Divide y vencerás

Algoritmos y estructuras de datos
Curso 2017/2018

Fernando Ortega, PhD.

Introducción (I)

- El método divide y vencerás está basado en la resolución de un problema dividiéndolo en dos o más subproblemas independientes entre sí de igual tipo o similar.
- El proceso de división continua hasta que el problema llega a ser lo suficientemente sencillo como para poderlo resolver directamente.
- Finalmente se recombinan las soluciones parciales de los subproblemas para obtener la solución del problema original.

Introducción (II)

- Tipo de problemas:
  - Dimensión grande.
  - La solución se puede plantear como la combinación de las soluciones de una serie de subproblemas similares al original.
  - Ejemplos: ordenación por mezcla, quicksort, búsqueda de un elemento dentro de un vector ordenado, torres de Hanoi,...
- Ventaja: proporciona soluciones eficientes.
- Cuidado: gasto por recursividad.

Esquema general

```
FUNCION divide_y_venceras(problema, solucion, ...)

    SI suficientemente_pequeño(problema) ENTONCES
        resolver(solucion)

    SINO
        dividir(problema, prob_1, ..., prob_k)

        divide_y_venceras(prob_1, sol_1)
        ...
        divide_y_venceras(prob_k, sol_k)

        combinar(sol_1, ..., sol_k)
```

Pasos

1. Identificar el tamaño del problema resoluble:
   - Determinar cuál es la solución del problema en el caso “base”.

2. Determinar cómo dividir el problema original y en cuántas partes.

3. Combinar las soluciones parciales para obtener la solución total.

Ejemplo (I)

- Cálculo del valor máximo y mínimo de un conjunto de elementos implementado por medio de un array de N elementos.
- Enfoque clásico:

```
void buscarMinMax (int * array, int n, int * max, int * min) {
    *max = array[0];
    *min = array[0];

    for (int i = 1; i < n; i++) {
        if (array[i] > max) max = vector[i]:
        if (array[i] < min) min = vector[i];
    }
}
```

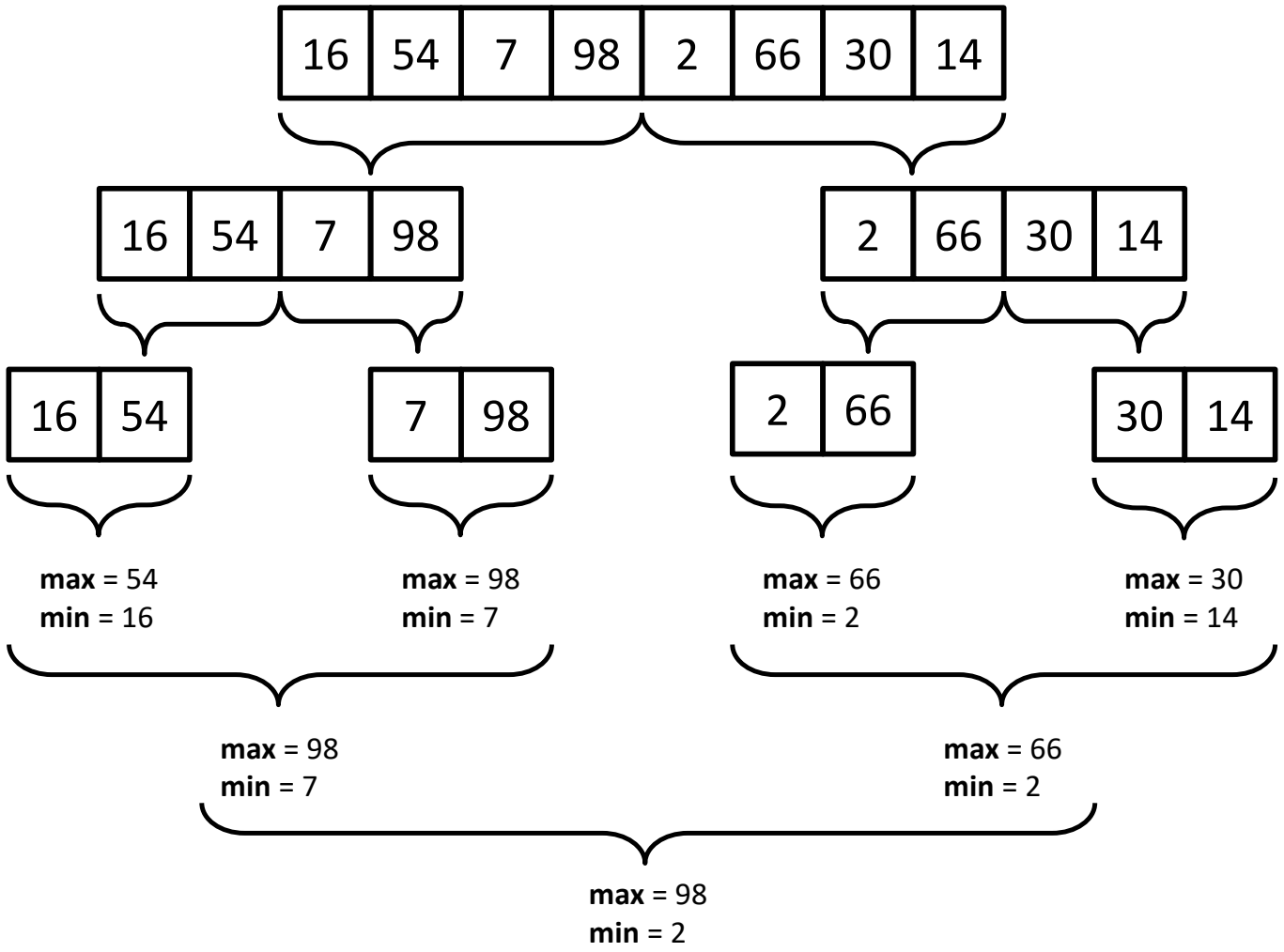
Ejemplo (II)

1. Identificar el tamaño del problema resoluble:
   - Si sólo tenemos 2 números a comprar, el máximo será el mayor y el mínimo el menor.
   - Si sólo tenemos 1 número, éste será tanto el máximo como el mínimo.

2. Determinar cómo dividir el problema original y en cuántas partes.
   - Podemos dividir el array en 2 subarrays de ”igual” tamaño.

3. Combinar las soluciones parciales para obtener la solución total.
   - El máximo será el mayor valor de los máximos de ambos subarrays.
   - El mínimo será el menor valor de los mínimos de ambos subarrays.

Ejemplo (III)



Ejemplo (IV)

```
void calcularMinMax (int * array, int *max, int *min, int sup, int inf) {
    if (sup - inf <= 1) {
        if (array[inf] >= array[sup]) {
            *max = array[inf];
            *min = array[sup];
        } else {
            *max = array[sup];
            *min = array[inf];
        }
    } else {
        int medio = (inf + sup) / 2;
        int max1, min1, max2, min2;

        calcularMinMax(array, &max1, &min1, medio, inf);
        calcularMinMax(array, &max2, &min2, sup, medio+1);

        *max = maximo(max1, max2);
        *min = minimo(min1, min2);
    }
}
```