MERGESORT

Integrantes: JORGE SIQUEIRA SERRÃO

DIVIDIR E CONQUISTAR

- O mergesort consiste de um algoritmo que divide um problema maior em vários subproblemas menores recursivamente e então os combina para encontrar a solução final.

- O mergesort é divido em três passos sendo eles:
- **Dividir**: Quebrar o problema maior em subproblemas.
- **Conquistar**: Resolver os problemas recursivamente.
- **Combinar**: Combinar as soluções para chegar no resultado final.

FUNCIONAMENTO

75	29	57	12	39	89	65	41

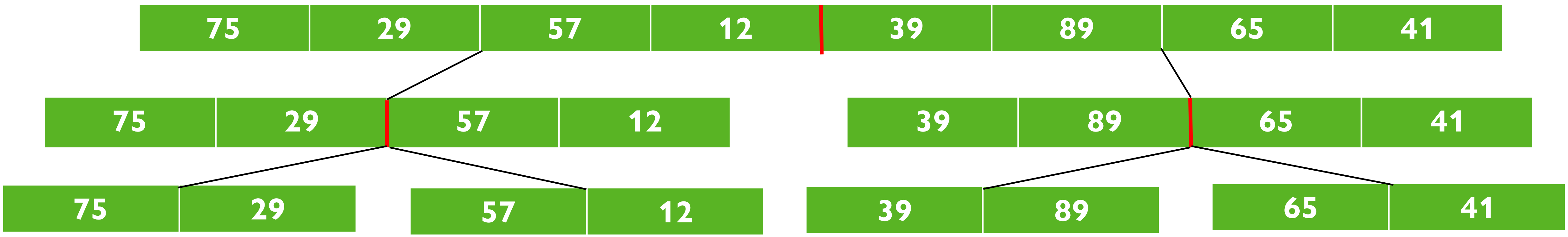
FUNCIONAMENTO

75	29	57	12	39	89	65	41

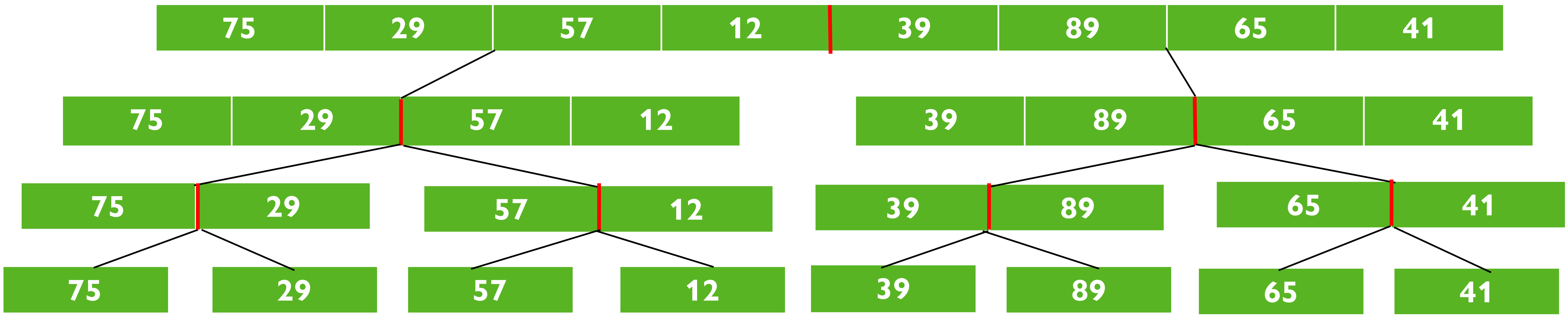
75	29	57	12

39	89	65	41

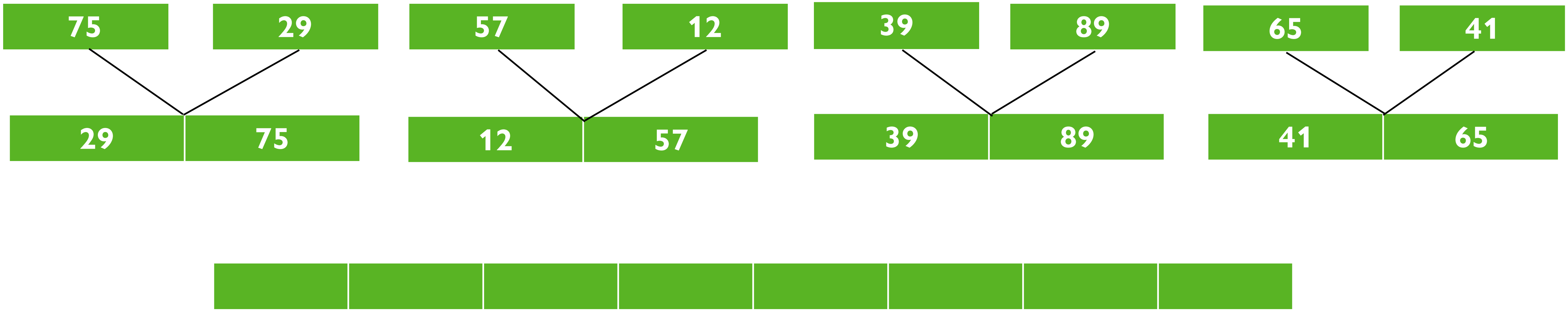
FUNCIONAMENTO



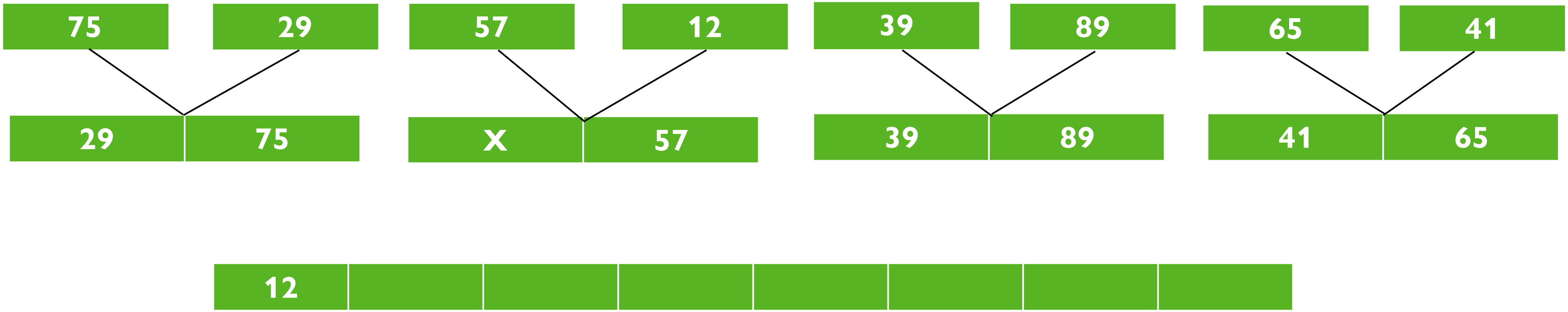
FUNCIONAMENTO



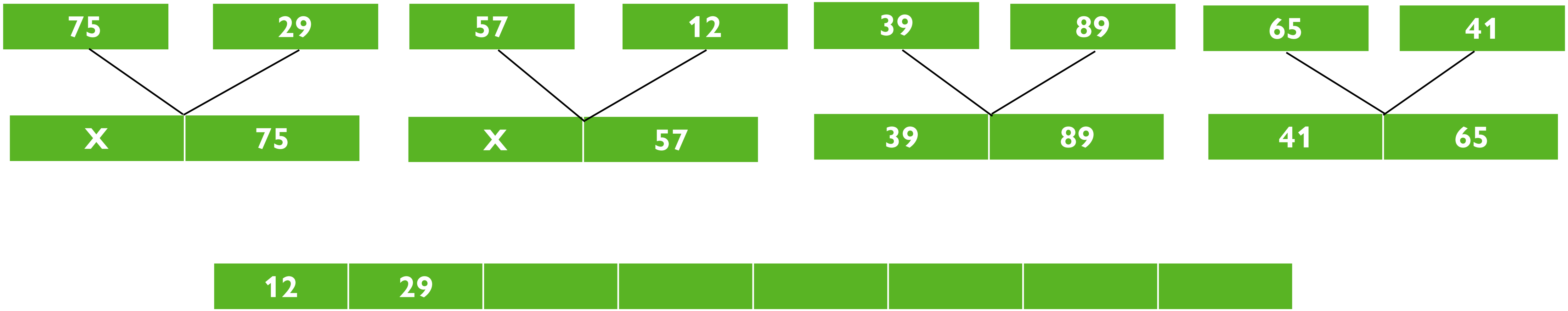
FUNCIONAMENTO



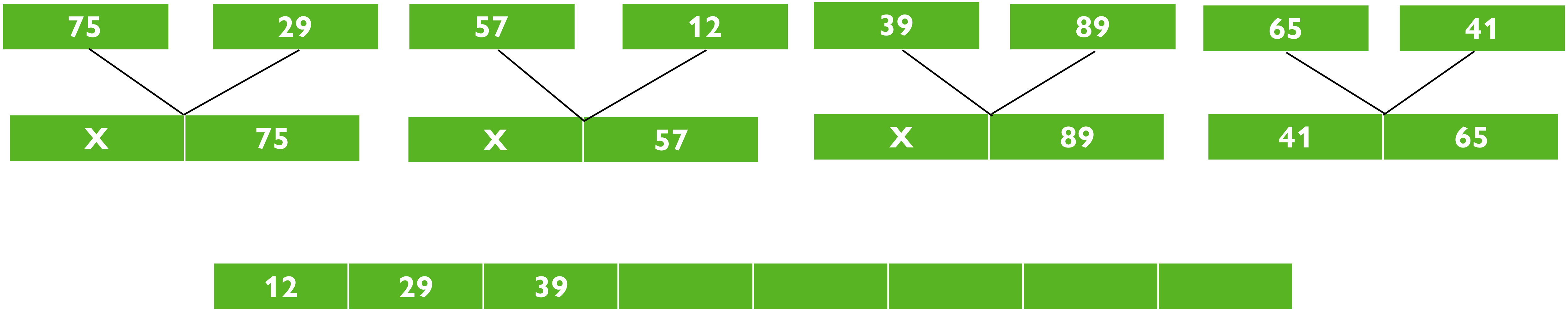
FUNCIONAMENTO



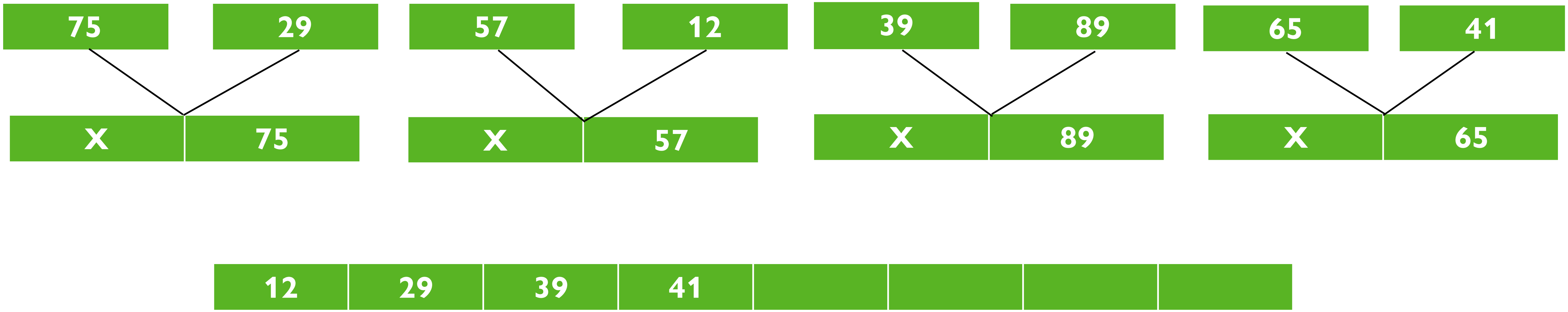
FUNCIONAMENTO



