ASA 4

DIVIDE AND CONQUER

DIVIDE AND CONQUER

Divide and Conquer adalah paradigma pemrograman yang membagi suatu permasalahan besar menjadi sub-masalah yang lebih kecil, menyelesaikan masing-masing sub-masalah, dan menggabungkan hasilnya untuk mendapatkan solusi akhir.

TAHAPAN

1. Divide (Pemisahan)
    - Memecah masalah menjadi dua atau lebih sub-masalah yang lebih kecil dengan karakteristik yang sama atau serupa.
2. Conquer (Penyelesaian)
    - Menyelesaikan setiap sub-masalah secara rekursif. Jika sub-masalah cukup kecil, langsung selesaikan secara langsung (base case).
3. Combine (Penggabungan)
    - Menggabungkan hasil dari setiap sub-masalah untuk mendapatkan solusi dari permasalahan awal.

MERGE SORT

1. Divide (Pemisahan)
    - Membagi masalah dari mengurutkan n element array menjadi 2 sub-problem yaitu mengurutkan 2 buah n/2 element array.
2. Conquer (Penyelesaian)
    - Menyelesaikan task sort n/2 element array secara rekursif, base case ketika array kosong atau berelemen sebanyak 1.
3. Combine (Penggabungan)
    - Menggabungkan 2 array berukuran n/2 yang sudah terurut.

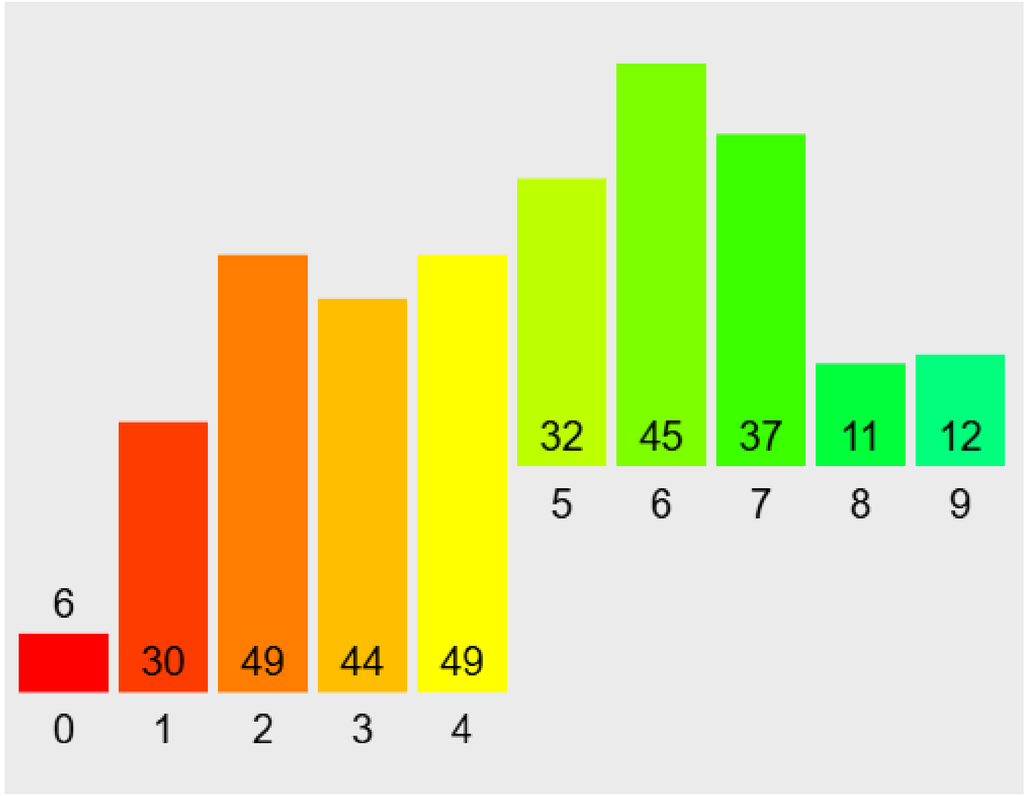
ALGORITMA MERGE SORT

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    # Divide: Membagi array menjadi dua bagian
    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])

