1.– Calcule la complejidad tanto de forma analítica como experimental de los siguientes métodos de ordenación: *Burbuja*, *Inserción Directa*, *Selección Directa*, *Montículo*, *Quicksort, Mergesort* y método de *Shell*. Los conjuntos de datos sobre los que se trabajará serán de 1000, 10000 y 100000 elementos enteros. Recopile datos sobre distintas máquinas.

2.– En una habitación oscura se tienen dos cajones en uno de los cuales hay *n* tornillos de varios tamaños, y en el otro las correspondientes *n* tuercas. Es necesario emparejar cada tornillo con su tuerca correspondiente, pero debido a la oscuridad no se pueden comparar tornillos con tornillos ni tuercas con tuercas, y la única comparación posible es la de intentar enroscar una tuerca en un tornillo para comprobar si es demasiado grande, demasiado pequeña, o se ajusta perfectamente al tornillo. Desarrolle un algoritmo **Divide y Vencerás** para emparejar los tornillos con las tuercas, que use O(*n log n*) comparaciones en término medio.

3.– Se tiene un conjunto de palabras que se quieren agrupar según su grado de semejanza. Desarrolle una **heurística voraz** de manera que las palabras queden agrupadas hasta un umbral dado δ, sabiendo que al unir dos grupos, la semejanza del nuevo grupo con todos los demás es el mínimo de los dos valores de semejanza.

La **semejanza** o proximidad de dos palabras $a = a_1a_2 \ldots a_m$ y $b = b_1b_2 \ldots b_n$ está dada por:

$$\frac{2Card(A \cap B)}{Card(A) + Card(B)}, \text{ siendo } A = \{ a_ia_{i+1} / 1 \le i \le m\text{-}1\} \text{ y } B = \{ b_ib_{i+1} / 1 \le i \le n\text{-}1\}$$

Nótese que *A* y *B* son conjuntos, y por tanto, no tienen elementos repetidos.

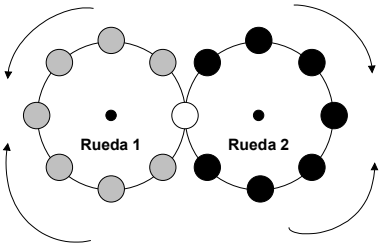
4.– El Tío Antonio tiene en su granja dos vacas: *Devoradora* y *Listilla*. Antes de ordeñarlas las alimenta llenando una hilera de *N* cubos con pienso (*N* es par). Cada cubo *i* contiene una cantidad p_i de pienso indicada en el cubo (todas las cantidades son distintas). Cada vaca en su turno debe elegir el cubo de uno de los extremos, y comerse su contenido. El cubo se retira y el turno pasa a la otra vaca. Se sigue así hasta agotar los cubos. La vaca que comienza comiendo se determina a priori por un procedimiento cualquiera. El objetivo de ambas vacas es comer en total lo máximo posible. La estrategia de *Devoradora* consiste en pensar poco y escoger el cubo de los extremos que esté más lleno. En cambio, *Listilla* prefiere pensárselo un poco más, para lo que ha adquirido lo último en computadoras vacunas portátiles.

a) Demuestre que la estrategia de *Devoradora* no es óptima, incluso cuando le toca elegir en primer lugar.

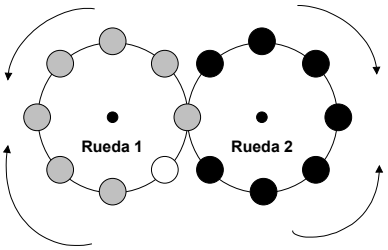
b) *Listilla* ha cursado un *master en Informática por la Escuela Superior Bovina*, pero en dicha escuela no estudian la **programación dinámica**, por lo que solicita su ayuda para diseñar un algoritmo utilizando dicha técnica, suponiendo que le toca empezar a escoger.

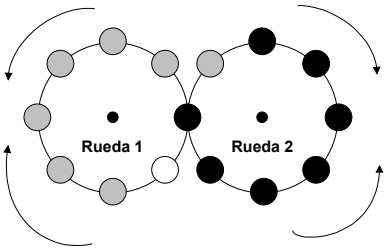
c) Diseñe un algoritmo de coste lineal de forma que *Listilla*, siempre que comience comiendo ella, coma al menos tanto como *Devoradora*, independientemente de la estrategia que siga esta última. ¿Es óptima la nueva estrategia?

5.– **Puzzle de las ruedas**. Diseñe un algoritmo de vuelta atrás que sea capaz de resolver el puzzle de las ruedas. Este puzzle se compone de dos ruedas giratorias y *N* bolas por cada rueda. De tal manera que al final del algoritmo queden en la siguiente configuración:



Para ver el funcionamiento, observe que tras una giro de la rueda 1 en sentido horario y otro de la rueda 2 en el mismo sentido las configuraciones serían:





6.– Tenemos un conjunto de *n* componentes electrónicas para colocar en *n* posiciones sobre una placa. Se nos dan dos matrices cuadradas *C* y *D*, de orden *n*, donde C_{ij} indica el número de conexiones necesarias entre la componente *i* y la componente *j*, y D_{pq} indica la distancia sobre la placa entre la posición *p* y la posición *q* (ambas matrices son simétricas y con diagonales nulas). Un cableado $(x_1,...,x_n)$ de la placa consiste en la colocación de cada componente *i* en una posición x_i. La longitud total de este cableado viene dado por la fórmula:

$$\sum_{i<j} C_{ij} D_{x_i x_j}$$

Diseñe un algoritmo **Backtracking** que devuelva el **cableado de longitud mínima**.

7.– **Juego**. La *Devoradora* del tío Antonio aprende rápido, y tras varios días de competir por la comida con la *Listilla*, ha cambiado de estrategia y ahora las dos se han convertido en jugadoras "*de la leche*", y buscan como único fin comer más que la otra, de hecho las dos puede que utilicen la misma estrategia. El juego consiste en que como antes cada una en su turno elige un cubo de uno de los extremos, se lo come y es retirado, ganando la que come más. Diseñe un algoritmo que para una posición del mismo evalúe el juego e indique la jugada a realizar.