cin.ufpe.br

Algoritmos e Estruturas de Dados

Mergesort

d) Considere a execução do MergeSort sobre a entrada $V = (3,4,7,6,2,1,0,5)$. Imediatamente após a ***terceira*** execução da função ***merge***, o vetor estará na forma $V = (3,4,6,7,1,2,0,5)$.

d) Considere a execução do MergeSort sobre a entrada $V = (3,4,7,6,2,1,0,5)$. Imediatamente após a ***terceira*** execução da função ***merge***, o vetor estará na forma $V = (3,4,6,7,1,2,0,5)$.

0	1	2	3	4	5	6	7
3	4	6	7	2	1	0	5

falso: fica assim (ver ilustração das iterações a seguir)

a) O MergeSort é um algoritmo do tipo ***dividir para conquistar***.

b) O MergeSort tem complexidade de ***tempo*** $\Theta(n)$ no ***melhor*** caso.

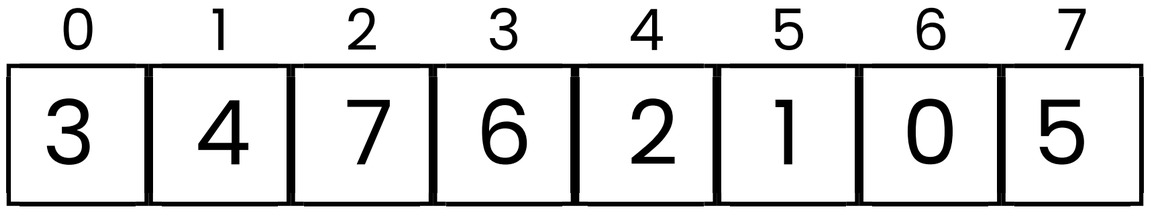
c) O MergeSort tem complexidade de ***espaço*** $\Theta(n)$ no ***pior*** caso.

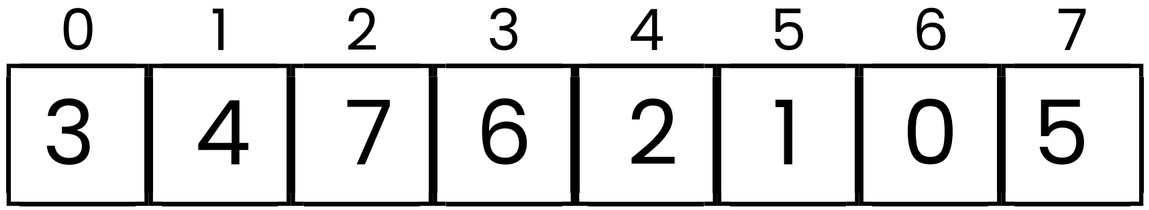
a) O MergeSort é um algoritmo do tipo ***dividir para conquistar***. V

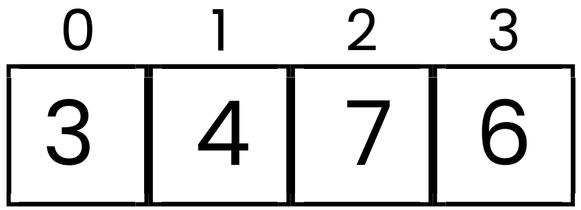
b) O MergeSort tem complexidade de ***tempo*** $\Theta(n)$ no ***melhor*** caso. F: theta(n*lg(n))

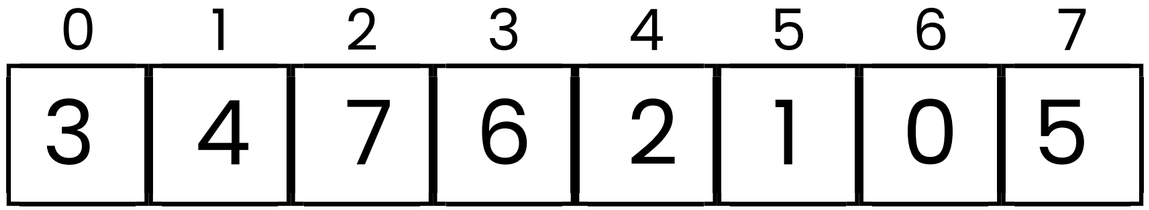
c) O MergeSort tem complexidade de ***espaço*** $\Theta(n)$ no ***pior*** caso.

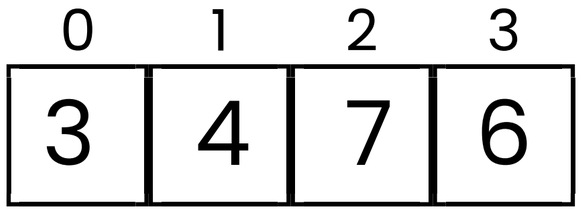
V: ver ilustrações para entender por que não é O(n*lg(n))

