운영체제론 실습

- 다중 스레드 정렬 응용

목 차

예제 코드 다운로드 경로

아래 명령어를 linux 환경에서 치면 다운받을 수 있음.

```
$ wget http://ce.hanyang.ac.kr/week9.zip
$ unzip week9.zip
```

합병 정렬 알고리즘

합병 정렬(Merge Sort)

- **정렬이란?**
  - 순서없이 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 복잡도가 달라짐.
    즉, 똑같은 데이터를 정렬하는데 많은 시간 차이를 보이기도 함 (**O(n), O(nlogn), O(n*n)**).

- **정렬을 어떻게 효율적으로 할까?**
  - **Divide & Conquer (분할정복)** 알고리즘
  - 대표적으로 **합병 정렬**과 **쾌속 정렬**이 있음

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

시간 복잡도

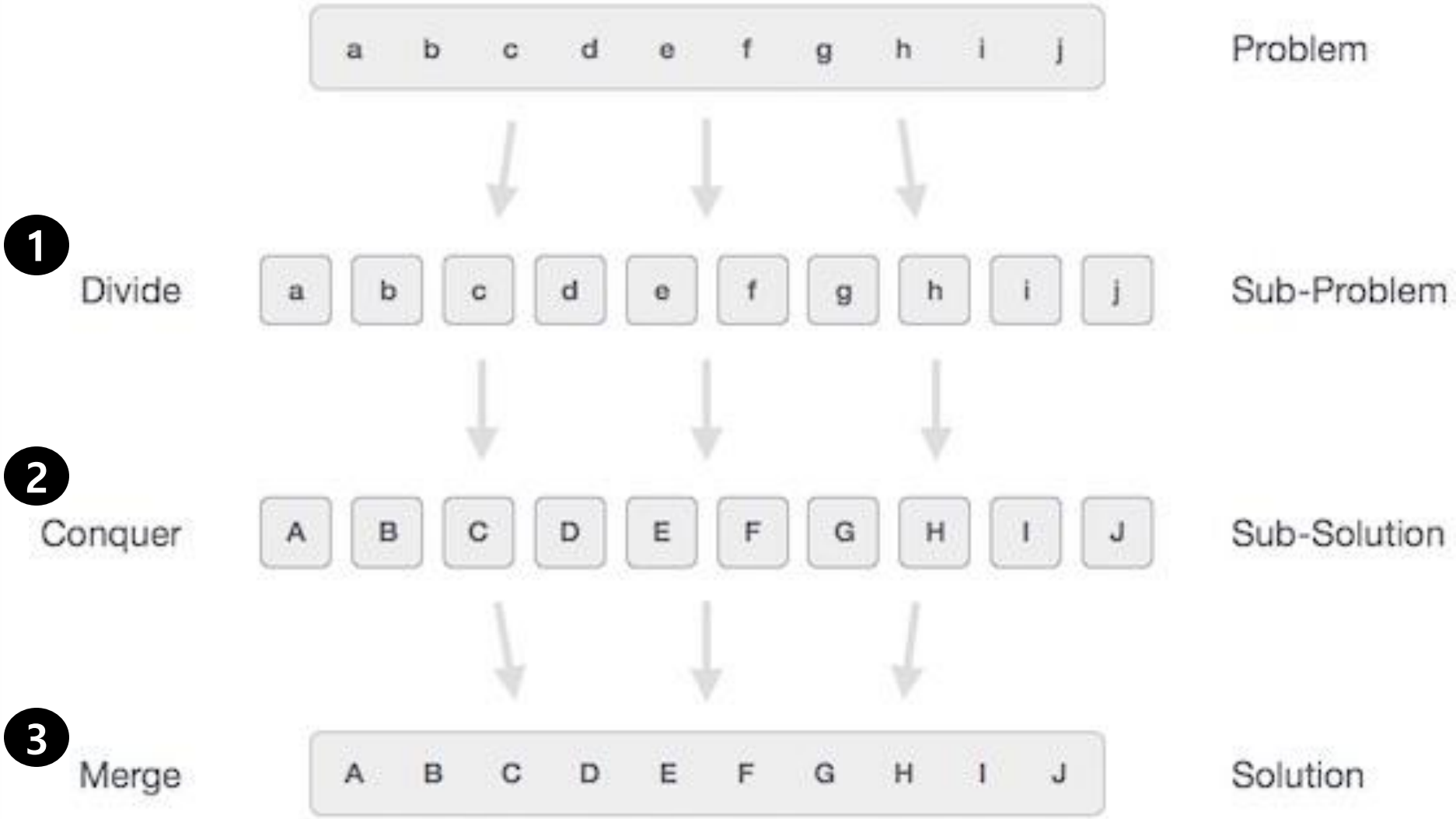
정렬 알고리즘에 따라
시간 복잡도가 많이 달라지는 것을 볼 수 있음

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)

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

분할 정복 알고리즘 (Divide & Conquer)