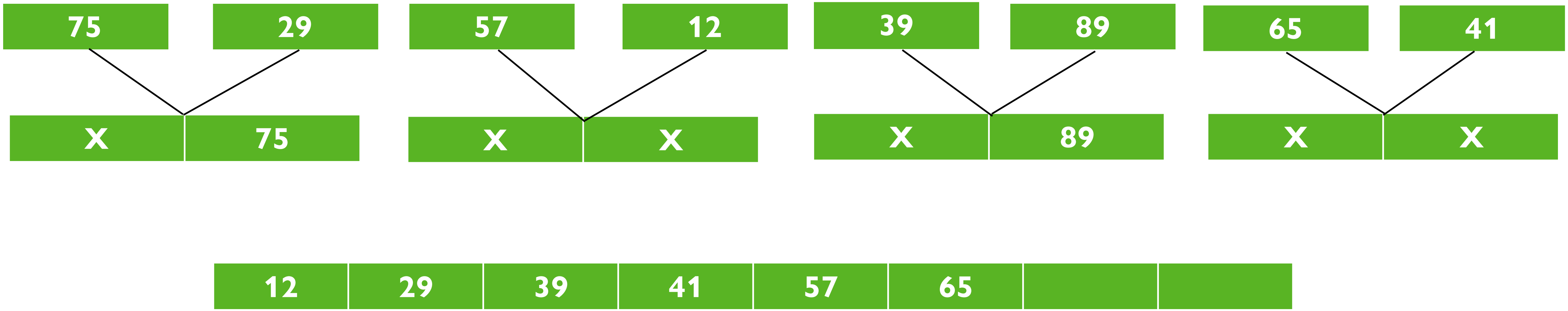
FUNCIONAMENTO



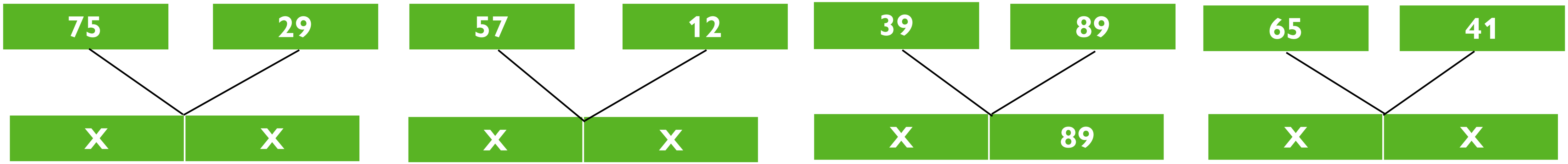
FUNCIONAMENTO



FUNCIONAMENTO

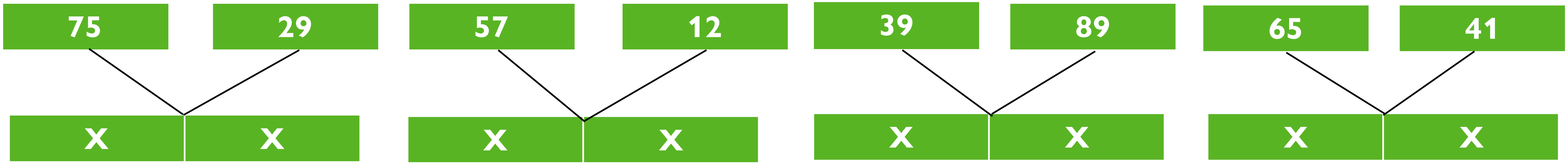


FUNCIONAMENTO



12	29	39	41	57	65	75	

FUNCIONAMENTO



12	29	39	41	57	65	75	89

RELAÇÃO DE RECORRÊNCIA

- T(n) = {1 se n = 1} e {3T(n/3) + n + 1 se n > 1}

$$T(n) = 3T(n/3) + n + 1$$

$$T(n/3) = 3T[3T(n/9) + (n/3) + 1] + n + 1$$

$$T(n/3) = 9T(n/9) + 2n + 3 + 1$$

$$T(n/9) = 9T[3T(n/27) + (n/9) + 1] + 2n + 3 + 1$$

