estudio, considerando variables reales como la cantidad de horas disponibles, la importancia de cada materia, el dominio previo del estudiante, y el tiempo mínimo requerido para que el estudio tenga un efecto real. En lugar de planear al azar, la idea es usar un algoritmo que evalúe múltiples combinaciones y escoja la mejor.

La herramienta se inspira en el problema clásico de la mochila 0/1, donde se busca maximizar el valor de los elementos dentro de una capacidad limitada. Aquí, los “objetos” son las materias, el “peso” es el número de horas necesarias, y el “valor” es el beneficio académico. El objetivo es llenar el tiempo disponible con las materias que mayor rendimiento generen.

Un enfoque voraz (greedy) no funciona bien en este contexto porque toma decisiones inmediatas sin considerar el panorama completo. Por ejemplo, elegir solo la materia con mejor valor/hora puede descartar combinaciones de materias que, juntas, generen más beneficios. La PD, en cambio, permite comparar todas las posibilidades y construir soluciones óptimas acumulando decisiones pasadas.

# Modelado del problema

Definición del estado:  
dp[i][j] = mayor ganancia total al considerar las primeras i materias con j horas disponibles.

Parámetros:  
- n: número de materias  
- H: horas totales disponibles  
- h[i]: horas mínimas requeridas para la materia i  
- v[i]: valor académico (beneficio) de estudiar i

Función de recurrencia:  
Si j ≥ h[i]: dp[i][j] = max(dp[i-1][j], dp[i-1][j - h[i]] + v[i])  
Si j < h[i]: dp[i][j] = dp[i-1][j]

Casos base: dp[0][j] = 0 para todo j, y dp[i][0] = 0 para todo i.

Memorización: Tabla (n+1) x (H+1), donde se guarda el valor óptimo en cada paso.

# Ejemplo numérico

Horas disponibles: 5  
Materias:  
- Cálculo (3h, valor 8)  
- Álgebra (2h, valor 5)  
- Historia (1h, valor 3)

Tabla dp[i][j]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Materias / Horas | 0 | 1 | 2 | 3 | 4 | 5 |
| Ninguna | 0 | 0 | 0 | 0 | 0 | 0 |
| Cálculo | 0 | 0 | 0 | 8 | 8 | 8 |
| Álgebra | 0 | 0 | 5 | 8 | 8 | 13 |
| Historia | 0 | 3 | 5 | 8 | 11 | 13 |

Explicación:  
La mejor combinación es Cálculo + Álgebra = 13 puntos. Historia no aporta más valor en ese caso.

# Conclusión

Este proyecto busca aplicar programación dinámica a un problema real del entorno estudiantil. Su implementación no solo ayuda a optimizar resultados académicos, sino también a reducir la ansiedad asociada a la planificación del estudio. La herramienta resultante podría evolucionar en una app o sistema automatizado que recomiende horarios personalizados.

# Referencias

- https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/  
- https://www.programiz.com/dsa/dynamic-programming  
- Kleinberg, J., & Tardos, É. (2006). \*Algorithm Design\*. Pearson.  
- Erickson, J. http://jeffe.cs.illinois.edu/teaching/algorithms/