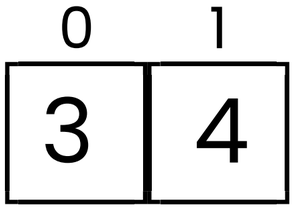

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

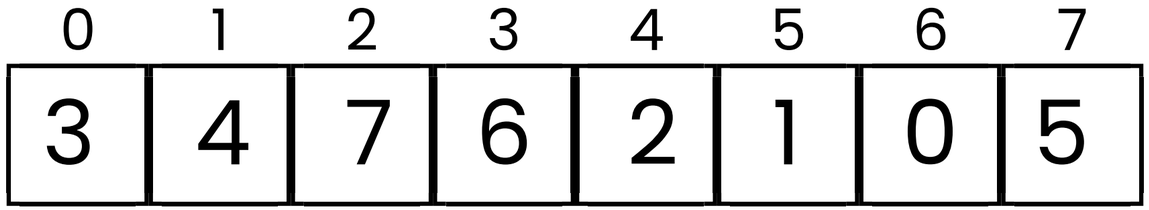

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

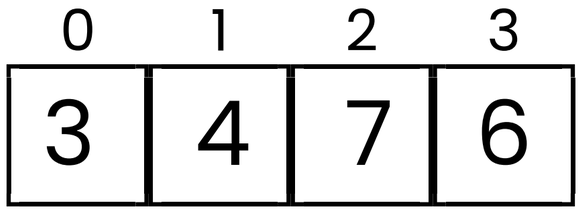

0 1 2 3
3 4 7 6

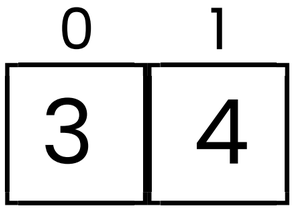

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

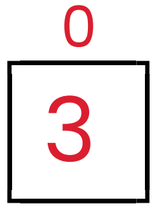

0 1 2 3
3 4 7 6

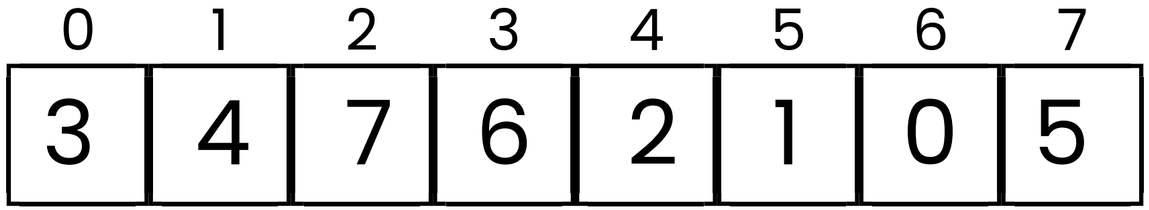

0 1
3 4

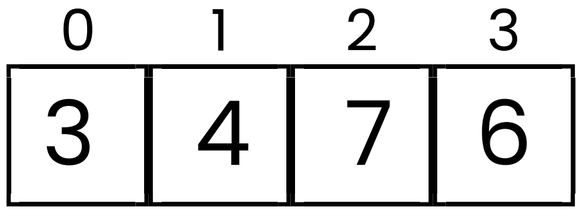

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

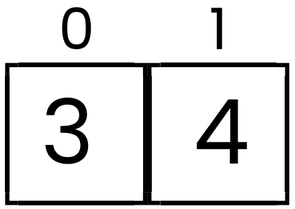

0 1 2 3
3 4 7 6

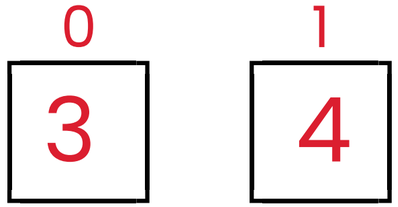

0 1
3 4

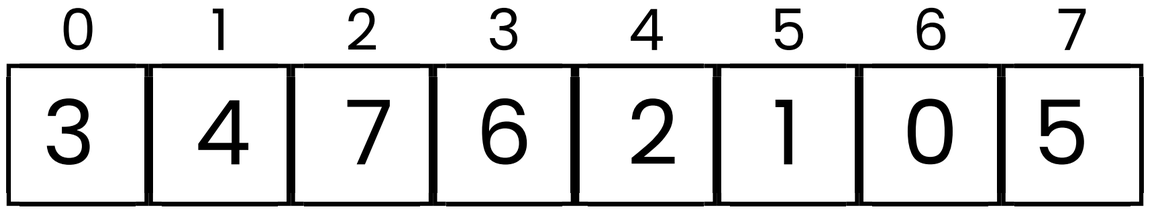

