Lab. 5

System Information

Duksu Kim

공통사항

- **모든 System call 및 Standard C function 사용 가능**
  - 단, 외부 라이브러리 사용 불가
- **Source code 및 실행파일 이름은 문제 번호 사용**
  - 예) **6_1.c** 및 **6_1.out** 등
- **시간 측정 필요 시, 자신이 만든 측정 함수 사용**

- **EL 제출 시,**
  - 모든 source code를 모은 한글/워드 파일 별도 제출

Class 01

Type A

3조

박주찬, 이건환, 강한별

Lab 6-1. 여러가지 정렬의 평균 시간

- **문제 설명**
  - 선택 정렬, 삽입 정렬, 버블 정렬, 퀵 정렬, 셸 정렬을 함수로 구현하여 평균 시간 구하기.
  - 정렬을 할 때 마다 완료 시간을 출력하기.
  - 각 정렬을 10번 실행 반복을 하여 평균 소요 시간을 구하기.
  - 어떤 정렬이 가장 소요시간이 적게 걸리는지 확인하기.

Lab 6-1. 여러가지 정렬의 평균 시간

- **조건**
  - CLOCK 함수를 사용하지 않고, gettimeofday를 사용하여 초 값을 읽어온다. 정확한 값을 읽어 오기 위해 micro-sec까지 측정한다.
  - 초 단위로 읽어온 시간 정보를 사람이 이해하기 편한 형태로 변환하는 tm구조체를 사용한다.
  - 정렬을 한번만 해보면 정확한 결과를 도출할 수 없으므로 최소 10번 이상 반복 실행하여 평균값을 구한다.
  - 정렬할 배열의 원소의 개수는 60000개로 하고, rand() 함수를 사용해 초기값을 설정한다.

Lab 6-1. 여러가지 정렬의 평균 시간

- **실행 1)**

```
-------[1]번째-------
시작 시간 : 2018년 10월 20일 20시 2분 56초.165275
배열 원소의 개수 : 60000

선택 정렬
완료 시간 : 2018년 10월 20일 20시 3분 0초.107432
소요 시간 : 3.942157sec

삽입 정렬
완료 시간 : 2018년 10월 20일 20시 3분 2초.292110
소요 시간 : 2.184678sec

버블 정렬
완료 시간 : 2018년 10월 20일 20시 3분 12초.302642
소요 시간 : 10.010532sec

퀵 정렬
완료 시간 : 2018년 10월 20일 20시 3분 12초.312891
소요 시간 : 0.010249sec

셸 정렬
완료 시간 : 2018년 10월 20일 20시 3분 12초.332108
소요 시간 : 0.019217sec
```

Lab 6-1. 여러가지 정렬의 평균 시간

- **실행 2)**

```
-------[10]번째-------
시작 시간 : 2018년 10월 20일 20시 27분 52초.551627
배열 원소의 개수 : 60000

선택 정렬
완료 시간 : 2018년 10월 20일 20시 27분 56초.476996
소요 시간 : 3.925369sec

삽입 정렬
완료 시간 : 2018년 10월 20일 20시 27분 58초.660405
소요 시간 : 2.183409sec

버블 정렬
완료 시간 : 2018년 10월 20일 20시 28분 8초.681437
소요 시간 : 10.021032sec

퀵 정렬
완료 시간 : 2018년 10월 20일 20시 28분 8초.691578
소요 시간 : 0.010141sec

셸 정렬
완료 시간 : 2018년 10월 20일 20시 28분 8초.710113
소요 시간 : 0.018535sec



결론
선택 정렬의 평균 시간 : 3.958847
삽입정렬의 평균 시간 : 2.181928
버블정렬의 평균 시간 : 10.042262
퀵 정렬의 평균 시간 : 0.010303
셸 정렬의 평균 시간 : 0.018603
```

Lab 6-1. 여러가지 정렬의 평균 시간

- **Hints 1)**
  - **tm data structure**

$ man -s 3 gmtime

```
struct tm
{
    int tm_sec;         /* Seconds.[0-60] (1 leap second) */
    int tm_min;         /* Minutes.[0-59] */
    int tm_hour;        /* Hours.[0-23] */
    int tm_mday;        /* Day.[1-31] */
    int tm_mon;         /* Month.[0-11] */
    int tm_year;        /* Year- 1900.  */
    int tm_wday;        /* Day of week.[0-6] */
    int tm_yday;        /* Days in year.[0-365]*/
    int tm_isdst;       /* DST.[-1/0/1]*/
}
```

Lab 6-1. 여러가지 정렬의 평균 시간

- **Hints 2)**

```c
#include <sys/time.h>
#include <stdio.h>

int main(void) {
    struct timeval tv;

    gettimeofday(&tv, NULL);
    printf("Time(sec) : %d\n", (int)tv.tv_sec);
    printf("Time(micro-sec) : %d\n", (int)tv.tv_usec);

    return 0;
}
```

```
Time(sec) : 1539103396
Time(micro-sec) : 279892
```

현재 초 : (double) tv.tv_sec + (double)tv.tv_usec / 1000000

Lab 6-2. 시간 복잡도 분석

- **문제 설명**
  - O(1), O(n), O(n^2)의 시간복잡도를 갖는 세 함수를 만들어준다.
  - 세 함수를 실행시켜서 걸리는 시간을 비교한다.

