운영체제론 실습 9주차

CPS LAB

다중 스레드 정렬 응용

CHAPTER 1.
합병 정렬 알고리즘

- 분할 정복 알고리즘
- 합병 정렬 예시
- 합병 정렬 예제 실습

합병 정렬 (Merge Sort)

- **정렬이란?**
  - 순서 없이 나열된 자료(record)를 특정한 **키(key) 값**에 따라
    오름차순/내림차순으로 **재배열**하는 것을 의미함

- **정렬을 효율적으로 해야 하는 이유는?**
  - 정렬 알고리즘에 따른 **시간 복잡도**가 달라짐.
  - 즉, 똑같은 데이터를 정렬하는데 많은 시간 차이를 보이기도 함 → **O(n), O(nlogn), O(n*n)**

- **어떻게 효율적으로 정렬할까?**
  - **Divide & Conquer (분할정복)** 알고리즘
    - 대표적으로 **합병 정렬**과 **쾌속 정렬(Quick Sort)**이 있음

- **어떻게 하면 쉽게 구현할까?**
  - **Recursive function (재귀함수)** 사용

시간 복잡도

Example

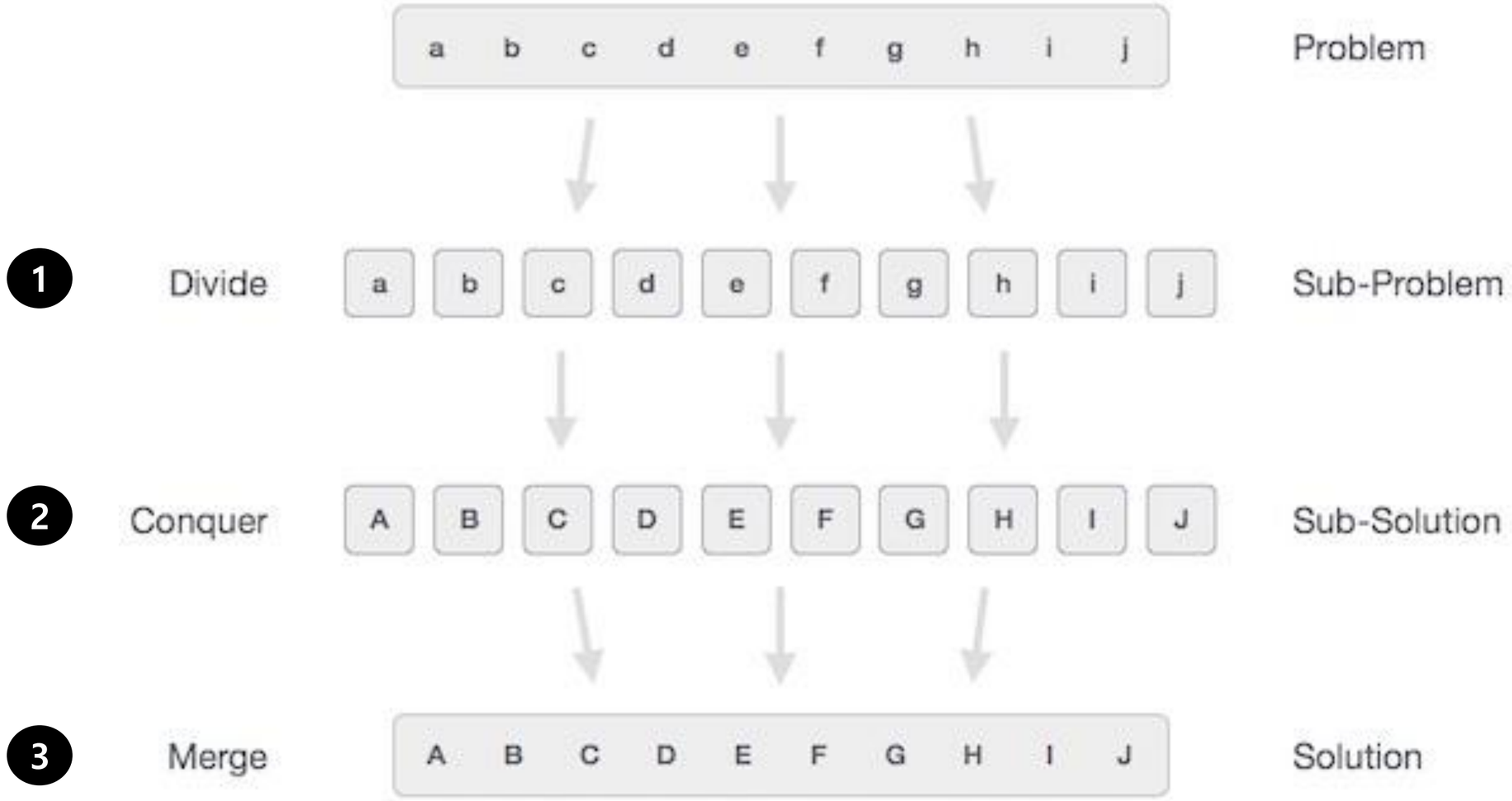
- O(n) = 24시간 (1일)
- O(n log(n)) = 1일 9시간
- O(n^2) = 24일