0
3

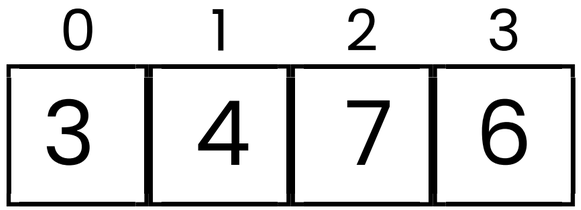

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

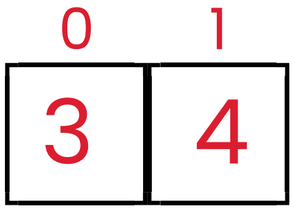

0 1 2 3
3 4 7 6


0 1
3 4

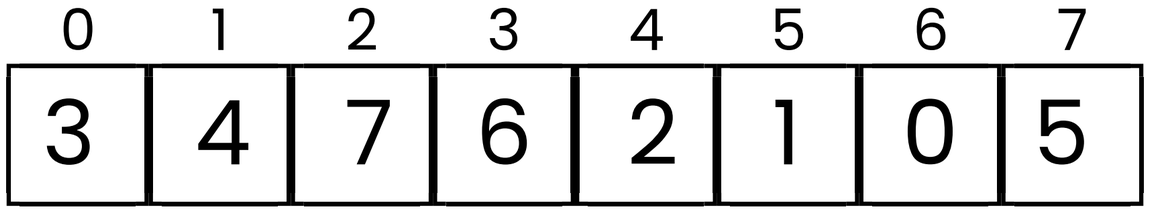

0 1
3 4

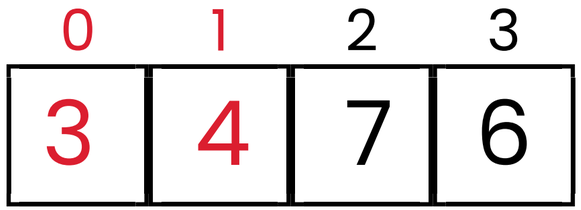


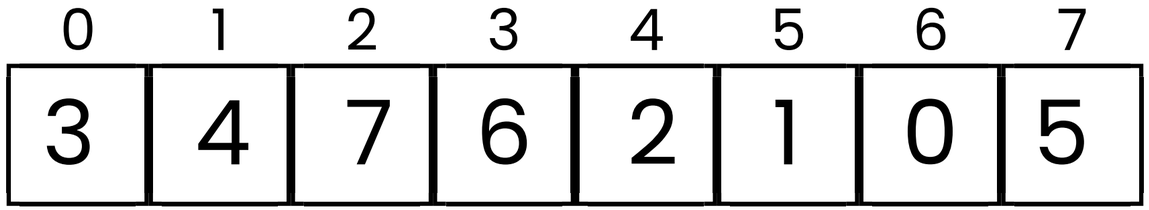


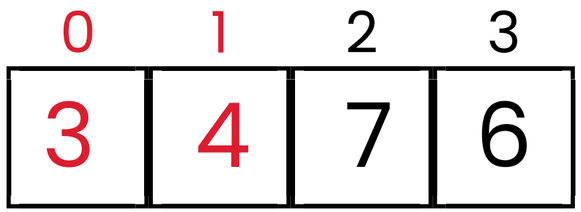


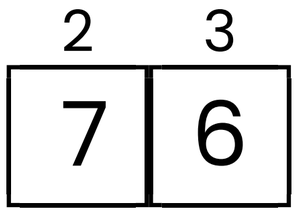
merge #1

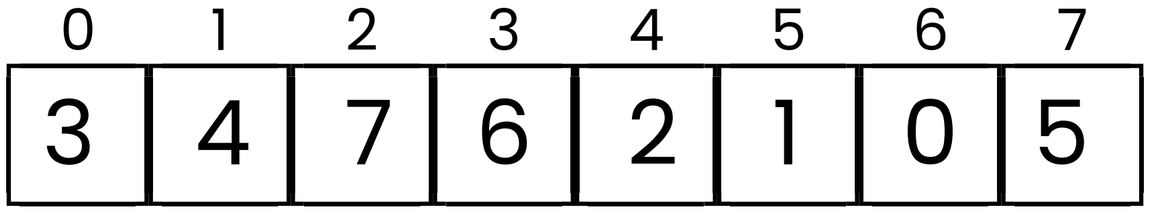

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

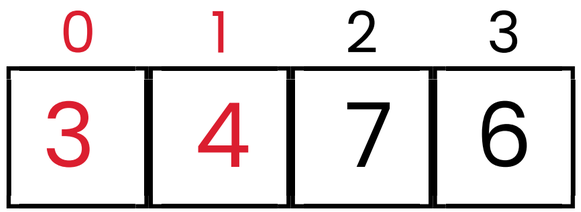

0 1 2 3
3 4 7 6

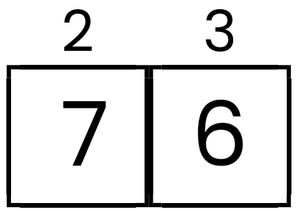

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

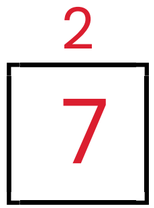