$$T(n/9)\ 27T(n/27) + 3n + 9 + 3 + 1$$

RELAÇÃO DE RECORRÊNCIA

- $T(n) = \{1 \text{ se } n = 1\} \text{ e } \{3T(n/3) + n + 1 \text{ se } n > 1\}$

$$T(n/27) = 27T[3T(n/81) + (n/27) + 1] + 3n + 9 + 3 + 1$$

$$T(n/27) = 81T(n/81) + 4n + 27 + 9 + 3 + 1$$

$$T(n) = (3^4)T(\ n/(3^4)\ ) + 4n + 3^3 + 3^2 + 3^1 + 3^0$$

$$T(n) = (3^k)T(\ n/(3^k)\ ) + k*n + (\ (\text{somatorio}(\Sigma)\ \text{de } i=0 \text{ até } k - 1)\ 3^i\ )$$

$$T(n) = (3^k)T(\ n/(3^k)\ ) + k*n + (\ (\ 1 - 3^{(k-1)}\ )/(1 - 3)\ )$$

$$T(n) = (3^k)T(\ n/(3^k)\ ) + k*n + (\ (\ (\ 3^{(k - 1)}\ )\ -1\ )/\ 2\ )$$

RELAÇÃO DE RECORRÊNCIA

- Fazendo $n = 3^k$ e $\log n$ na base $3 = k$:

$$T(n) = (3^k)T(1) + k*n + ((3^{(k - 1)}) - 1) / 2$$

$$T(n) = 3^k + k*n + ((3^{(k-1)}) - 1) / 2$$

$$T(n) = n + n*\log n \text{ base } 3 + ((3^{\log n-1 \text{ base } 3}) -1) / 2$$

$$T(n) = n + n*\log n \text{ base } 3 + (n - 3) / 6 \Rightarrow O(n*\log n \text{ base } 3)$$

