

# 보고서

학과 : 컴퓨터공학과  
학번: 21720903  
이름 : 조태식

# 목차

- 1.개요
- 2.알고리즘 및 구현
- 3.실험결과 분석 및 결론

## 1.개요

수업시간에 디스크 스케줄러에 대해서 배워보았다. FCFS부터 SSTF 그리고 SCAN과 LOOK 에션바흐 등등 여러 가지 기법들이 있었다. 각각의 상황에 따라 최적화 기법은 다 달랐는데, 이것을 우리 컴퓨터는 착착 잘해낸다는 것이 새삼 좋았다. 그러면서 직접 배웠던 코드들을 작성해보고 실행시켜가면서 어떤 상황일 때 우세한 기법인지 상황에 따른 기법들을 찾아보고 더 나아가 나만의 스케줄러 기법을 만들어보고자 한다.

## 2.알고리즘 및 구현

```
#include <stdio.h>      //입출력 헤더
#include <stdlib.h>      //난수 생성
#include <time.h>        //난수 현재시각반영
#include <math.h>        //제곱근 함수 사용
//배운 기법들: FCFS SSTF SCAN C-SCAN C-LOOK LOOK F-SCAN N-STEP SCAN
//에션바흐 섹터큐잉 SLTF SPTF SATF FD-SCAN P-SCAN GSS-SCHEDUAL
//EDF-SCHEDUL SCAN-EDF-SCHE
//필요한 변수 선언
int which = 0; //몇번째기법을 선택할건지
int headerposition = 0; //헤더포지션 위치
int track[10] = { 0, }; //무작의 트랙의 위치
int count = 0; //무작의 트랙의 개수
int seek_distance[10] = { 0, }; //이동거리 모음집
double ave_seek_time = 0; //평균탐색시간
double ave_rotational_latency = 0; //평균 회전지연시간
int total_seek_distance = 0; //전체이동거리
double deviation[10] = { 0, }; //편차 배열
double deviation2[10] = { 0, }; //편차 제곱의 배열
double variance = 0; //분산
double total_deviation2 = 0; //편차제곱의 합
double standard_deviation = 0; //표준편차
int just = 0; //그냥 잠시 쓰는 변수
int sstf_distance[10] = { 0, }; //sstf기법 다룰때 쓰는 각각의 이동거리 매번 이동거리
//최소값 구할때 사용
int real_sstf_distance[10] = { 0, }; //진짜 sstf 기법 이동거리
int small = 0; //sstf기법 쓸때 거리 최소값
```

```

int sstf_track[10] = { 0, }; //sstf기법쓸때 거리에 따른 track 리빌딩
int track_left = 0; //sstf 기법쓸때 남은 track 개수
int remain = 0; //sstf 기법쓸때 잠시기억장소
int number = 0; //sstf기법쓸때 while문안에들어갈 number
int direction = 0; //look에서 방향결정함수
int look_track[10] = { 0, }; //look에서 track
int look_distance[10] = { 0, }; //look에서 거리
int look_order[10] = { 0, }; //look에서 order
int seek_time = 0; //seek_time 입력
double total_rotational_latency = 0; //총회전지연시간
double rotational_latency[10] = { 0, }; //회전지연시간
double rpm = 0; //분당회전수
int sltf_track[10] = { 0, }; //sltf 순서정렬
int main(void) {
    printf("어떤 디스크 스케줄링 기법을 선택하시겠습니까?\n(1.FCFS기법 2.SSTF기법
3.LOOK기법 4.섹터큐잉 기법 5.시기의기법):");
    scanf("%d", &which);
    if (which == 1) { //FCFS
        printf("FCFS기법을 선택하셨습니다.");
        printf("\n디스크 헤더의 초기위치를 입력하세요:(0부터 199까지):");
        scanf("%d", &headerposition);
        printf("seek_time 기준값을 입력해주세요:");
        scanf("%d", &seek_time);
        srand(time(NULL)); //랜덤함수
        printf("무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터~최대
10까지)");

        count = rand() % 9 + 1;
        printf("\n트랙의 개수가 %d로 정해졌습니다.", count);
        printf("실린더의 범위를 0~199로 한정했습니다.");
        printf("이제 %d개가 랜덤으로정해집니다.", count);
        for (int i = 0; i < count; i++) {
            track[i] = rand() % 200;
        }
        printf("\n정해진 트랙번호의 순서: ");
        for (int i = 0; i < count; i++) {
            printf("%d, ", track[i]);
        }
        printf("\nFCFS기법을 선택했으므로 순서는 들어온 순서대로이므로:");
        for (int i = 0; i < count; i++) {
            printf("\n%d번째로 %d 입니다.", i+1, track[i]); //번호를 0번째보다
1번째로 보는게 편해서

```

```

    }
    for (int i = 0; i < count; i++) {
        if (i == 0) { //처음일때는 헤더의위치와 거리를 재므로
            just = track[i] - headerposition;
            seek_distance[i] = abs(just);
        }
        else {
            just = track[i] - track[i - 1];
            seek_distance[i] = abs(just);
        }
    }
    printf("\n");
    for (int i = 0; i < count; i++) {
        if (i == 0) {
            printf("%d와 %d의 이동거리는 %d입니다.\n",
headerposition, track[i], seek_distance[i]);
        }
        else {
            printf("%d와 %d의 이동거리는 %d입니다.\n", track[i],
track[i-1], seek_distance[i]);
        }
    }
    for (int i = 0; i < count; i++) {
        total_seek_distance = total_seek_distance + seek_distance[i];
    }
    printf("전체 이동거리는 %d입니다.", total_seek_distance);
    printf("\nseek_time이 %d이므로 총 탐색시간은 %d입니다. 즉
회전지연시간이 없으므로 총 실행시간은 %d입니다.", seek_time, total_seek_distance *
seek_time, total_seek_distance * seek_time);
    ave_seek_time = (total_seek_distance * seek_time)/count;
    printf("그리고 평균 탐색시간은 %lf입니다.\n", ave_seek_time);
    for (int i = 0; i < count; i++) {
        deviation[i] = (seek_distance[i] * seek_time) - ave_seek_time;
        printf("각 %d 번째 탐색시간 %d에 대한 편차는 %lf입니다.",
i+1, seek_distance[i] * seek_time, deviation[i]);
        deviation2[i] = (deviation[i]*deviation[i]);
        printf("이 편차의 제곱은 %lf입니다.\n", deviation2[i]);
        total_deviation2 = total_deviation2 + deviation2[i];
    }
    variance = total_deviation2 / count;
    printf("총 편차의 합이 %lf이므로 분산은 %lf 입니다.", total_deviation2,

```

```

variance);

    standard_deviation = sqrt(variance);
    printf("그러므로 실행시간의 표준편차는 %lf입니다.", standard_deviation);
}
else if (which == 2) { //SSTF
    printf("SSTF기법을 선택하셨습니다.");
    printf("\n디스크 헤더의 초기위치를 입력하세요:(0부터 199까지):");
    scanf("%d", &headerposition);
    printf("seek_time 기준값을 입력해주세요:");
    scanf("%d", &seek_time);
    srand(time(NULL));
    printf("무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터~최대
10까지));

    count = rand() % 9 + 1;
    printf("\n트랙의 개수가 %d로 정해졌습니다.", count);
    printf("실린더의 범위를 0~199로 한정했습니다.");
    printf("%d개가 랜덤으로정해집니다.", count);
    for (int i = 0; i < count; i++) {
        track[i] = rand() % 200;
        sstf_track[i] = track[i];
    }
    printf("\n정해진 트랙번호: ");
    for (int i = 0; i < count; i++) {
        printf("%d ", track[i]);
    }
    printf("\nSSTF기법을 선택했으므로 순서는:\n");
    track_left = count; //총 트랙의 개수 저장
    int just1 = 0;
    while (track_left > 0) { //트랙의 개수가 0이 될때까지
        if (track_left == count) { //첫순서를 정할때 헤더위치와
비교해야하므로
            for (int i = 0; i < count; i++) {
                sstf_distance[i] = abs(headerposition -
sstf_track[i]); //각각의 이동거리 저장
            }
            small = 200; //최솟값을 찾기위해 임의로 200 배정
            for (int i = 0; i < count; i++) {
                if (sstf_distance[i] < small) {
                    small = sstf_distance[i];
                    seek_distance[number] = small;
//최솟값을 찾기위해 다시 저장

```

```

just = i; //최솟값일때의 i를 just에 저장
다음번부터는 이친구는 배제하고 거리재야하므로
real_sstf_distance[just1] =
sstf_distance[i]; //최솟값일때의 그 탐색거리를 real배열에 저장
remain = sstf_track[just];
//remain이라는 곳에 최솟값일때 트랙번호를 저장
    }
}
printf("%d번째가 %d로 젤 작으므로\n", number+1,
remain);

sstf_track[just] = 1000; //다음부터는 배제하고
거리재므로 1000이라는 큰숫자 입력
    }
else { //첫순서를 제외하고는 전에 실행됐던 위치와
비교해야하므로

    for (int i = 0; i < count; i++) {
        sstf_distance[i] = abs(remain - sstf_track[i]);
//remain에 저장된값이 전에 했던 트랙번호이므로
    }
    small = 200;
    for (int i = 0; i < count; i++) {
        if (sstf_distance[i] < small) {
            small = sstf_distance[i];
            seek_distance[number] = small;
            just = i;
            real_sstf_distance[just1] =
sstf_distance[i];

            remain = sstf_track[just];

        }
    }
    printf("%d번째가 %d로 젤 작으므로\n", number+1,
remain);

    sstf_track[just] = 1000;
}
number++; //몇번째 체크
track_left--; //남은 트랙체크
just1++; //이동거리
}
for (int i = 0; i < count; i++) {
    total_seek_distance = total_seek_distance + seek_distance[i];

```

```

    }
    printf("전체 이동거리는:%d", total_seek_distance);
    printf("\nseek_time이 %d이므로 총 탐색시간은
%d입니다",seek_time,total_seek_distance* seek_time);
    printf("즉 회전지연시간이 없으므로 총 실행시간은
%d입니다.",total_seek_distance* seek_time);
    ave_seek_time = (total_seek_distance * seek_time) / count;
    printf("평균 탐색시간은 %lf입니다.\n", ave_seek_time);
    for (int i = 0; i < count; i++) {
        deviation[i] = (seek_distance[i] * seek_time) - ave_seek_time;
        printf("각 %d번째 탐색시간 %d에 대한 편차는 %lf입니다.", i + 1,
seek_distance[i] * seek_time, deviation[i]);
        deviation2[i] = (deviation[i] * deviation[i]);
        printf("이 편차의 제곱은 %lf입니다.\n", deviation2[i]);
        total_deviation2 = total_deviation2 + deviation2[i];
    }
    variance = total_deviation2 / count;
    printf("총 편차의 합이 %lf이므로 분산은 %lf입니다.", total_deviation2,
variance);

    standard_deviation = sqrt(variance);
    printf("그러므로 표준편차는 %lf입니다.", standard_deviation);
}
else if (which == 3) { //LOOK
    printf("LOOK기법을 선택하셨습니다.");
    printf("\n디스크 헤더의 초기위치를 입력하세요:(0부터 199까지):");
    scanf("%d", &headerposition);
    printf("seek_time 기준값을 입력해주세요:");
    scanf("%d", &seek_time);
    srand(time(NULL));
    printf("무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터~최대
10까지));

    count = rand() % 9 + 1;
    printf("트랙의 개수가 %d로 정해졌습니다.", count);
    printf("실린더의 범위를 0~199로 한정했습니다.");
    printf("\n%d개가 랜덤으로정해집니다.", count);
    for (int i = 0; i < count; i++) {
        track[i] = rand() % 200;
        look_track[i] = track[i];
    }
    printf("\n정해진 트랙번호의 순서: ");
    for (int i = 0; i < count; i++) {

```



```

        printf("%d ", track[i]);
    }
    printf("\n오름차순으로 재배열된 순서:");    //headerposition 보다 큰거랑
작은거 나눠서 재배열하기.
    for (int i = 0; i < count; i++) {
        for (int j = 0; j < count - 1; j++) {
            if (look_track[j] > look_track[j+1]) {
                just = look_track[j];
                look_track[j] = look_track[j + 1];
                look_track[j+1] = just;
            }
        }
    }
    for (int i = 0; i < count; i++) {
        printf("%d ", look_track[i]);
    }
    printf("\n어느 방향으로 헤드를 시작할지 방향을 정하세요.(1.올라가는방향
2.내려가는방향):");

