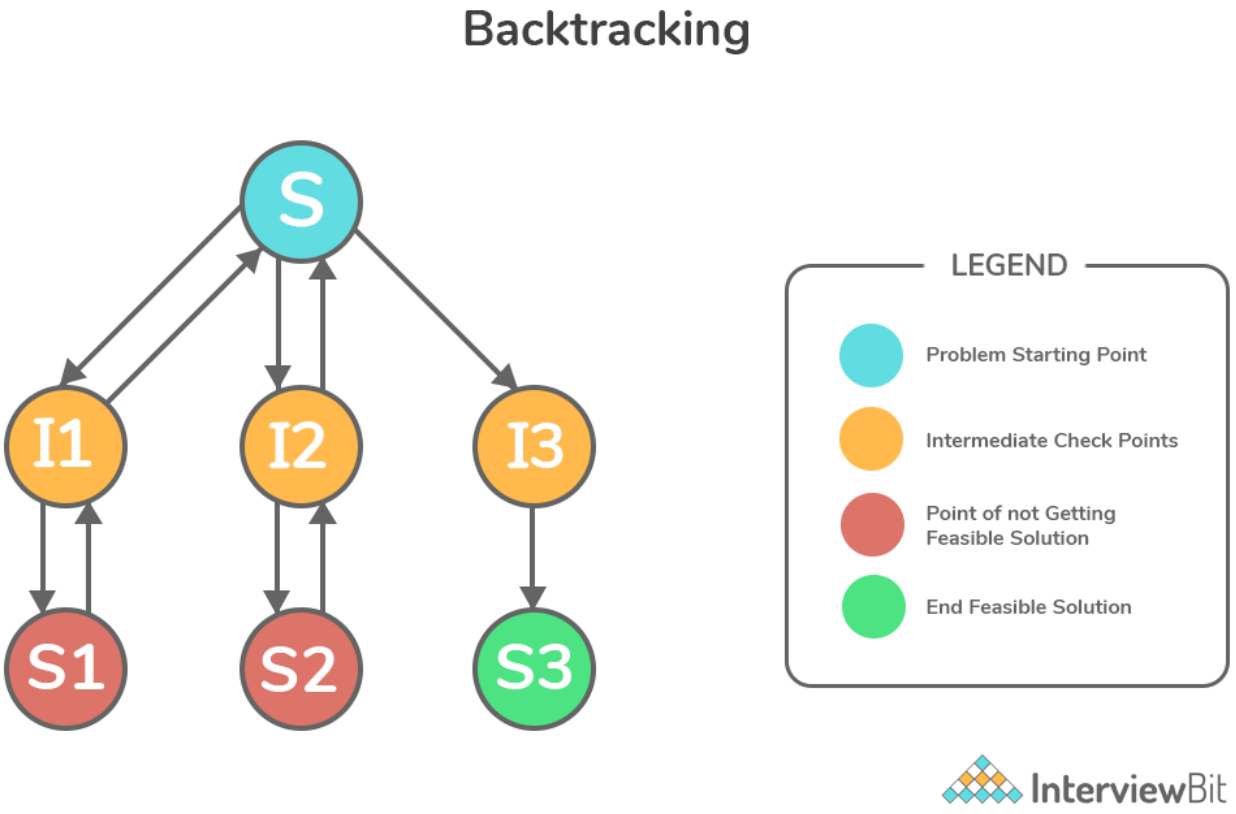
MEMORIA CASO 4 METODOLOGÍA



Integrantes:
Paulino Bermúdez
Andres Díaz
Kevin Gómez
Germán Pajarero

ÍNDICE

Análisis De Complejidad De Nuestro Algoritmo:

El algoritmo que hemos utilizado es una implementación recursiva de backtracking para buscar la solución óptima a un problema de selección de becas.
Para hallar la complejidad del algoritmo, analizamos las operaciones que se realizan en cada llamada recursiva y cuántas llamadas recursivas se realizan.

En cada llamada, se itera sobre todas las becas disponibles para averiguar si son válidas y no solapan con becas ya asignadas en nuestra solución parcial.

Cada beca válida se agrega a la solución parcial, se llama recursivamente con la solución actualizada y se remueve la beca de la solución parcial.
Operaciones que se realizan en cada llamada:

- Iterar sobre todas las becas disponibles (**O(n)**).
- Verificar si la beca es válida y no se solapa con las becas ya asignadas (**O(n)**).
- Agregar la beca a la solución parcial (**O(1)**).
- Realizar una llamada recursiva con la solución actualizada (**O(1)**)
  - En el peor de los casos (sin solución), la complejidad sería **O(1)** porque solo se realizaría llamada a resolver().
  - En el mejor de los casos (la primera solución es óptima), la complejidad sería **O(n)**, recorre la lista de becas una vez (1 vez * n becas = n).
- Remover la beca de la solución parcial (**O(1)**).