0 1 2 3
3 4 7 6

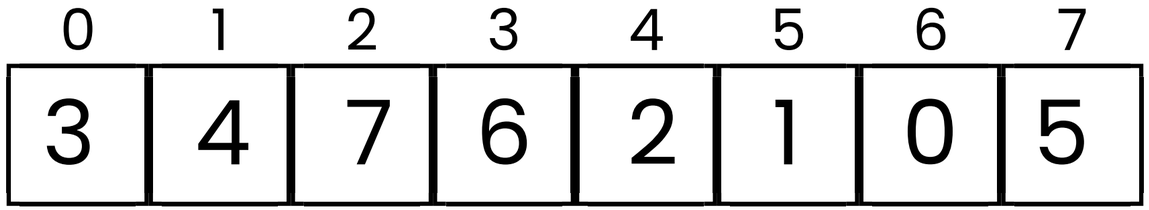

2 3
7 6

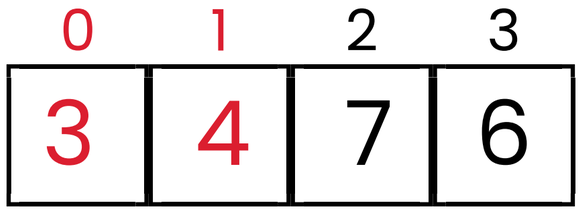

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

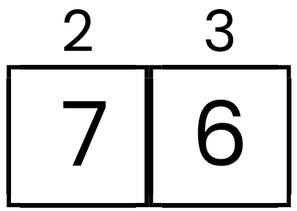

0 1 2 3
3 4 7 6

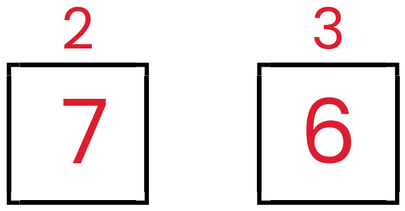

2 3
7 6

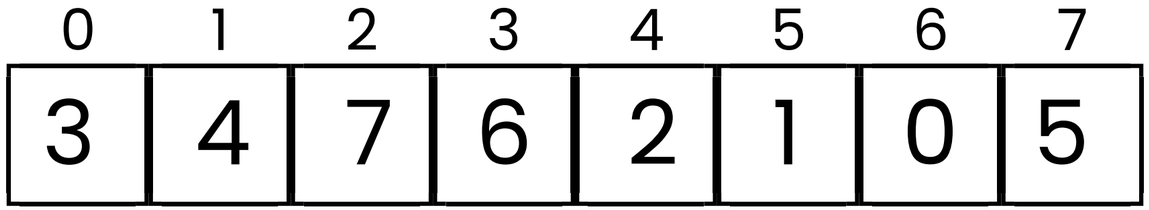

2
7

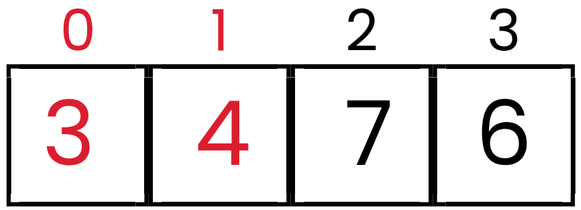

0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5

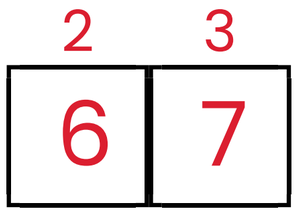

0 1 2 3
3 4 7 6


2 3
7 6

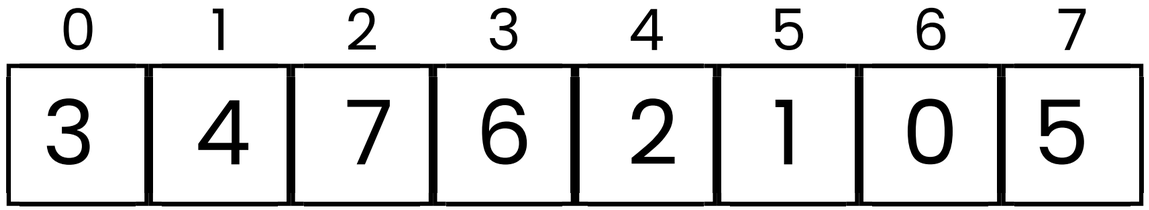

2 3
7 6

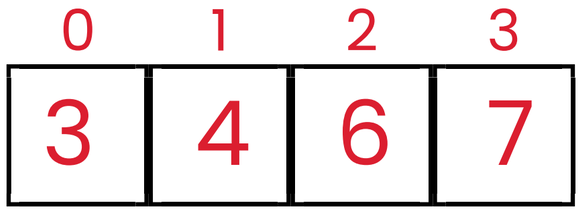




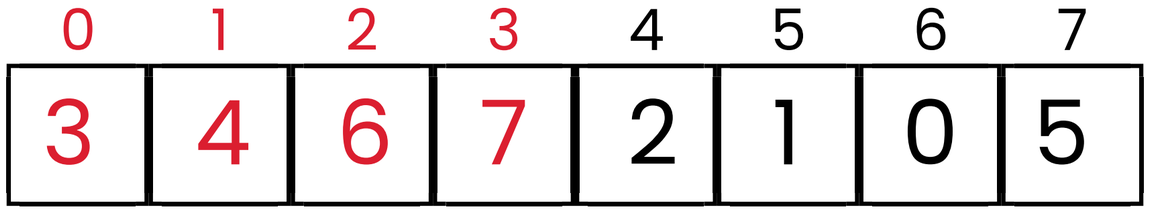


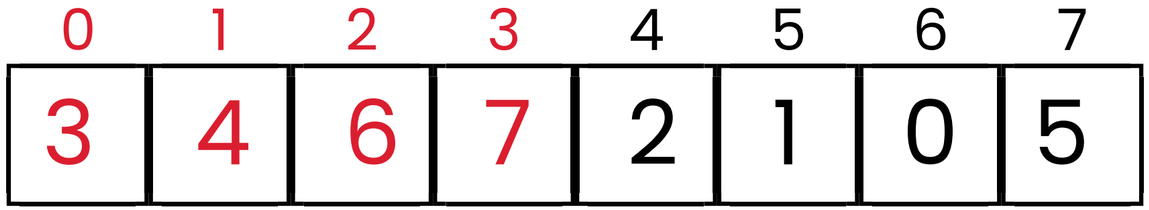
merge #2