Lab 6-2. 시간 복잡도 분석

- **조건**
  - Clock함수를 사용하지 않고, tms구조체, times함수를 사용한다.
  - O(1) O(n) O(n^2)함수는 1~n까지의 덧셈을 반환하는 함수를 사용한다.
  - 출력은 O(1) O(n) O(n^2) 세 개를 동시에 비교를 해서 출력하는 것과 O(1) O(n) 두 개를 동시에 비교를 해서 출력하도록 한다.

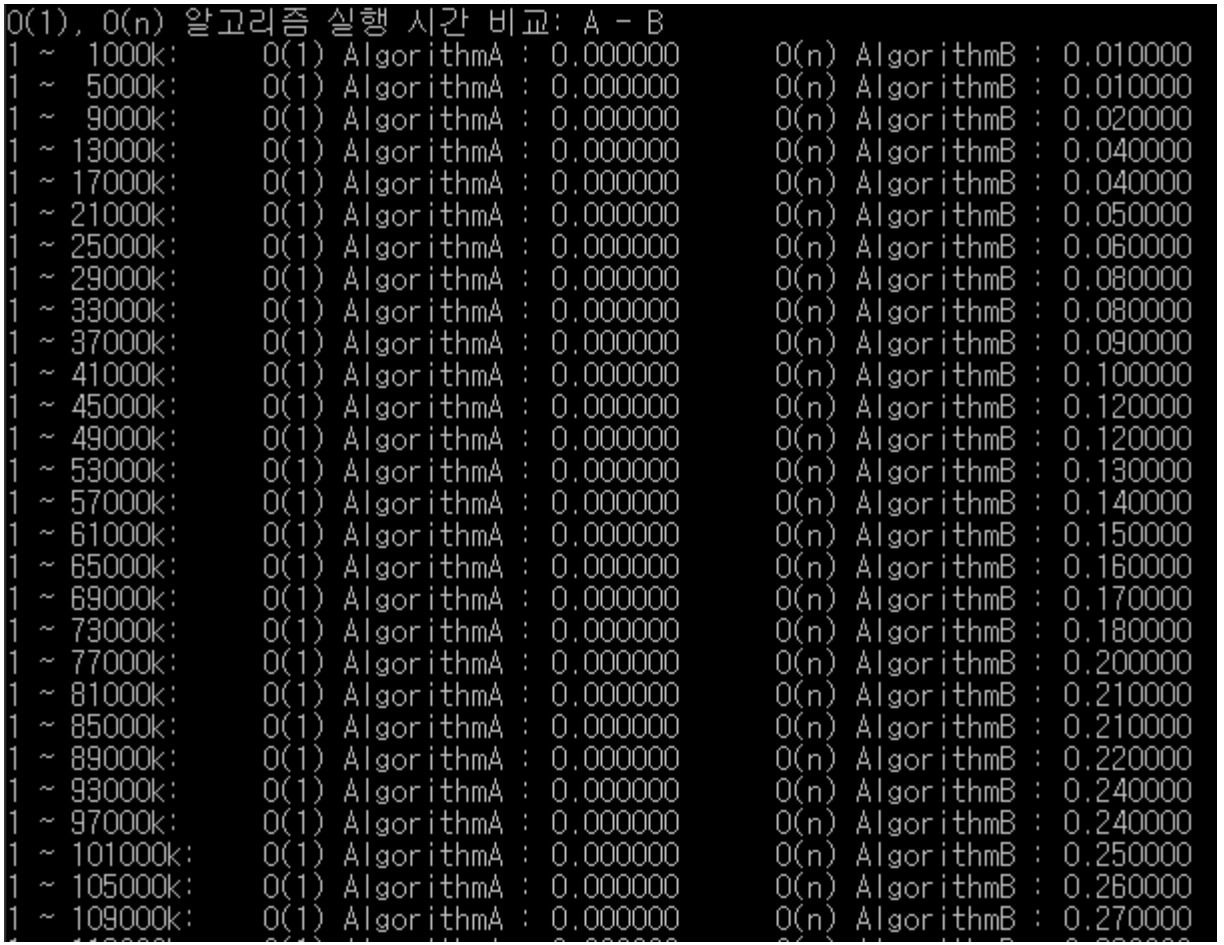
Lab 6-2. 시간 복잡도 분석

- 실행 1)

```
O(1), O(n), O(n^2) 알고리즘 실행 시간 비교: A - B - C
1 ~ 11000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.080000
1 ~ 13000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.120000
1 ~ 15000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.160000
1 ~ 17000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.210000
1 ~ 19000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.250000
1 ~ 21000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.330000
1 ~ 23000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.380000
1 ~ 25000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.450000
1 ~ 27000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.530000
1 ~ 29000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.600000
1 ~ 31000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.690000
1 ~ 33000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.790000
1 ~ 35000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.880000
1 ~ 37000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 0.980000
1 ~ 39000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 1.090000
1 ~ 41000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 1.210000
1 ~ 43000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 1.330000
1 ~ 45000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 1.480000
1 ~ 47000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 1.610000
1 ~ 49000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 1.790000
1 ~ 51000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 1.900000
1 ~ 53000:      O(1) AlgorithmA : 0.000000      O(n) AlgorithmB : 0.000000      O(n^2) AlgorithmC : 2.010000
```