    # Combine: Menggabungkan hasil pengurutan kedua bagian
    return merge(left, right)
```

ALGORITMA MERGE SORT

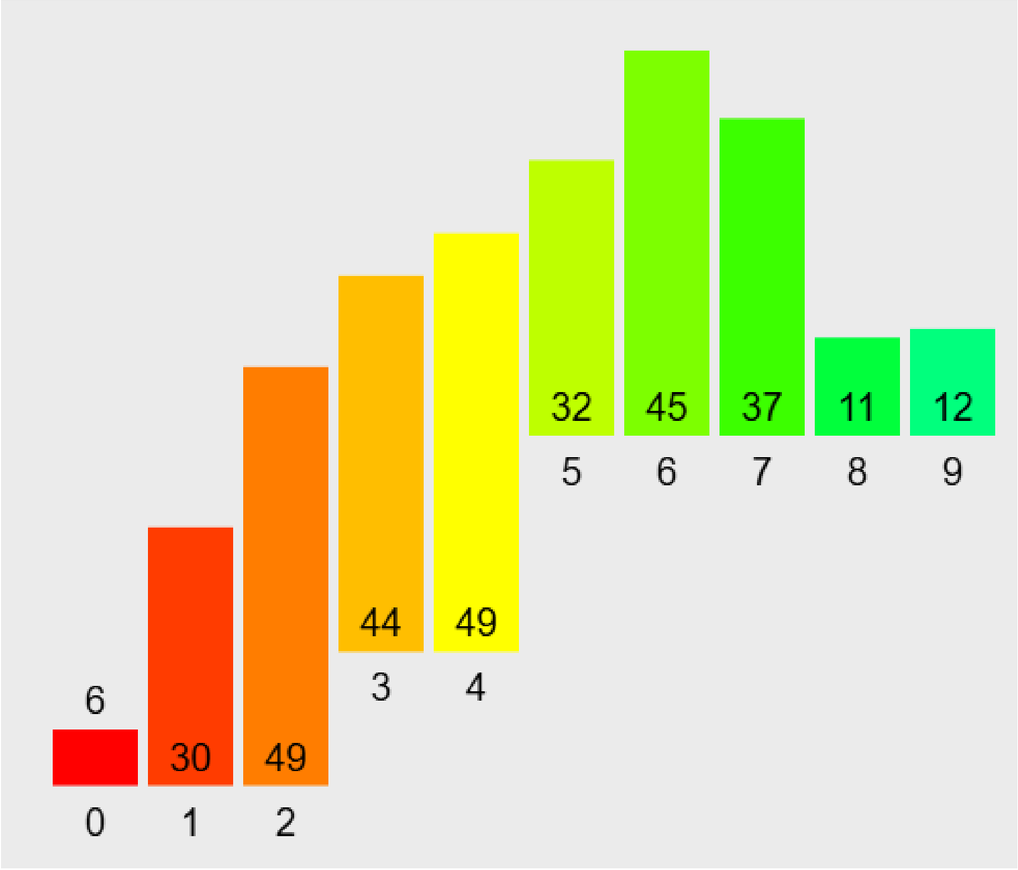
```python
def merge(left, right):
    result = []
    i = j = 0