```

```

scanf("%d", &direction);
remain = 0;
number = 0;
if (direction == 1) {    //올라가는방향일때
    printf("LOOK기법을 선택고 방향은 올라가는방향이므로 순서는:");
    for (int i = 0; i < count; i++) {    //헤더포지션보다 큰값들을

```

순서대로

```

        if (headerposition <= look_track[i]) {
            look_order[remain] = look_track[i];
            remain++;
        }
    }
    number = remain;
    for (int i = 0; i < count; i++) {    //헤더포지션보다 작은값들을

```

순서대로

```

        if (headerposition > look_track[i]) {
            look_order[remain] = look_track[i];
            remain++;
        }
    }
    for (int i = number; i < count; i++) {    //오름차순 정렬
        for (int j = number; j < count - 1; j++) {
            if (look_order[j] < look_order[j + 1]) {

```

```

        just = look_order[j];
        look_order[j] = look_order[j + 1];
        look_order[j + 1] = just;
    }
}
}
for (int i = 0; i < count; i++) {
    printf("%d ", look_order[i]);
}
for (int i = 0; i < count; i++) {
    if (i == 0) {
        look_distance[i] = abs(headerposition -
look_order[i]);
    }
    else {
        look_distance[i] = abs(look_order[i] -
look_order[i - 1]);
    }
}
printf("\n총 이동거리는: ");
for (int i = 0; i < count; i++) {
    total_seek_distance = total_seek_distance +
look_distance[i];
}
printf("%d", total_seek_distance);
printf("\nseek_time이 %d이므로 총 탐색시간은 %d입니다.",
seek_time, total_seek_distance * seek_time);
}
else if (direction == 2) { //내려가는방향
    printf("LOOK기법을 선택했고 방향은 내려가는방향이므로
순서는:");

    for (int i = 0; i < count; i++) {
        if (headerposition >= look_track[i]) {
            look_order[remain] = look_track[i];
            remain++;
        }
    }
    number = remain;
    for (int i = 0; i < count; i++) {
        if (headerposition < look_track[i]) {
            look_order[remain] = look_track[i];

```

```

        remain++;
    }
}
for (int i = 0; i < number; i++) {
    for (int j = 0; j < number - 1; j++) {
        if (look_order[j] < look_order[j + 1]) {
            just = look_order[j];
            look_order[j] = look_order[j + 1];
            look_order[j + 1] = just;
        }
    }
}
for (int i = 0; i < count; i++) {
    printf("%d ", look_order[i]);
}
for (int i = 0; i < count; i++) {
    if (i == 0) {
        look_distance[i] = abs(headerposition -
look_order[i]);
    }
    else {
        look_distance[i] = abs(look_order[i] -
look_order[i - 1]);
    }
}
printf("\n총 이동거리는: ");
for (int i = 0; i < count; i++) {
    total_seek_distance = total_seek_distance +
look_distance[i];
}
printf("%d", total_seek_distance);
printf("\nseek_time이 %d이므로 총 탐색시간은 %d입니다",
seek_time, total_seek_distance * seek_time);
}
else {
    printf("올바르지 않은 방향입니다.");
}
printf("즉 회전지연시간이 없으므로 총 실행시간은 %d입니다.",
total_seek_distance * seek_time);
ave_seek_time = (total_seek_distance * seek_time) / count;
printf("평균 탐색시간은 %lf입니다.\n", ave_seek_time);

```

```

        for (int i = 0; i < count; i++) {
            deviation[i] = (look_distance[i] * seek_time) - ave_seek_time;
            printf("각 %d 번째 탐색시간 %d에 대한 편차는 %lf입니다.", i +
1, look_distance[i] * seek_time, deviation[i]);
            deviation2[i] = (deviation[i] * deviation[i]);
            printf("이 편차의 제곱은 %lf입니다.\n", deviation2[i]);
            total_deviation2 = total_deviation2 + deviation2[i];
        }
        variance = total_deviation2 / count;
        printf("총 편차의 합이 %lf이므로 분산은 %lf 입니다.", total_deviation2,
variance);

        standard_deviation = sqrt(variance);
        printf("\n그러므로 표준편차는 %lf입니다.", standard_deviation);
    }
    else if (which == 4) { //섹터큐잉
        printf("섹터큐잉 기법을 선택하셨습니다.");
        printf("\n디스크 헤더의 초기위치를 입력하세요:(0부터 199까지):");
        scanf("%d", &headerposition);
        printf("디스크의 rpm 기준값을 입력해주세요(5400,6000,7200 등등):");
        scanf("%lf", &rpm);
        srand(time(NULL));
        printf("무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터~최대
10까지).");

        count = rand() % 9 + 1;
        printf("트랙의 개수가 %d로 정해졌습니다.", count);
        printf("실린더의 범위를 0~199로 한정했습니다.");
        printf("%d개가 랜덤으로정해집니다.", count);
        for (int i = 0; i < count; i++) {
            track[i] = rand() % 200;
        }
        printf("\n정해진 트랙번호의 순서:");
        for (int i = 0; i < count; i++) {
            printf("%d ", track[i]);
        }
        printf("\n");
        double sec_rpm = rpm / 60; //초당 회전수
        double ms_rpm = sec_rpm / 1000; //ms당 회전수
        double clock = 1 / ms_rpm; //1회전하는데 걸리는시간 단위 ms
        printf("디스크가 한바퀴 회전하는데 걸린시간 %lf입니다.", clock);
        int sector_number = 0; //섹터개수
        printf("\n섹터큐잉을 선택하셨습니다.트랙이 하나이므로 seek time이

```

```

없습니다. 섹터의 개수를 입력하세요(4,8,10중 입력):");
scanf("%d", &sector_number);
if ((sector_number == 4) || (sector_number == 8) || (sector_number
== 10)) {
    printf("올바른 섹터의 개수입니다.섹터의 개수 %d를
입력하셨으므로 0~199까지를 분류하겠습니다. ", sector_number);
}
else {
    printf("섹터 오류입니다.");
}
if (sector_number == 4) {
    int width = 200 / 4;    //한섹터당
    int counter = 0;
    int left = 0;    //가로세로
    int up = 0;    //위아래
    int headersector = 0;    //처음 입력한 헤더섹터의 위치
    double real_clock = clock / 4; //섹터하나도는데 걸리는시간;
    printf("\n섹터가 4개입니다.");
    for (int j = 0; j < 4; j++) {    //헤드섹터구하기
        if (j * width <= headerposition && headerposition <=
(j + 1) * width - 1) {
            headersector = j;
        }
    }
    printf("초기에 헤더포지션을 %d로 선택하셨습니다.하지만 sector의
개수를 4개로 정하셨으니, 현재 헤더의 위치는 섹터%d입니다\n", headerposition,
headersector);

    int sector[4][10] = { 0, };
    int sector_number[4][10] = { 0, };
    int sector_order[10] = { 0, };
    for (int j = 0; j < 4; j++) { //섹터에맞게 분할하기
        for (int i = 0; i < count; i++) {
            if (j * width <= track[i] && track[i] <= (j +
1) * width - 1) {
                sector[left][up] = track[i];
                sector_number[left][up] = j + 1;
                up++;
            }
        }
        left++;
        up = 0;
    }
}

```

```

    }
    printf("0의 값은 트랙이 없다는것입니다.\n");
    for (int i = 0; i < 4; i++) {
        printf("%d에서부터 %d까지는 섹터 %d \t ", i * width,
(i + 1) * width - 1, i);

        for (int j = 0; j < 10; j++) {
            printf("%d ", sector[i][j]);
        }
        printf("\n");
    }
    printf("큐를 섹터에 따라 분류했으니 섹터큐잉에 따른 순서는: ");
    counter = 0;
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 4; j++) {
            if (sector[j][i] != 0) {
                sltf_track[counter] = sector[j][i];
                sector_order[counter] =
sector_number[j][i];

                counter++;
            }
        }
    }
    for (int i = 0; i < count; i++) {
        printf("%d ", sltf_track[i]);
    }
    printf("\n");
    printf("디스크가 한바퀴 회전하는데 걸린시간 %lf입니다.\n즉
1/4바퀴 도는데 걸리는 시간은 %lf이고 \n2/4바퀴 도는데 걸리는 시간은 %lf이고 \n3/4바퀴
도는데 걸리는 시간은 %lf입니다. \n", clock,clock/4,clock/2,clock*3/4);
    printf("그렇다면 회전지연시간은? 처음 헤더위치가 %d섹터이므로
", headersector);

    for (int i = 0; i < count; i++) {
        if (i == 0) {
            if (sector_order[i] >= headersector) {
                rotational_latency[i] = real_clock *
abs(sector_order[i] - headersector);

            }
            else {
                rotational_latency[i] = real_clock *
(4 - abs(sector_order[i] - headersector));
            }
        }
    }
}

```

```

        }
        else {
            if (sector_order[i] >= sector_order[i - 1]) {
                rotational_latency[i] = real_clock *
abs(sector_order[i] - sector_order[i - 1]);
            }
            else {
                rotational_latency[i] = real_clock *
(4 - abs(sector_order[i] - sector_order[i - 1]));
            }
        }
    }
    printf("\n");
    for (int i = 0; i < count; i++) {
        printf("%lf ", rotational_latency[i]);
    }
    printf("\n");
    for (int i = 0; i < count; i++) {
        total_rotational_latency = total_rotational_latency +
rotational_latency[i];
    }
    printf("전체 회전시간은 %lf입니다", total_rotational_latency);
}
else if (sector_number == 8) {
    int width = 200 / 8;
    int counter = 0;
    int left = 0;    //가로세로
    int up = 0;      //위아래
    int headersector = 0;
    double real_clock = clock / 8; //섹터하나도는데 걸리는시간;
    printf("\n섹터가 8개입니다.");
    for (int j = 0; j < 8; j++) {
        if (j * width <= headerposition && headerposition <=
(j + 1) * width - 1) {
            headersector = j;
        }
    }
    printf("초기에 헤더포지션을 %d로 선택하셨습니다.하지만 sector의
개수를 8개로 정하셨으니,현재 헤더의 위치는 섹터%d입니다\n", headerposition,
headersector);

    int sector[8][10] = { 0, };

```

```

int sector_number[8][10] = { 0, };
int sector_order[10] = { 0, };
for (int j = 0; j < 8; j++) {
    for (int i = 0; i < count; i++) {
        if (j * width <= track[i] && track[i] <= (j +
1) * width - 1) {

            sector[left][up] = track[i];
            sector_number[left][up] = j + 1;
            up++;

        }
        left++;
        up = 0;
    }
    printf("0은 트랙이 없는겁니다.\n");
    for (int i = 0; i < 8; i++) {
        printf("%d에서부터 %d까지는 섹터 %d \t ", i * width,
(i + 1) * width - 1, i);

        for (int j = 0; j < 10; j++) {
            printf("%d ", sector[i][j]);
        }
        printf("\n");
    }
    printf("큐를 섹터에 따라 분류했으니 섹터큐잉에 따른 순서는:");
    counter = 0;
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 8; j++) {
            if (sector[j][i] != 0) {
                sltf_track[counter] = sector[j][i];
                sector_order[counter] =
sector_number[j][i];

                counter++;
            }
        }
    }
    for (int i = 0; i < count; i++) {
        printf("%d ", sltf_track[i]);
    }
    printf("\n디스크가 한바퀴 회전하는데 걸린시간 %lf입니다.\n즉
1/8바퀴 도는데 걸리는 시간은 %lf이고 \n2/8바퀴 도는데 걸리는 시간은 %lf이고 \n3/8바퀴
도는데 걸리는 시간은 %lf이고\n", clock, clock / 8, clock*2 / 8, clock * 3 / 8);

```



```

        printf("4/8바퀴 도는데 걸리는 시간은 %lf이고\n5/8바퀴 도는데
        걸리는 시간은 %lf이고 \n6/8바퀴 도는데 걸리는 시간은 %lf이고\n7/8바퀴 도는데 걸리는
        시간은 %lf입니다\n", clock*4/8, clock*5 / 8, clock * 6 / 8, clock * 7 / 8);
        printf("그렇다면 회전지연시간은? 처음 헤더위치가 %d섹터이므로
        ", headersector);

        for (int i = 0; i < count; i++) {
            if (i == 0) {
                if (sector_order[i] >= headersector) {
                    rotational_latency[i] = real_clock *
abs(sector_order[i] - headersector);
                }
                else {
                    rotational_latency[i] = real_clock *
(8 - abs(sector_order[i] - headersector));
                }
            }
            else {
                if (sector_order[i] >= sector_order[i - 1]) {
                    rotational_latency[i] = real_clock *
abs(sector_order[i] - sector_order[i - 1]);
                }
                else {
                    rotational_latency[i] = real_clock *
(8 - abs(sector_order[i] - sector_order[i - 1]));
                }
            }
        }
        printf("\n");
        for (int i = 0; i < count; i++) {
            printf("%lf ", rotational_latency[i]);
        }
        printf("\n");
        for (int i = 0; i < count; i++) {
            total_rotational_latency = total_rotational_latency +
rotational_latency[i];
        }
        printf("전체 회전시간은 %lf입니다", total_rotational_latency);
    }
    else {
        int width = 200 / 10;
        int counter = 0;