- Para o melhor caso, pior caso e caso médio o custo será $O(n*\log n)$

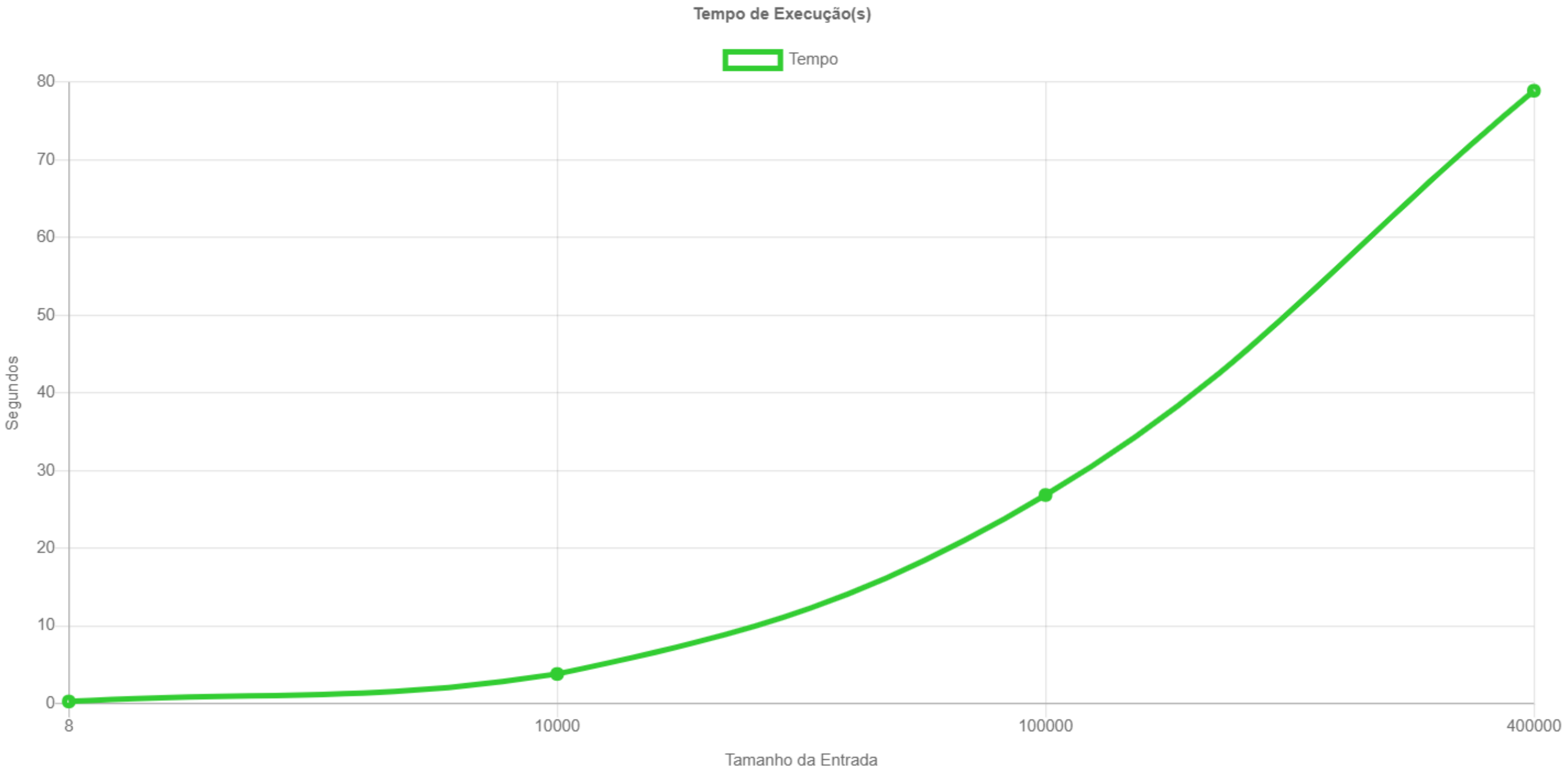


GRÁFICO DO TEMPO DE EXECUÇÃO DO MERGERSORT

- Entrada 1: 75, 29, 57, 12, 39, 89, 65, 41.

  Tempo: 0,277s.

- Entrada 2: array desordenado com 10000 números.

  Tempo: 3,817s.

- Entrada 3: array desordenado com 100000 números.

  Tempo: 26,858s.

- Entrada 4: array desordenado com 400000 números.

  Tempo: 78,898s.

ANALISANDO QUICKSORT

- $T(n) = 2T(n/2) + O(n)$

$$T(n) = 2T(n/2) + n$$

$$T(n/2) = 2[2T((n/2)/2) + n/2] + n$$

$$T(n/2) = 4T(n/4) + n + n$$

$$T(n/4) = 4[2T((n/4)/2) + n/4] + n + n$$

$$T(n/4) = 8T(n/8) + n + n + n$$

$$T(n) = (2^k)T(n/2^k) + kn, \text{ com } n = 2^k$$

$$T(n) = nT(n/n) + kn, \text{ com } k = \log(n)$$

$$T(n) = nT(1) + n\log(n), \text{ com } T(1) = 0$$

ANALISANDO QUICKSORT

- $T(n) = 2T(n/2) + O(n)$

$$T(n) = n\log(n)$$

Complexidade $O(n*\log n)$

- O quicksort neste caso possui uma complexidade similar ao mergesort.