※ 표기법

- Big-Omega (Ω) : 최선의 경우
- Big-Theta (θ) : 평균 시간 복잡도
- Big-O (O) : 최악의 경우

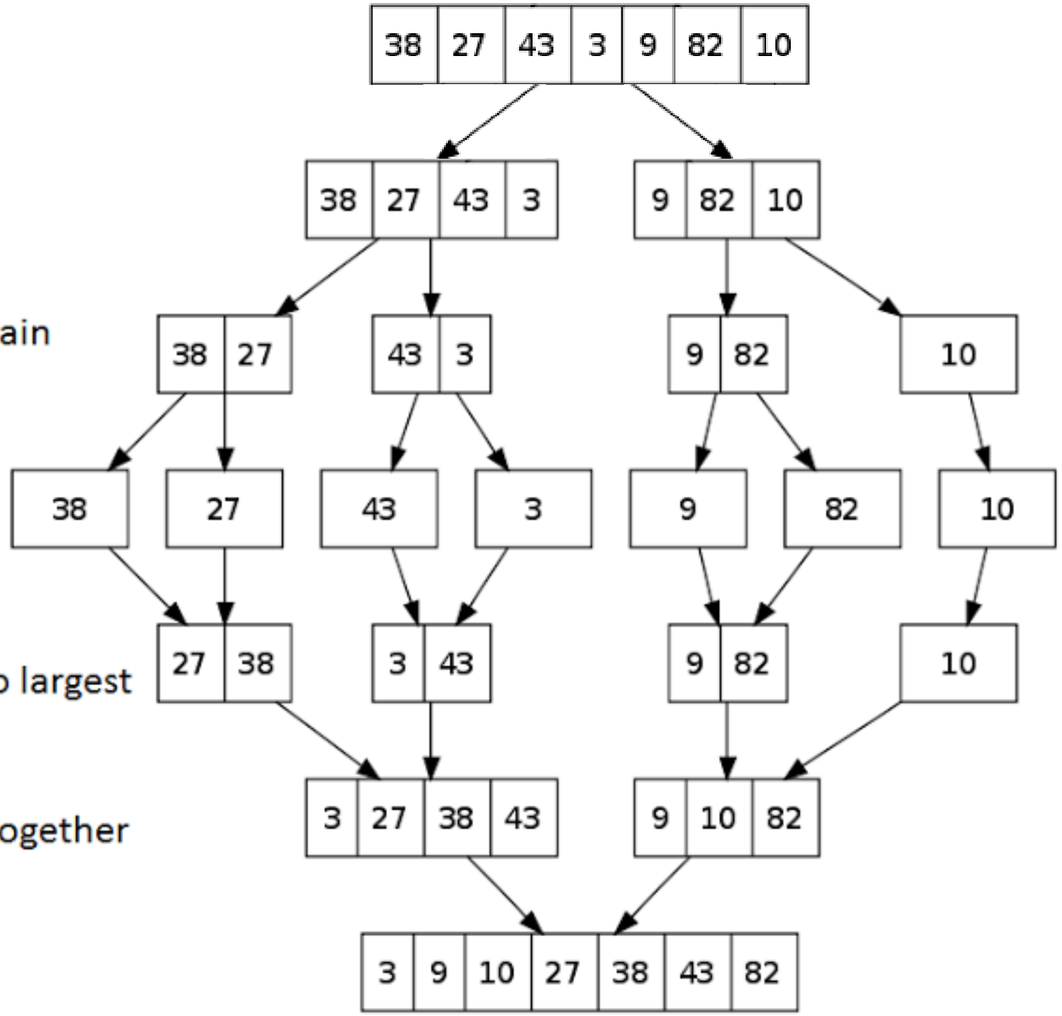
Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