```

```

int left = 0;    //가로세로
int up = 0;     //위아래
int headersector = 0;
double real_clock = clock / 10; //섹터하나도는데 걸리는시간;
printf("\n섹터가 10개입니다.");
for (int j = 0; j < 10; j++) {
    if (j * width <= headerposition && headerposition <=
(j + 1) * width - 1) {
        headersector = j;
    }
}
printf("초기에 헤더포지션을 %d로 선택하셨습니다.하지만 sector의
개수를 10개로 정하셨으니, 현재 헤더의 위치는 섹터%d입니다\n", headerposition,
headersector);

int sector[10][10] = { 0, };
int sector_number[10][10] = { 0, };
int sector_order[10] = { 0, };
for (int j = 0; j < 10; j++) {
    for (int i = 0; i < count; i++) {
        if (j * width <= track[i] && track[i] <= (j +
1) * width - 1) {
            sector[left][up] = track[i];
            sector_number[left][up] = j + 1;
            up++;
        }
    }
    left++;
    up = 0;
}
printf("0인것은 트랙번호가 없단 것입니다.\n");
for (int i = 0; i < 10; i++) {
    printf("%d에서부터 %d까지는 섹터 %d \t ", i * width,
(i + 1) * width - 1, i);

    for (int j = 0; j < 10; j++) {
        printf("%d ", sector[i][j]);
    }
    printf("\n");
}
printf("큐를 섹터에 따라 분류했으니 섹터큐잉에 따른 순서는:");
counter = 0;
for (int i = 0; i < 10; i++) {

```

```

        for (int j = 0; j < 10; j++) {
            if (sector[j][i] != 0) {
                sltf_track[counter] = sector[j][i];
                sector_order[counter] =
sector_number[j][i];

                counter++;
            }
        }
        for (int i = 0; i < count; i++) {
            printf("%d ", sltf_track[i]);
        }
        printf("\n디스크가 한바퀴 회전하는데 걸린시간 %lf입니다.\n즉
1/10바퀴 도는데 걸리는 시간은 %lf이고 \n2/10바퀴 도는데 걸리는 시간은 %lf이고
\n3/10바퀴 도는데 걸리는 시간은 %lf이고\n", clock, clock / 10, clock * 2 / 10, clock
* 3 / 10);

        printf("4/10바퀴 도는데 걸리는 시간은 %lf이고 \n5/10바퀴
도는데 걸리는 시간은 %lf이고 \n6/10바퀴 도는데 걸리는 시간은 %lf이고\n7/10바퀴 도는데
걸리는 시간은 %lf이고\n", clock * 4 / 10, clock * 5 / 10, clock * 6 / 10, clock * 7
/ 10);

        printf("8/10바퀴 도는데 걸리는 시간은 %lf이고 \n9/10바퀴
도는데 걸리는 시간은 %lf입니다.\n", clock * 8 / 10, clock * 9 / 10);
        printf("그렇다면 회전지연시간은? 처음 헤더위치가 %d섹터이므로
", headersector);

        for (int i = 0; i < count; i++) {
            if (i == 0) {
                if (sector_order[i] >= headersector) {
                    rotational_latency[i] = real_clock *
abs(sector_order[i] - headersector);
                }
                else {
                    rotational_latency[i] = real_clock *
(10 - abs(sector_order[i] - headersector));
                }
            }
            else {
                if (sector_order[i] >= sector_order[i - 1]) {
                    rotational_latency[i] = real_clock *
abs(sector_order[i] - sector_order[i - 1]);
                }
                else {

```

```

        rotational_latency[i] = real_clock *
(10 - abs(sector_order[i] - sector_order[i - 1]));
    }
}

}
printf("\n");
for (int i = 0; i < count; i++) {
    printf("%lf ", rotational_latency[i]);
}
printf("\n");
for (int i = 0; i < count; i++) {
    total_rotational_latency = total_rotational_latency +
rotational_latency[i];
}
printf("전체 회전시간은 %lf입니다", total_rotational_latency);
}
printf("총 회전지연시간은 %lf입니다. 즉 탐색시간이 없으므로 총 실행시간은
%lf입니다.", total_rotational_latency, total_rotational_latency);
ave_rotational_latency = total_rotational_latency / count;
printf("평균 회전지연시간은 %lf입니다.\n", ave_rotational_latency);
for (int i = 0; i < count; i++) {
    deviation[i] = rotational_latency[i] - ave_rotational_latency;
    printf("각 %d 번째 회전지연시간%lf 에 대한 편차는 %lf입니다.",
i + 1, rotational_latency[i], deviation[i]);
    deviation2[i] = (deviation[i] * deviation[i]);
    printf("이 편차의 제곱은 %lf입니다.\n", deviation2[i]);
    total_deviation2 = total_deviation2 + deviation2[i];
}
variance = total_deviation2 / count;
printf("총 편차의 합이 %lf이므로 분산은 %lf 입니다.", total_deviation2,
variance);

standard_deviation = sqrt(variance);
printf("그러므로 표준편차는 %lf입니다.", standard_deviation);
printf("\n\n");
}
else if (which == 5) { //시기의기법
    printf("시기의기법을 선택하셨습니다. 교수님의 수업을 들던중 P-SCAN과
SATF기법은 있지만 우선순위와 데이터 전송시간을 동시에 고려한 기법은 아직
없는것같았다.\n\n시기의기법: 아무리 우선순위가 높아도 데이터전송시간이 ");
    printf("느려지면 뒤에 있는 것들이 늦게걸리므로 우선순위와 데이터크기를

```

```

조화롭게 처리할수잇는 방법을 선택해보았다. ");
    printf("\n시기의기법을 선택하셨습니다.");
    printf("\n디스크 헤더의 초기위치를 입력하세요:(0부터 199까지):");
    scanf("%d", &headerposition);
    printf("seek_time 기준값을 입력해주세요:");
    scanf("%d", &seek_time);
    printf("디스크의 rpm 기준값(디스크의 분당 회전수)을
입력해주세요(요새하드디스크는 3600rpm부터 5400rpm 7200rpm):");
    scanf("%lf", &rpm);
    srand(time(NULL));
    printf("무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터~최대
10까지).");

    count = rand() % 9 + 1;
    printf("트랙의 개수가 %d로 정해졌습니다.", count);
    printf("실린더의 범위를 0~199로 한정했습니다.");
    printf("\n%d개가 랜덤으로정해집니다.", count);
    int a[10][4] = { 0, }; //트랙, 우선순위 데이터
    for (int i = 0; i < count; i++) {
        a[i][0] = rand() % 200;
    }
    printf("\n정해진 트랙번호의 순서: ");
    for (int i = 0; i < count; i++) {
        printf("%d ", a[i][0]);
    }
    double sec_rpm = rpm / 60; //초당 회전수
    double ms_rpm = sec_rpm / 1000; //ms초당 회전수
    double one_time = 1 / ms_rpm; //1바퀴당 걸리는 ms
    printf("\n디스크가 1초당 회전하는 횟수는 %lf바퀴입니다.", sec_rpm);
    printf("디스크가 1ms초당 회전하는 횟수는 %lf바퀴입니다.", ms_rpm);
    printf("디스크가 1바퀴를 도는데걸리는 시간은 %lf ms입니다.", one_time);
    double track_byte = 0;
    printf("\n트랙크기를 정해주세요(10^3 이상 10^8이하):");
    scanf("%lf", &track_byte);
    printf("한트랙당 크기는 %lf byte입니다.", track_byte);
    double dump = one_time / 1000;
    printf("\n디스크가 %lf초당 전송하는 양은 %lf byte입니다.", one_time /
1000, track_byte);
    printf("\n디스크가 1초당 전송하는 양은 %lf byte입니다.", track_byte *
(1 / dump));

    double mega = track_byte * (1 / dump) / 1000000;
    printf("\n디스크가 1초당 전송하는 양은 %lf Mbyte입니다.", mega);

```

```

printf("\n이제 각각의 우선순위와 데이터크기를 입력합니다.\n");
printf("트랙번호 우선순위 데이터크기(단위 Mbyte)(입력하고 tab을
눌러주세요)\n");
for (int i = 0; i < count; i++) {
    printf("%d의 우선순위 입력, 데이터크기 입력: ", a[i][0]);
    scanf("%d %d", &a[i][1], &a[i][2]);
}
for (int i = 0; i < count; i++) {
    a[i][3] = a[i][1] * a[i][2];
}

printf("\n~~~~~\n");
);
for (int i = 0; i < count; i++) {
    printf("%d의 우선순위 %d 데이터크기 %d Mbyte 결론 %d \n",
a[i][0], a[i][1], a[i][2], a[i][3]);
}
int user = 0;
int trash3 = 0;
int trash2 = 0;
int trash1 = 0;
int trash0 = 0;
for (int i = 0; i < count; i++) { //결론에 따른 순서 변경
    for (int j = 0; j < count - 1; j++) {
        if (a[j][3] > a[j + 1][3]) {
            trash3 = a[j][3];
            a[j][3] = a[j + 1][3];
            a[j + 1][3] = trash3;
            trash2 = a[j][2];
            a[j][2] = a[j + 1][2];
            a[j + 1][2] = trash2;
            trash1 = a[j][1];
            a[j][1] = a[j + 1][1];
            a[j + 1][1] = trash1;
            trash0 = a[j][0];
            a[j][0] = a[j + 1][0];
            a[j + 1][0] = trash0;
        }
    }
}
}

```

```

printf("~~~~~\n우선순위와
데이터의 양에 따른 순서를 정하면:\n");
    for (int i = 0; i < count; i++) {
        printf("%d번째로는 결론이 %d인 %d가 시작된다.\n",i+1, a[i][3],
a[i][0]);
    }
    for (int i = 0; i < count; i++) {
        if (i == 0) {
            seek_distance[i] = abs(headerposition - a[i][0]);
        }
        else {
            seek_distance[i] = abs(a[i-1][0] - a[i][0]);
        }
    }
    printf("~~~~~\n");
    for (int i = 0; i < count; i++) {
        if (i == 0) {
            printf("%d 에서 %d로 이동한다. 즉 이동거리는%d
이다\n",headerposition,a[i][0],seek_distance[i]);
        }
        else {
            printf("%d 에서 %d로 이동한다.즉 이동거리는
%d이다\n", a[i-1][0], a[i][0],seek_distance[i]);
        }
    }
    for (int i = 0; i < count; i++) {
        total_seek_distance = total_seek_distance + seek_distance[i];
    }
    printf("\n전체 이동거리는 %d입니다.\n", total_seek_distance);
    double taesik_seek_time[10] = { 0, };
    for (int i = 0; i < count; i++) {
        taesik_seek_time[i] = seek_distance[i] * seek_time;
    }
    double transport_time[10] = { 0, };//데이터 전송시간
    for (int i = 0; i < count; i++) {
        transport_time[i] = a[i][2] / mega;
        printf("%d번째의 %d Mbyte 데이터 전송시간은 %lf이다.\n", i +
1, a[i][2], transport_time[i]);
    }
    double response_time[10] = { 0, };//응답시간
    printf("~~~~~\n");

```

```

        for (int i = 0; i < count; i++) {
            if (i == 0) {
                response_time[i] = taesik_seek_time[i] +
transport_time[i];
            }
            else {
                response_time[i] = taesik_seek_time[i] +
transport_time[i] + response_time[i - 1];
            }

            printf("%d번째의 %d 번호의 응답시간 %lf이다.\n", i + 1, a[i][0],
response_time[i]);
        }
        printf("\n총 응답시간은 %lf입니다. 즉 회전지연시간이 없으므로 총
응답시간은 %lf입니다.", response_time[count-1], response_time[count-1]);
        double ave_response_time = 0;
        ave_response_time = response_time[count-1] / count;
        printf("\n평균 응답시간은 %lf입니다.\n", ave_response_time);
        for (int i = 0; i < count; i++) {
            deviation[i] = response_time[i] - ave_response_time;
            printf("각 %d 번째 응답시간%lf 에 대한 편차는 %lf입니다.", i +
1, response_time[i], deviation[i]);
            deviation2[i] = (deviation[i] * deviation[i]);
            printf("이 편차의 제곱은 %lf입니다.\n", deviation2[i]);
            total_deviation2 = total_deviation2 + deviation2[i];
        }
        variance = total_deviation2 / count;
        printf("총 편차의 합이 %lf이므로 분산은 %lf 입니다.", total_deviation2,
variance);

        standard_deviation = sqrt(variance);
        printf("그러므로 표준편차는 %lf입니다.", standard_deviation);
    }
    else {
        printf("당신이 선택한 기법은 없는 디스크 스케줄링 기법입니다.");
    }
    return 0;
}

```



### 3. 실험결과 분석

아직은 코드를 짜는것에 대해 미흡하고 솔직히 이 디스크 스케줄링 기법에 있어서 정확히 이해를 하고있는 것이 아닌것같다. 그래도 열심히 해서 코드를 작성하고 돌려보았고 곧 다가올 여름방학때 좀더 공부를 할생각이다.

