Merge Sort

Algoritma Pemrograman 2

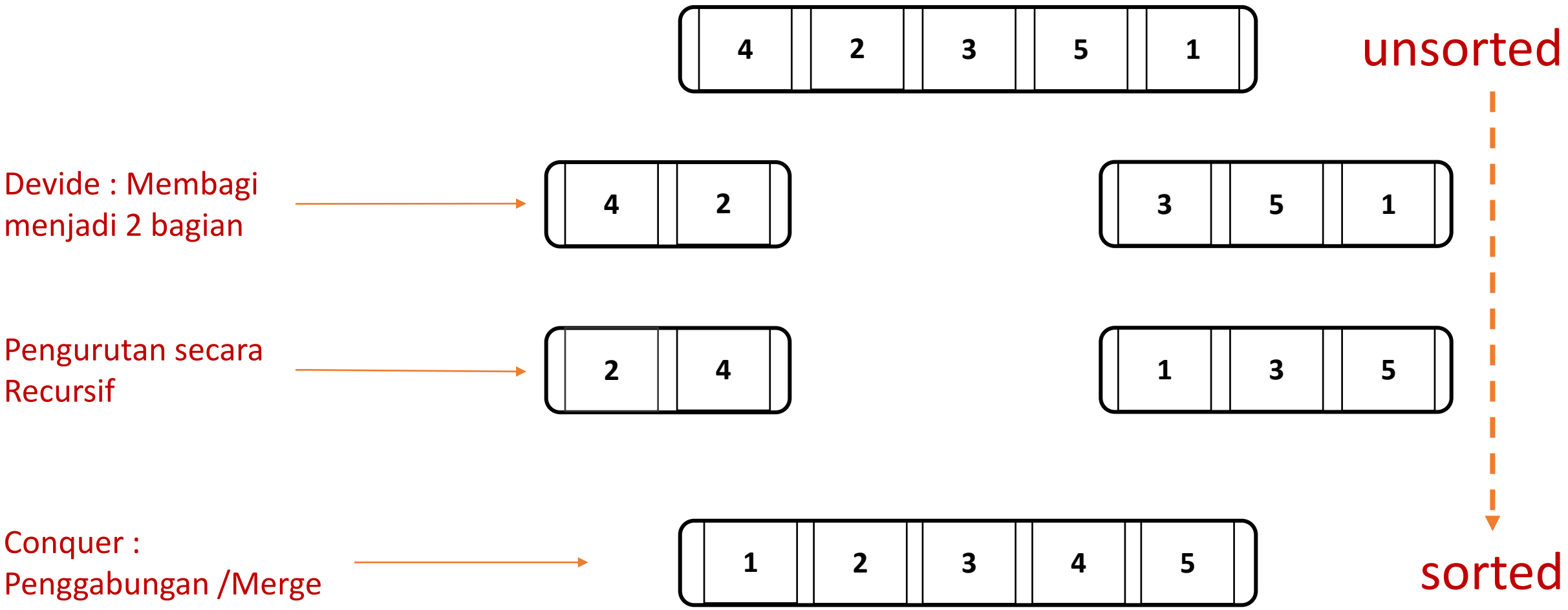
Program Studi Teknik Informatika

Institut Teknologi Sumatera

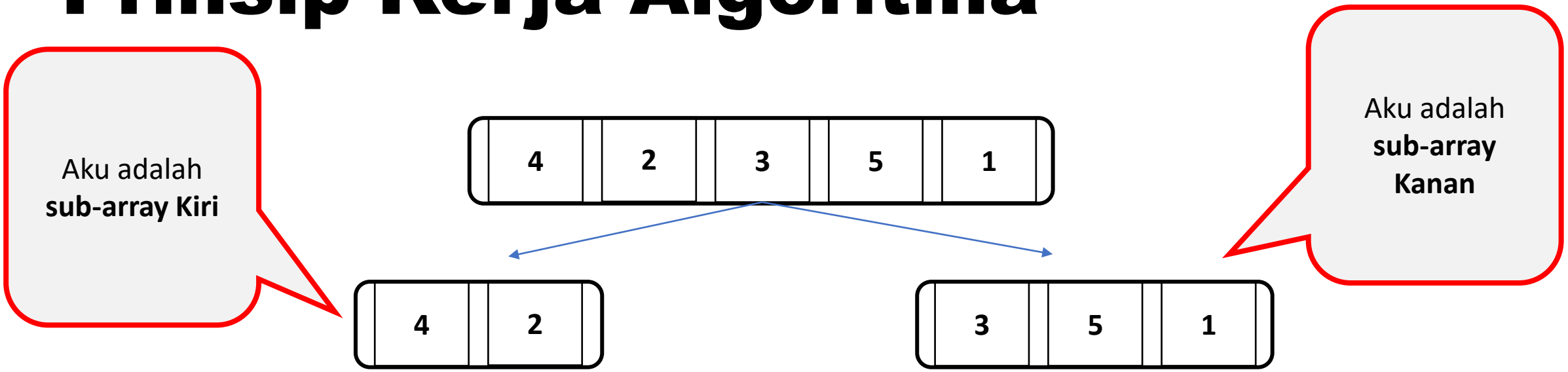
Merge Sort

- **Merge sort** adalah sort yang dilakukan dengan teknik merge (menggabungkan)
- **Algoritma merge sort** membagi array menjadi sub-array yang ukurannya sama besar. Masing-masing sub-array diurutkan secara rekursif (***divide***), dan kemudian digabungkan kembali untuk membentuk array yang terurut (***conquer***)
- Divide step
  - Bagilah array menjadi sub-array (idealnya berukuran hampir sama)
  - Urutkan kedua bagian secara rekursif
- Conquer step
  - Gabungkan kedua bagian untuk membentuk array yang diurutkan

