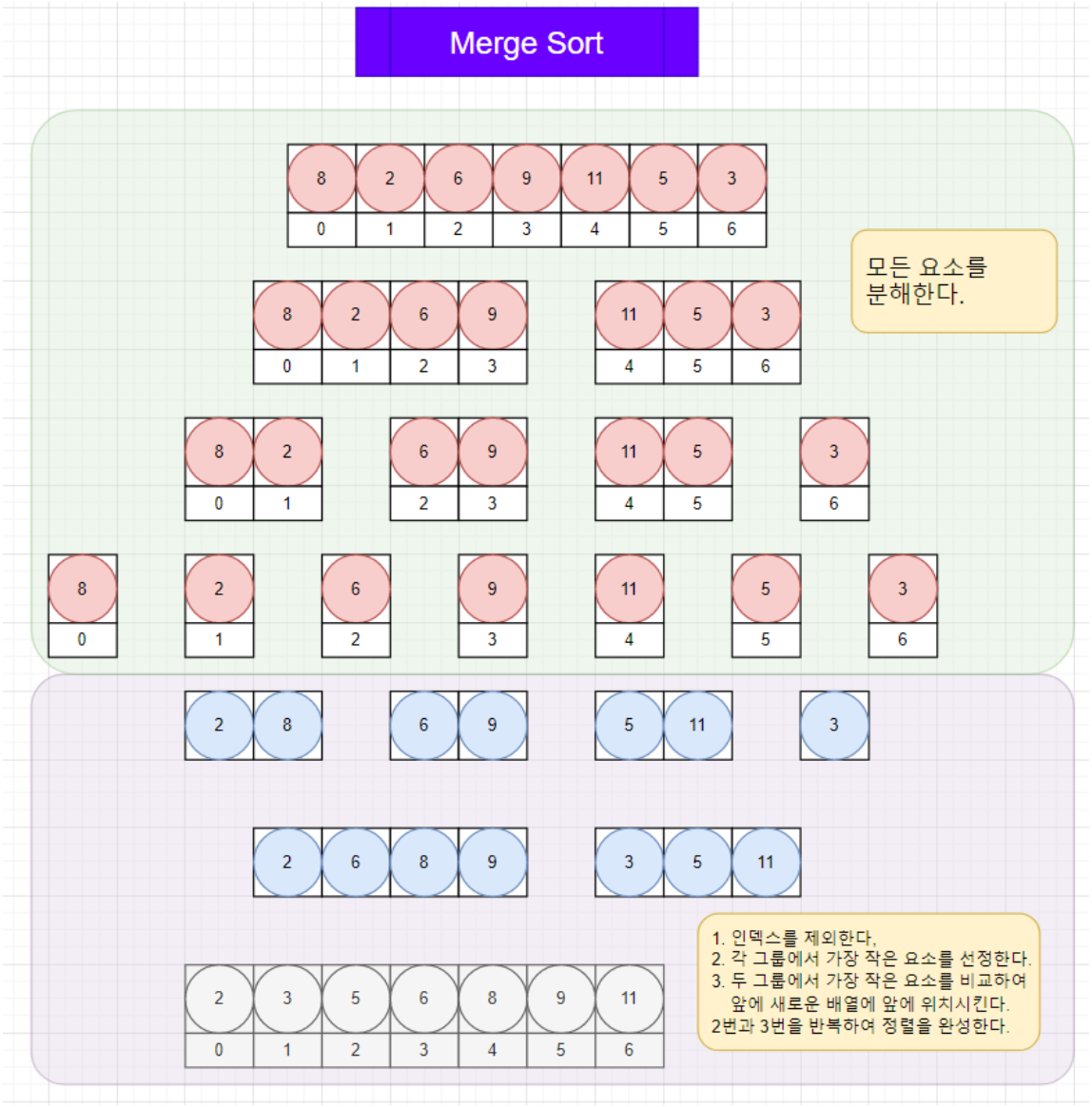
Merge Sort

▼ 병합정렬_Merge Sort

> 👉 분할 정복(Divide and Conquer)을 사용하는 방법이다. 분할 정복은 주어진 문제를 해결하기 쉬운 단계까지 분할 한 후에 분할된 문제를 해결하고 그 결과를 다시 결합하는 알고리즘이다.