분할 정복 알고리즘 (Divide & Conquer)



합병 정렬 예시

1. Divide the array into two parts
2. Divide the array into two parts again
3. Break each element into single parts
4. Sort the elements from smallest to largest
5. Merge the divided sorted arrays together
6. The array has been sorted



예제) Merge Sort

week9/1. merge_sort/merge_sort.c

```c
#include <stdio.h>
#define DATA_SIZE 8

void mergeSort(int data[], int p, int r);
void merge(int data[], int p, int q, int r);

int main() {
  int data[DATA_SIZE] = {5, 2, 4, 7, 1, 3, 2, 6};
  mergeSort(data, 0, DATA_SIZE-1);
        int i;
  for (i = 0; i < DATA_SIZE; i++)
    printf("%d ", data[i]);
        printf("\n");
  return 0;
}
```

재귀 함수

```c
void mergeSort(int data[], int p, int r) {
  int q;
  if (p < r) {
    q = (p + r) / 2;
    mergeSort(data, p, q);
    mergeSort(data, q + 1, r);
    merge(data, p, q, r);
  }
}
```

```c
void merge(int data[], int p, int q, int r) {
  int i = p, j = q + 1, k = p;
  int tmp[DATA_SIZE];
  while (i <= q && j <= r) {
    if (data[i] <= data[j])
      tmp[k++] = data[i++];
    else
      tmp[k++] = data[j++];
  }
  while (i <= q)
    tmp[k++] = data[i++];
  while (j <= r)
    tmp[k++] = data[j++];
  for (int a = p; a <= r; a++)
    data[a] = tmp[a];
}
```

1. Divide
2. Conquer
3. Merge

CHAPTER 2.
다중 스레드 정렬 응용

- pthread 함수 사용법
- 프로젝트 흐름
- 프로젝트 실습

pthread_create()

```
#include <pthread.h>
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                                      void *(*start_routine) (void *), void *arg);
```

- **새로운 thread를 생성한다.**
  1) ***thread*** : 생성이 성공한 경우 생성된 thread의 ID가 저장됨
  2) ***attr*** : thread 속성 (본 실습 시간에는 NULL 사용)
  3) ***start_routine*** : thread가 수행할 함수명
  4) ***arg*** : 수행할 함수에 전달할 인자의 포인터 (주소값)
- **사용 예제** *(각 인자를 이해시키기 위한 예제이므로 작동하지 않음)*

```
pthread_t tid;
void *some_function(void *data){
      …
}
struct some_struct arg;
arg.index = 0;
arg.data = data;
pthread_create(&tid, NULL, some_function, &arg);
```

pthread_exit()

```
#include <pthread.h>
void pthread_exit(void *retval);
```

- **thread가 자신을 종료 시킬 때 사용한다.** (thread가 호출하는 함수 내에 사용)
  1) ***retval*** : thread의 반환 값을 넘겨주기 위함 **(pthread_join의 retval 공간에 쓰기)**

- **사용 예제** *(각 인자를 이해시키기 위한 예제이므로 작동하지 않음)*

```
void *some_function(void *data){

      …

      pthread_exit(0);
}
```

pthread_join()

```
#include <pthread.h>
int pthread_join(pthread_t thread, void **retval);
```

- **thread의 종료를 기다린다.**
  1) ***thread*** : 해당 ID를 가진 thread를 기다림
  2) ***retval*** : thread의 반환 값을 가져오기 위함 **(pthread_exit의 retval 값을 가져옴)**