#### 1. FCFS기법

```
C:\Windows\system32\cmd.exe
어떤 디스크 스케줄링 기법을 선택하시겠습니까?
(1 FCFS기법 2 SSTF기법 3 LOOK기법 4 선택큐잉 기법 5 시기의기법):1
FCFS기법을 선택하셨습니다.
디스크 헤드의 초기위치를 입력하세요:(0부터 199까지):123
seek_time 기준값을 입력해주세요:5
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지)
트랙의 개수가 4로 정해졌습니다. 실린더의 범위를 0-199로 한정했습니다. 이제 4개가 랜덤으로정해집니다.
정해진 트랙번호의 순서: 156, 33, 75, 9,
FCFS기법을 선택했으므로 순서는 들어온 순서대로이므로:
1번째가 156입니다.
2번째가 33입니다.
3번째가 75입니다.
4번째가 9입니다.
123과 156의 이동거리는 33입니다.
33과 156의 이동거리는 123입니다.
75와 33의 이동거리는 42입니다.
9와 75의 이동거리는 66입니다.
전체 이동거리는 249입니다.
seek_time이 5이므로 총 탐색시간은 1320입니다. 즉 회전지연시간이 없으므로 총 실행시간은 1320입니다. 그리고 평균 탐색시간은 330.000000입니다.
1번째 탐색시간 156에 대한 편차는 -165.000000입니다. 이 편차의 제곱은 27225.000000입니다.
2번째 탐색시간 33에 대한 편차는 -285.000000입니다. 이 편차의 제곱은 81225.000000입니다.
3번째 탐색시간 75에 대한 편차는 -120.000000입니다. 이 편차의 제곱은 14400.000000입니다.
4번째 탐색시간 9에 대한 편차는 0.000000입니다. 이 편차의 제곱은 0.000000입니다.
총 편차의 합이 123330.000000이므로 분산은 30712.500000 입니다. 그러므로 실행시간의 표준편차는 175.249622입니다. 계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
어떤 디스크 스케줄링 기법을 선택하시겠습니까?
(1 FCFS기법 2 SSTF기법 3 LOOK기법 4 선택큐잉 기법 5 시기의기법):1
FCFS기법을 선택하셨습니다.
디스크 헤드의 초기위치를 입력하세요:(0부터 199까지):171
seek_time 기준값을 입력해주세요:3
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지)
트랙의 개수가 8로 정해졌습니다. 실린더의 범위를 0-199로 한정했습니다. 이제 8개가 랜덤으로정해집니다.
정해진 트랙번호의 순서: 31, 79, 113, 32, 100, 185, 17, 121,
FCFS기법을 선택했으므로 순서는 들어온 순서대로이므로:
1번째가 31입니다.
2번째가 79입니다.
3번째가 113입니다.
4번째가 32입니다.
5번째가 100입니다.
6번째가 185입니다.
7번째가 17입니다.
8번째가 121입니다.
171와 31의 이동거리는 140입니다.
79와 31의 이동거리는 48입니다.
113와 79의 이동거리는 34입니다.
32와 113의 이동거리는 81입니다.
100과 32의 이동거리는 68입니다.
185와 100의 이동거리는 85입니다.
17와 185의 이동거리는 168입니다.
121과 17의 이동거리는 104입니다.
전체 이동거리는 728입니다.
seek_time이 3이므로 총 탐색시간은 2184입니다. 즉 회전지연시간이 없으므로 총 실행시간은 2184입니다. 그리고 평균 탐색시간은 273.000000입니다.
1번째 탐색시간 420에 대한 편차는 147.000000입니다. 이 편차의 제곱은 21609.000000입니다.
2번째 탐색시간 144에 대한 편차는 -129.000000입니다. 이 편차의 제곱은 16641.000000입니다.
3번째 탐색시간 102에 대한 편차는 -171.000000입니다. 이 편차의 제곱은 29241.000000입니다.
4번째 탐색시간 243에 대한 편차는 -30.000000입니다. 이 편차의 제곱은 900.000000입니다.
5번째 탐색시간 204에 대한 편차는 -59.000000입니다. 이 편차의 제곱은 4781.000000입니다.
6번째 탐색시간 255에 대한 편차는 -18.000000입니다. 이 편차의 제곱은 324.000000입니다.
7번째 탐색시간 504에 대한 편차는 231.000000입니다. 이 편차의 제곱은 53361.000000입니다.
8번째 탐색시간 312에 대한 편차는 39.000000입니다. 이 편차의 제곱은 1521.000000입니다.
총 편차의 합이 123339.000000이므로 분산은 16044.750000 입니다. 그러므로 실행시간의 표준편차는 126.667873입니다. 계속하려면 아무 키나 누르십시오 . . .
```

#### 2. SSTF기법

```
C:\Windows\system32\cmd.exe
어떤 디스크 스케줄링 기법을 선택하시겠습니까?
(1 FCFS기법 2 SSTF기법 3 LOOK기법 4 선택큐잉 기법 5 시기의기법):2
SSTF기법을 선택하셨습니다.
디스크 헤드의 초기위치를 입력하세요:(0부터 199까지):111
seek_time 기준값을 입력해주세요:2
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지)
트랙의 개수가 7로 정해졌습니다. 실린더의 범위를 0-199로 한정했습니다. 7개가 랜덤으로정해집니다.
정해진 트랙번호: 141, 6, 79, 3, 192, 199, 143
SSTF기법을 선택했으므로 순서는:
1번째가 141로 제일 작은값으로
2번째가 143으로 제일 작은값으로
3번째가 192로 제일 작은값으로
4번째가 199로 제일 작은값으로
5번째가 79로 제일 작은값으로
6번째가 6로 제일 작은값으로
7번째가 3로 제일 작은값으로
전체 이동거리는 284
seek_time이 2이므로 총 탐색시간은 568입니다. 즉 회전지연시간이 없으므로 총 실행시간은 568입니다. 평균 탐색시간은 81.000000
입니다.
1번째 탐색시간 60에 대한 편차는 -21.000000입니다. 이 편차의 제곱은 441.000000입니다.
2번째 탐색시간 4에 대한 편차는 -77.000000입니다. 이 편차의 제곱은 5929.000000입니다.
3번째 탐색시간 98에 대한 편차는 17.000000입니다. 이 편차의 제곱은 289.000000입니다.
4번째 탐색시간 14에 대한 편차는 -57.000000입니다. 이 편차의 제곱은 4489.000000입니다.
5번째 탐색시간 24에 대한 편차는 159.000000입니다. 이 편차의 제곱은 25281.000000입니다.
6번째 탐색시간 146에 대한 편차는 65.000000입니다. 이 편차의 제곱은 4225.000000입니다.
7번째 탐색시간 5에 대한 편차는 -75.000000입니다. 이 편차의 제곱은 5625.000000입니다.
총 편차의 합이 46279.000000이므로 분산은 6611.285714입니다. 그러므로 표준편차는 81.308119입니다. 계속하려면 아무 키나 누르
십시오 . . .
```

```

C:\Windows\system32\cmd.exe
어떤 디스크의 스케줄링 기법을 선택하시겠습니까?
(1. FDFS기법 2. SSTF기법 3. LOOK기법 4. 선택큐잉 기법 5. 시기의기법):2
SSTF기법을 선택하셨습니다.
디스크의 헤더의 순기번호를 입력하세요.(0부터 199까지):100
seek_time 기준값을 입력해주세요:3
무작위로 트래크의 개수를 정해드리겠습니다(최소 1부터-최대 10까지)
트래크의 개수가 8로 정해졌습니다.실린더의 범위를 0-199로 한정했습니다.8개가 랜덤으로정해집니다.
정해진 트래크번호:34 178 181 32 178 50 19 149
SSTF기법을 선택했으므로 순서는
1번짜리 149로
2번짜리 178로
3번짜리 178로
4번짜리 181로
5번짜리 50로
6번짜리 34로
7번짜리 32로
8번짜리 19로
이동거리:243
seek_time이 30이므로 총 탐색시간은 729입니다.즉 회전지연시간이 없으므로 총 실행시간은 729입니다.평균 탐색시간은 91.000000입니다.
1번짜리 탐색시간 147에 대한 편차는 56.000000입니다.이 편차의 제곱은 3136.000000입니다.
2번짜리 탐색시간 81에 대한 편차는 -10.000000입니다.이 편차의 제곱은 100.000000입니다.
3번짜리 탐색시간 61에 대한 편차는 -45.000000입니다.이 편차의 제곱은 2025.000000입니다.
4번짜리 탐색시간 81에 대한 편차는 -32.000000입니다.이 편차의 제곱은 1024.000000입니다.
5번짜리 탐색시간 33에 대한 편차는 302.000000입니다.이 편차의 제곱은 91204.000000입니다.
6번짜리 탐색시간 46에 대한 편차는 -43.000000입니다.이 편차의 제곱은 1849.000000입니다.
7번짜리 탐색시간 3에 대한 편차는 -95.000000입니다.이 편차의 제곱은 9025.000000입니다.
8번짜리 탐색시간 39에 대한 편차는 -52.000000입니다.이 편차의 제곱은 2704.000000입니다.
총 편차의 합이 120167.000000이므로 분산은 15020.875000입니다.그러므로 표준편차는 122.559679입니다.계속하려면 아무 키나 누르십시오 . . .

```

### 3. LOOK기법

#### (1)올라가는 방향

```

C:\Windows\system32\cmd.exe
어떤 디스크의 스케줄링 기법을 선택하시겠습니까?
(1. FDFS기법 2. SSTF기법 3. LOOK기법 4. 선택큐잉 기법 5. 시기의기법):3
LOOK기법을 선택하셨습니다.
디스크의 헤더의 순기번호를 입력하세요.(0부터 199까지):100
seek_time 기준값을 입력해주세요:10
무작위로 트래크의 개수를 정해드리겠습니다(최소 1부터-최대 10까지)트래크의 개수가 8로 정해졌습니다.실린더의 범위를 0-199로 한정했습니다.
정해진 트래크번호의 순서:3 125 8 152 89 192 143 183
오름차순으로 재배열된 순서:3 8 89 125 143 152 183 192
이는 방향으로 헤드를 시작할지 방향을 정하세요.(1. 올라가는방향 2. 내려가는방향):1
LOOK기법을 선택하고 방향은 올라가는방향이므로 순서는:125 143 152 183 192 89 8 3
총 이동거리:281
seek_time이 100이므로 총 탐색시간은 2810입니다.즉 회전지연시간이 없으므로 총 실행시간은 2810입니다.평균 탐색시간은 351.000000입니다.
1번짜리 탐색시간 250에 대한 편차는 -101.000000입니다.이 편차의 제곱은 10201.000000입니다.
2번짜리 탐색시간 180에 대한 편차는 -171.000000입니다.이 편차의 제곱은 29241.000000입니다.
3번짜리 탐색시간 90에 대한 편차는 -261.000000입니다.이 편차의 제곱은 68121.000000입니다.
4번짜리 탐색시간 110에 대한 편차는 -41.000000입니다.이 편차의 제곱은 1681.000000입니다.
5번짜리 탐색시간 90에 대한 편차는 -261.000000입니다.이 편차의 제곱은 68121.000000입니다.
6번짜리 탐색시간 1030에 대한 편차는 679.000000입니다.이 편차의 제곱은 460141.000000입니다.
7번짜리 탐색시간 810에 대한 편차는 439.000000입니다.이 편차의 제곱은 210681.000000입니다.
8번짜리 탐색시간 50에 대한 편차는 -391.000000입니다.이 편차의 제곱은 90681.000000입니다.
총 편차의 합이 939688.000000이므로 분산은 117461.000000입니다.
그러므로 표준편차는 342.725838입니다.계속하려면 아무 키나 누르십시오 . . .