Merge Sort : Ilustrasi

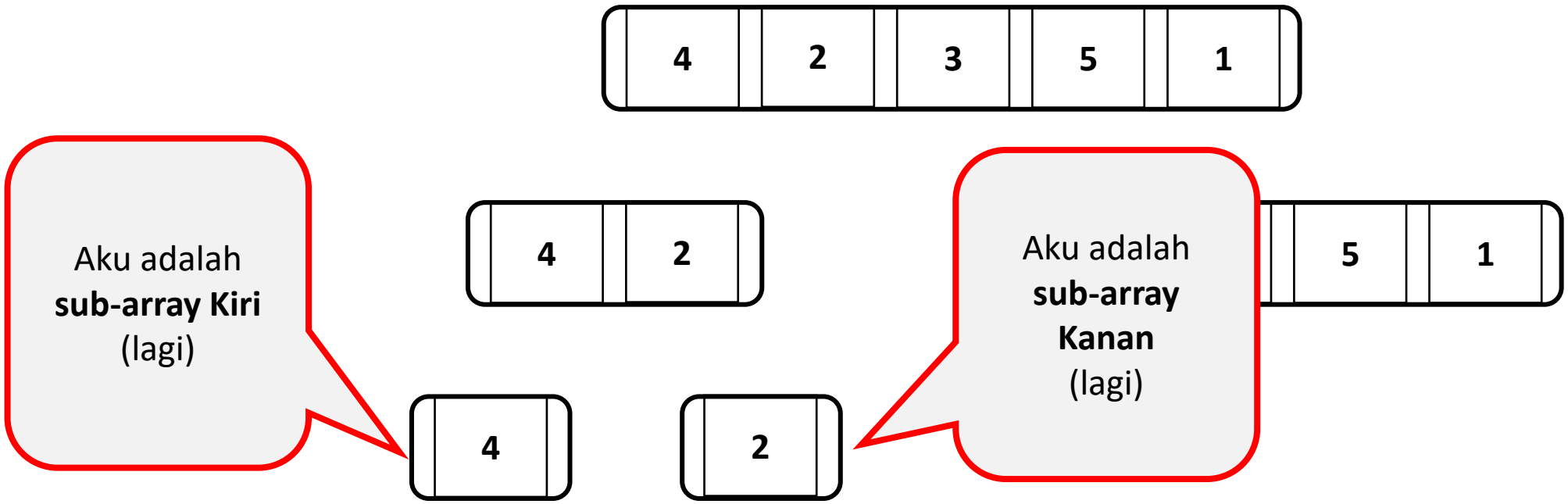


Prinsip Kerja Algoritma



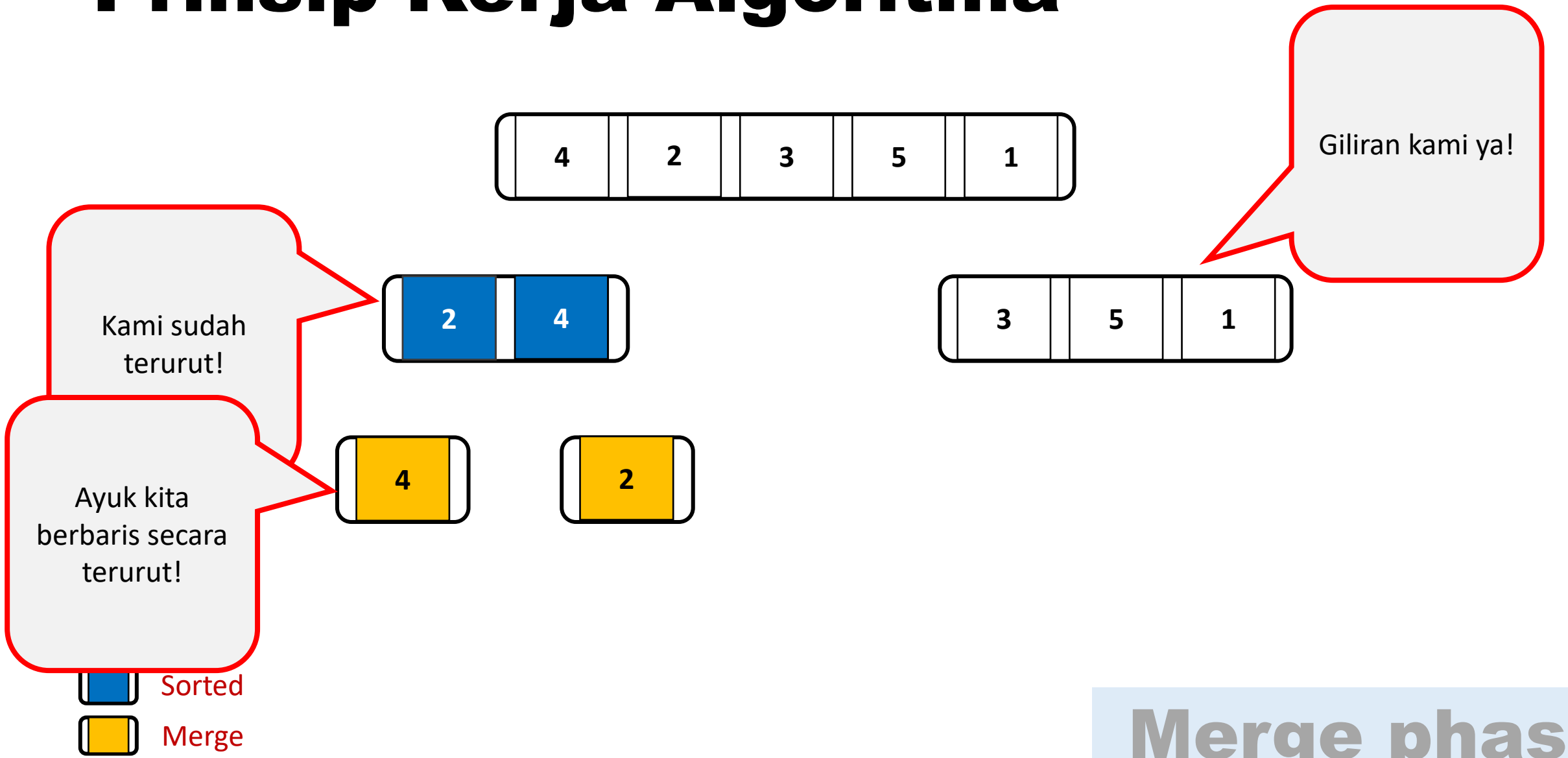
Split phase

Prinsip Kerja Algoritma



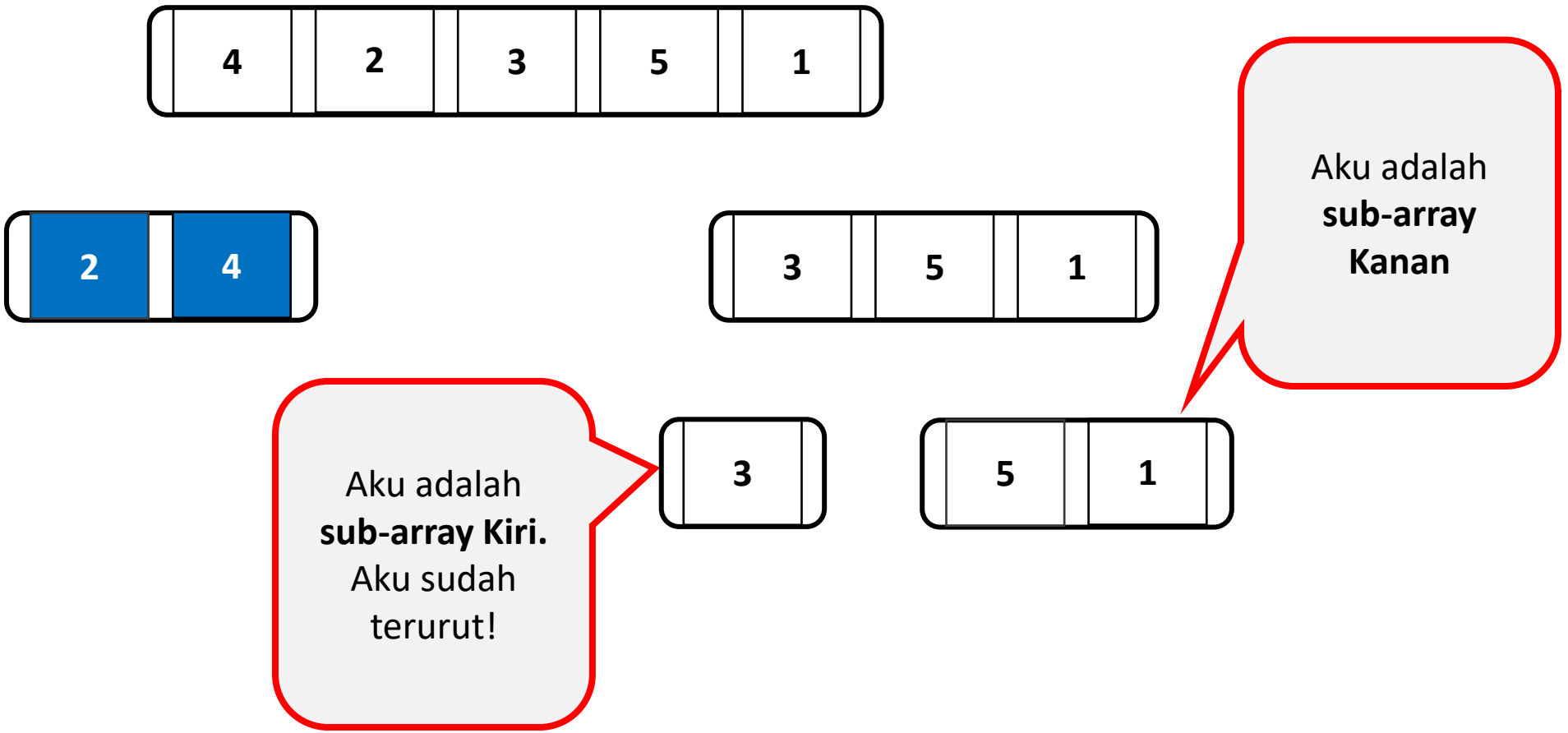
Split phase

Prinsip Kerja Algoritma