    # Menggabungkan dua array terurut menjadi satu
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    # Menambahkan elemen yang tersisa
    result.extend(left[i:])
    result.extend(right[j:])
    return result
```

ALGORITMA MERGE SORT



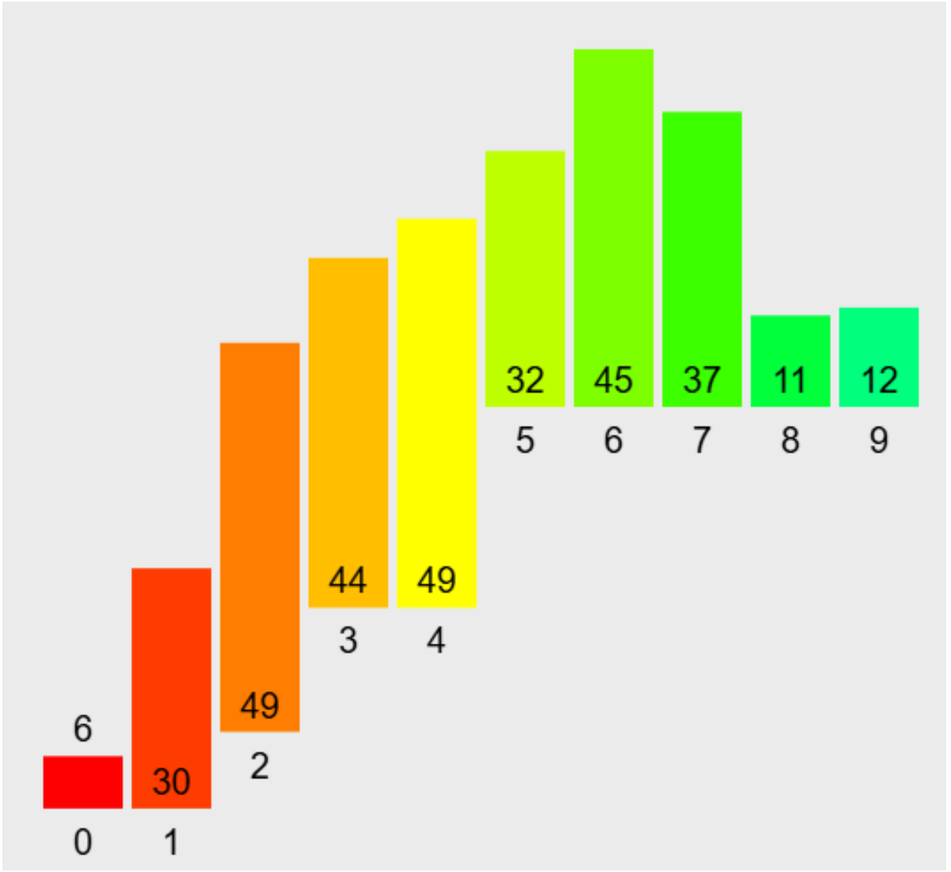
Divide: Bagi array menjadi 2 pada index [0, 9] menjadi index [0,4] dan [5,9]

ALGORITMA MERGE SORT



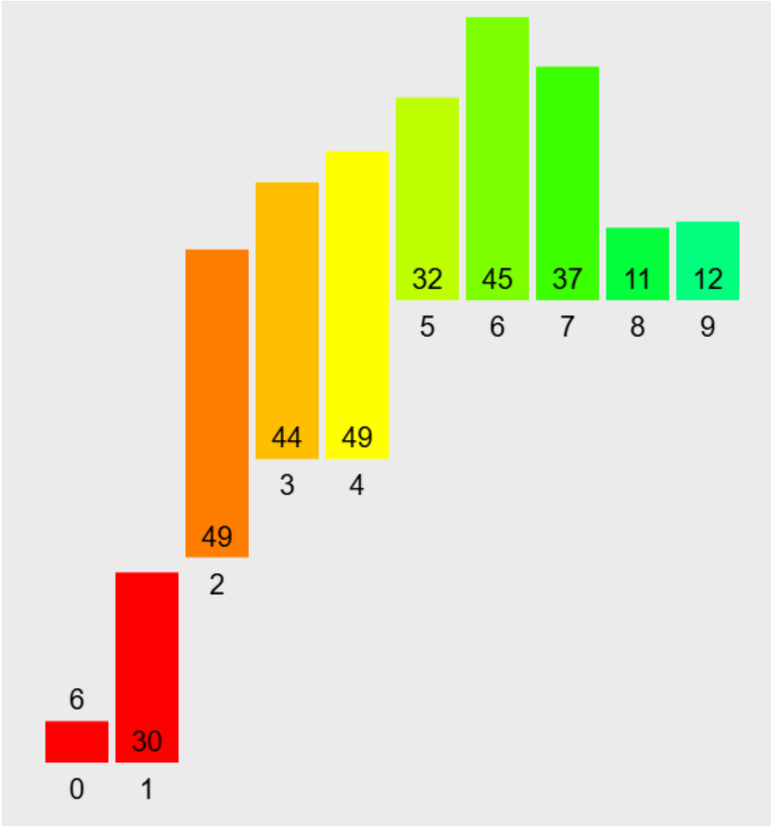
Divide: Bagi array menjadi 2 pada index [0, 4] menjadi index [0,2] dan [3,4]

ALGORITMA MERGE SORT



Divide: Bagi array menjadi 2 pada index [0, 2] menjadi index [0,1] dan [2,2]]

ALGORITMA MERGE SORT

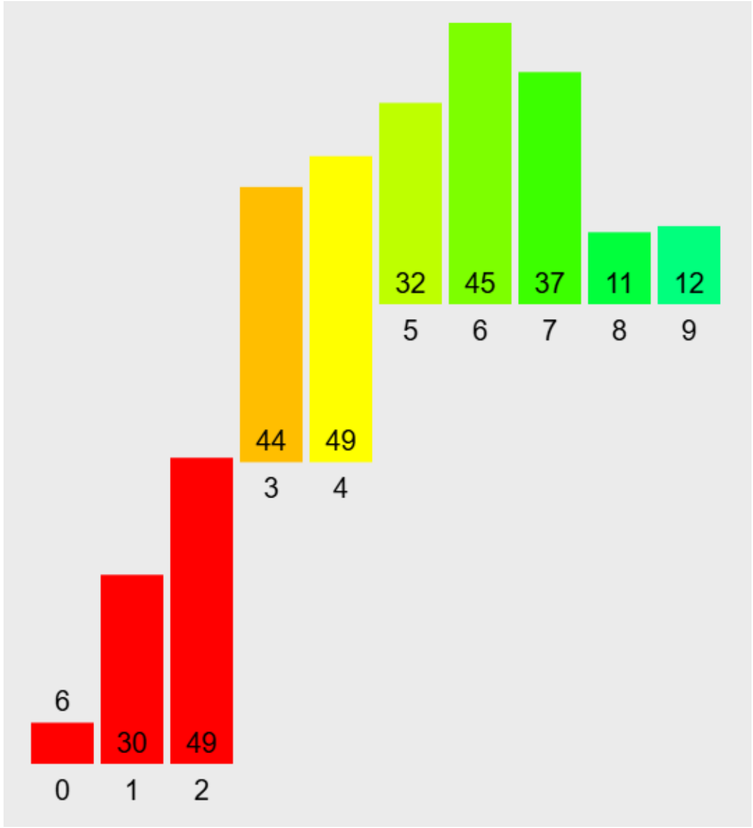


Conquer

Karena elemen array [0,1] sudah 2, kita bisa lakukan perbandingan secara langsung..

Pada contoh ini elemen array [0,1] sudah sesuai.

ALGORITMA MERGE SORT

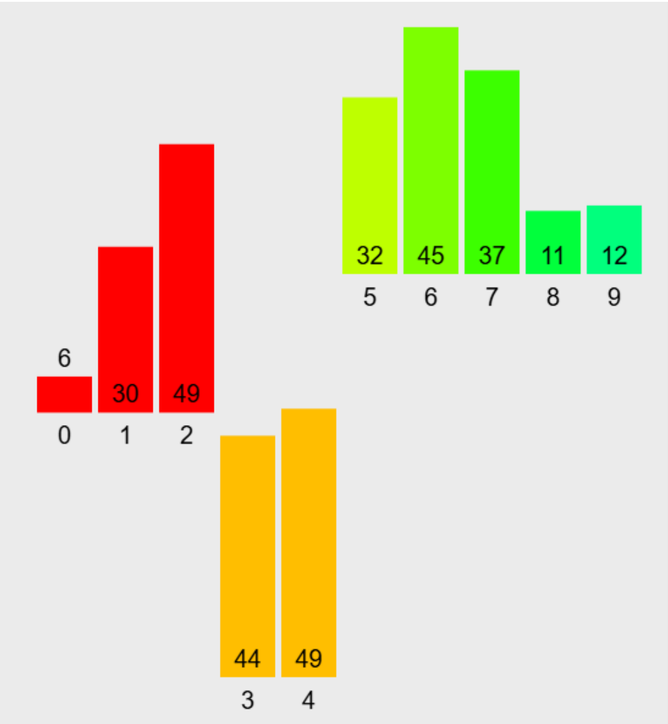


Merge

Gabungkan elemen array [0,1] dan [2,2]]

Hasil: array [0,2] terurut

ALGORITMA MERGE SORT

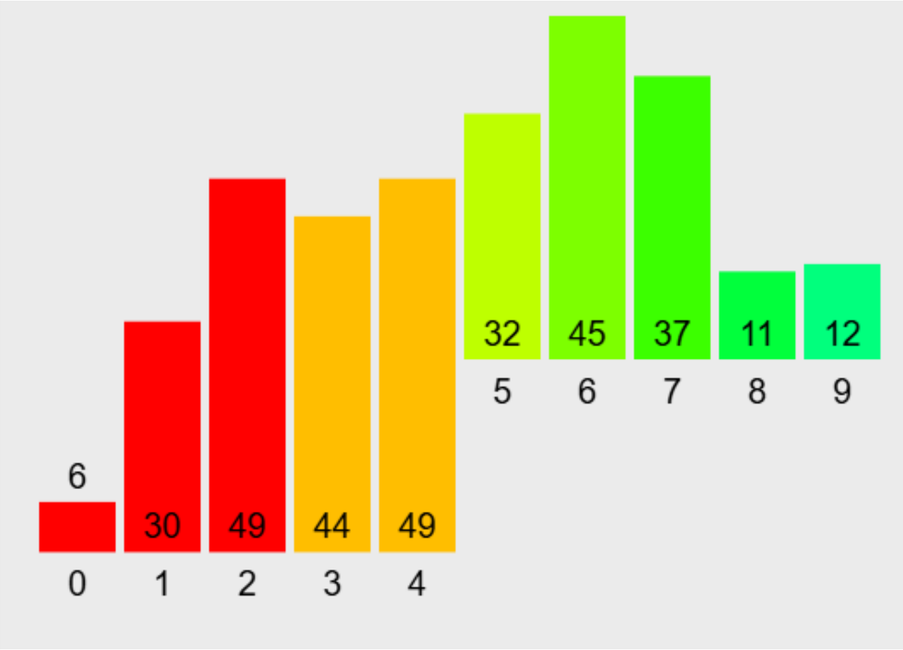


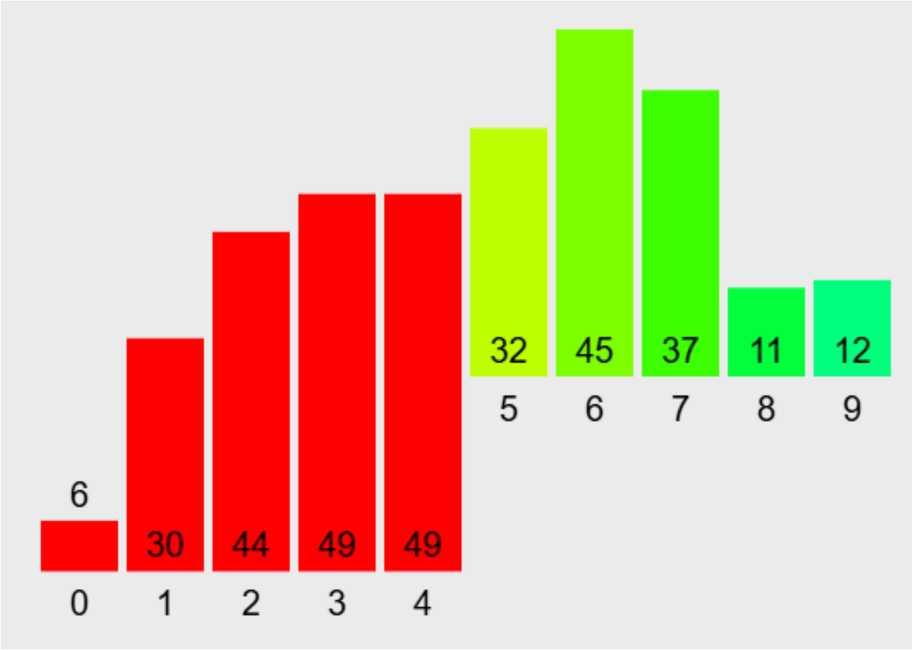
Conquer

Karena elemen array [3,4] sudah 2, kita bisa lakukan perbandingan secara langsung..