```

```

C:\Windows\system32\cmd.exe
어떤 디스크의 스케줄링 기법을 선택하시겠습니까?
(1. FDFS기법 2. SSTF기법 3. LOOK기법 4. 선택큐잉 기법 5. 시기의기법):3
LOOK기법을 선택하셨습니다.
디스크의 헤더의 순기번호를 입력하세요.(0부터 199까지):133
seek_time 기준값을 입력해주세요:19
무작위로 트래크의 개수를 정해드리겠습니다(최소 1부터-최대 10까지)트래크의 개수가 9로 정해졌습니다.실린더의 범위를 0-199로 한정했습니다.
정해진 트래크번호의 순서:96 128 110 149 44 43 20 182 106
오름차순으로 재배열된 순서:20 43 44 96 106 110 128 149 182
이는 방향으로 헤드를 시작할지 방향을 정하세요.(1. 올라가는방향 2. 내려가는방향):1
LOOK기법을 선택하고 방향은 올라가는방향이므로 순서는:149 182 128 110 116 96 44 43 20
총 이동거리:211
seek_time이 190이므로 총 탐색시간은 4009입니다.즉 회전지연시간이 없으므로 총 실행시간은 4009입니다.평균 탐색시간은 445.000000입니다.
1번짜리 탐색시간 304에 대한 편차는 -141.000000입니다.이 편차의 제곱은 19881.000000입니다.
2번짜리 탐색시간 627에 대한 편차는 182.000000입니다.이 편차의 제곱은 33124.000000입니다.
3번짜리 탐색시간 1025에 대한 편차는 581.000000입니다.이 편차의 제곱은 337561.000000입니다.
4번짜리 탐색시간 342에 대한 편차는 -103.000000입니다.이 편차의 제곱은 10609.000000입니다.
5번짜리 탐색시간 759에 대한 편차는 -369.000000입니다.이 편차의 제곱은 136161.000000입니다.
6번짜리 탐색시간 190에 대한 편차는 -255.000000입니다.이 편차의 제곱은 65025.000000입니다.
7번짜리 탐색시간 985에 대한 편차는 543.000000입니다.이 편차의 제곱은 294849.000000입니다.
8번짜리 탐색시간 198에 대한 편차는 -426.000000입니다.이 편차의 제곱은 181476.000000입니다.
9번짜리 탐색시간 437에 대한 편차는 -8.000000입니다.이 편차의 제곱은 64.000000입니다.
총 편차의 합이 1078750.000000이므로 분산은 119861.111111입니다.
그러므로 표준편차는 346.206635입니다.계속하려면 아무 키나 누르십시오 . . .

```

(2)내려가는 방향

```
C:\Windows\system32\cmd.exe
아직 디스크 스케줄링 기법을 선택하시겠습니까?
(1.FDFS기법 2.SSTF기법 3.LOOC기법 4.섹터큐잉 기법 5.시기의기법):3
LOOC기법을 선택하셨습니다.
디스크 헤더의 조각위치를 입력하세요:(0부터 199까지):33
seek_time 기준값을 입력해주세요:5
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지)트랙의 개수가 8로 정해졌습니다.실린더의 범위를 0-199로 한
정했습니다.
6개가 랜덤으로정해집니다.
정해진 트랙번호의 순서: 161 126 92 109 69 195 109 143
오름차순으로 재배열된 순서:69 92 109 109 126 143 161 195
이제 방향으로 헤드를 시작할지 방향을 정하세요.(1.올라가는방향 2.내려가는방향):2
LOOC기법을 선택했고 방향은 내려가는방향이므로 순서는:69 92 109 109 126 143 161 195
총 이동거리는: 162
seek_time이 50이므로 총 탐색시간은 810입니다즉 회전지연시간이 없으므로 총 실행시간은 810입니다.평균 탐색시간은 101.000000
입니다.
1번재 탐색시간 180에 대한 편차는 79.000000입니다.이 편차의 제곱은 6241.000000입니다.
2번재 탐색시간 115에 대한 편차는 14.000000입니다.이 편차의 제곱은 196.000000입니다.
3번재 탐색시간 85에 대한 편차는 -16.000000입니다.이 편차의 제곱은 256.000000입니다.
4번재 탐색시간 0에 대한 편차는 -101.000000입니다.이 편차의 제곱은 10201.000000입니다.
5번재 탐색시간 85에 대한 편차는 -16.000000입니다.이 편차의 제곱은 256.000000입니다.
6번재 탐색시간 85에 대한 편차는 -16.000000입니다.이 편차의 제곱은 256.000000입니다.
7번재 탐색시간 90에 대한 편차는 -11.000000입니다.이 편차의 제곱은 121.000000입니다.
8번재 탐색시간 170에 대한 편차는 69.000000입니다.이 편차의 제곱은 4761.000000입니다.
총 편차의 합이 22268.000000이므로 분산은 2786.000000 입니다.
그러므로 표준편차는 52.782573입니다.계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
아직 디스크 스케줄링 기법을 선택하시겠습니까?
(1.FDFS기법 2.SSTF기법 3.LOOC기법 4.섹터큐잉 기법 5.시기의기법):3
LOOC기법을 선택하셨습니다.
디스크 헤더의 조각위치를 입력하세요:(0부터 199까지):99
seek_time 기준값을 입력해주세요:1
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지)트랙의 개수가 6로 정해졌습니다.실린더의 범위를 0-199로 한
정했습니다.
6개가 랜덤으로정해집니다.
정해진 트랙번호의 순서: 122 126 189 92 7 141
오름차순으로 재배열된 순서:7 92 122 126 141 189
이제 방향으로 헤드를 시작할지 방향을 정하세요.(1.올라가는방향 2.내려가는방향):2
LOOC기법을 선택했고 방향은 내려가는방향이므로 순서는:92 7 122 126 141 189
총 이동거리는: 274
seek_time이 1이므로 총 탐색시간은 274입니다즉 회전지연시간이 없으므로 총 실행시간은 274입니다.평균 탐색시간은 45.000000
입니다.
1번재 탐색시간 7에 대한 편차는 -38.000000입니다.이 편차의 제곱은 1444.000000입니다.
2번재 탐색시간 85에 대한 편차는 40.000000입니다.이 편차의 제곱은 1600.000000입니다.
3번재 탐색시간 115에 대한 편차는 70.000000입니다.이 편차의 제곱은 4900.000000입니다.
4번재 탐색시간 4에 대한 편차는 -41.000000입니다.이 편차의 제곱은 1681.000000입니다.
5번재 탐색시간 15에 대한 편차는 -30.000000입니다.이 편차의 제곱은 900.000000입니다.
6번재 탐색시간 48에 대한 편차는 3.000000입니다.이 편차의 제곱은 9.000000입니다.
총 편차의 합이 10834.000000이므로 분산은 1755.666667 입니다.
그러므로 표준편차는 41.900678입니다.계속하려면 아무 키나 누르십시오 . . .
```

4.섹터큐잉 기법

(1) 5400 RPM 섹터 4개

```
C:\Windows\system32\cmd.exe
아직 디스크 스케줄링 기법을 선택하시겠습니까?
(1.FDFS기법 2.SSTF기법 3.LOOC기법 4.섹터큐잉 기법 5.시기의기법):4
섹터큐잉 기법을 선택하셨습니다.
디스크 헤더의 조각위치를 입력하세요:(0부터 199까지):100
디스크의 rpm 기준값을 입력해주세요:(5400 6000 7200 등):5400
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지).트랙의 개수가 5로 정해졌습니다.실린더의 범위를 0-199로
정했습니다.5개가 랜덤으로정해집니다.
정해진 트랙번호의 순서:9 65 104 155 99
디스크가 한바퀴 회전하는데 걸린시간 11.111111입니다.
섹터큐잉을 선택하셨습니다.트랙이 하나이므로 seek_time이 없습니다.섹터의 개수를 입력하세요(4,8,10중 입력):4
올라가는 방향으로 섹터의 개수입니다.섹터의 개수 4를 입력하셨습니다.0-199까지를 분포하겠습니다.
디스크가 정해집니다.조기에 헤더보지션을 100로 선택하셨습니다.하지만 sector의 개수를 4개로 정하셨으니, 현재 헤더의 위치는 섹
터2입니다.
92) 같은 트랙이 없다는것입니다.
9 0 0 0 0 0 0 0 0 0
100에서부터 99까지는 섹터 1 65 98 0 0 0 0 0 0 0
150에서부터 149까지는 섹터 2 104 0 0 0 0 0 0 0 0
100에서부터 199까지는 섹터 3 155 0 0 0 0 0 0 0 0
150에서부터 199까지는 섹터 4 9 65 104 155 99
디스크가 한바퀴 회전하는데 걸린시간 11.111111입니다.
1/4바퀴 도는데 걸리는 시간은 2.777778이고
3/4바퀴 도는데 걸리는 시간은 5.555556이고
1/4바퀴 도는데 걸리는 시간은 9.333333입니다.
그렇다면 회전지연시간은2 제곱 헤더위치가 2번타이므로
8.333333 2.77778 2.77778 2.77778 5.55556
즉 회전지연시간은 22.222222입니다총 회전지연시간은 22.222222입니다.즉 탐색시간이 없으므로 총 실행시간은 22.222222입니다.평
균 회전지연시간은 4.444444입니다.
1번재 회전지연시간218.333333에 대한 편차는 3.888889입니다.이 편차의 제곱은 15.123457입니다.
2번재 회전지연시간2.77778에 대한 편차는 -1.888889입니다.이 편차의 제곱은 3.555556입니다.
3번재 회전지연시간2.77778에 대한 편차는 -1.888889입니다.이 편차의 제곱은 3.555556입니다.
4번재 회전지연시간2.77778에 대한 편차는 -1.888889입니다.이 편차의 제곱은 3.555556입니다.
5번재 회전지연시간5.555556에 대한 편차는 1.111111입니다.이 편차의 제곱은 1.234568입니다.
총 편차의 합이 24.691358이므로 분산은 4.938272 입니다.그러므로 표준편차는 2.222222입니다.
계속하려면 아무 키나 누르십시오 . . .
```







```
C:\Windows\system32\cmd.exe
비록 디스크 스케줄링 기법을 선택하시겠습니까?
(1) FFS기법 (2) SST기법 (3) LOOK기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하십시오.
디스크 헤드의 초기 위치를 입력하십시오: (0부터 199까지):123
디스크의 rpm 기준값을 입력하십시오(5400, 6000, 7200 등등): 5400
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지). 트랙의 개수가 8로 정해졌습니다. 실린더의 범위를 0-199로
확장했습니다. 8개가 랜덤으로정해집니다.
현재의 트랙번호의 순서:157 124 44 123 93 140 175 152
디스크가 한번씩 회전하는데 걸린시간 11.111111입니다.
섹터큐잉을 선택하십시오. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하십시오(4,8,10중 입력):10
올바른 섹터의 개수입니다. 섹터의 개수 10를 입력하셨으므로 0-199까지를 분류하겠습니다.
섹터가 10개입니다. 초기에 헤드포지션을 123로 선택하였습니다. 하지만 sector의 개수를 10개로 정하였으니, 현재 헤드의 위치는 섹터
6입니다.
0인것은 트랙번호가 없다는 것입니다.
00에서부터 199까지는 섹터 0 0 0 0 0 0 0 0 0 0
200에서부터 399까지는 섹터 1 0 0 0 0 0 0 0 0 0
400에서부터 599까지는 섹터 2 44 0 0 0 0 0 0 0 0
600에서부터 799까지는 섹터 3 0 0 0 0 0 0 0 0 0
800에서부터 999까지는 섹터 4 93 0 0 0 0 0 0 0 0
1000에서부터 1199까지는 섹터 5 0 0 0 0 0 0 0 0 0
1200에서부터 1399까지는 섹터 6 124 123 0 0 0 0 0 0 0
1400에서부터 1599까지는 섹터 7 157 140 152 0 0 0 0 0 0 0
1600에서부터 1799까지는 섹터 8 175 0 0 0 0 0 0 0 0
1800에서부터 1999까지는 섹터 9 0 0 0 0 0 0 0 0 0
큐를 섹터에 따라 분류했으나 섹터큐잉에 따른 순서는 44 93 124 157 175 129 140 152
디스크가 한번씩 회전하는데 걸린시간 11.111111입니다.
즉 17/10바퀴 도는데 걸리는 시간은 11.111111이고
2/10바퀴 도는데 걸리는 시간은 2.222222이고
3/10바퀴 도는데 걸리는 시간은 3.333333이고
4/10바퀴 도는데 걸리는 시간은 4.444444이고
5/10바퀴 도는데 걸리는 시간은 5.555556이고
6/10바퀴 도는데 걸리는 시간은 6.666667이고
7/10바퀴 도는데 걸리는 시간은 7.777778이고
8/10바퀴 도는데 걸리는 시간은 8.888889이고
9/10바퀴 도는데 걸리는 시간은 10.000000입니다.
그렇다면 회전지연시간은? 처음 헤드위치가 6섹터이므로
7.77778 2.22222 2.22222 1.11111 1.11111 8.88889 1.11111 0.00000
현재 회전시간은 24.44444입니다.총 회전지연시간은 24.44444입니다. 즉 탐색시간이 없으므로 총 실행시간은 24.44444입니다. 평균
회전지연시간은 3.666667입니다.
1번 회전지연시간7.77778에 대한 편차는 4.722222입니다. 이 편차의 제곱은 22.299389입니다.
2번 회전지연시간2.22222에 대한 편차는 -0.833333입니다. 이 편차의 제곱은 0.694444입니다.
3번 회전지연시간2.22222에 대한 편차는 -0.833333입니다. 이 편차의 제곱은 0.694444입니다.
4번 회전지연시간1.11111에 대한 편차는 -1.844444입니다. 이 편차의 제곱은 3.402593입니다.
5번 회전지연시간1.11111에 대한 편차는 -1.844444입니다. 이 편차의 제곱은 3.402593입니다.
6번 회전지연시간8.88889에 대한 편차는 5.633333입니다. 이 편차의 제곱은 31.727778입니다.
7번 회전지연시간1.11111에 대한 편차는 -1.844444입니다. 이 편차의 제곱은 3.402593입니다.
8번 회전지연시간0.00000에 대한 편차는 -3.055556입니다. 이 편차의 제곱은 9.338421입니다.
9번 회전지연시간0.00000에 대한 편차는 -3.055556입니다. 이 편차의 제곱은 9.338421입니다.
총 편차의 합이 76.395062이므로 분산은 9.799383 입니다. 그러므로 표준편차는 3.13087입니다.
```