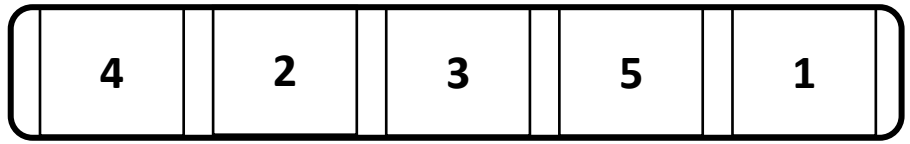


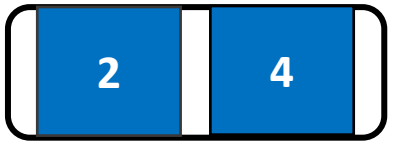
Merge phase

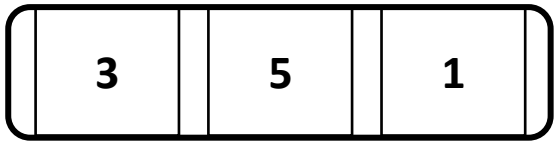
Prinsip Kerja Algoritma

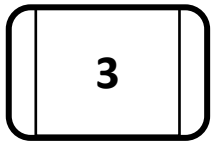


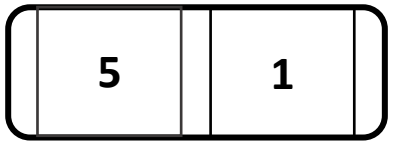
Prinsip Kerja Algoritma

