

Optimización - 2013 - Práctica 1
Algoritmos Recursivos, Divide and Conquer

1. Implementar una función que dado un arreglo V de elementos devuelva un conjunto con todas las permutaciones posibles de V . Una posibilidad es, si V tuviese n elementos, tomar los primeros $n - 1$ elementos, obtener todas las permutaciones posibles de esos $n - 1$ elementos (acá se usaría la recursión) y después tratar de agregar a todas las permutaciones generadas el elemento que se omitió. Estudiar la complejidad.
2. Escribir un procedimiento que resuelva el problema de las Torres de Hanoi con n discos. Determinar la complejidad.
3. Escribir un procedimiento que resuelva la variante del problema de las Torres de Hanoi con n discos según la cual los movimientos de un disco entre las clavijas de los extremos están prohibidas, es decir, cada movimiento debe ser hacia o desde la clavija del medio. Determinar la complejidad.
4. El algoritmo Insertion Sort se puede implementar de manera recursiva de la siguiente forma: para ordenar $A[1..n]$, primero ordenamos (recursivamente) $A[1..n - 1]$ y luego insertamos $A[n]$ en el arreglo ordenado. Escribir el algoritmo y calcular la complejidad.
5. Para resolver un problema de tamaño n se consideran tres algoritmos, cada uno de ellos con las siguientes características:
 - A1: Divide el problema en 5 instancias de la mitad de tamaño, que se resuelven recursivamente. Posteriormente, en tiempo lineal se combinan las subsoluciones.
 - A2: Considera dos subproblemas de tamaño $n - 1$ que se resuelven recursivamente y se combinan las soluciones en tiempo constante.
 - A3: Divide el problema en 9 instancias de tamaño $n/3$, que se resuelven recursivamente. En tiempo cuadrático se combinan las subsoluciones.

¿Cuál es el mejor algoritmo en términos de complejidad temporal?
6. Se tiene un arreglo A de n elementos enteros, todos distintos entre sí y ordenados. Diseñar un algoritmo utilizando la técnica Divide and Conquer que determine si existe un índice i tal que $A[i] = i$, con una complejidad temporal de $O(\log_2(n))$.
7. Hay que organizar un torneo que involucre a n competidores. Cada competidor debe jugar exactamente una vez contra cada uno de sus oponentes. Además cada competidor debe jugar un partido por día. Suponga que n es potencia de 2, defina un algoritmo que utilice la técnica de Divide and Conquer para armar un fixture de $n - 1$ días.
8. Implementar en Python el algoritmo visto en la clase teórica para poder determinar, dado un conjunto de puntos en el plano euclídeo, la distancia mínima entre cualquier par de ellos.