(4) 6000 RPM 섹터 4개

```
C:\Windows\system32\cmd.exe
비록 디스크 스케줄링 기법을 선택하시겠습니까?
(1) FFS기법 (2) SST기법 (3) LOOK기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하십시오.
디스크 헤드의 초기 위치를 입력하십시오: (0부터 199까지):126
디스크의 rpm 기준값을 입력하십시오(5400, 6000, 7200 등등): 6000
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지). 트랙의 개수가 2로 정해졌습니다. 실린더의 범위를 0-199로
확장했습니다. 2개가 랜덤으로정해집니다.
현재의 트랙번호의 순서:3 28
디스크가 한번씩 회전하는데 걸린시간 10.000000입니다.
섹터큐잉을 선택하십시오. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하십시오(4,8,10중 입력):4
올바른 섹터의 개수입니다. 섹터의 개수 4를 입력하셨으므로 0-199까지를 분류하겠습니다.
섹터가 4개입니다. 초기에 헤드포지션을 126로 선택하였습니다. 하지만 sector의 개수를 4개로 정하였으니, 현재 헤드의 위치는 섹터
2입니다.
0인것은 트랙이 없다는 것입니다.
00에서부터 499까지는 섹터 0 3 28 0 0 0 0 0 0 0 0
500에서부터 999까지는 섹터 1 0 0 0 0 0 0 0 0 0
1000에서부터 1499까지는 섹터 2 0 0 0 0 0 0 0 0 0
1500에서부터 1999까지는 섹터 3 0 0 0 0 0 0 0 0 0
큐를 섹터에 따라 분류했으나 섹터큐잉에 따른 순서는: 3 28
디스크가 한번씩 회전하는데 걸린시간 10.000000입니다.
즉 17/4바퀴 도는데 걸리는 시간은 5.000000이고
2/4바퀴 도는데 걸리는 시간은 5.000000이고
3/4바퀴 도는데 걸리는 시간은 7.500000입니다.
그렇다면 회전지연시간은? 처음 헤드위치가 2섹터이므로
5.00000 0.00000
현재 회전시간은 7.500000입니다.총 회전지연시간은 7.500000입니다. 즉 탐색시간이 없으므로 총 실행시간은 7.500000입니다. 평균
회전지연시간은 3.750000입니다.
1번 회전지연시간7.50000에 대한 편차는 3.750000입니다. 이 편차의 제곱은 14.062500입니다.
2번 회전지연시간0.00000에 대한 편차는 -3.750000입니다. 이 편차의 제곱은 14.062500입니다.
총 편차의 합이 28.125000이므로 분산은 14.062500 입니다. 그러므로 표준편차는 3.750000입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
비록 디스크 스케줄링 기법을 선택하시겠습니까?
(1) FFS기법 (2) SST기법 (3) LOOK기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하십시오.
디스크 헤드의 초기 위치를 입력하십시오: (0부터 199까지):145
디스크의 rpm 기준값을 입력하십시오(5400, 6000, 7200 등등): 6000
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지). 트랙의 개수가 5로 정해졌습니다. 실린더의 범위를 0-199로
확장했습니다. 5개가 랜덤으로정해집니다.
현재의 트랙번호의 순서:1 18 165 46 141
디스크가 한번씩 회전하는데 걸린시간 10.000000입니다.
섹터큐잉을 선택하십시오. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하십시오(4,8,10중 입력):4
올바른 섹터의 개수입니다. 섹터의 개수 4를 입력하셨으므로 0-199까지를 분류하겠습니다.
섹터가 4개입니다. 초기에 헤드포지션을 145로 선택하였습니다. 하지만 sector의 개수를 4개로 정하였으니, 현재 헤드의 위치는 섹터
2입니다.
0인것은 트랙이 없다는 것입니다.
00에서부터 499까지는 섹터 0 1 18 46 0 0 0 0 0 0
500에서부터 999까지는 섹터 1 0 0 0 0 0 0 0 0 0
1000에서부터 1499까지는 섹터 2 141 0 0 0 0 0 0 0 0
1500에서부터 1999까지는 섹터 3 165 0 0 0 0 0 0 0 0
큐를 섹터에 따라 분류했으나 섹터큐잉에 따른 순서는: 1 141 165 18 46
디스크가 한번씩 회전하는데 걸린시간 10.000000입니다.
즉 17/4바퀴 도는데 걸리는 시간은 5.000000이고
2/4바퀴 도는데 걸리는 시간은 5.000000이고
3/4바퀴 도는데 걸리는 시간은 7.500000입니다.
그렇다면 회전지연시간은? 처음 헤드위치가 2섹터이므로
7.50000 5.00000 2.50000 5.00000 0.00000
현재 회전시간은 17.500000입니다.총 회전지연시간은 17.500000입니다. 즉 탐색시간이 없으므로 총 실행시간은 17.500000입니다. 평균
회전지연시간은 3.500000입니다.
1번 회전지연시간7.50000에 대한 편차는 4.000000입니다. 이 편차의 제곱은 16.000000입니다.
2번 회전지연시간5.00000에 대한 편차는 1.500000입니다. 이 편차의 제곱은 2.250000입니다.
3번 회전지연시간2.50000에 대한 편차는 -1.000000입니다. 이 편차의 제곱은 1.000000입니다.
4번 회전지연시간5.00000에 대한 편차는 -1.000000입니다. 이 편차의 제곱은 1.000000입니다.
5번 회전지연시간0.00000에 대한 편차는 -5.000000입니다. 이 편차의 제곱은 25.000000입니다.
총 편차의 합이 32.500000이므로 분산은 6.500000 입니다. 그러므로 표준편차는 2.549510입니다.
계속하려면 아무 키나 누르십시오 . . .
```

## (5) 6000 RPM 섹터 8개

```
C:\Windows\system32\cmd.exe
어떤 디스크 스케줄링 방법을 선택하시겠습니까?
(1. FFS기법 2. SSTF기법 3. LIFO기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하셨습니다.
디스크 헤드의 초기화치를 입력하세요:(0부터 199까지):58
디스크의 rpm 기준값을 입력해주세요:(5400,6000,7200 등중):6000
최소부터 트래크의 개수를 정해드립니다(최소 1부터-최대 10까지). 트래크의 개수가 3로 정해졌습니다.실린더의 범위를 0-199로
정해드립니다.3번가.변경으로정해집니다.
정확한 트랙번호의 순서:185 194 20
디스크가 한번회전하는데 걸린시간 10.000000입니다.
섹터큐잉을 선택하셨습니다.트랙이 하나이므로 seek_time이 없습니다. 섹터의 개수를 입력하세요(4,8,10중 입력):8
트랙은 섹터의 개수입니다.섹터의 개수 8를 입력하시므로 0-199까지를 분류하였습니다.
섹터가 8개입니다. 초기에 헤드포지션을 58로 선택하셨습니다.하지만 sector의 개수를 8개로 정하였으니,현재 헤드의 위치는 섹터
2입니다.
0은 트랙이 없으므로
0에서부터 240까지는 섹터 0 20 0 0 0 0 0 0 0 0 0
25에서부터 49까지는 섹터 1 0 0 0 0 0 0 0 0 0 0
50에서부터 74까지는 섹터 2 0 0 0 0 0 0 0 0 0 0
75에서부터 99까지는 섹터 3 0 0 0 0 0 0 0 0 0 0
100에서부터 124까지는 섹터 4 0 0 0 0 0 0 0 0 0 0
125에서부터 149까지는 섹터 5 0 0 0 0 0 0 0 0 0 0
150에서부터 174까지는 섹터 6 0 0 0 0 0 0 0 0 0 0
175에서부터 199까지는 섹터 7 185 194 0 0 0 0 0 0 0
트랙 섹터에 따라 분류했으나 섹터큐잉에 따른 순서는 20 185 194
디스크가 한번회전하는데 걸린시간 10.000000입니다.
즉 1/8바퀴 도는데 걸린 시간은 1.2500000이고
2/8바퀴 도는데 걸린 시간은 2.5000000이고
3/8바퀴 도는데 걸린 시간은 3.7500000이고
4/8바퀴 도는데 걸린 시간은 5.0000000이고
5/8바퀴 도는데 걸린 시간은 6.2500000이고
6/8바퀴 도는데 걸린 시간은 7.5000000이고
7/8바퀴 도는데 걸린 시간은 8.7500000입니다.
그렇다면 회전지연시간은? 처음 헤드위치가 2섹터이므로
8.750000 6.750000 0.000000
총 회전지연시간은 17.500000입니다. 즉 탐색시간이 없으므로 총 실행시간은 17.500000입니다.
평균 회전지연시간은 5.933333입니다.
각 1 변회전 회전지연시간 8.750000 에 대한 편차는 2.916667입니다. 이 편차의 제곱은 8.506944입니다.
각 2 변회전 회전지연시간 7.500000 에 대한 편차는 2.916667입니다. 이 편차의 제곱은 8.506944입니다.
각 3 변회전 회전지연시간 6.000000 에 대한 편차는 -5.933333입니다. 이 편차의 제곱은 34.027778입니다.
총 편차의 합이 51.041667이므로 분산은 17.013889 입니다. 그러므로 표준편차는 4.124730입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
어떤 디스크 스케줄링 방법을 선택하시겠습니까?
(1. FFS기법 2. SSTF기법 3. LIFO기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하셨습니다.
디스크 헤드의 초기화치를 입력하세요:(0부터 199까지):76
디스크의 rpm 기준값을 입력해주세요:(5400,6000,7200 등중):6000
최소부터 트래크의 개수를 정해드립니다(최소 1부터-최대 10까지). 트래크의 개수가 8로 정해졌습니다.실린더의 범위를 0-199로
정해드립니다.8번가.변경으로정해집니다.
정확한 트랙번호의 순서:176 167 128 150 114 120 35 12
디스크가 한번회전하는데 걸린시간 10.000000입니다.
섹터큐잉을 선택하셨습니다.트랙이 하나이므로 seek_time이 없습니다. 섹터의 개수를 입력하세요(4,8,10중 입력):8
트랙은 섹터의 개수입니다.섹터의 개수 8를 입력하시므로 0-199까지를 분류하였습니다.
섹터가 8개입니다. 초기에 헤드포지션을 76로 선택하셨습니다.하지만 sector의 개수를 8개로 정하였으니,현재 헤드의 위치는 섹터
2입니다.
0은 트랙이 없으므로
25에서부터 240까지는 섹터 0 12 0 0 0 0 0 0 0 0 0
250에서부터 490까지는 섹터 1 35 0 0 0 0 0 0 0 0 0
500에서부터 740까지는 섹터 2 0 0 0 0 0 0 0 0 0 0
750에서부터 990까지는 섹터 3 0 0 0 0 0 0 0 0 0 0
1000에서부터 1240까지는 섹터 4 114 120 0 0 0 0 0 0 0 0
1250에서부터 1490까지는 섹터 5 128 0 0 0 0 0 0 0 0 0
1500에서부터 1740까지는 섹터 6 167 150 0 0 0 0 0 0 0
1750에서부터 1990까지는 섹터 7 176 0 0 0 0 0 0 0 0
트랙 섹터에 따라 분류했으나 섹터큐잉에 따른 순서는 12 35 114 128 167 176 120 150
디스크가 한번회전하는데 걸린시간 10.000000입니다.
즉 1/8바퀴 도는데 걸린 시간은 1.2500000이고
2/8바퀴 도는데 걸린 시간은 2.5000000이고
3/8바퀴 도는데 걸린 시간은 3.7500000이고
4/8바퀴 도는데 걸린 시간은 5.0000000이고
5/8바퀴 도는데 걸린 시간은 6.2500000이고
6/8바퀴 도는데 걸린 시간은 7.5000000이고
7/8바퀴 도는데 걸린 시간은 8.7500000입니다.
그렇다면 회전지연시간은? 처음 헤드위치가 3섹터이므로
7.500000 1.500000 3.750000 1.250000 1.250000 1.250000 6.250000 2.500000
총 회전지연시간은 25.000000입니다. 즉 탐색시간이 없으므로 총 실행시간은 25.000000입니다.
평균 회전지연시간은 3.125000입니다.
각 1 변회전 회전지연시간 7.500000 에 대한 편차는 4.375000입니다. 이 편차의 제곱은 19.140625입니다.
각 2 변회전 회전지연시간 1.500000 에 대한 편차는 -1.875000입니다. 이 편차의 제곱은 3.515625입니다.
각 3 변회전 회전지연시간 3.750000 에 대한 편차는 0.625000입니다. 이 편차의 제곱은 0.390625입니다.
각 4 변회전 회전지연시간 1.250000 에 대한 편차는 -1.875000입니다. 이 편차의 제곱은 3.515625입니다.
각 5 변회전 회전지연시간 2.500000 에 대한 편차는 -1.875000입니다. 이 편차의 제곱은 3.515625입니다.
각 6 변회전 회전지연시간 6.250000 에 대한 편차는 -1.875000입니다. 이 편차의 제곱은 3.515625입니다.
각 7 변회전 회전지연시간 2.500000 에 대한 편차는 3.125000입니다. 이 편차의 제곱은 9.765625입니다.
총 편차의 합이 43.750000이므로 분산은 5.468750 입니다. 그러므로 표준편차는 2.338536입니다.
계속하려면 아무 키나 누르십시오 . . .
```