Pada contoh ini elemen array [3,4] sudah sesuai.

ALGORITMA MERGE SORT



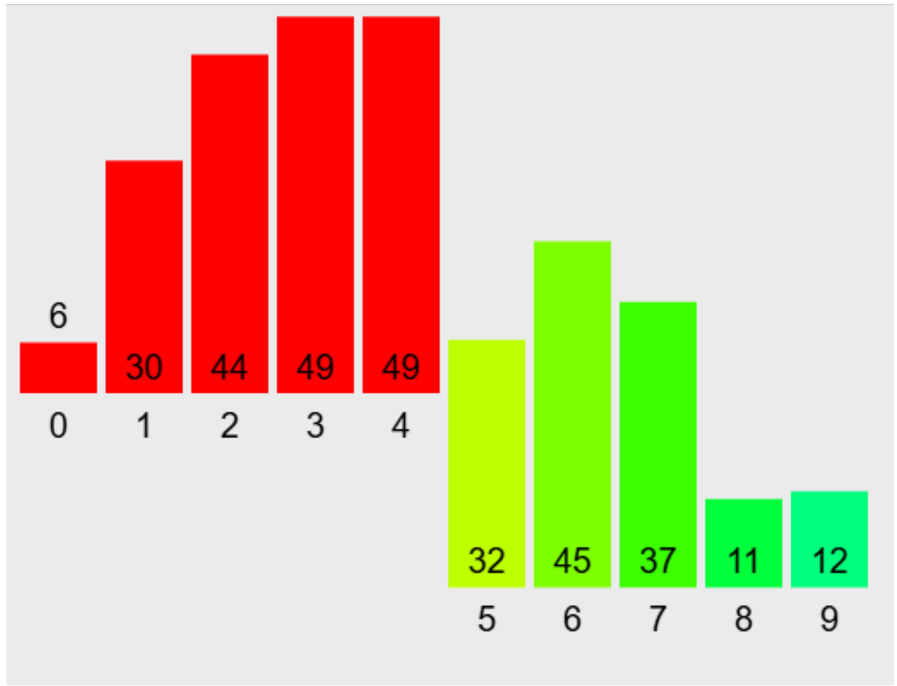


Merge

Kita sudah memiliki 2 array terurut [0,2] dan [3,4], lakukan penggabungan

Hasil: arraay [0,4] terurut

ALGORITMA MERGE SORT



Karena elemen [0,4] sudah terurut, sisa mengurutkan elemen [5,9]

Untuk elemen sisanya, lakukan eksplorasi pada link: https://visualgo.net/en/sorting

Lalu pilih pada bagian Merge Sort

MASTER THEOREM

Master Theorem berlaku untuk persamaan rekursif dalam bentuk:

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d$$

Untuk menentukan bentuk pasti dari $T(n)$, terdapat 3 kasus:

1. **Kasus 1**: $a > b^d$, maka:
$$T(n) = \Theta(n^{\log_b a})$$
2. **Kasus 2**: $a = b^d$, maka:
$$T(n) = \Theta(n^d \log n)$$
3. **Kasus 3**: $a < b^d$, maka:
$$T(n) = \Theta(n^d)$$

MASTER THEOREM - MERGE SORT

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    # Divide: Membagi array menjadi dua bagian
    mid = len(arr) // 2
    left = merge_sort(arr[:mid]) # rekursi: T(n/2)
    right = merge_sort(arr[mid:]) # rekursi: T(n/2)

    # Combine: Menggabungkan hasil pengurutan kedua bagian
    return merge(left, right) # O(n)
```