Lab 6-2. 시간 복잡도 분석

- 실행 2)



Lab 6-2. 시간 복잡도 분석

- **Hints 1)**
  - O(1)은 공식n*(n-1)/2을 사용한다.
  - O(n)은 1+2+3+ … + n으로 구현한다.
  - O(n^2)은 1 + (1+1) + (1+1+1) + (1+1+1+1) + … 으로 구현한다.

Class 02

Type B

6조

김지연 김형찬 이재홍 강동균

Lab 6-1. π를 근사하는 시간비교

- **문제 설명**
  - π값을 근사하는 식을 이용하여 π값을 구하며, 근사식 (소수점 20자리까지) 의 계산시간을 비교한다.
  - 출력: 시작시간, 종료시간, 근사식1(결과, 시간), 근사식2(결과, 시간)
- **Hints**
  - 근사식1(Leibniz formula for π): $\frac{\pi}{4} = \sum_{i=0}^{n} \frac{(-1)^i}{2i+1}$
  - 근사식2(Euler formula for π): $\frac{\pi^2}{6} = \sum_{i=1}^{n} \frac{1}{i^2}$
  - #include <math.h>
  - gcc –o Lab6_1.out Lab6_1.c **-lm**
  - 변수 타입: long double
- **실행의 예**

```
$ ./Lab6_1.out 999999999
start time: 2018year 10month 20day 01hour 40minute 40second

Number of polynomials = 999999999

Leibniz formula for PI(to 20 decimal place): 3.14159265258805042720
processing time: 1.33s
Euler formula for PI(to 20 decimal place): 3.14159264498238988139
processing time: 1.34s

end time: 2018year 10month 20day 01hour 40minute 42second
```

Lab 6-2. PS

- **Linux ps 명령어를 구현해보자!**

- **힌트 1 : /proc/[pid]/status 정보 파싱**
- **힌트 2 : dirent.h, pwd.h**
- **조건 1 : PID 기준으로 정렬되어 있어야함**

```
$ ./6_1
zero@zero:~/Desktop$ ./6_1
[+] Current PID : 31459 PPID : 30263
UID                  PID            PGID           State          CMD
root                 1              1              S              systemd
root                 2              0              S              kthreadd
root                 4              0              S              kworker/0:0H
root                 6              0              S              mm_percpu_wq
root                 7              0              S              ksoftirqd/0
root                 8              0              S              rcu_sched
root                 9              0              S              rcu_bh
root                 10             0              S              migration/0
…
```

Common Problem

Lab 6-3

By DS Kim

Lab 6-3. myTimer 만들기

- **include 해서 사용할 수 있게, 별도의 header 파일로 작성**
- **myTimer의 기능 (각 기능을 함수로 작성)**
  - int **myTimer_init** (int numTimers)
    - 지원하는 timer 수를 인자로 지정
    - Timer 사용을 위한 초기화 작업 수행
  - int **myTimer_finalize** (void)
    - Timer에 할당된 메모리 모두 해제
  - int **myTimer_on** (int timerID)
    - ID가 timerID인 timer on (시간측정 시작)
    - 기존에 측정된 시간이 있으면, 시간 누적
  - int **myTimer_off** (int timerID)
    - ID가 timerID인 timer off (시간측정 종료)
  - int **muTimer_print** (void)
    - 측정된 시간들을 milliseconds 단위(소수점 2자리까지)로 출력

Lab 6-3. myTimer 만들기

- **Timer를 테스트 하기 위한 임의의 코드를 작성하고 Timer 사용 결과를 함께 제출**
- **Hints**
  - gettimeofday()
  - Milliseconds = 10^{-3} 초

Lab 6-3. myTimer 만들기

- 사용의 예

```
#include  " DSTimer.h"

int main(void)
{
     DSTimer_init(2);// timer initialize

     DSTimer_on(0); // Turn on timer 0
     // do something
     DSTimer_off(0);  // Turn off timer 0

     DSTimer_on(1); // Turn on timer 1
     // do something
     DSTimer_off(1); // Turn off timer 1

     DSTimer_print();
     DSTimer_finalize()
}
```

$ DSTimerTest.out
[DSTimer]
Timer 0 : 12.33 ms
Timer 1 : 31.23 ms