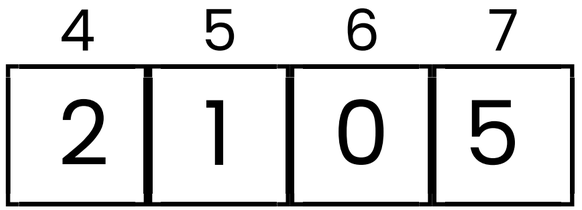


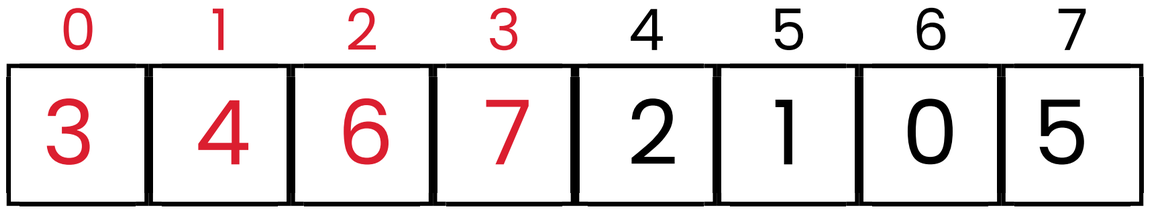


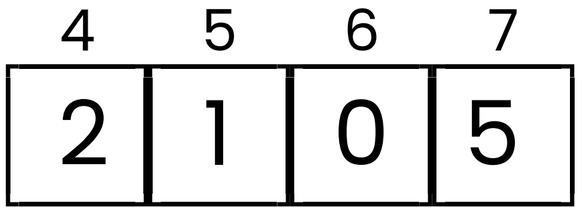
merge #3

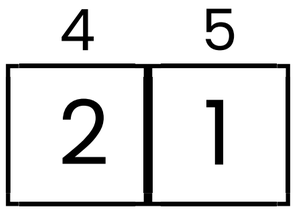

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

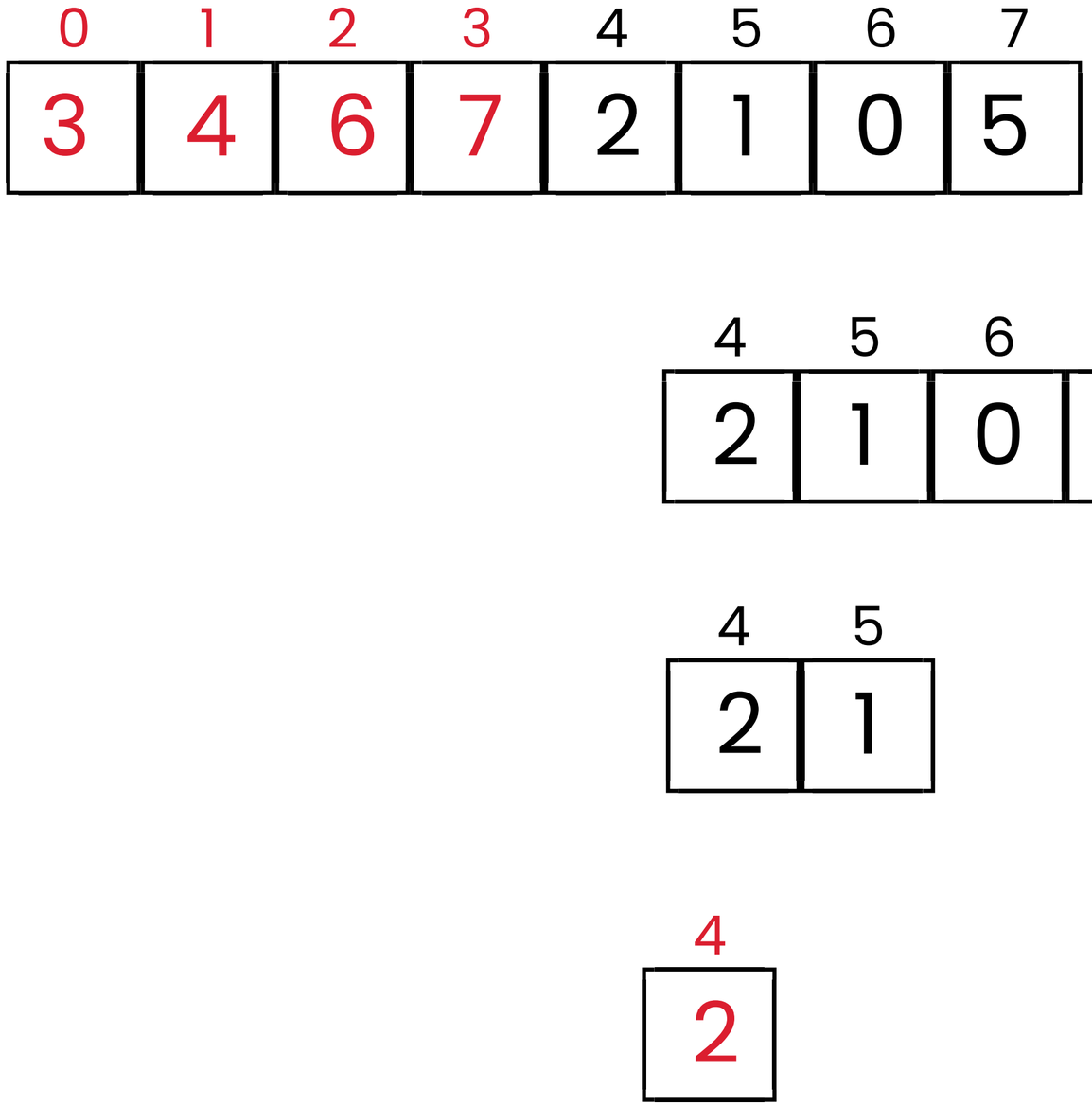

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

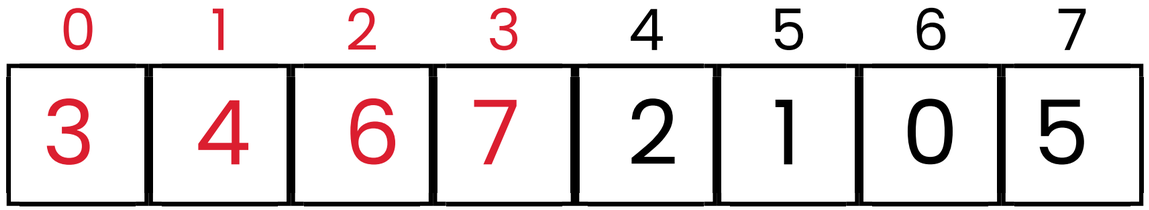

4 5 6 7
2 1 0 5

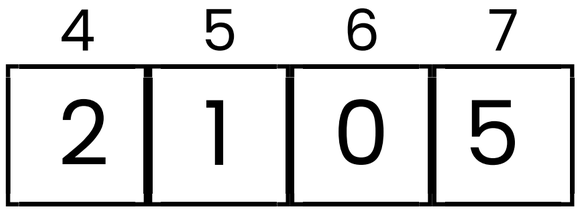

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

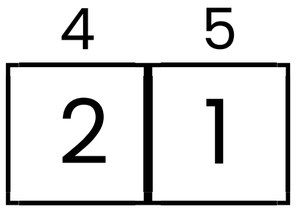

4 5 6 7
2 1 0 5

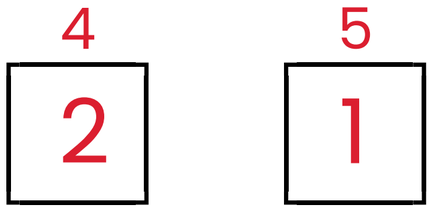

4 5
2 1

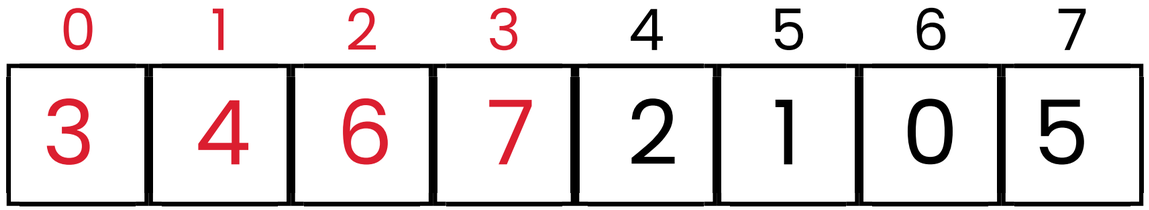

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5
4 5 6 7
2 1 0 5
4 5
2 1
4
2