- **사용 예제** *(각 인자를 이해시키기 위한 예제이므로 작동하지 않음)*

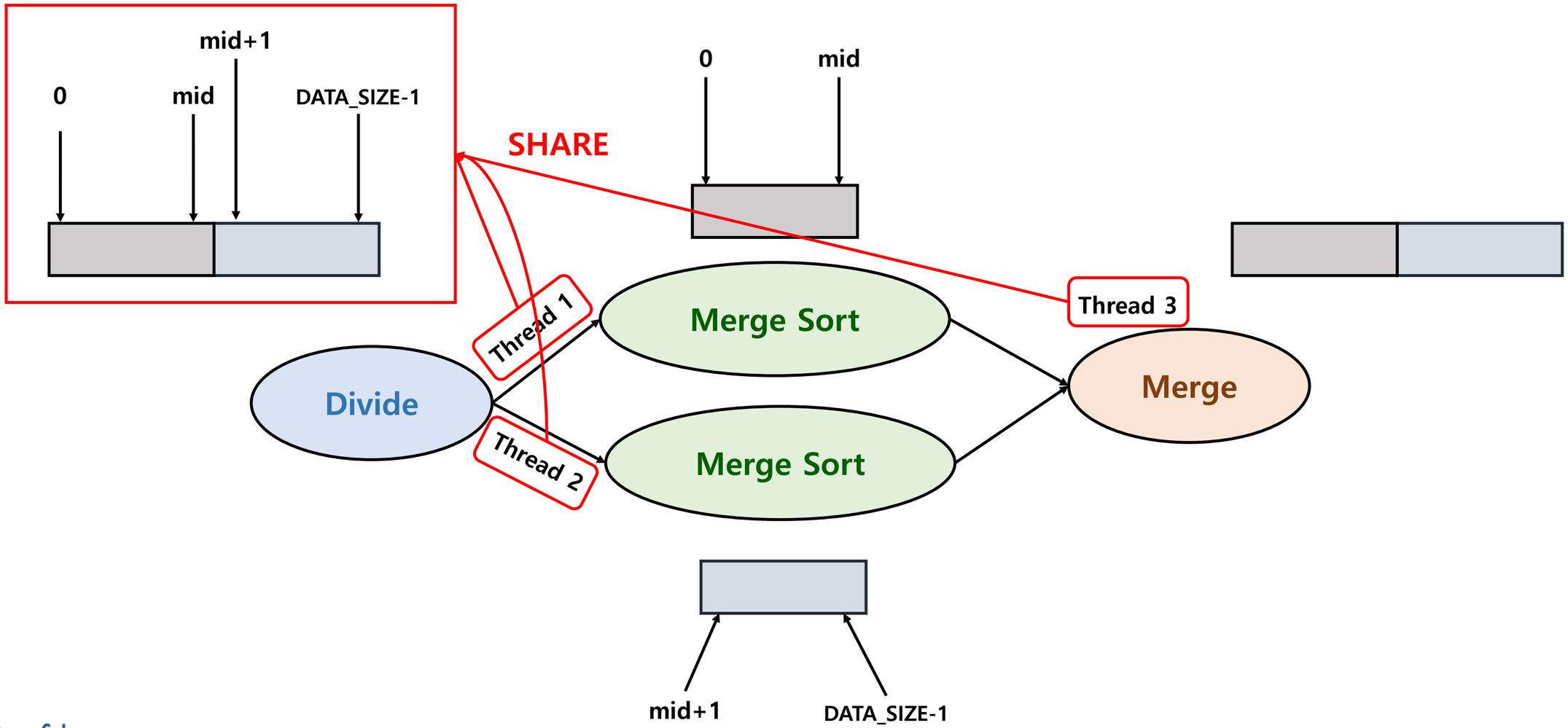
```
pthread_t tid;

      …
/* thread가 생성되고 종료됨 */
      …

pthread_join(tid, NULL);
```

