PRÁCTICA DE ESTRATEGIAS ALGORÍTMICAS

En esta práctica veremos otro aspecto de nuestra temática de este curso: Juego de Tronos. Harren el Negro gobierna la Isla de Hierro, cuyos habitantes viven principalmente del pirateo. Harren debe encargar a cada uno de sus galeones el ataque a una de las ciudades de Westeros, pero algunas de las tripulaciones son más efectivas que otras y además el botín que se puede obtener en cada ciudad es distinto. Por tanto, les interesa optimizar los beneficios que se obtiene en cada día de ataques. Los nombres de las ciudades y galeones que corresponden a cada índice son los siguientes:

	Ciudades	**Galeones**
1	King's Landing	Iron Victory
2	Lannisport	Grief
3	Oldtown	Lord Quellon
4	White Harbor	Lamentation
5	Seagard	Kite
6	Storm's End	Dagger

En esta práctica realizaréis la implementación en C de dos estrategias algorítmicas para la resolución del problema de asignación de "ataques a ciudades" a "galeones" similar al 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:

GALEONES \ CIUDADES	1	2	3
1	11	17	8
2	9	7	6
3	13	15	16

Para n=6:

GALEONES \ CIUDADES	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 (Ejercicio vuelta atrás (asignación) resuelto, diapositiva 13) sin vector **usadas** y modificaciones en diapositivas 15-17 para utilizar el vector **usadas**). De hecho, podéis probar con la misma matriz para comprobar que el árbol generado y la solución sean correctos.

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 atacar una ciudad a un galeón.
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 asignación 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. Inicialmente, podéis probar vuestro programa con el ejercicio resuelto en clase de teoría (Ejercicio ramificación y poda resuelto) para comprobar que vais generando bien los nodos y sus cotas, que sigue el esquema algorítmico de la diapositiva 80 de teoría (MB-LIFO).

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 (imprimiendo el 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.

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	

Ejemplo de salida del algoritmo de ramificación y poda: Podéis probar vuestro programa con el ejemplo visto en teoría: En este caso se ha escrito en pantalla mucha información para poder seguir el flujo del programa, pero en vuestro caso no es necesario que escribáis toda esta información si el número de nodos es correcto.

```
---------- RAMIFICACION Y PODA ----------

---------- RESULTADOS CON ESTIMACIONES TRIVIALES ----------
Nodos explorados con estimaciones triviales:
        Nodo 1: CI=0    BE=13.5 CS=27    -->entra en LNV
Ramifico nodo 1
        Nodo 2: CI=4    BE=13.0 CS=22    -->entra en LNV
        Nodo 3: CI=9    BE=18.0 CS=27    -->entra en LNV
        Nodo 4: CI=1    BE=10.0 CS=19    -->entra en LNV
Ramifico nodo 3
        Nodo 5: CI=16   BE=20.5 CS=25    -->entra en LNV
        Nodo 6: CI=12   BE=16.5 CS=21    -->entra en LNV
Ramifico nodo 5
        Nodo 7: CI=21   BE=21.0 CS=21    -->es solución=21
Podo nodo 6
Ramifico nodo 2
        Nodo 8: CI=6    BE=10.5 CS=15   -->no entra en LNV
        Nodo 9: CI=7    BE=11.5 CS=16   -->no entra en LNV
Podo nodo 4

LNV VACÍA

Tareas asignadas:
        Persona 0 --> Tarea 1
        Persona 1 --> Tarea 0
        Persona 2 --> Tarea 2
Solución óptima=21
Número de nodos explorados: 9

---------- RESULTADOS CON ESTIMACIONES PRECISAS ----------
Nodos explorados con estimaciones precisas:
        Nodo   1:       CI=21   BE=21.5 CS=22    -->entra en LNV
Ramifico nodo 1
        Nodo   2:       CI=  10 BE=  11 CS=  12  -->NO entra en LNV
        Nodo   3:       CI=  21 BE=  22 CS=  22  -->entra en LNV
        Nodo   4:       CI=  11 BE=  12 CS=  14  -->NO entra en LNV
Ramifico nodo 3
        Nodo   5:       CI=  21 BE=  21 CS=  21 -->Solución voraz
        Asignación voraz: Nodo 6: tarea: 2       -->es solución=21
        Nodo   7:       CI=  18 BE=  18 CS=  18  -->NO entra en LNV

LNV VACÍA

Tareas asignadas:
        Persona 0 --> Tarea 1
        Persona 1 --> Tarea 0
        Persona 2 --> Tarea 2
Solución óptima=21
Número de nodos explorados: 7
```