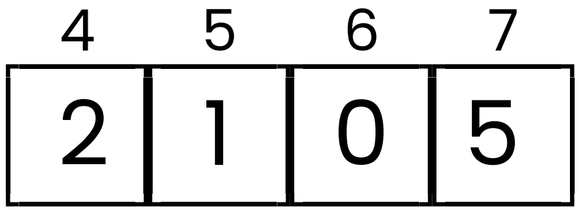

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

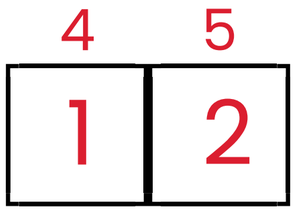

4 5 6 7
2 1 0 5

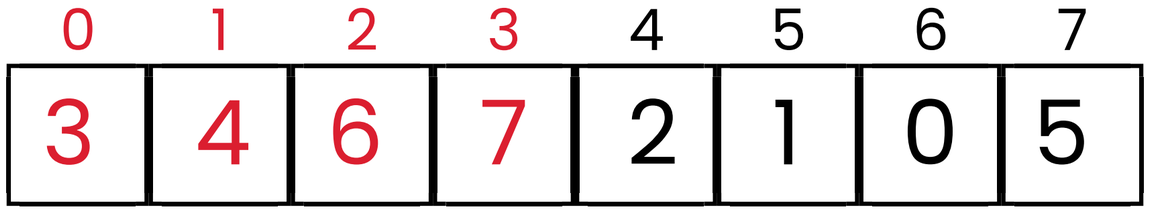

4 5
2 1

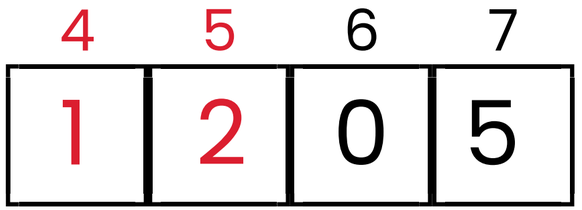

4 5
2 1

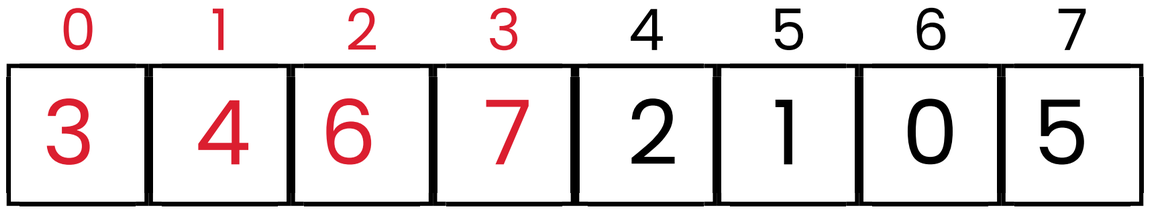

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

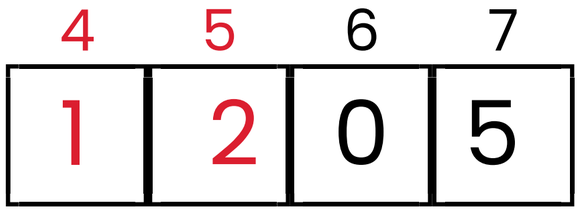

4 5 6 7
2 1 0 5

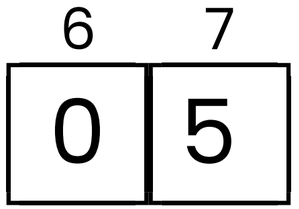

4 5
1 2

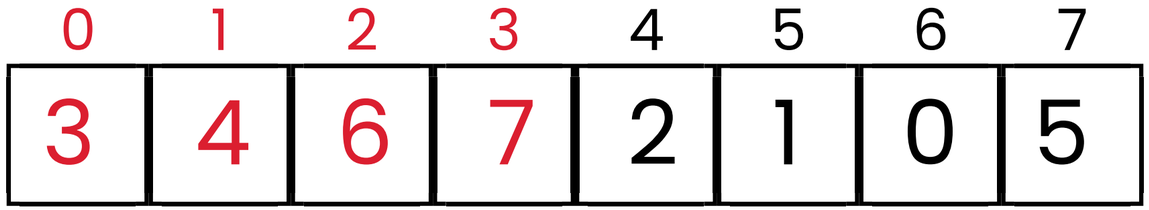

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

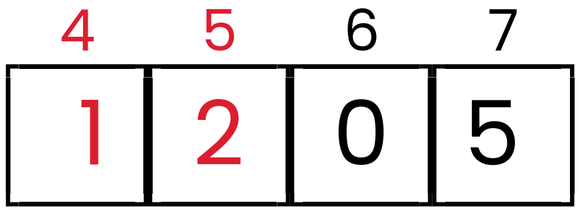

4 5 6 7
1 2 0 5

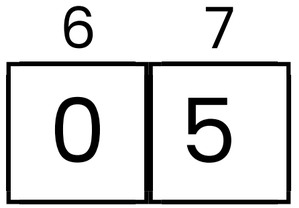

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

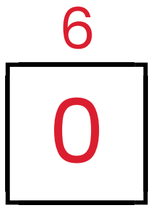

4 5 6 7
1 2 0 5