큰 문제를 잘게 쪼개면
쉽게 해결할 수 있음



합병 정렬 예

1

1. Divide the array into two parts

2. Divide the array into two parts again

3. Break each element into single parts

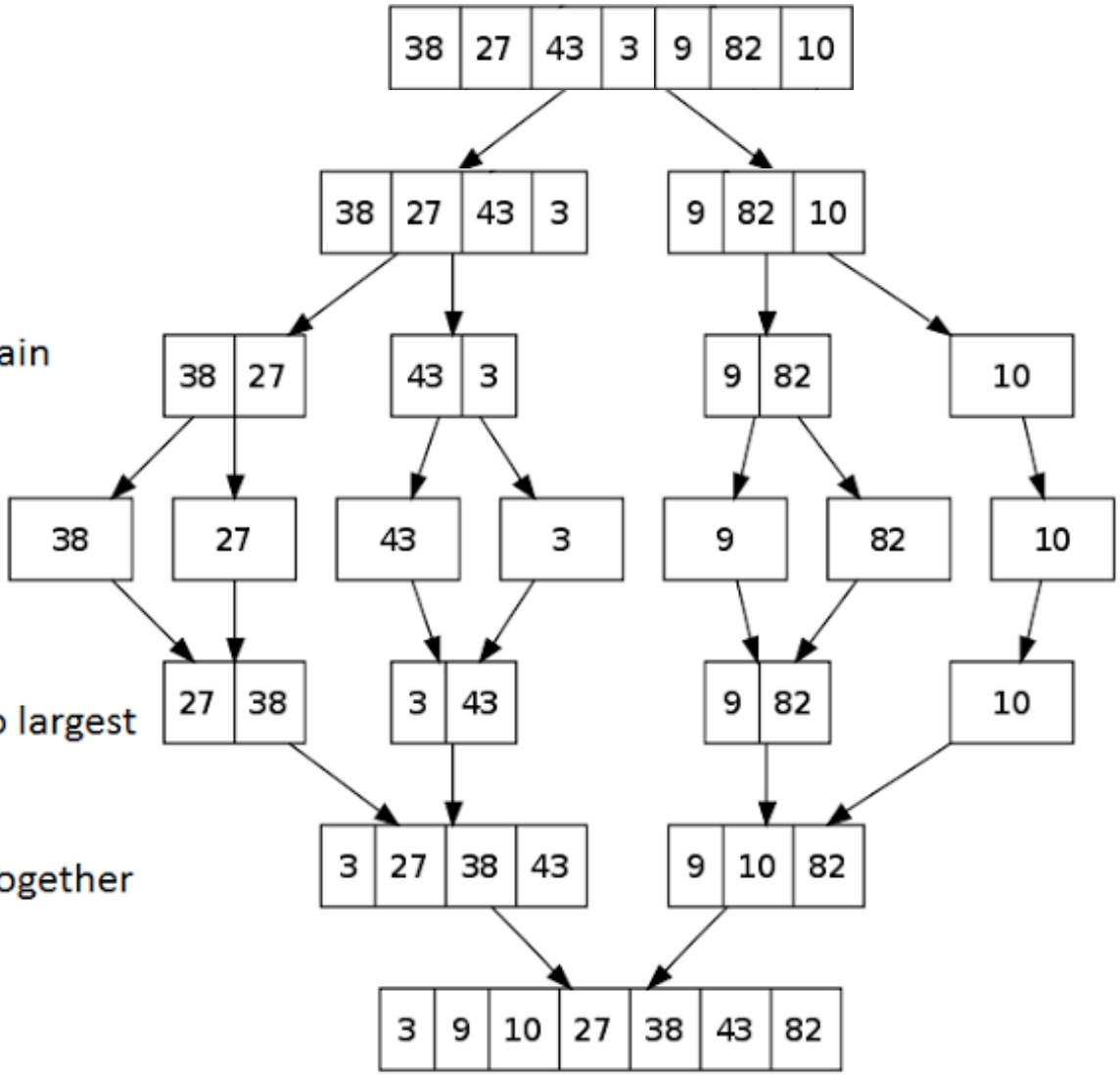
2

4. Sort the elements from smallest to largest

3

5. Merge the divided sorted arrays together

6. The array has been sorted



Example 1) Merge Sort

```
 1 #include <stdio.h>
 2 #define DATA_SIZE 8
 3
 4 void mergeSort(int data[], int p, int r);
 5 void merge(int data[], int p, int q, int r);
 6
 7 int main() {
 8   int data[DATA_SIZE] = {5, 2, 4, 7, 1, 3, 2, 6};
 9   mergeSort(data, 0, DATA_SIZE-1);
10   int i;
11   for (i = 0; i < DATA_SIZE; i++)
12     printf("%d ", data[i]);
13   printf("\n");
14   return 0;
15 }
16
17 void mergeSort(int data[], int p, int r) {
18   int q;
19   if (p < r) {
20     q = (p + r) / 2;
21     mergeSort(data, p, q);
22     mergeSort(data, q + 1, r);
23     merge(data, p, q, r);
24   }
25 }
26
```

재귀함수

1. Divide

2. Conquer

3. Merge

Example 1) Merge Sort

```
27 void merge(int data[], int p, int q, int r) {
28   int i = p, j = q + 1, k = p;
29   int tmp[DATA_SIZE];
30   while (i <= q && j <= r) {
31     if (data[i] <= data[j])
32       tmp[k++] = data[i++];
33     else
34       tmp[k++] = data[j++];
35   }
36   while (i <= q)
37     tmp[k++] = data[i++];
38   while (j <= r)
39     tmp[k++] = data[j++];
40   for (int a = p; a <= r; a++)
41     data[a] = tmp[a];
42 }
```

프로젝트:
다중 스레드 정렬 응용

pthread_create() 사용법

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

- 새로운 thread를 생성한다.
  - *thread* : 생성이 성공한 경우 생성된 thread의 ID가 저장됨
  - *attr* : thread 속성 (본 실습 시간에는 NULL 사용)
  - *start_routine* : thread가 수행할 함수명
  - *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() 사용법