Multi-threaded 프로그래밍 시 고려사항

Operating System Concepts 9th p.165 4.2.1, 10th p.162 4.2.1

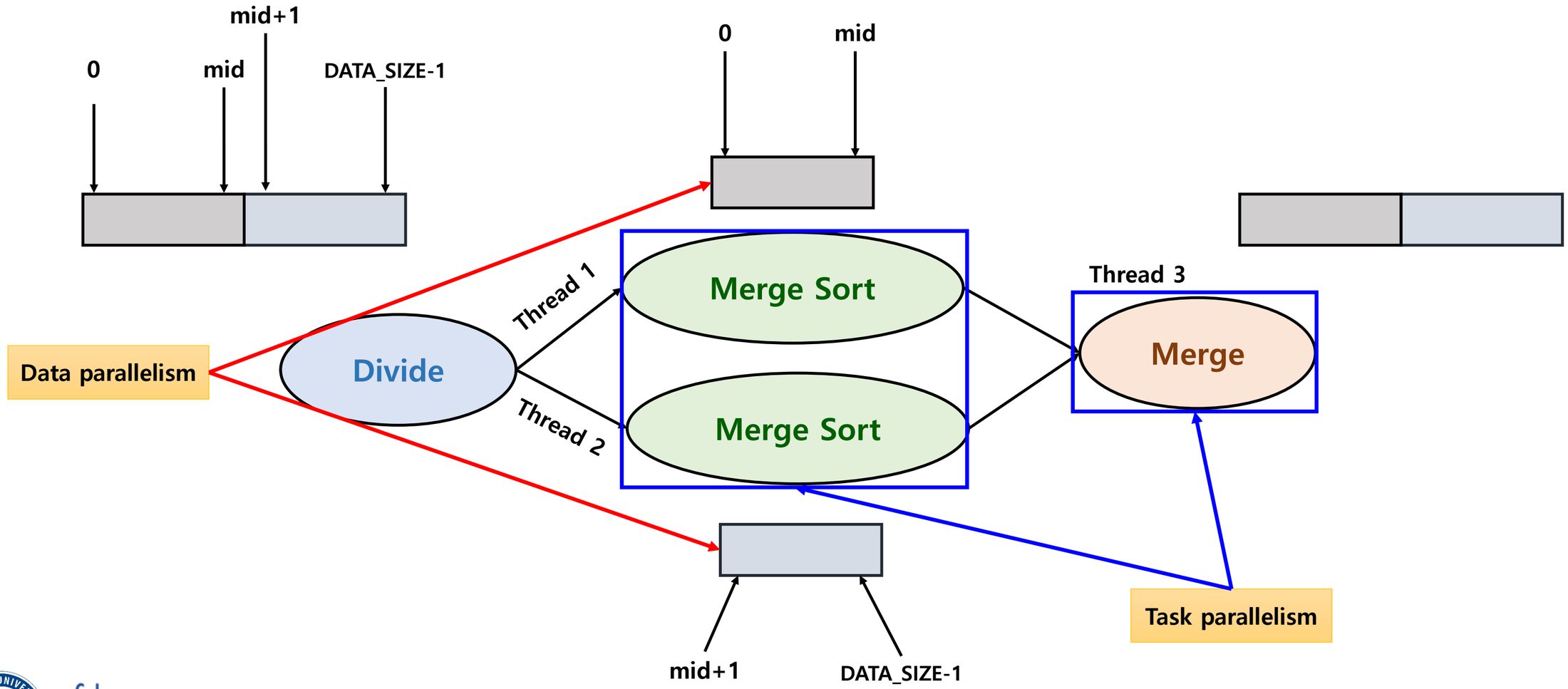
1. **태스크 인식(Identifying tasks) :** 프로세스를 병행가능한 태스크로 나눠야 함
2. **균형(Balance) :** 전체 작업에 **균등한 기여도**를 가지도록 태스크를 나눠야 함
3. **데이터 분리(Data splitting) :** 데이터 또한 개별 코어에서 사용하도록 나눠야 함
4. **데이터 종속성(Data dependency) :**
   - 태스크가 접근하는 데이터는 둘 이상의 태스크 사이에 **종속성이 없는지 검토**되어야 함
   - 태스크 사이에 데이터 종속성이 존재할 경우, 수용 가능하도록 **동기화** 해줘야 함
5. **시험 및 디버깅(Testing and Debugging) :**
   - 다중 스레드로 인해 다양한 경로가 존재할 수 있는데,
     단일 스레드 애플리케이션에 대하여 시험 및 디버깅하는 것보다 훨씬 어려움

프로젝트 흐름


mid+1
0
mid
DATA_SIZE-1
SHARE
0
mid
Thread 3
Thread 1
Merge Sort
Divide
Merge
Thread 2
Merge Sort
mid+1
DATA_SIZE-1