** n es el número de becas*

La complejidad del algoritmo depende del número de llamadas recursivas.

- En el peor caso, se realizará una llamada por cada combinación posible de becas, lo cual sería **$O(2^n)$**, ya que cada beca puede estar o no en la solución final.
- Sin embargo, gracias a las podas que realizamos al comprobar si la solución actual tiene mejor salario y si las fechas se solapan, la cantidad de llamadas se reduce.

En resumen, la complejidad del algoritmo es **$O(2^n)$** en el peor caso.

Contraejemplo Utilizando Fuerza Bruta:

La complejidad del algoritmo usando backtracking es menor que con fuerza bruta ya que permite reducir el número de soluciones a evaluar para encontrar la óptima.

Demostración:

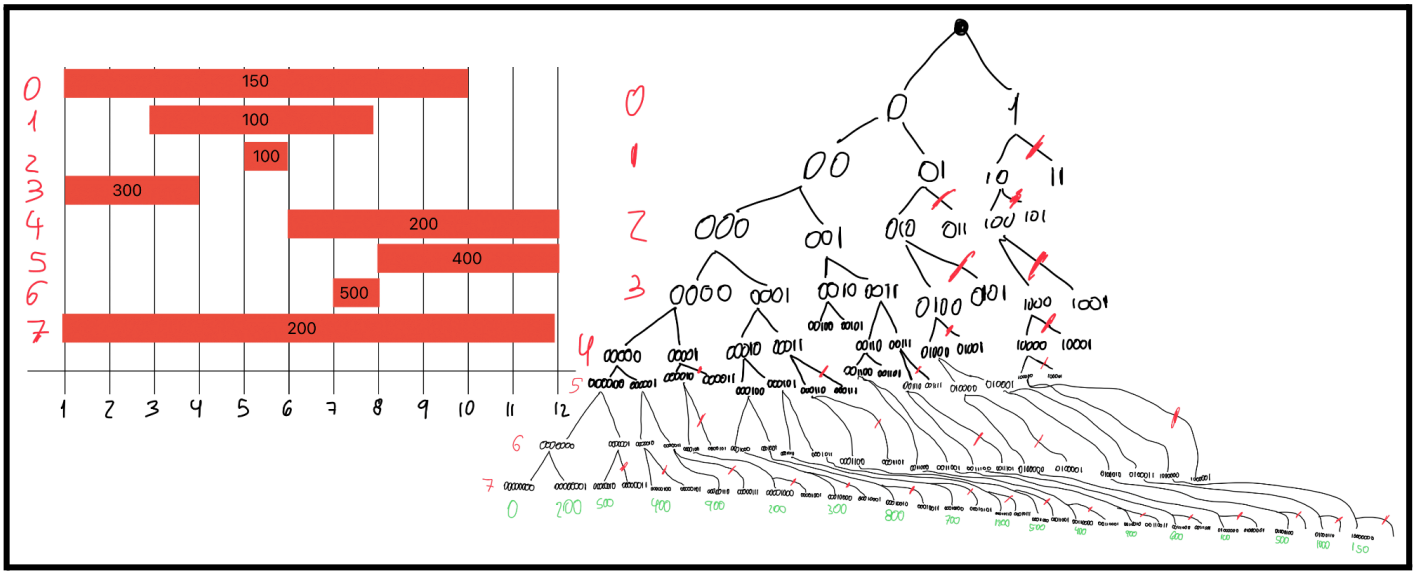
Supongamos que tenemos 5 becas.
Si utilizamos fuerza bruta, debemos considerar todas las combinaciones posibles de becas para encontrar la solución óptima.

En este caso, la cantidad total de combinaciones sería 2^5, lo que equivale a 32 combinaciones posibles.

En el peor de los casos, habría que evaluar 32 soluciones distintas para encontrar la óptima.

Por otro lado, si utilizamos backtracking podemos reducir en gran medida el número de soluciones a evaluar.

En el peor de los casos, habría que evaluar todas las combinaciones posibles, pero gracias a la poda podemos descartar combinaciones no compatibles, lo que reduce el total de soluciones a evaluar.



Esquema realizado a la hora de realizar el proyecto, en el que se ve como backtracking "poda" soluciones no compatibles.