Divide y Vencerás

Análisis y Diseño de Algoritmos

Dr. Jaime Osorio Ubaldo

Técnicas de diseño de algoritmos

El diseño de algoritmos es un área central en la Ciencia de la Computación, existen diversas técnicas o patrones de diseño de algoritmos que son muy útiles para la solución de diversos problemas, las principales son:

1. Divide y vencerás.
2. Programación dinámica.
3. Algoritmo Greedy (voraz, goloso, devorador o ávido).
4. Backtracking.
5. Ramificación y poda.

Divide y vencerás

Consiste en resolver el problema a partir de la solución de subproblemas del mismo tipo que el original. Si el problema x es de tamaño n y los tamaños de los subproblemas $x_1, x_2, \cdots, x_k$ son, respectivamente $n_1, n_2, \cdots, n_k$, el tiempo de ejecución $T(n)$ está definido por

$$T(n) = \begin{cases} g(n), & n \leq n_o \\ \sum_{j=1}^{k} T(n_j) + f(n), & n > n_o \end{cases}$$

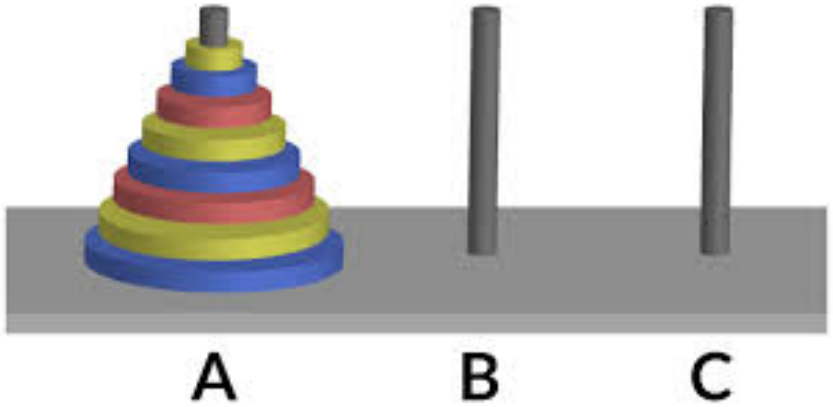
donde

1. n_o y $g(n)$ son, respectivamente es el tamaño y el costo temporal del caso base.
2. $f(n)$ es el tiempo de descomponer en subproblemas y combinar el resultado de estos.

Procedimiento

1. Plantear el problema de forma que pueda descomponerse en k subproblemas del mismo tipo, pero de menor tamaño y que no exista traslape entre ellos (si hay traslape se puede usar otra técnica llamada programación dinámica),
2. Resolver independientemente todos los subproblemas, ya sea directamente si son elementales o bien de forma recursiva.
3. Por último, combinar las soluciones obtenidas en el paso anterior para construir la solución del problema original.