MASTER THEOREM - MERGE SORT

Sebagai contoh untuk kasus merge sort, bentuk persamaannya adalah

$$T(n) = 2T\left(\frac{n}{2}\right) + 1 \cdot n^1$$

Diketahui $a = 2,\ b = 2,\ c = 1,\ d = 1$ dan $2 = 2^1$ maka menggunakan kasus kedua.

$$T(n) = \Theta(n^1 \log n) = \Theta(n \log n)$$

QUICK SORT

1. Divide (Pemisahan)
    - Memilih pivot (bebas, sebagai contoh ambil saja elemen array pertama) lalu bagi array menjadi 3 bagian: < pivot; = pivot; > pivot dan urutkan array yang < pivot dan > pivot.
2. Conquer (Penyelesaian)
    - Menyelesaikan task sort array < pivot dan > pivot dengan base case elemen array < 1.
3. Combine (Penggabungan)
    - Menggabungkan 3 bagian array yang sudah terurut.

ALGORITMA QUICK SORT

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)
```

ALGORITMA QUICK SORT

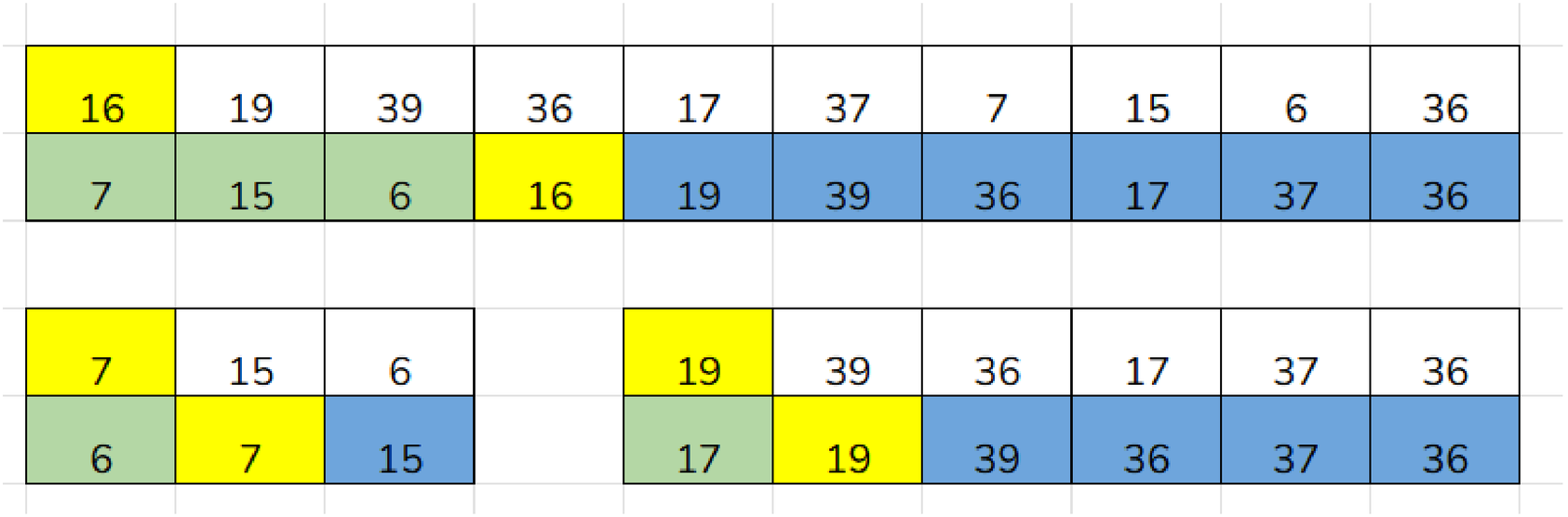
16	19	39	36	17	37	7	15	6	36
7	15	6	16	19	39	36	17	37	36

Pilih pivot, sebagai contoh pada elemen pertama.

Lalu posisikan elemen yang < pivot dan > pivot.

Sebagai contoh elemen berwarna hijau sebagai < pivot dan warna biru sebagai > pivot.

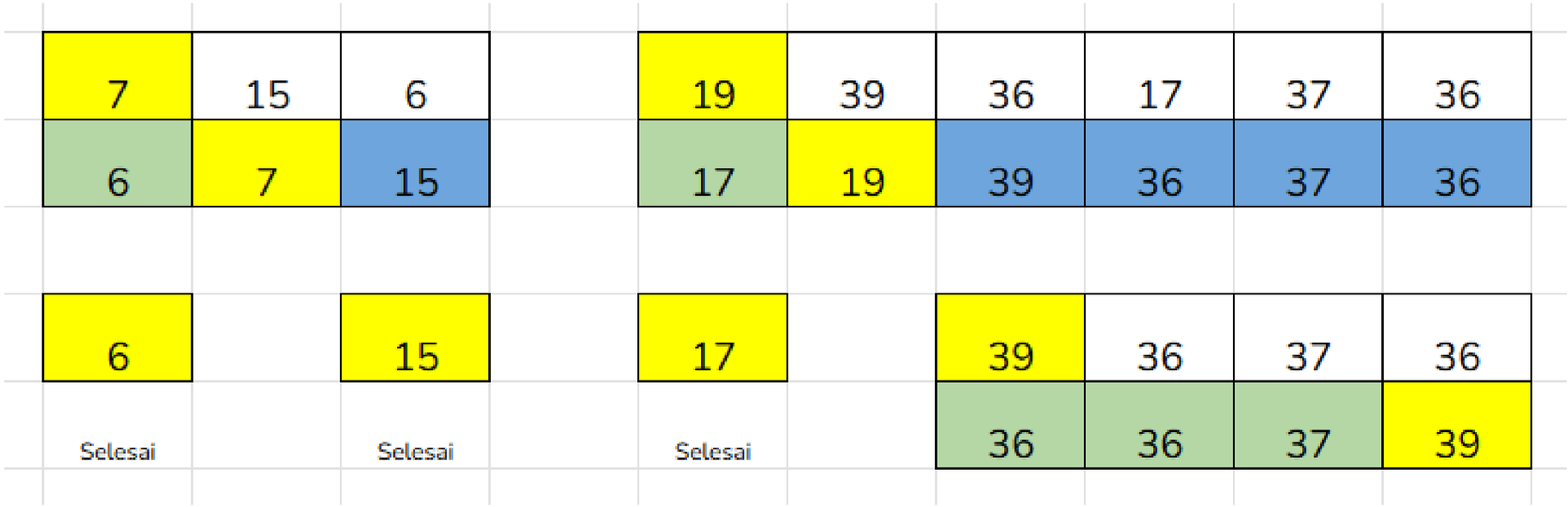