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

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

- **사용 예제**

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

        …

        pthread_exit(0);
}
```

pthread_join() 사용법

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

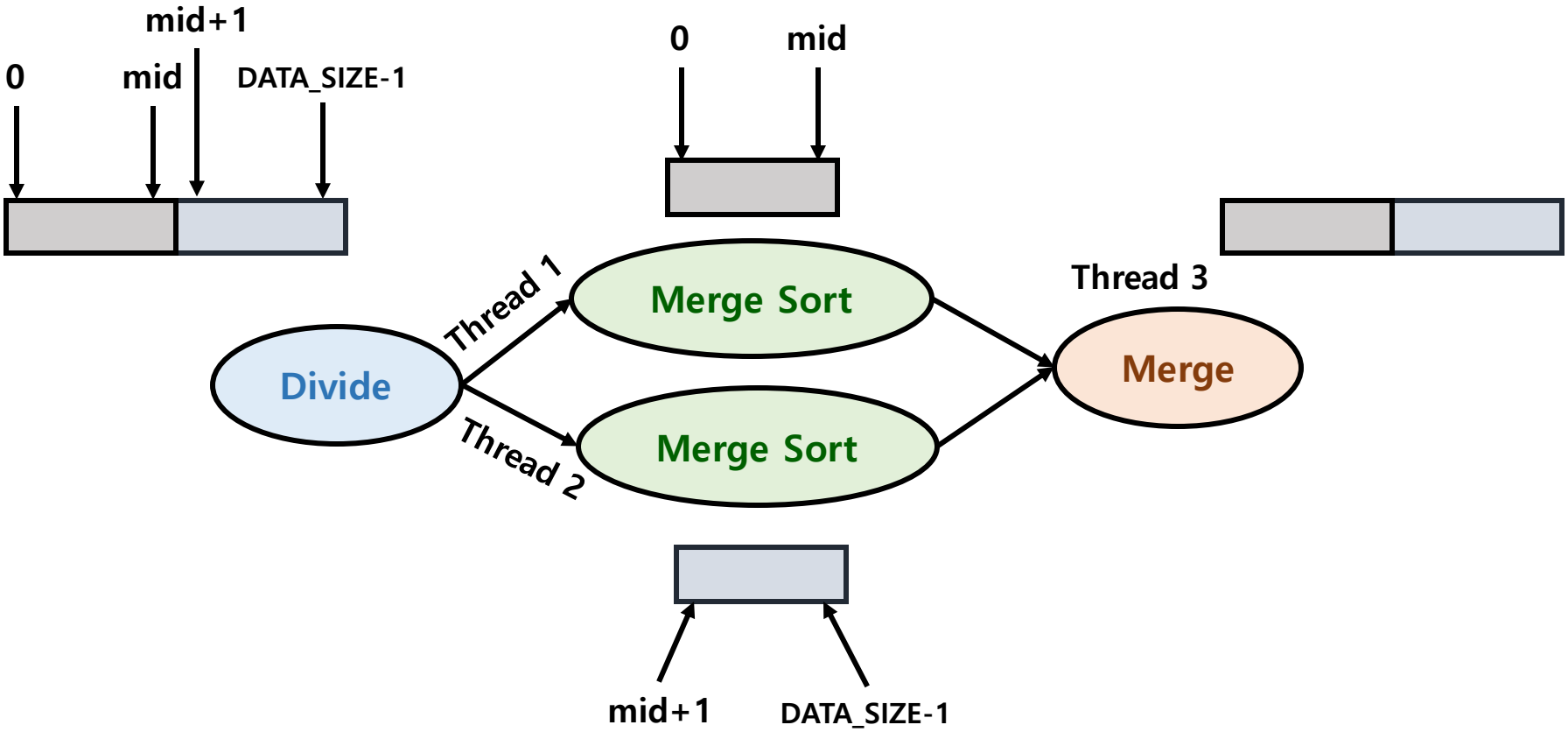
- thread의 종료를 기다린다.
  - *thread* : 해당 ID를 가진 thread를 기다림
  - *retval* : thead의 반환 값을 가져오기 위함 (**pthread_exit의 retval 값을 가져옴**)
- **사용 예제**

```
pthread_t tid;

        ...
/* thread가 생성되고 종료됨 */
        ...

pthread_join(&tid, NULL);
```

프로젝트 흐름 설명


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

프로젝트: 다중 스레드 정렬 응용

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>

#define DATA_SIZE 16

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

void *sort_thread(void *param);
void *merge_thread(void *param);

void *t_print(void *data);

struct range{
  char *t_name;
  int index;
  int end;
  int *data;
};

int main(int argc, char **argv)
{
  int data[DATA_SIZE] =
    {5, 16, 4, 7, 1, 3, 2, 6, 8, 13, 11, 9, 10, 12, 15, 14};
  pthread_t tid[3];
  int thr_id;
  int status;
  char tm[] = "thread_m";

  t_print((void *)tm);