## (7) 7200 RPM 섹터 4개

```
C:\Windows\system32\cmd.exe
어떤 디스크 스케줄링 기법을 선택하시겠습니까?
(1. FCFS기법 2. SSTF기법 3. LRU기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하셨습니다.
디스크 헤드의 초기 위치를 입력하세요: (0부터 199까지):28
디스크의 rpm 기준값을 입력해주세요(5400, 6000, 7200 등등):7200
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지). 트랙의 개수가 3로 정해졌습니다. 실린더의 범위를 0-199로
정해드리겠습니다. 3개가 랜덤으로 정해집니다.
정해진 트랙번호의 순서: 21 131 193
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
섹터큐잉을 선택하셨습니다. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하세요(4,8,10중 입력):4
클러스터 섹터의 개수입니다. 섹터의 개수 4를 입력하셨으므로 0-199까지를 분류하겠습니다.
섹터가 4개입니다. 초기에 헤더포지션을 28로 선택하셨습니다. 하지만 sector의 개수를 4개로 정하였으니, 현재 헤더의 위치는 섹
터1입니다.
0번 트랙이 없다는것입니다.
0에서부터 49까지는 섹터 0 21 0 0 0 0 0 0 0 0
50에서부터 99까지는 섹터 1 0 0 0 0 0 0 0 0 0
100에서부터 149까지는 섹터 2 131 0 0 0 0 0 0 0 0
150에서부터 199까지는 섹터 3 193 0 0 0 0 0 0 0 0
큐를 섹터에 따라 분류했으나 섹터큐잉에 따른 순서는: 21 131 193
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
트럭 1/4바퀴 도는데 걸리는 시간은 2.083333이고
2/4바퀴 도는데 걸리는 시간은 4.166667이고
3/4바퀴 도는데 걸리는 시간은 6.250000입니다.
그렇다면 회전지연시간은? 처음 헤더위치가 0섹터이므로
2.083333 4.166667 2.083333
전체 회전시간은 8.333333입니다 총 회전지연시간은 8.333333입니다. 즉 탐색시간이 없으므로 총 실행시간은 8.333333입니다. 평균
회전지연시간은 2.777778입니다.
트럭 1번쪽 회전지연시간2.083333에 대한 편차는 -0.694444입니다. 이 편차의 제곱은 0.482253입니다.
트럭 2번쪽 회전지연시간4.166667에 대한 편차는 1.388889입니다. 이 편차의 제곱은 1.929012입니다.
트럭 3번쪽 회전지연시간2.083333에 대한 편차는 -0.694444입니다. 이 편차의 제곱은 0.482253입니다.
총 편차의 합이 2.893519이므로 분산은 0.964506 입니다. 그러므로 표준편차는 0.982099입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
어떤 디스크 스케줄링 기법을 선택하시겠습니까?
(1. FCFS기법 2. SSTF기법 3. LRU기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하셨습니다.
디스크 헤드의 초기 위치를 입력하세요: (0부터 199까지):67
디스크의 rpm 기준값을 입력해주세요(5400, 6000, 7200 등등):7200
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지). 트랙의 개수가 6로 정해졌습니다. 실린더의 범위를 0-199로
정해드리겠습니다. 6개가 랜덤으로 정해집니다.
정해진 트랙번호의 순서: 90 53 193 153 85 168
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
섹터큐잉을 선택하셨습니다. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하세요(4,8,10중 입력):4
클러스터 섹터의 개수입니다. 섹터의 개수 4를 입력하셨으므로 0-199까지를 분류하겠습니다.
섹터가 4개입니다. 초기에 헤더포지션을 67로 선택하셨습니다. 하지만 sector의 개수를 4개로 정하였으니, 현재 헤더의 위치는 섹
터1입니다.
0번 트랙이 없다는것입니다.
0에서부터 49까지는 섹터 0 23 0 0 0 0 0 0 0 0
50에서부터 99까지는 섹터 1 90 85 0 0 0 0 0 0 0
100에서부터 149까지는 섹터 2 0 0 0 0 0 0 0 0 0
150에서부터 199까지는 섹터 3 193 153 168 0 0 0 0 0 0
큐를 섹터에 따라 분류했으나 섹터큐잉에 따른 순서는: 23 90 193 85 153 168
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
트럭 1/4바퀴 도는데 걸리는 시간은 2.083333이고
2/4바퀴 도는데 걸리는 시간은 4.166667이고
3/4바퀴 도는데 걸리는 시간은 6.250000입니다.
그렇다면 회전지연시간은? 처음 헤더위치가 1섹터이므로
0.000000 2.083333 4.166667 4.166667 4.166667 0.000000
전체 회전시간은 14.583333입니다 총 회전지연시간은 14.583333입니다. 즉 탐색시간이 없으므로 총 실행시간은 14.583333입니다. 평균
회전지연시간은 2.430556입니다.
트럭 1번쪽 회전지연시간0.000000에 대한 편차는 -2.430556입니다. 이 편차의 제곱은 5.907600입니다.
트럭 2번쪽 회전지연시간2.083333에 대한 편차는 -0.347222입니다. 이 편차의 제곱은 0.120563입니다.
트럭 3번쪽 회전지연시간4.166667에 대한 편차는 1.736111입니다. 이 편차의 제곱은 3.014082입니다.
트럭 4번쪽 회전지연시간4.166667에 대한 편차는 1.736111입니다. 이 편차의 제곱은 3.014082입니다.
트럭 5번쪽 회전지연시간4.166667에 대한 편차는 1.736111입니다. 이 편차의 제곱은 3.014082입니다.
트럭 6번쪽 회전지연시간0.000000에 대한 편차는 -2.430556입니다. 이 편차의 제곱은 5.907600입니다.
총 편차의 합이 20.378008이므로 분산은 3.496335 입니다. 그러므로 표준편차는 1.869649입니다.
계속하려면 아무 키나 누르십시오 . . .
```

## (8) 7200 RPM 섹터 8개

```
C:\Windows\system32\cmd.exe
어떤 디스크 스케줄링 기법을 선택하시겠습니까?
(1. FCFS기법 2. SSTF기법 3. LRU기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하셨습니다.
디스크 헤드의 초기 위치를 입력하세요: (0부터 199까지):97
디스크의 rpm 기준값을 입력해주세요(5400, 6000, 7200 등등):7200
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지). 트랙의 개수가 3로 정해졌습니다. 실린더의 범위를 0-199로
정해드리겠습니다. 3개가 랜덤으로 정해집니다.
정해진 트랙번호의 순서: 120 118 140
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
섹터큐잉을 선택하셨습니다. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하세요(4,8,10중 입력):8
클러스터 섹터의 개수입니다. 섹터의 개수 8를 입력하셨으므로 0-199까지를 분류하겠습니다.
섹터가 8개입니다. 초기에 헤더포지션을 97로 선택하셨습니다. 하지만 sector의 개수를 8개로 정하였으니, 현재 헤더의 위치는 섹터
3입니다.
0번 트랙이 없다는것입니다.
0에서부터 24까지는 섹터 0 0 0 0 0 0 0 0 0
25에서부터 49까지는 섹터 1 0 0 0 0 0 0 0 0
50에서부터 74까지는 섹터 2 0 0 0 0 0 0 0 0
75에서부터 99까지는 섹터 3 0 0 0 0 0 0 0 0
100에서부터 124까지는 섹터 4 120 118 0 0 0 0 0 0
125에서부터 149까지는 섹터 5 140 0 0 0 0 0 0 0
150에서부터 174까지는 섹터 6 0 0 0 0 0 0 0 0
175에서부터 199까지는 섹터 7 0 0 0 0 0 0 0 0
큐를 섹터에 따라 분류했으나 섹터큐잉에 따른 순서는: 120 140 118
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
트럭 1/8바퀴 도는데 걸리는 시간은 1.041667이고
2/8바퀴 도는데 걸리는 시간은 2.083333이고
3/8바퀴 도는데 걸리는 시간은 3.125000이고
4/8바퀴 도는데 걸리는 시간은 4.166667이고
5/8바퀴 도는데 걸리는 시간은 5.208333이고
6/8바퀴 도는데 걸리는 시간은 6.250000이고
7/8바퀴 도는데 걸리는 시간은 7.291667입니다.
그렇다면 회전지연시간은? 처음 헤더위치가 3섹터이므로
2.083333 1.041667 1.291667
전체 회전시간은 10.416667입니다 총 회전지연시간은 10.416667입니다. 즉 탐색시간이 없으므로 총 실행시간은 10.416667입니다. 평균
회전지연시간은 3.472222입니다.
트럭 1번쪽 회전지연시간1.041667에 대한 편차는 -1.388889입니다. 이 편차의 제곱은 1.929012입니다.
트럭 2번쪽 회전지연시간1.291667에 대한 편차는 -2.430556입니다. 이 편차의 제곱은 5.907600입니다.
트럭 3번쪽 회전지연시간7.291667에 대한 편차는 3.819444입니다. 이 편차의 제곱은 14.588156입니다.
총 편차의 합이 22.424769이므로 분산은 7.474923 입니다. 그러므로 표준편차는 2.734031입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
디스크의 스캐쥬링 방법을 선택하시겠습니까?
1. FDFS기법 2. SSIF기법 3. LOOK기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하셨습니다.
디스크의 헤더의 추가값을 입력하세요 (0부터 199까지):77
디스크의 rom 기준값을 입력해주세요(5400,6000,7200 등등):7200
무지극으로 트랙의 개수를 정해드립니다(최소 1부터-최대 10까지). 트랙의 개수가 9로 정해졌습니다. 실린더의 범위를 0-199로
정해드립니다. 9개가 범위에서 정해집니다.
현재 트랙번호의 순서: 187 0 93 95 142 134 147 60 164
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
섹터큐잉을 선택하셨습니다. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하세요(4,8,10중 입력):8
헤더의 개수입니다. 섹터의 개수 8을 입력하셨습니다. 0-199까지를 분류하겠습니다.
디스크가 8개입니다. 초기에 헤더포지션을 77로 선택하셨습니다. 하지만 sector의 개수를 8개로 정하였으니, 현재 헤더의 위치는 섹터
8입니다.
0은 트랙이 없습니다.
0에서부터 240까지는 섹터 0 0 0 0 0 0 0 0 0 0
10에서부터 480까지는 섹터 1 93 0 0 0 0 0 0 0 0
20에서부터 720까지는 섹터 2 60 0 0 0 0 0 0 0 0
30에서부터 960까지는 섹터 3 93 0 0 0 0 0 0 0 0
40에서부터 1200까지는 섹터 4 0 0 0 0 0 0 0 0 0
50에서부터 1440까지는 섹터 5 142 134 147 0 0 0 0 0 0
60에서부터 1680까지는 섹터 6 164 0 0 0 0 0 0 0 0
70에서부터 1920까지는 섹터 7 187 0 0 0 0 0 0 0 0
디스크 섹터에 따라 분류했습니다. 섹터큐잉에 따른 순서는:35 60 93 142 164 187 134 147 0
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
즉 1/8바퀴 도는데 걸린 시간은 1.041667)고
2/8바퀴 도는데 걸린 시간은 2.083333)고
3/8바퀴 도는데 걸린 시간은 3.125000)고
4/8바퀴 도는데 걸린 시간은 4.166667)고
5/8바퀴 도는데 걸린 시간은 5.208333)고
6/8바퀴 도는데 걸린 시간은 6.250000)고
7/8바퀴 도는데 걸린 시간은 7.291667)고
그렇다면 회전지연시간은? 처음 헤더위치가 3섹터이므로
7.291667 1.041667 1.041667 2.083333 1.041667 1.041667 6.250000 0.000000 2.083333
회전지연시간은 21.875000입니다. 즉 회전지연시간은 21.875000입니다. 즉 탐색시간이 없으므로 총 실행시간은 21.875000입니다. 평균
회전지연시간은 2.437500입니다.
1.1바퀴 회전지연시간 2.437500에 대한 편차는 4.861111입니다. 이 편차의 제곱은 23.630401입니다.
2.2바퀴 회전지연시간 4.166667에 대한 편차는 -1.388889입니다. 이 편차의 제곱은 1.929012입니다.
3.3바퀴 회전지연시간 4.166667에 대한 편차는 -1.388889입니다. 이 편차의 제곱은 1.929012입니다.
4.4바퀴 회전지연시간 2.083333에 대한 편차는 0.347222입니다. 이 편차의 제곱은 0.120563입니다.
5.5바퀴 회전지연시간 0.000000에 대한 편차는 -2.437500입니다. 이 편차의 제곱은 5.937500입니다.
6.6바퀴 회전지연시간 0.000000에 대한 편차는 -1.388889입니다. 이 편차의 제곱은 1.929012입니다.
7.7바퀴 회전지연시간 6.250000에 대한 편차는 3.814444입니다. 이 편차의 제곱은 14.58156입니다.
8.8바퀴 회전지연시간 0.000000에 대한 편차는 -2.437500입니다. 이 편차의 제곱은 5.937500입니다.
9.9바퀴 회전지연시간 2.083333에 대한 편차는 -0.347222입니다. 이 편차의 제곱은 0.120563입니다.
총 편차의 합이 52.083333)으로 분산은 5.767037 입니다. 그러므로 표준편차는 2.402626입니다.
```