ALGORITMA QUICK SORT



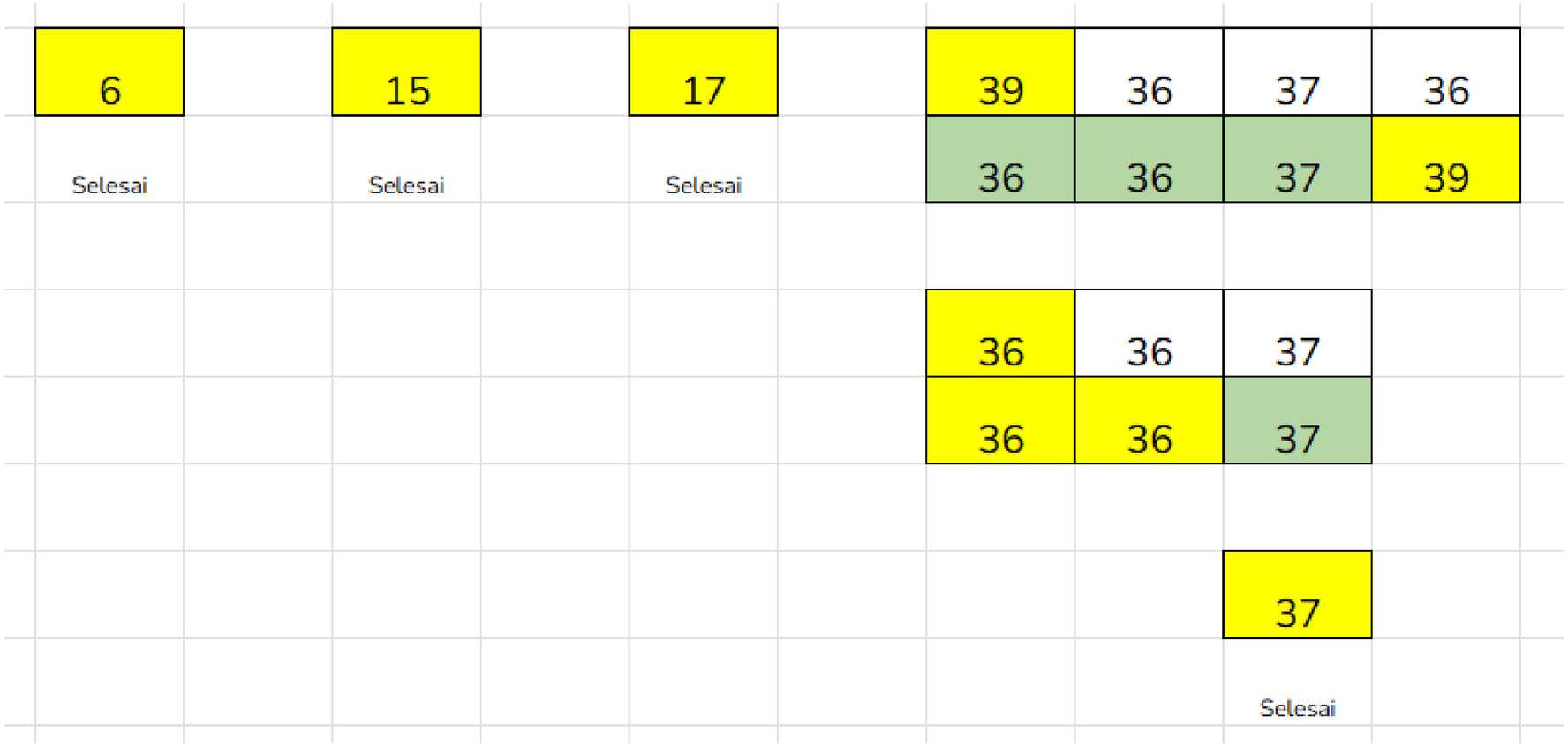
Untuk elemen < pivot dan > pivot lakukan rekursif dan pilih pivot unttuk masing-masing array.

Lakukan langkah yang sama terus menerus sampai tersisa satu elemen.

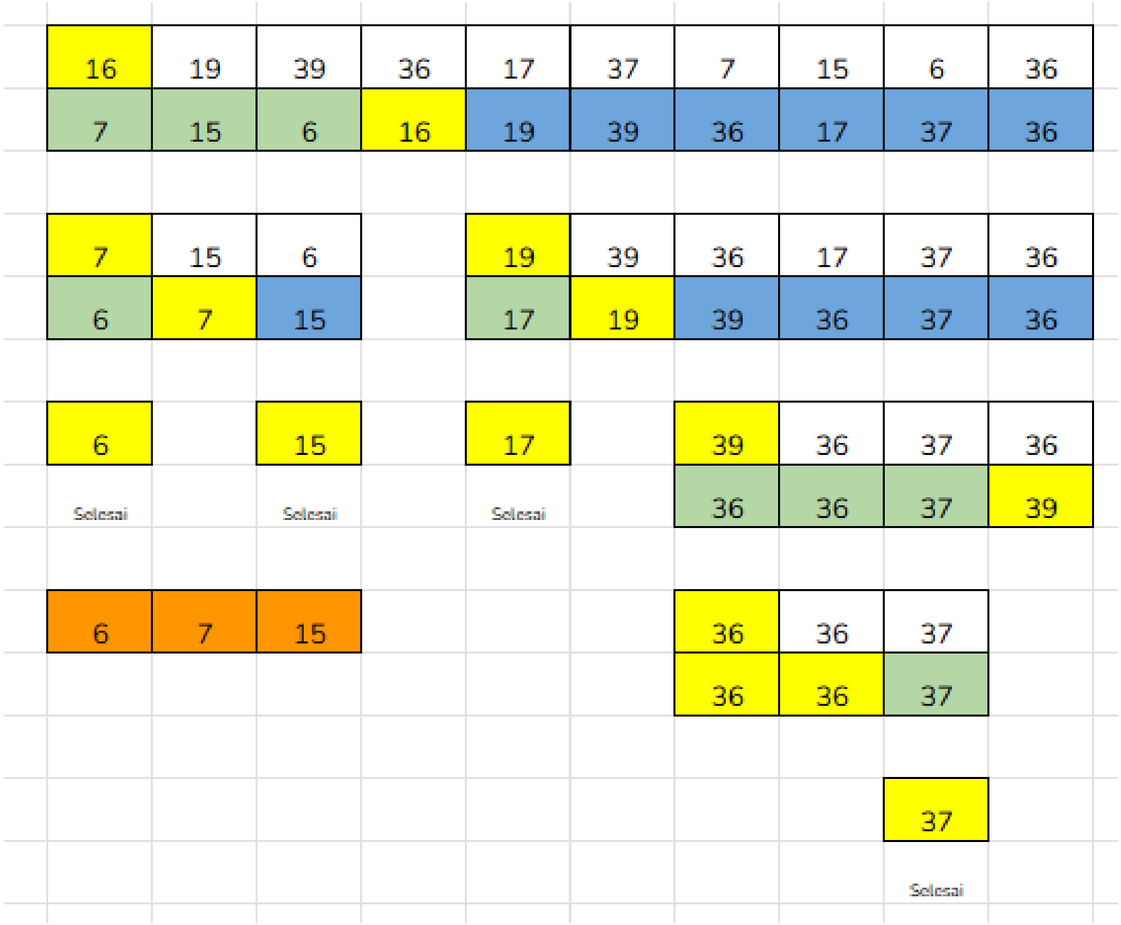
ALGORITMA QUICK SORT



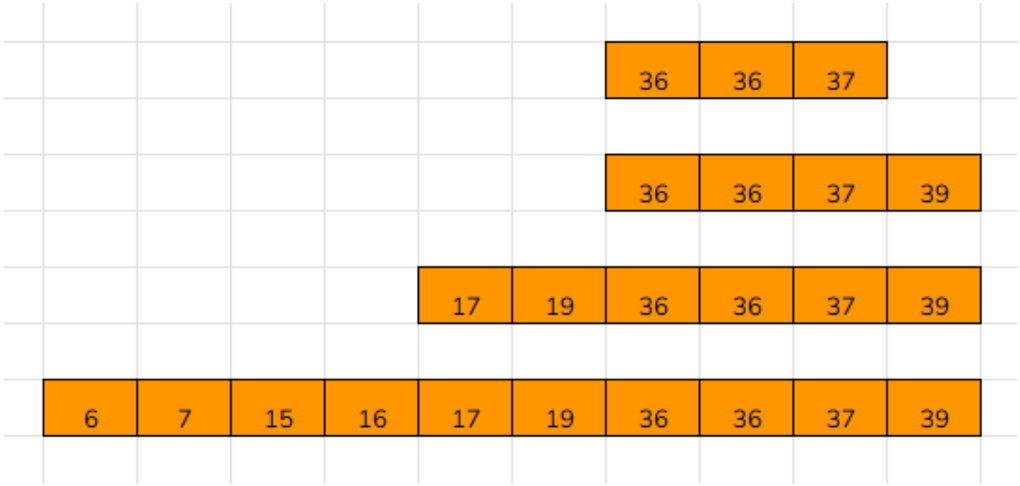
ALGORITMA QUICK SORT



ALGORITMA QUICK SORT



Setelah semua proses divide selesai, lakukan penggabungan dari < pivot, = pivot, dan > pivot.

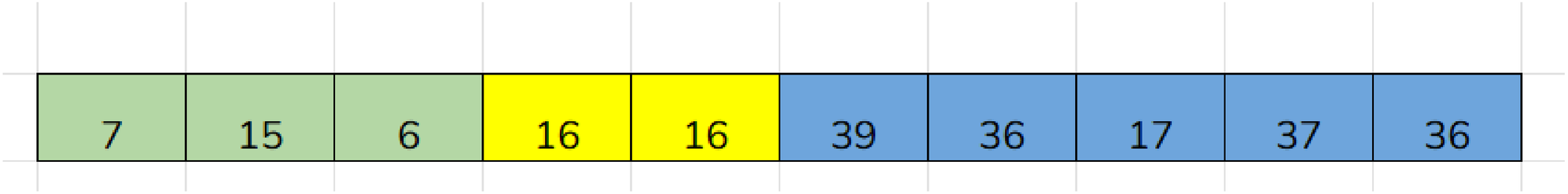


TOM SEDANG LAPAR

