PRÁCTICA DE ESTRATEGIAS ALGORÍTMICAS

En esta práctica realizaréis la implementación en C de dos estrategias algorítmicas para la resolución del problema de asignación de tareas a personas resuelto en clase de teoría con las dos versiones (con y sin vector `usadas[]`).

Debéis probar con n=3 según el ejercicio resuelto en clase (con esa tabla de beneficios) y con un n mayor (n=6) según las tablas de beneficios siguientes:

Para n=3:

TAREAS

B	1	2	3
1	11	17	8
2	9	7	6
3	13	15	16

Para n=6:

TAREAS

PERSONAS	B	1	2	3	4	5	6
	1	11	17	8	16	20	14
	2	9	7	6	12	15	18
	3	13	15	16	12	16	18
	4	21	24	28	17	26	20
	5	10	14	12	11	15	13
	6	12	20	19	13	22	17

Primera parte: implementación de la estrategia de vuelta atrás (backtracking)

Será necesario implementar el algoritmo de vuelta atrás visto en teoría con y sin el vector `usadas`. Podéis utilizar el pseudocódigo del ejercicio de asignación resuelto en clase para implementar las distintas funciones.

El programa debe imprimir el número de nodos generados por el algoritmo en cada caso (los que cumplen el criterio).

Debéis determinar el orden de complejidad del algoritmo. Analizar las funciones más costosas, comprobando e imprimiendo el número de pasos. Comprobar si se reduce (y cuánto) la complejidad del algoritmo y el número de pasos al utilizar el vector `usadas`.

Debes entregar el programa que resuelve el problema y además un informe explicando:

1. Qué representan las componentes del vector `s`.
2. Qué función realiza la asignación de una tarea a una persona.

3. Qué comprueba la función `MasHermanos()`.
4. Cuál es el objetivo de la función `Retroceder()`.
5. Las conclusiones (teóricas y prácticas) en cuanto a la complejidad de la tarea y el número de pasos cuando se emplea el vector `usadas` y cuando no se emplea.

Para sacar vuestras conclusiones, vuestro programa debe calcular e imprimir los siguientes datos para cada caso:

n	Vector `usadas`	Nº nodos visitados	Nº pasos `Criterio`	Nº pasos `Generar`	Nº pasos `Solución`	Nº pasos `MasHermanos`	Nº pasos `Retroceder`
3	No						
	Si						
6	No						
	Si						

Segunda parte: implementación de la estrategia de ramificación y poda

El objetivo de esta segunda parte es comprobar el efecto de la poda en el número de nodos que se visitan en el árbol implícito de exploración de soluciones.

Especificaciones:

- La estrategia de ramificación será MB-LIFO.
- Deben probarse las dos estrategias para la estimación de las cotas: estimaciones triviales y estimaciones precisas, y comparar la diferencia en la efectividad de la poda (imprimiendoel número de nodos explorados en cada uno de los casos).

Una vez resuelto el ejercicio, debéis indicar los cambios que habría que hacer para convertir el problema en uno de minimización (si la matriz B representase costes en lugar de beneficios). No es necesario implementar la solución para este caso.
El programa debe imprimir el número de nodos generados por el algoritmo en cada caso (los que cumplen el criterio).

Debes entregar el programa que resuelve el problema y además un informe explicando:

1. Conclusiones sobre los efectos de la poda: relación entre la reducción de nodos y la complejidad del cálculo y comparación del número de nodos generados con la práctica de *vuelta atrás*.
2. Explicación de cambios para el caso de minimización: cambios en condición de poda y cálculo de C, cambios en estrategia de ramificación y valores iniciales, cambios en las estimaciones de las cotas inferior y superior.

Para sacar vuestras conclusiones, vuestro programa debe calcular e imprimir los siguientes datos para cada caso, usando siempre el vector `usadas`:

N	Nº de nodos explorados
3	
6	