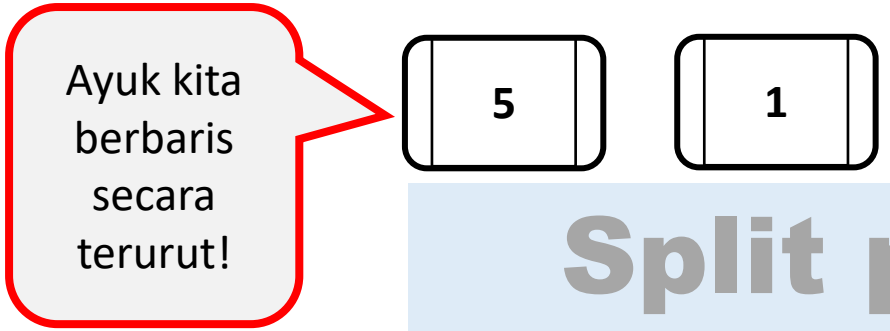


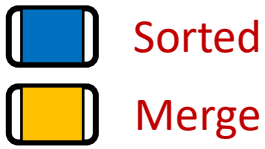






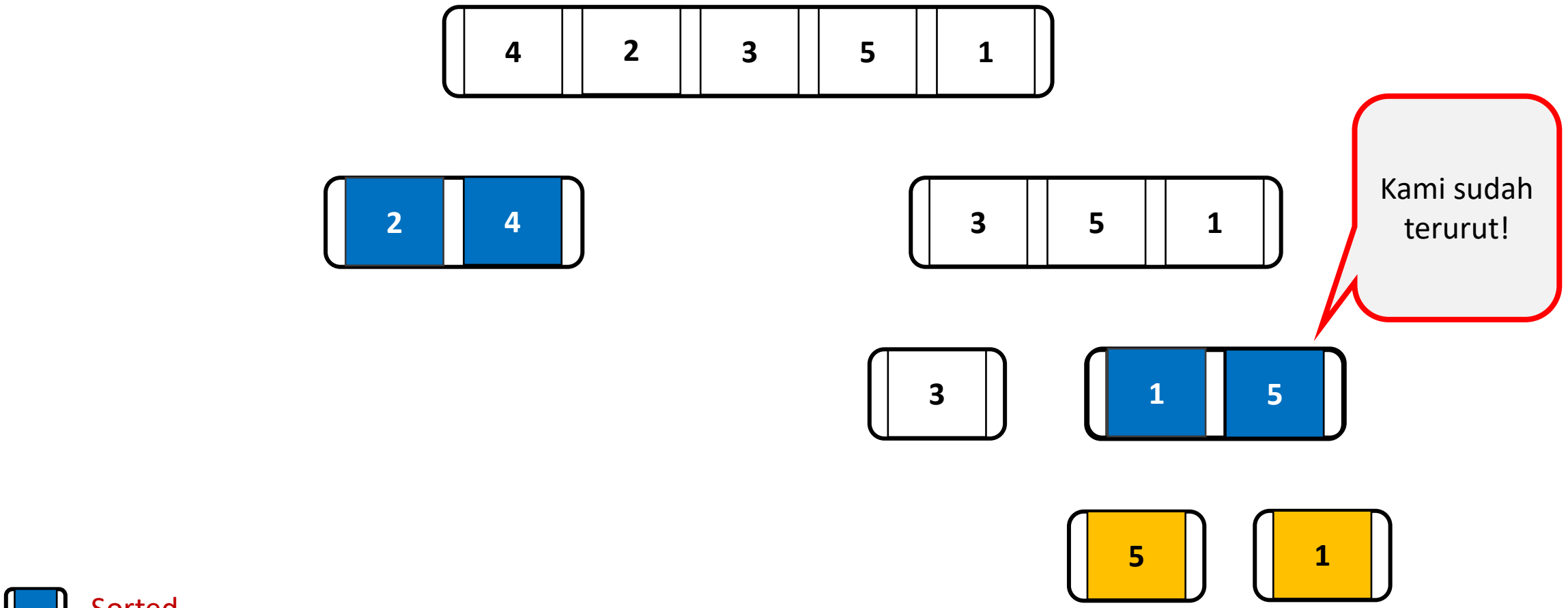




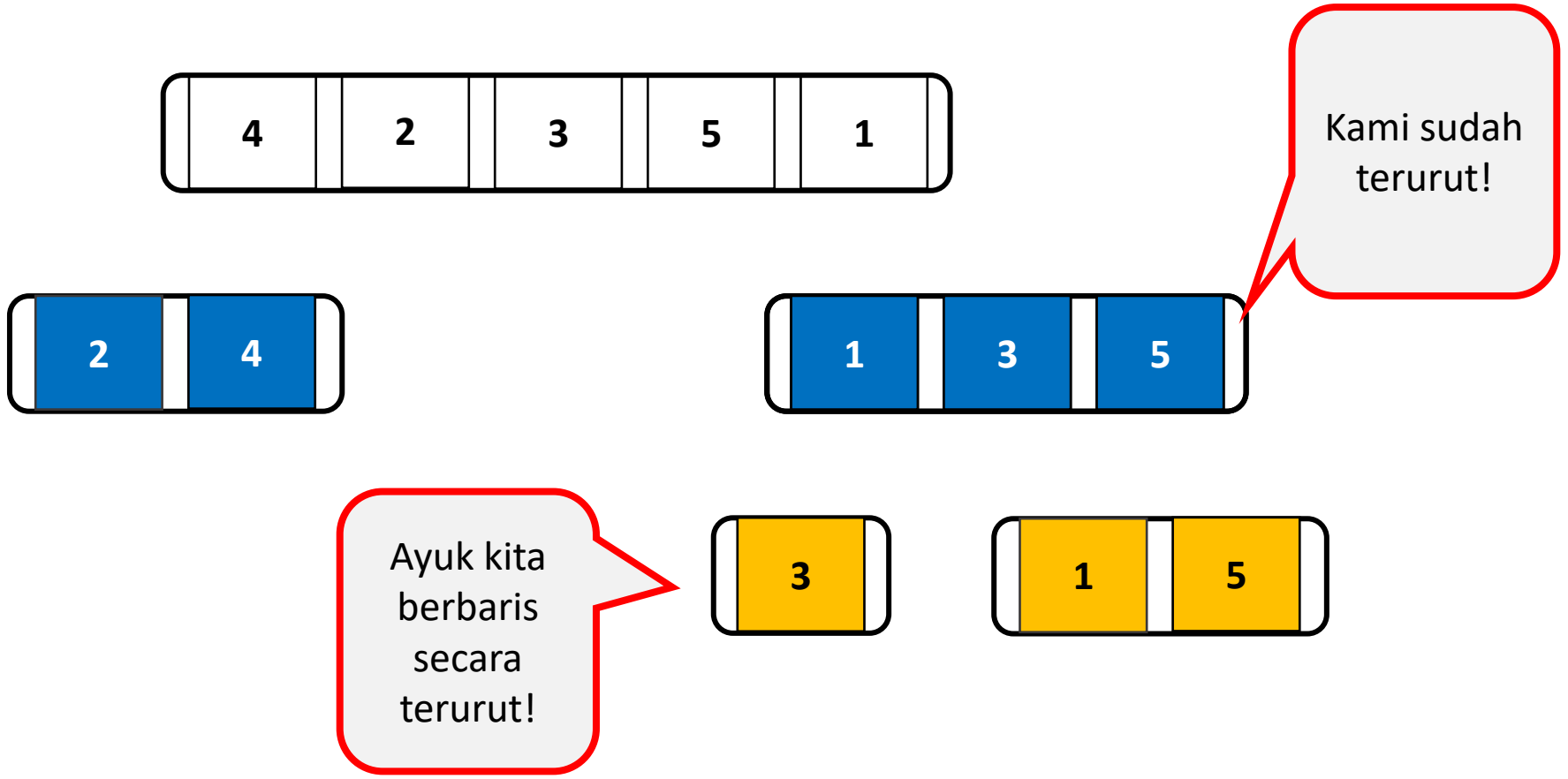


Split phase

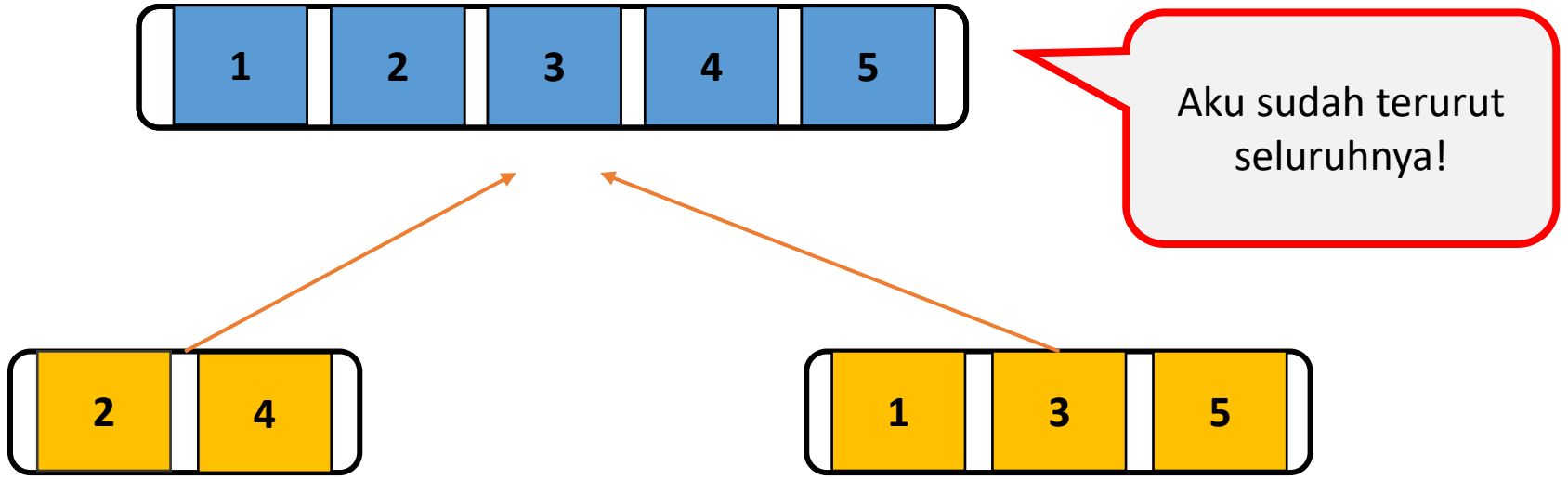
Prinsip Kerja Algoritma



Prinsip Kerja Algoritma



Prinsip Kerja Algoritma



Sorted

Merge

Merge phase

Merge Sort : Implementasi

```
void mergeSort(int array[ ], int awal, int akhir)
 {
        if (awal<akhir)     {
                int tengah = (awal+akhir) / 2 ;
                // Membagi menjadi sub-array dan pengurutan secara rekrusif
                mergeSort(array, awal, tengah) ;
                mergeSort(array, tengah+1, akhir) ;
                // menggabungkan sub-array yang telah diurutkan
                merge(array, awal, tengah, akhir) ;       }
 }
```

Kelebihan Vs Kekurangan Merge Sort

- Dibandingkan dengan algoritma lain, merge sort ini termasuk algoritma yang sangat efisien dalam penggunaannya sebab setiap list selalu dibagi-bagi menjadi list yang lebih kecil, kemudian digabungkan lagi sehingga tidak perlu melakukan banyak perbandingan