```

프로젝트: 다중 스레드 정렬 응용

```
34    /* 1.DIVIDE */
35    struct range first_half, second_half, merge_range;
36    int mid = DATA_SIZE/2;
37
38    first_half.t_name = "thread_1";
39    first_half.index = 0;
40    first_half.end = mid;
41    first_half.data = data;
42
43    second_half.t_name = "thread_2";
44    second_half.index = mid+1;
45    second_half.end = DATA_SIZE-1;
46    second_half.data = data;
47
48    merge_range.t_name = "thread_3";
49    merge_range.index = mid;
50    merge_range.end = DATA_SIZE-1;
51    merge_range.data = data;
52
53    /* 2.SORT */
54    /* TODO: 1st Thread / Sort first half of data */
55    /* TODO: 2nd Thread / Sort second half of data */
56
57    /* 3.MERGE */
58    /* TODO: 3rd Thread / Merge the result of two halfs  */
59
60    /* TODO: waits for the first thread   */
61    /* TODO: waits for the second thread  */
62    /* TODO: waits for the third thread   */
63
64    int i;
65    for(i = 0; i < DATA_SIZE; i++)
66      printf("%d ", data[i]);
67    printf("\n");
68    return 0;
69 }
```

생각해보기:

Thread 1,2가 종료하길 기다리는 코드는 어디에 위치하는게 맞을까요?

프로젝트: 다중 스레드 정렬 응용

merge_sort()와 merge() 함수는
예제 1)의 함수와 동일

```
117 void *t_print(void *data) {
118   pid_t pid;  // process id
119   pthread_t tid;
120
121   pid = getpid();
122   tid = pthread_self();
123
124   char* thread_name = (char*)data;
125
126   printf("[%s] pid: %u, tid: %x\n",
127       thread_name, (unsigned int)pid, (unsigned int)tid);
128   sleep(1);
129 }
```

프로젝트: 결과화면 (잘못된 구현 결과)

```
jsbaik@jsbaik:~/OS2019/week9$ ./mthread_sort
[thread_m] pid: 10221, tid: fba89740
[thread_1] pid: 10221, tid: fb26a700
[thread_3] pid: 10221, tid: fa268700
[thread_2] pid: 10221, tid: faa69700
1 2 3 4 5 6 7 8 13 9 10 11 12 14 15 16
jsbaik@jsbaik:~/OS2019/week9$ ./mthread_sort
[thread_m] pid: 10225, tid: 1e6d7740
[thread_2] pid: 10225, tid: 1d6b7700
[thread_3] pid: 10225, tid: 1ceb6700
[thread_1] pid: 10225, tid: 1deb8700
5 9 10 11 12 13 14 15 16 4 7 1 3 2 6 8
jsbaik@jsbaik:~/OS2019/week9$ ./mthread_sort
[thread_m] pid: 10229, tid: 518b740
[thread_1] pid: 10229, tid: 496c700
[thread_2] pid: 10229, tid: 416b700
[thread_3] pid: 10229, tid: 396a700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
jsbaik@jsbaik:~/OS2019/week9$ ./mthread_sort
[thread_m] pid: 10233, tid: 51195740
[thread_1] pid: 10233, tid: 50976700
[thread_2] pid: 10233, tid: 50175700
[thread_3] pid: 10233, tid: 4f974700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

순서대로 정렬되지 않음

프로젝트: 결과화면 (항상 같은 결과를 출력)

```
jsbaik@jsbaik:~/OS2019/week9$ ./mthread_sort
[thread_m] pid: 10465, tid: 32cf6740
[thread_1] pid: 10465, tid: 324d7700
[thread_2] pid: 10465, tid: 31cd6700
[thread_3] pid: 10465, tid: 31cd6700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
jsbaik@jsbaik:~/OS2019/week9$ ./mthread_sort
[thread_m] pid: 10469, tid: 76988740
[thread_2] pid: 10469, tid: 75968700
[thread_1] pid: 10469, tid: 76169700
[thread_3] pid: 10469, tid: 75968700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
jsbaik@jsbaik:~/OS2019/week9$ ./mthread_sort
[thread_m] pid: 10473, tid: 2faa8740
[thread_2] pid: 10473, tid: 2ea88700
[thread_1] pid: 10473, tid: 2f289700
[thread_3] pid: 10473, tid: 2ea88700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
jsbaik@jsbaik:~/OS2019/week9$ ./mthread_sort
[thread_m] pid: 10477, tid: 12c63740
[thread_1] pid: 10477, tid: 12444700
[thread_2] pid: 10477, tid: 11c43700
[thread_3] pid: 10477, tid: 11c43700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

수고하셨습니다.

- 다음 시간:

  잠자는 수업 조교