1.– Sean los conjuntos C_1 = (*Burbuja*, *Inserción Directa*, *Selección Directa*,), C_2 = (*Montículo*, método de *Shell*) y C_3 = (*Quicksort*, *Mergesort*). Calcule la complejidad tanto de forma analítica como experimental de UN MÉTODO de cada uno de los tres conjuntos anteriores. Los conjuntos de datos sobre los que se trabajará serán de 10.000, 100.000 y 1.000.000 elementos enteros. Recopile datos sobre distintas máquinas.

2.– Se consideran un vector *v* de *N* elementos y dos enteros positivos *k* y *p*. Diseñe un algoritmo que **devuelva los *p* elementos de *v* más próximos al k-ésimo menor elemento de *v* luego de ordenarlo**. La complejidad del algoritmo desarrollado debe ser lineal en el peor de los casos. La complejidad no es necesario que se calcule analíticamente, pero debe justificarse.

3.– **La granja de Pepito**. Una comarca tiene N_G de parcelas para pastar el ganado, N_C parcelas de cultivo y N_A parcelas de arboleda. Al final de la temporada cada una de ellas aporta un beneficio B_G, B_C y B_A, respectivamente. Así, si un granjero tiene *G* parcelas de pastos, *C* parcelas para cultivar y *A* parcelas de arboleda, el beneficio total al final de la temporada viene dado por:

$$Beneficio = G*B_G + C*B_C + A*B_A$$

Al comienzo de cada temporada, el granjero puede decidir comprar una (y sólo una) nueva zona con sus ahorros, teniendo cada tipo de zona un precio diferente: P_G, P_C y P_A.

Además, tenemos que:

a) cada parcela de ganado nueva supone inutilizar media parcela cultivable para dar de comer a los animales,

b) cada parcela cultivable adicional supone inutilizar media de arboleda, y

c) cada parcela de arboleda supone inutilizar media parcela de ganado para su plantación.

Dado un presupuesto inicial *P*, unas zonas iniciales de cada tipo G_0, C_0 y A_0, se trata de maximizar el beneficio que podemos obtener al cabo de *N* temporadas. Resuelva el problema mediante un ***algoritmo voraz***.

4.– Un barco de transporte de mercancías, con una capacidad máxima de carga *C*, ha sido adquirido por una empresa que quiere maximizar los beneficios obtenidos con su utilización. El barco transporta mercancías entre una lista de *P* puertos. En cada puerto se pueden comprar y vender *M* productos diferentes. El precio de producto varía de un puerto a otro. En cada puerto lo que hace es vender el producto que lleva de carga (vende toda su carga) y carga otro producto de ese puerto (la cantidad a cargar dependerá de la capacidad del barco, y de la disponibilidad de capital). Conocemos los días que tarda el barco en ir desde cada puerto a todos los demás (se incluye el tiempo de carga y descarga). El capitán dispone de un capital inicial *D* para comprar mercancías.

El problema consiste en encontrar la ruta que maximiza los beneficios en un tiempo *T* (en días), teniendo en cuenta que la ruta debe comenzar y terminar en el mismo puerto, y además, no se puede repetir ningún puerto. Resuelva el problema utilizando ***Programación Dinámica***.

5.– **Ascensores Inteligentes**. Un edificio de *P* plantas dispone de dos ascensores, cada uno con una capacidad de *C* personas, que tardan un tiempo de *t* segundos en desplazarse de una planta a la siguiente. Se ha detectado que la hora punta es entre las 11:00 y las 11:05 habiendo un tránsito masivo de personas que se desplazan a través de las distintas plantas. Cada persona que desea hacer un trayecto, indica la hora a la que desea coger un ascensor, la planta origen y la planta destino.

Supuesto que tenemos la lista de *N* trayectos solicitados, correspondientes a otras tantas personas (esto es, una persona solo solicita un trayecto), se trata de planificar el funcionamiento de los ascensores para que

se minimice el tiempo de espera para entrar en el ascensor del conjunto de las personas. Diseñe e implemente un algoritmo basado en ***Backtracking*** que resuelva el problema.

Resuelva el problema si se trata de minimizar el tiempo de servir las *N* peticiones, sin que una persona sobrepase un cierto tiempo de espera *TEsperaMaximo* antes de subir al ascensor, ni esté en el ascensor más de un tiempo *TDentroMaximo*.

Nota: Despréciese el tiempo de apertura y cerrado de puertas, y el tiempo empleado en entrar y salir del ascensor.

6.– **Juego**. Dos jugadores juegan al siguiente juego. Cada jugador tiene un cartón formado por huecos y operadores (suma, resta, multiplicación y división) de forma alternativa. El cartón tiene un número par de huecos *H* y *H-1* operadores. Cada jugador puede ir rellenando los huecos del cartón, de izquierda a derecha, con diferentes números comprendidos entre *1* y *H/2*. Además cada jugador no puede jugar dos veces el mismo número. Cuando un jugador coloca un número N comprendido entre *1* y *H/2*, al rival se le coloca de forma automática el valor *(H/2 + 1) – N* en su mismo hueco del cartón.

Gana el juego aquel que tras evaluar el cartón, el resultado de las operaciones (sin tener en cuenta la precedencia) se queda más cercano de un valor V dado.

Diseñe una función que dada una posición del juego la evalúe, y devuelva la decisión a tomar.

Ejemplo:

Cartón:

	+		*		-		*		+		-		/		*		+	

H = 10, V = 25
Valores posibles que se pueden jugar [1-5]
Comienza el jugador 1, y gana el jugador que se queda más cerca de *V = 25*.

El desarrollo del juego podría ser:
J1 -> 3, J2 -> 5, J1 -> 5, J2 -> 1, J1 -> 1, J2 -> 3, J1 -> 2, J2 -> 4, J1 -> 4, J2 -> 4.

3	+	1	*	5	-	5	*	1	+	3	-	2	/	4	*	4	+	2	=18
3	+	5	*	1	-	1	*	5	+	3	-	4	/	2	*	2	+	4	=38

Jugador 1 -> Resultado = 18 -> |25-18| = 7 --------> GANADOR
Jugador 2 -> Resultado = 38 -> |25-38| = 13

Otro ejemplo:

5	+	3	*	1	-	4	*	4	+	2	-	2	/	5	*	3	+	1	=10,6
1	+	3	*	5	-	2	*	2	+	4	-	4	/	5	*	3	+	1	=22,6

Jugador 1 -> Resultado = 10,6 -> |25-10,6| = 14,4
Jugador 2 -> Resultado = 22,6 -> |25-22,6| = 2,4 --------> GANADOR

7.– **Práctica Optativa**. Resuelva el problema número cuatro mediante un algoritmo basado en **Backtracking**.