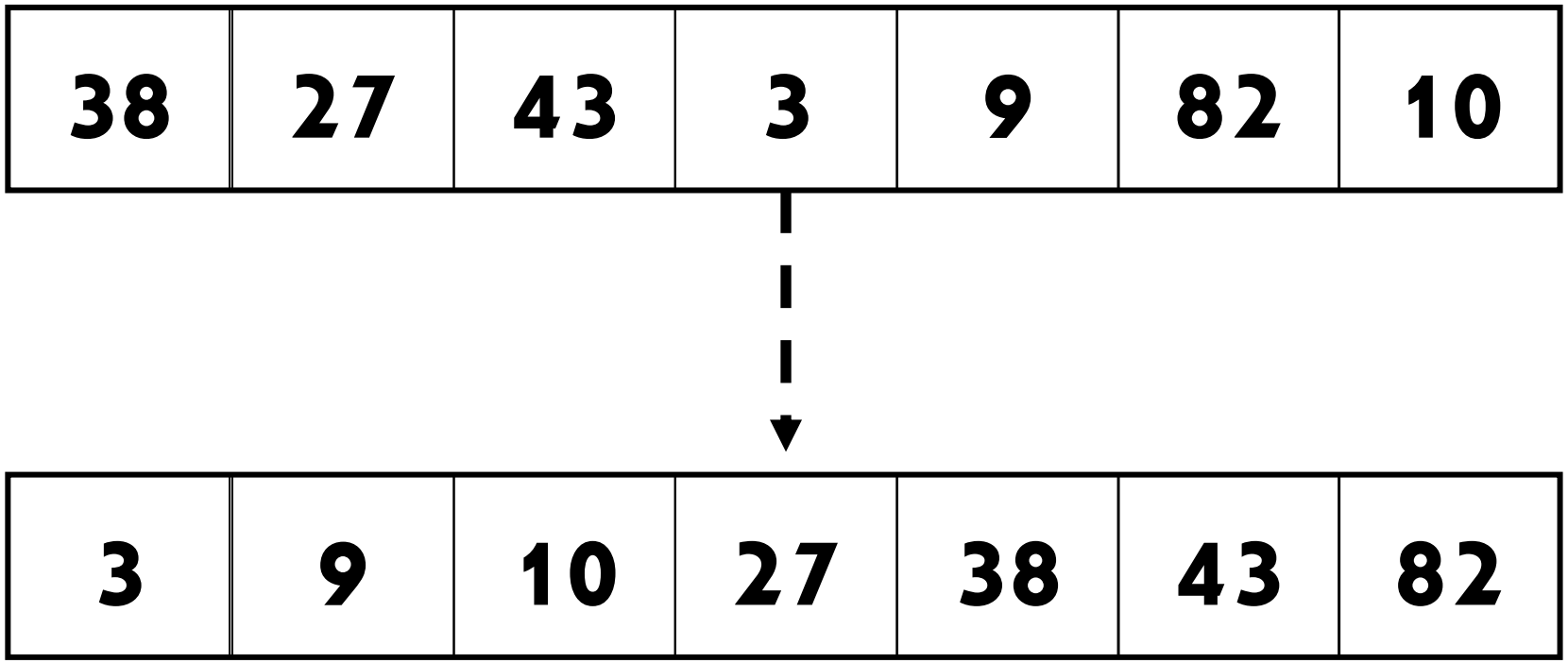
- Kelemahan utama merge sort adalah algoritma ini mebutuhkan setidaknya ruang atau memori dua kali lebih besar karena dilakukan secara rekrusif dan memakai dua elemen terpisah

SEKIAN

Selamat Belajar! ☺

Tugas

- Lakukan simulasi pengurutan elemen pada *array* berikut dengan menggunakan algoritma ***merge-sort***.



Tugas

- Berdasarkan prinsip kerja yang telah disampaikan, Implementasikan algoritma ***merge-sort*** dengan menggunakan bahasa C++!

```c
/ * l = indeks kiri dan r = indeks kanan dari
sub-array yang akan disortir * /
void mergeSort(int *data, int l, int r) {
  if (l < r) {
    int m = (l + (r)) / 2;

    // Split first and second halves
    mergeSort(data, l, m);
    mergeSort(data, m + 1, r);

    // Finally merge first and second halves
    merge(data, l, m, r);
  }
}
```

```c
void merge(int *data, int l, int m, int r) {
  int i, j, k;
  int n1 = m - l + 1;
  int n2 = r - m;

  /* create temp data arrays */
  int L[n1], R[n2];

  /* Copy data to temp data arrays L[] and R[] */
  for (i = 0; i < n1; i++)
    L[i] = data[l + i];
  for (j = 0; j < n2; j++)
    R[j] = data[m + 1 + j];

  /* Merge the temp data arrays back into data[l..r]*/
  i = 0; // Initial index of first subarray
  j = 0; // Initial index of second subarray
  k = l; // Initial index of merged subarray
```

```c
while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
      data[k] = L[i];
      i++;
    } else {
      data[k] = R[j];
      j++;
    }
    k++;
  }

  /* Copy the remaining elements of L[], if there are any */
  while (i < n1) {
    data[k] = L[i];
    i++;
    k++;
  }

  /* Copy the remaining elements of R[], if there are any */
  while (j < n2) {
    data[k] = R[j];
    j++;
    k++;
  }
}
```