Torres de Hanoi

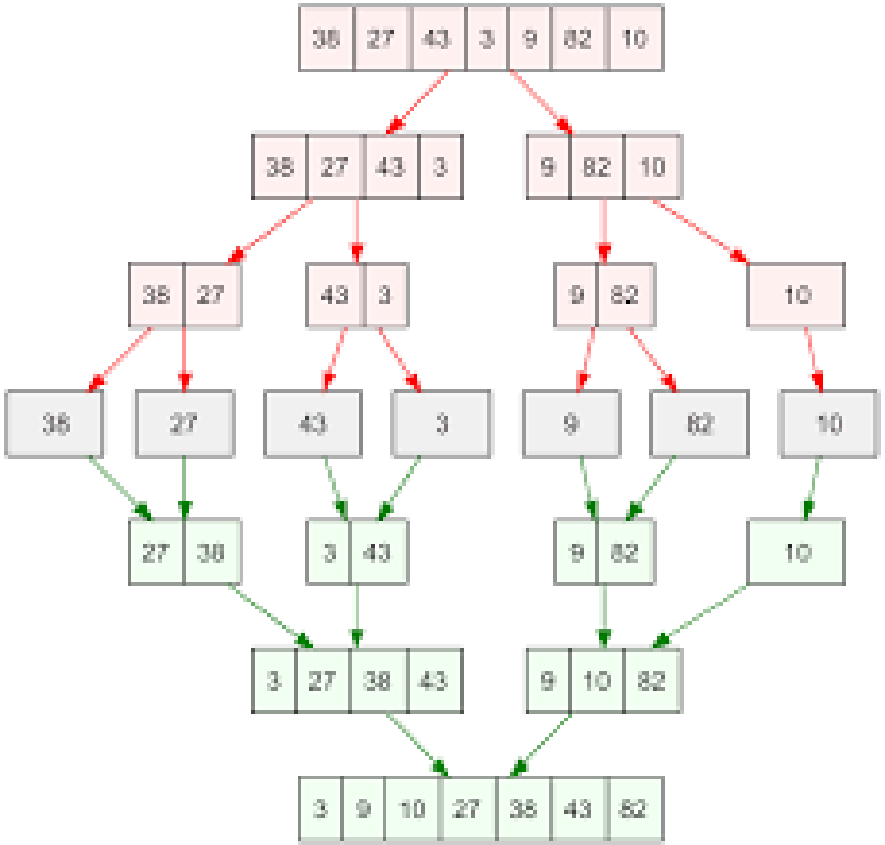


El tiempo de ejecución $T(n)$ para pasar n discos del poste A al poste B es la suma del tiempo $T(n-1)$ para pasar $n-1$ discos de A a C, más el tiempo $T(1)$ para pasar el disco n a B, más el tiempo $T(n-1)$ para pasar los $n-1$ discos de C a B. El caso base se presenta cuando solo hay que mover un disco, es decir

$$T(n) = \begin{cases} 1, & n = 1 \\ T(n-1) + 1 + T(n-1), & n > 1 \end{cases}$$

Mergesort (Ordenamiento por mezcla)

Su estrategia consiste en dividir un conjunto de datos en dos subconjuntos que sean de un tamaño tan similar como sea posible (a la mitad), dividir estas dos partes mediante llamadas recursivas, y finalmente, al llegar al caso base, mezclar los dos partes para obtener un arreglo ordenado.



Mergesort

Pasos

1. Dividir recursivamente la primera mitad del conjunto.
2. Dividir recursivamente la segunda mitad del conjunto.
3. Mezclar los dos subarreglos para ordenarlos.

Observación. Ignoramos los casos base.

Mergesort

Merge(A, B, C)
  $i = 1,\ j = 1$
  Para $k = 1 \cdots n$ *hacer*
    si $A[i] < B[j]$ *entonces*
      $C[k] = A[i]$
      $i = i + 1$
    sino
      $C[k] = B[j]$
      $j = j + 1$
    fin si
  Fin Para.

El costo temporal es $T_1 = 4n + 2$.

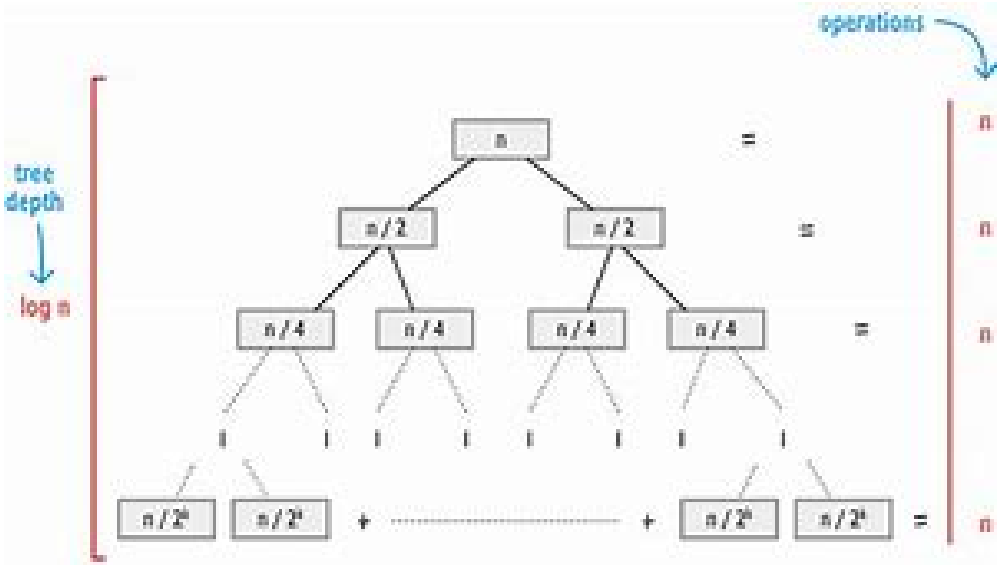
Mergesort

$MergeSort(A[1, \cdots, n])$
$si\ n = 1$
  $return\ A$
$m \approx n/2$
$B = MergeSort(A[1, \cdots, m])$
$C = MergeSort(A[m+1, \cdots, n])$
$Merge(B, C, A')$
$return\ A'$.

El costo temporal es

$$T(n) = 2T(\frac{n}{2}) + 4n + 2$$

Mergesort



De la figura

$$\frac{n}{2^k} = 1$$

luego

$$k = \log_2 n$$

Mergesort

$$
\begin{aligned}
T(n) &\leq 2T(n/2) + 6n \\
&= 2^2 T(n/4) + 2(6n) \\
&\vdots \\
&= 2^k T(1) + k(6n) \\
&= 2^{\log_2 n}(2) + 6n \log_2 n \\
&= 2n + 6n \log_2 n
\end{aligned}
$$

Por lo tanto, es de orden $\mathcal{O}(n \log_2 n)$.

Temas de investigación.

1. ¿Qué ventajas y desventajas presenta esta técnica de diseño?.
2. Explique una aplicación de esta técnica de diseño.