```python
def tom_sedang_lapar(arr, k):
    if len(arr) == 1:
        return arr[0]

    pivot = arr[0]
    len_pivot = len([x for x in arr if x == pivot])
    kiri = [x for x in arr if x < pivot]
    kanan = [x for x in arr if x > pivot]

    if k <= len(kiri):
        return tom_sedang_lapar(kiri, k)
    elif len(kiri)+1 <= k <= len(kiri)+len_pivot:
        return pivot
    else:
        return tom_sedang_lapar(kanan, k-len(kiri)-len_pivot)

n, k = map(int, input().split())
arr = [int(x) for x in input().split()]
print(tom_sedang_lapar(arr, n-k+1))
```

TOM SEDANG LAPAR

Cara yang digunakan serupa dengan algoritma Quick Sort, biasa dinamakan Quick Select. Untuk mempermudah akan menggunakan algoritma mencari nilai terkecil ke-k alih-alih mencari nilai terbesar ke-k.



Sebagai contoh untuk array ini

- Jika k <= 3, maka bilangan urutan ke-k pasti akan berada di kiri
- Jika 4 <= k <= 5, maka bilangan urutan ke-k pasti pivot itu sendiri
- Jika k >= 6, maka bilangan urutan ke-k pasti berada di kanan.

Perhatikan jika mengambil elemen di kanan untuk melakukan reset index (lihat contoh kode)

THANK YOU

FOR YOUR NICE ATTENTION

link canva: https://www.canva.com/design/DAGgzEmWcr4/w1SLJkahg0kE_aaVctk0bw/edit?utm_content=DAGgzEmWcr4&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton