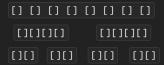
Divide et Impera

Mon. 4 Apr

Divide et Impera è una tecnica di programmazione ricorsiva, basata sullo spezzare il problema in parti più piccole.



Si possono individuare tre fasi:

- **▼** Divide
 - Divide il problema in n sottoproblemi, frazioni del problema iniziale
- **▼** Impera

Risolve i sottoproblemi in maniera indipendente [fase ricorsiva]

▼ Combina

Combina i sottoproblemi per ottenere la soluzione generale

Valutazione dei Tempi di Esecuzione



↑ Divide et Impera

Merge Sort

Mon, 4 Apr

Il Merge Sort è un algoritmo di ordinamento che sfrutta la tecnica di programmazione Divide et Impera. È un algoritmo di ordinamento stabile, ovvero che mantiene l'ordine che avevano relativo tra di loro valori uguali presenti nell'array di partenza, ma non in loco, ovvero necessita di variabili di appoggio che aumentano all'aumentare di n.

Istanza d'esempio

▼ *Merge Sort* applicato a due *array* di esempio

[Grigio] → Preso in considerazione · [Giallo] → Ordinato · [Viola] → Ignorato

10,	35,	03,	04,	30,	10,	12,	01,	07
						12,		
						12,		
10	35	03,	04	30,	10,	12,	01,	07

```
35
        03, 04
        03,
            04
10,
        03,
            04
                30, 10, 12, 01, 07
10,
            04
10,
        03
            04
10,
        03
03,
            35
                30, 10, 12, 01, 07
                         12, 01, 07
03,
    04,
        10,
                30,
                     10
    04,
        10,
                    10
                         12, 01, 07
                30
                    10
    04,
                30
                    10
    04,
        10,
03,
                30, 10
                         12, 01, 07
    04,
                30,
                             01, 07
    04,
        10,
                             01, 07
                30, 10
                         12
                             01
                                 07
03,
        10,
                30,
                             01 07
    04,
        10,
03,
    04,
                30,
                             01
                                 07
03,
    04,
                30,
                     10
                         12
                             01,
                30,
                         01, 07,
    04,
        10,
03,
        10,
                01,
                     07,
                         10,
                                  30
    03, 04, 07, 10, 10, 12, 30, 35
```

Valutazione tempi di esecuzione

▼ Pseudocodice dell'algoritmo

```
void MergeSort(A[n], int i, int f)
begin

if i < f

m = (i + f) /i 2
    divisione intera

MergeSort(A[], i, m)

MergeSort(A[], m+1, f)

Merge(A[], i, m, f)
endM</pre>
```

```
if A[i1] <= A[i2]
            B[ib] = A[i1]
            i1++
         else
            B[ib] = A[i2]
            i2++
         ib++
      end
      while i1 <= med
      begin
         B[ib] = A[i1]
         i1++
         ib+
      end
      while i2 <= p2
      begin
         B[ib] = A[i2]
         i2++
         ib+
      end
      for ib = p1 to p2
         A[ib] = B[ib]
      end
   end
Merge
Formula per il calcolo delle operazioni
   Tmerge(n) = 3c + 4cn + 2cn \approx n
Caso migliore:
```

Merge Sort

Formula per il calcolo delle operazioni

$$Tms(n) =$$

Tm(n) pprox n Caso peggiore: Tp(n) pprox n Tempo medio: $\Theta(n)$

Caso migliore:

Tm(n)

Caso peggiore:

Tp(n)

Tempo medio: $\Theta(nlogn)$

vedi lez 6.04.2022

↑ Merge Sort

Merge Sort applicato a due array di esempio

[Grigio] → Preso in considerazione

[Giallo] → Ordinato

[Viola] → Ignorato

↑ Merge Sort

$$m = (i + f) / i 2$$

divisione intera

↑ Merge Sort

in questo modo si prende l'elemento in i1, garantendo la stabilità dell'algoritmo