병렬 실행의 유형 (Types of Parallelism)

Operating System Concepts 9th p.166 4.2.2, 10th p.165 4.2.2



프로젝트) 다중 스레드 정렬 응용

week9/2. project/mthread_sort.c

```c
/* 1.DIVIDE */
struct range first_half, second_half, merge_range;
int mid = DATA_SIZE / 2;

first_half.t_name = "thread_1";
first_half.index = 0;
first_half.end = mid;
first_half.data = data;

second_half.t_name = "thread_2";
second_half.index = mid + 1;
second_half.end = DATA_SIZE - 1;
second_half.data = data;

merge_range.t_name = "thread_3";
merge_range.index = mid;
merge_range.end = DATA_SIZE - 1;
merge_range.data = data;

/* 2.SORT */
/* TODO 1: 1st Thread / Sort first half of data */
thr_id = <ENTER YOUR CODE>;
if (thr_id < 0)
{
        perror("thread create error : ");
        exit(0);
}

/* TODO 2: 2nd Thread / Sort second half of data */
thr_id = <ENTER YOUR CODE>;
if (thr_id < 0)
{
        perror("thread create error : ");
        exit(0);
}
```

```c
        /* 3.MERGE */
        /* TODO 3: 3rd Thread / Merge the result of two halfs  */
        thr_id = <ENTER YOUR CODE>;
        if (thr_id < 0)
        {
                perror("thread create error : ");
                exit(0);
        }

        /* TODO 4: waits for the first thread   */

        /* TODO 5: waits for the second thread  */

        /* TODO 6: waits for the third thread   */


        int i;
        for (i = 0; i < DATA_SIZE; i++)
                printf("%d ", data[i]);
        printf("\n");
        return 0;
}
```

프로젝트) 실행 결과 화면

```
os@os-virtual-machine:~/os-week/week9/mthread_sort_ans$ ./mthread_sort_ans
[thread_m] pid: 4876, tid: f3837740
[thread_2] pid: 4876, tid: f2814700
[thread_1] pid: 4876, tid: f3015700
[thread_3] pid: 4876, tid: f2013700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
os@os-virtual-machine:~/os-week/week9/mthread_sort_ans$ ./mthread_sort_ans
[thread_m] pid: 4880, tid: dbf6740
[thread_1] pid: 4880, tid: d3d4700
[thread_3] pid: 4880, tid: c3d2700
[thread_2] pid: 4880, tid: cbd3700
1 2 3 4 5 6 7 8 13 9 10 11 12 14 15 16
os@os-virtual-machine:~/os-week/week9/mthread_sort_ans$ ./mthread_sort_ans
[thread_m] pid: 4884, tid: c560b740
[thread_1] pid: 4884, tid: c4de9700
[thread_2] pid: 4884, tid: c45e8700
[thread_3] pid: 4884, tid: c3de7700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
os@os-virtual-machine:~/os-week/week9/mthread_sort_ans$ ./mthread_sort_ans
[thread_m] pid: 4888, tid: 4b096740
[thread_1] pid: 4888, tid: 4a874700
[thread_2] pid: 4888, tid: 4a073700
[thread_3] pid: 4888, tid: 49872700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
os@os-virtual-machine:~/os-week/week9/mthread_sort_ans$ ./mthread_sort_ans
[thread_m] pid: 4892, tid: 39a56740
[thread_1] pid: 4892, tid: 39234700
[thread_2] pid: 4892, tid: 38a33700
[thread_3] pid: 4892, tid: 38232700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
os@os-virtual-machine:~/os-week/week9/mthread_sort_ans$ ./mthread_sort_ans
[thread_m] pid: 4896, tid: 7caaa740
[thread_1] pid: 4896, tid: 7c288700
[thread_2] pid: 4896, tid: 7ba87700
[thread_3] pid: 4896, tid: 7b286700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

정렬 관련 참고 영상

- **15 Sorting Algorithms in 6 Minutes**

  https://www.youtube.com/watch?v=kPRA0W1kECg

- **Merge Sort vs Quick Sort**

  https://www.youtube.com/watch?v=es2T6KY45cA

감사합니다.

CPS LAB

Lim Jiseoup

jseoup@hanyang.ac.kr