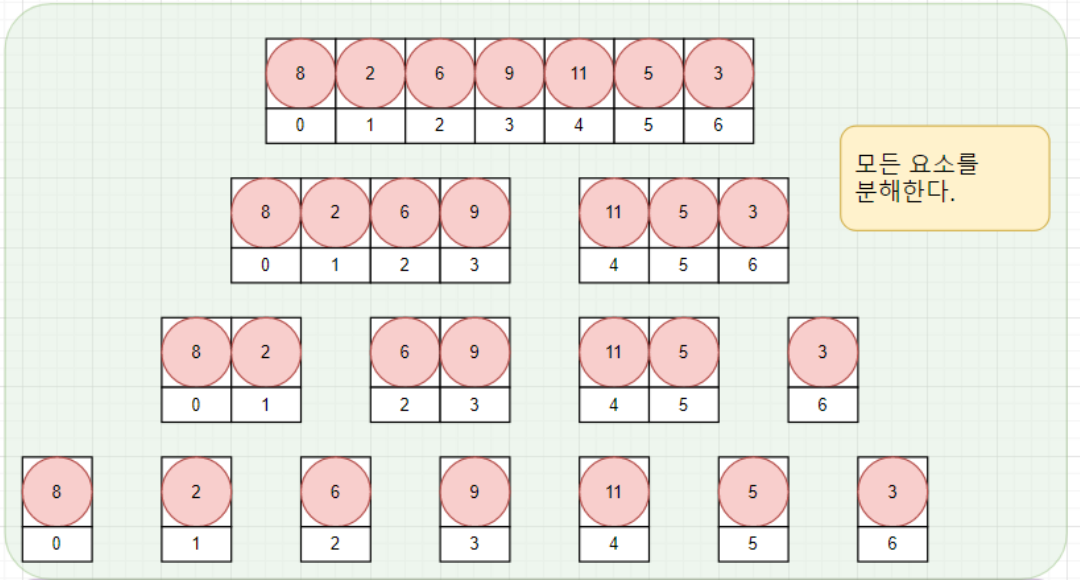
▼ Process



▼ Algorithm

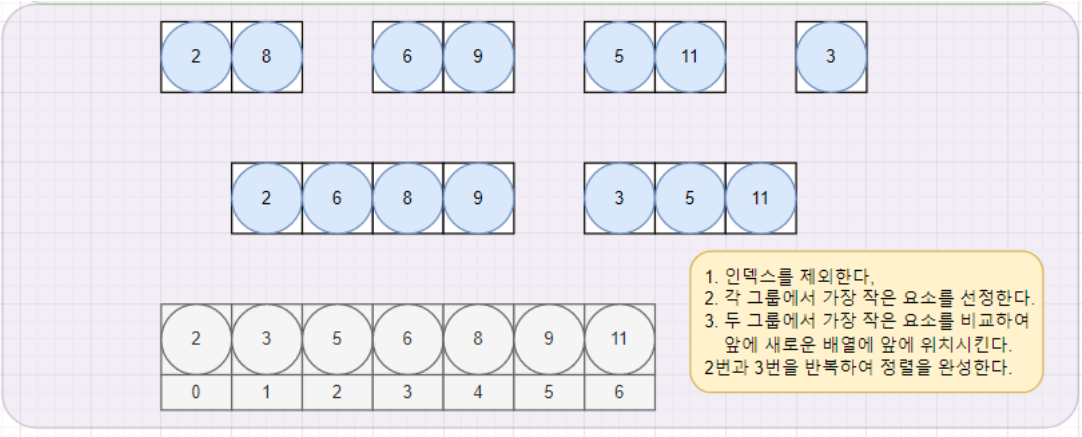
▼ (1) 데이터를 분해한다.

> 👉 정렬을 생각하지 않고 1개씩 될 때까지 나눈다. 나누는 방법은 배열을 반으로 분해하는 과정을 반복한다. 위에서는 7개의 데이터를 분해하기 위해 3번의 진행하였다.



▼ (2) 비교후 결합한다.

👉 분해가 완료된 후 데이터를 비교하면서 결합한다.
1. 제일 앞에 있는 2와 8을 다시 합치는데, 작은 수가 2가 앞으로 큰수 8이 뒤로 보낸 결합 상태가 된다.
2. 이때 각각의 그룹에서 먼저 제일 첫번째 값을 비교하여 작은 값은 추출한다.
3. 그 다음 다시 한번 각각의 그룹의 제일 왼쪽 즉 작은 값을 또 다시 비교하여 둘 줄 작은 값을 다시 추출한다.
4. 위의 과정을 반복한다.



▼ Java Code

```
package Merge_Sort;

public class MergeSort {
  public static int[] arr;
  public static int[] tmp;

  public static void main(String[] args) {
      arr = new int[]{1, 9, 8, 5, 4, 2, 3, 7, 6};
      tmp = new int[arr.length];
      printArray(arr);
      mergeSort(0, arr.length-1);
      printArray(arr);
  }

  public static void mergeSort(int start, int end) {
      if (start<end) {
          int mid = (start+end) / 2;
          mergeSort(start, mid);
          mergeSort(mid+1, end);
          int p = start;
          int q = mid + 1;
          int idx = p;

          while (p<=mid || q<=end) {
              if (q>end || (p<=mid && arr[p]<arr[q])) {
                  tmp[idx++] = arr[p++];
              } else {
                  tmp[idx++] = arr[q++];
              }
          }

          for (int i=start;i<=end;i++) {
              arr[i]=tmp[i];
          }
      }
  }

  public static void printArray(int[] a) {
      for (int i=0;i<a.length;i++)
          System.out.print(a[i]+" ");
      System.out.println();
  }
}
```