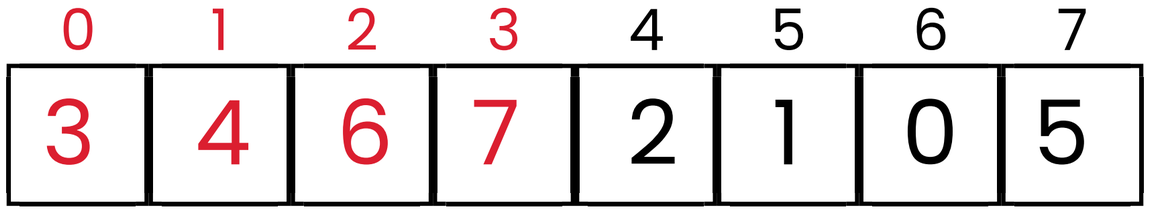

6 7
0 5

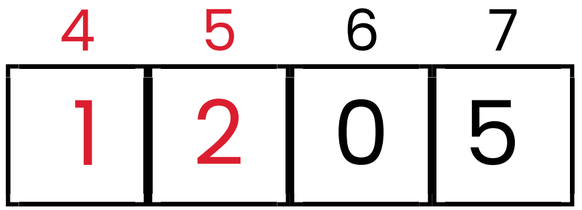

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

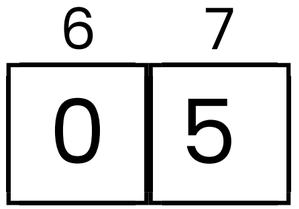

4 5 6 7
1 2 0 5

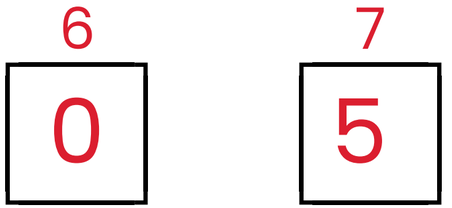

6 7
0 5

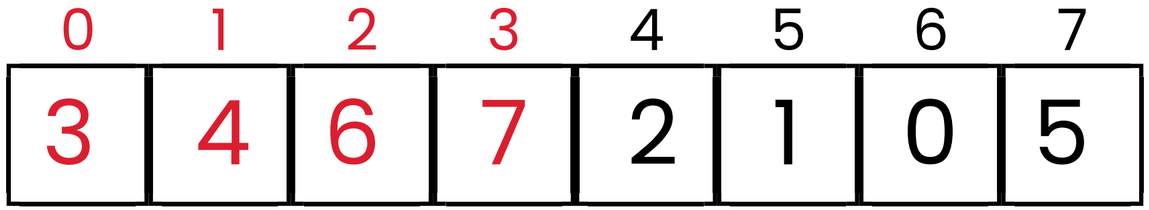

6
0

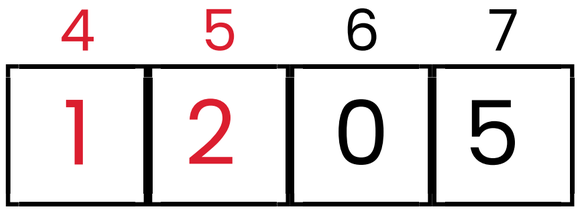

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

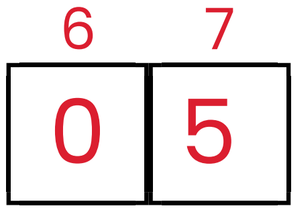

4 5 6 7
1 2 0 5

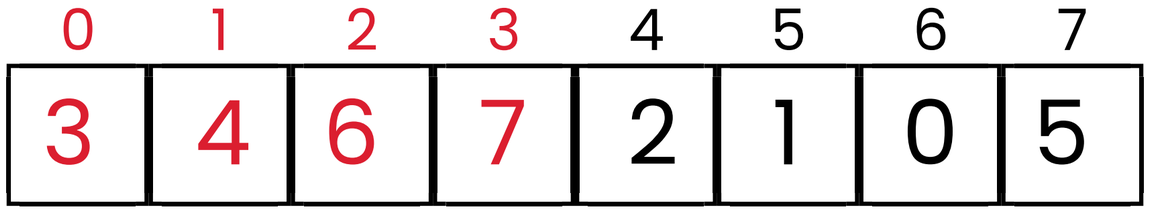

6 7
0 5

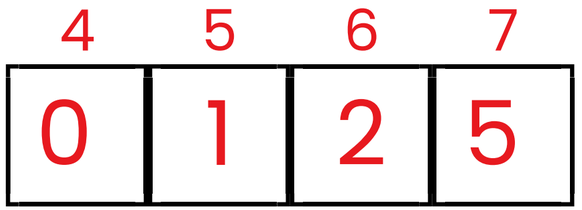

6
0
7
5

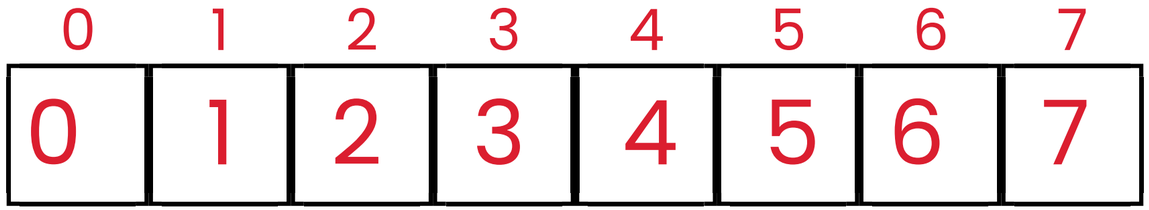

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5

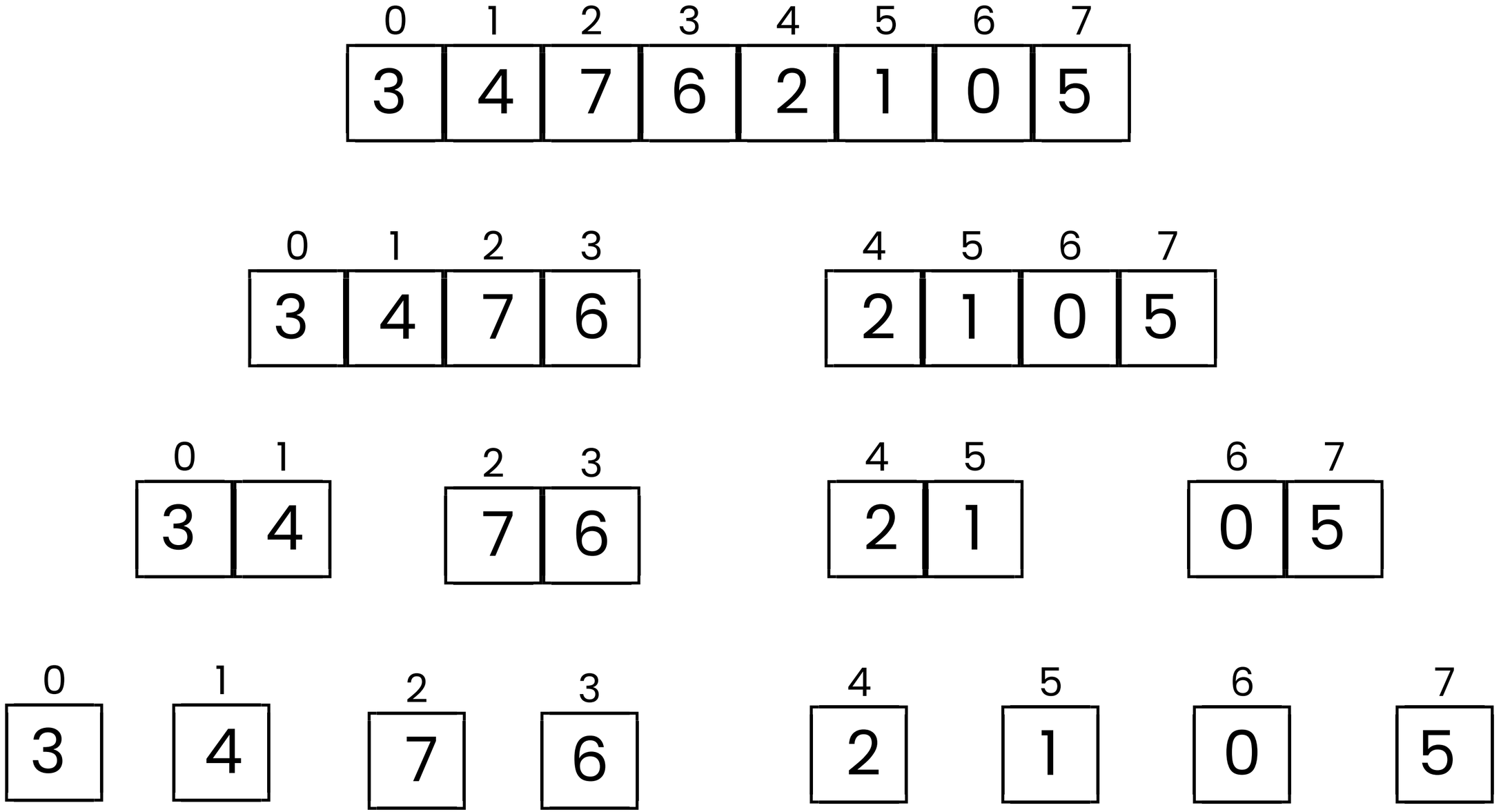

4 5 6 7
1 2 0 5

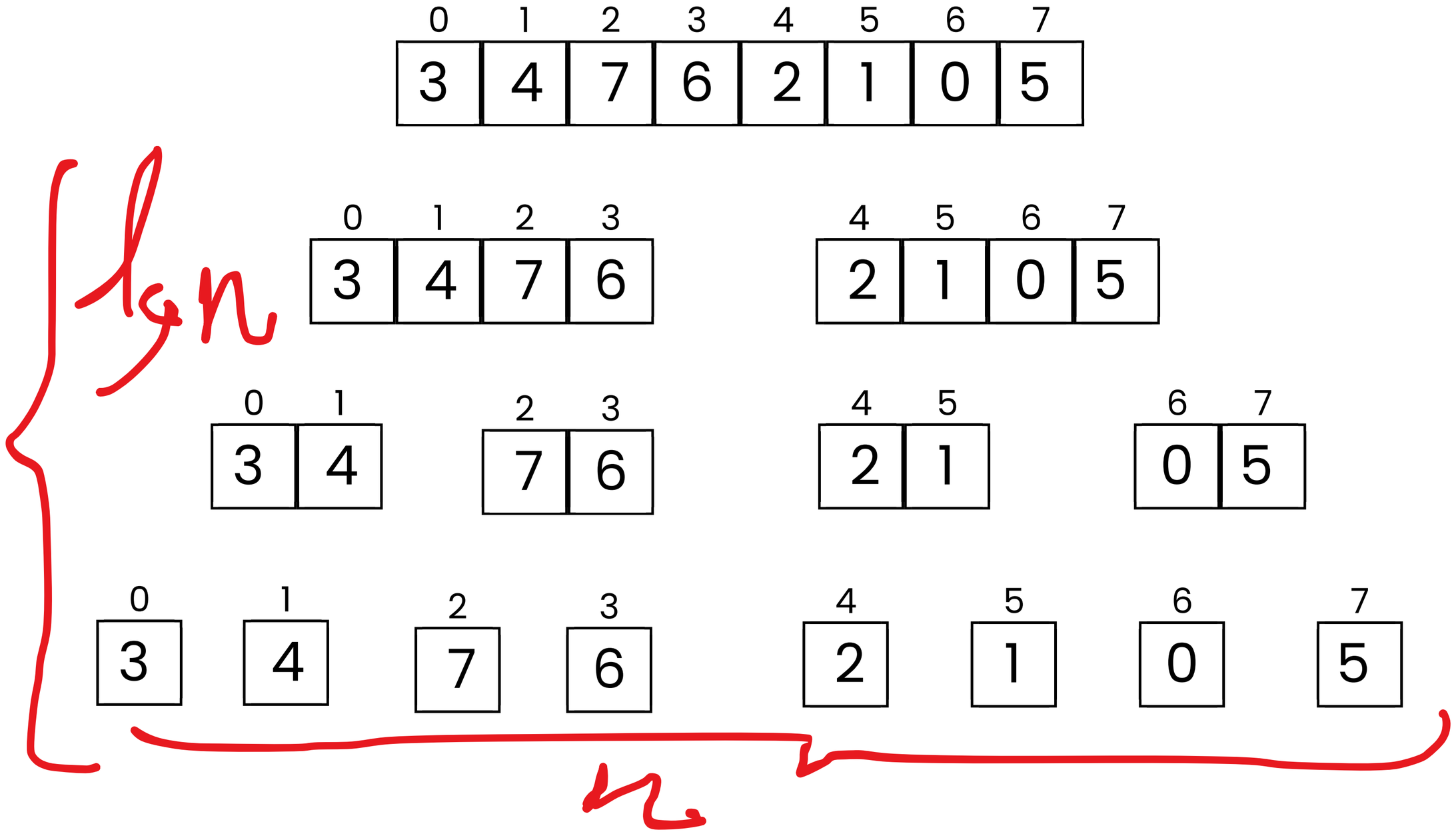

6 7
0 5

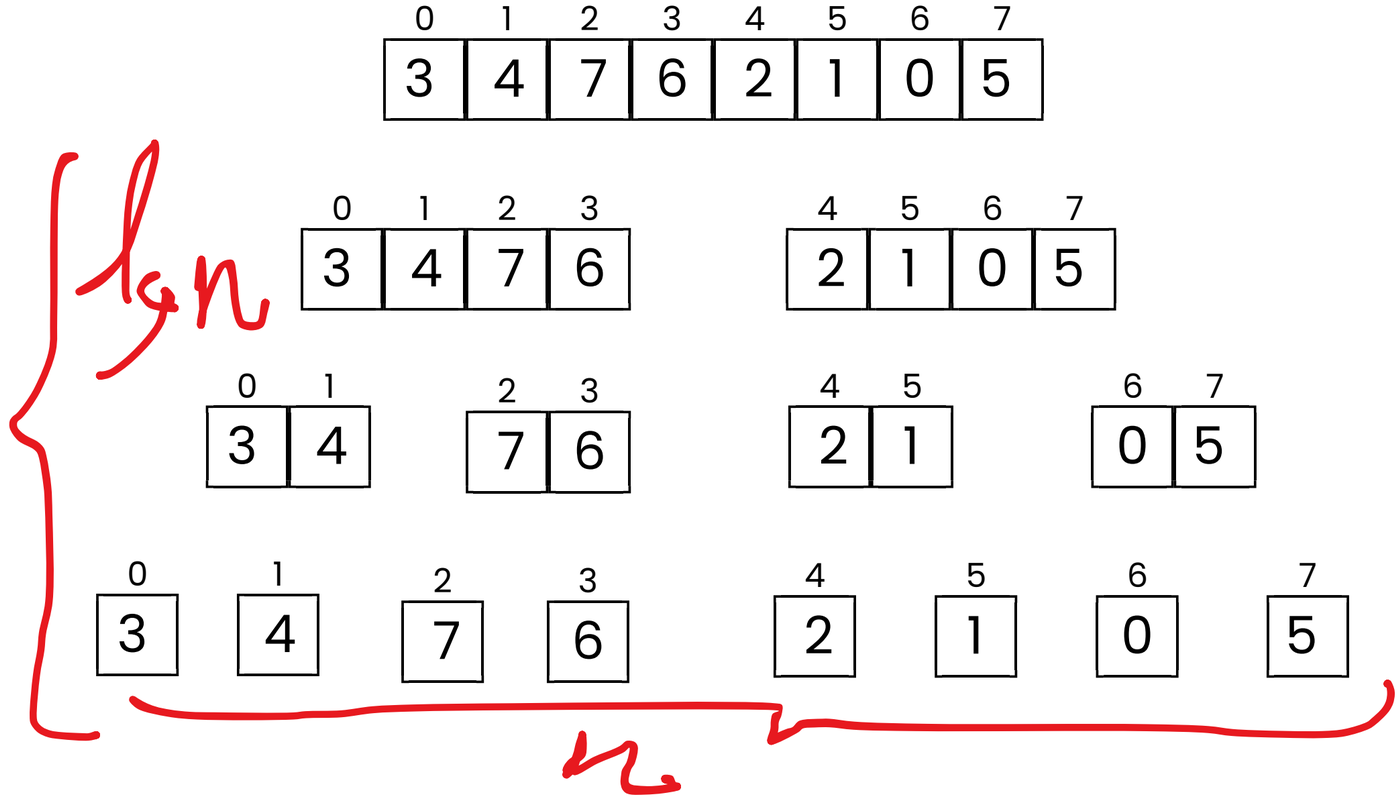

0 1 2 3 4 5 6 7
3 4 6 7 2 1 0 5


4 5 6 7
0 1 2 5


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5
0 1 2 3
3 4 7 6
4 5 6 7
2 1 0 5
0 1
3 4
2 3
7 6
4 5
2 1
6 7
0 5
0
3
1
4
2
7
3
6
4
2
5
1
6
0
7
5


0 1 2 3 4 5 6 7
3 4 7 6 2 1 0 5
0 1 2 3
3 4 7 6
4 5 6 7
2 1 0 5
0 1
3 4
2 3
7 6
4 5
2 1
6 7
0 5
0
3
1
4
2
7
3
6
4
2
5
1
6
0
7
5
log n
n



complexidade de tempo theta(n*lg(n)) vem daqui. Mas note que a memória auxiliar não armazena, em um **mesmo instante**, a árvore de recursão **inteira**. A complexidade de memória é theta(n)

- **Não deixem de fazer 2-3 provas passadas!!!**

https://www.cin.ufpe.br/~paguso/courses/if672ec/arquivo/

(**prova 1**, deixar prova 2 e finais para depois)

Add a subheading

- visualização de algoritmos de ordenação:

visualgo