(9) 7200 RPM 섹터 10개

```
C:\Windows\system32\cmd.exe
디스크의 스캐쥬링 방법을 선택하시겠습니까?
1. FDFS기법 2. SSIF기법 3. LOOK기법 4. 섹터큐잉 기법 5. 시기의기법):4
섹터큐잉 기법을 선택하셨습니다.
디스크의 헤더의 추가값을 입력하세요 (0부터 199까지):32
디스크의 rom 기준값을 입력해주세요(5400,6000,7200 등등):7200
무지극으로 트랙의 개수를 정해드립니다(최소 1부터-최대 10까지). 트랙의 개수가 4로 정해졌습니다. 실린더의 범위를 0-199로
정해드립니다. 4개가 범위에서 정해집니다.
현재 트랙번호의 순서: 139 39 153 2
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
섹터큐잉을 선택하셨습니다. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하세요(4,8,10중 입력):10
헤더의 개수입니다. 섹터의 개수 10을 입력하셨습니다. 0-199까지를 분류하겠습니다.
디스크가 10개입니다. 초기에 헤더포지션을 32로 선택하셨습니다. 하지만 sector의 개수를 10개로 정하였으니, 현재 헤더의 위치는
섹터 10입니다.
0은 트랙이 없습니다.
0에서부터 199까지는 섹터 0 2 0 0 0 0 0 0 0 0
10에서부터 399까지는 섹터 1 39 0 0 0 0 0 0 0 0
200에서부터 599까지는 섹터 2 0 0 0 0 0 0 0 0 0
300에서부터 799까지는 섹터 3 0 0 0 0 0 0 0 0 0
400에서부터 999까지는 섹터 4 0 0 0 0 0 0 0 0 0
500에서부터 1199까지는 섹터 5 0 0 0 0 0 0 0 0 0
600에서부터 1399까지는 섹터 6 139 0 0 0 0 0 0 0 0
700에서부터 1599까지는 섹터 7 153 0 0 0 0 0 0 0 0
800에서부터 1799까지는 섹터 8 0 0 0 0 0 0 0 0 0
900에서부터 1999까지는 섹터 9 0 0 0 0 0 0 0 0 0
디스크 섹터에 따라 분류했습니다. 섹터큐잉에 따른 순서는:2 39 139 153
디스크가 한바퀴 회전하는데 걸린시간 8.333333입니다.
즉 1/10바퀴 도는데 걸린 시간은 0.833333)고
2/10바퀴 도는데 걸린 시간은 1.666667)고
3/10바퀴 도는데 걸린 시간은 2.500000)고
4/10바퀴 도는데 걸린 시간은 3.333333)고
5/10바퀴 도는데 걸린 시간은 4.166667)고
6/10바퀴 도는데 걸린 시간은 5.000000)고
7/10바퀴 도는데 걸린 시간은 5.833333)고
8/10바퀴 도는데 걸린 시간은 6.666667)고
9/10바퀴 도는데 걸린 시간은 7.500000)입니다.
그렇다면 회전지연시간은? 처음 헤더위치가 1섹터이므로
0.833333 4.166667 0.833333
회전지연시간은 5.833333입니다. 즉 회전지연시간은 5.833333입니다. 즉 탐색시간이 없으므로 총 실행시간은 5.833333입니다. 평균
회전지연시간은 1.458333입니다.
1.1바퀴 회전지연시간 0.000000에 대한 편차는 -1.458333입니다. 이 편차의 제곱은 2.126736입니다.
2.2바퀴 회전지연시간 2.083333에 대한 편차는 0.625000입니다. 이 편차의 제곱은 0.390625입니다.
3.3바퀴 회전지연시간 4.166667에 대한 편차는 2.708333입니다. 이 편차의 제곱은 7.350694입니다.
4.4바퀴 회전지연시간 8.333333에 대한 편차는 -0.625000입니다. 이 편차의 제곱은 0.390625입니다.
총 편차의 합이 10.243750)으로 분산은 2.560764 입니다. 그러므로 표준편차는 1.600239입니다.
```

```
C:\Windows\system32\cmd.exe
여러 디스크 스캐쥬링 기법을 선택하시겠습니까?
1. FRS기법 2. SAT기법 3. L00기법 4. 섹터큐잉 기법 5. 시기의 기법) :4
데이터의 기법을 선택하십시오.
디스크 헤더의 초기 위치를 입력하십시오. (0부터 199까지) :10
디스크의 rpm 기본값을 입력하십시오 (5400, 6000, 7200 등등) :7200
무작위로 트랙의 개수를 정하십시오 (최소 1부터 최대 10까지) 트랙의 개수가 6로 정해졌습니다. 실린더의 범위를 0-199로
정확화합니다.
정해진 트랙번호의 순서: 71 144 103 42 99 142
디스크가 한바퀴 회전하는데 걸린 시간: 8.333333입니다.
섹터 큐잉을 선택하십시오. 트랙이 하나이므로 seek time이 없습니다. 섹터의 개수를 입력하십시오 (4, 8, 10중 입력) :10
올바른 섹터의 개수입니다. 섹터의 개수 10을 입력하셨으므로 0-199까지를 분류하겠습니다.
섹터가 10개입니다. 초기에 헤더 포지션을 10로 선택하였습니다. 하지만 sector의 개수를 10개로 정하였으니, 현재 헤더의 위치는
섹터 0입니다.
이것은 트랙번호가 없다 것입니다.
0에서부터 199까지는 섹터 0 0 0 0 0 0 0 0 0 0
200에서부터 399까지는 섹터 1 0 0 0 0 0 0 0 0 0
400에서부터 599까지는 섹터 2 42 0 0 0 0 0 0 0 0
600에서부터 799까지는 섹터 3 71 0 0 0 0 0 0 0 0
800에서부터 999까지는 섹터 4 99 0 0 0 0 0 0 0 0
1000에서부터 1199까지는 섹터 5 103 0 0 0 0 0 0 0 0
1200에서부터 1399까지는 섹터 6 0 0 0 0 0 0 0 0 0
1400에서부터 1599까지는 섹터 7 144 142 0 0 0 0 0 0 0
1600에서부터 1799까지는 섹터 8 0 0 0 0 0 0 0 0 0
1800에서부터 1999까지는 섹터 9 0 0 0 0 0 0 0 0 0
각 섹터에 따라 분류하니 섹터 큐잉에 따른 순서는: 42 71 99 103 144 142
디스크가 한바퀴 회전하는데 걸린 시간: 8.333333입니다.
즉 1/10바퀴 반는데 걸린 시간은 0.833333이고
2/10바퀴 반는데 걸린 시간은 1.666667이고
3/10바퀴 반는데 걸린 시간은 2.500000이고
4/10바퀴 반는데 걸린 시간은 3.333333이고
5/10바퀴 반는데 걸린 시간은 4.166667이고
6/10바퀴 반는데 걸린 시간은 5.000000이고
7/10바퀴 반는데 걸린 시간은 5.833333이고
8/10바퀴 반는데 걸린 시간은 6.666667이고
9/10바퀴 반는데 걸린 시간은 7.500000입니다.
그동안 회전지연시간은? 처음 헤더 위치가 0번이므로
2.500000 0.833333 0.833333 0.833333 1.666667 0.000000
현재 회전시간은 6.66667입니다. 총 회전지연시간은 6.66667입니다. 즉 탐색시간이 없으므로 총 실행시간은 6.66667입니다. 평균
회전지연시간은 1.111111입니다.
각 1바퀴를 회전지연시간: 2.500000에 대한 편차는 1.368889입니다. 이 편차의 제곱은 1.929012입니다.
각 2바퀴를 회전지연시간: 3.333333에 대한 편차는 -0.277778입니다. 이 편차의 제곱은 0.077160입니다.
각 3바퀴를 회전지연시간: 4.166667에 대한 편차는 -0.277778입니다. 이 편차의 제곱은 0.077160입니다.
각 4바퀴를 회전지연시간: 5.000000에 대한 편차는 -0.277778입니다. 이 편차의 제곱은 0.077160입니다.
각 5바퀴를 회전지연시간: 5.833333에 대한 편차는 0.555556입니다. 이 편차의 제곱은 0.308642입니다.
각 6바퀴를 회전지연시간: 6.666667에 대한 편차는 1.111111입니다. 이 편차의 제곱은 1.234568입니다.
총 편차의 합이 3.703704이므로 분산은 0.617284입니다. 그러므로 표준편차는 0.785674입니다.
계속하려면 아무 키나 누르십시오 . . .
```

## 5. 시기의 기법

### (1)3600RPM

```
C:\Windows\system32\cmd.exe
여러 디스크 스캐쥬링 기법을 선택하시겠습니까?
1. FRS기법 2. SAT기법 3. L00기법 4. 섹터큐잉 기법 5. 시기의 기법) :5
시기의 기법을 선택하십시오.
시기의 기법은 아예 없는 것입니다.
시기의 기법을 아무런 우선순위가 높아도 데이터 전송 시간이 느려지면 뒤에 있는 것들이 늦게 걸리므로 우선순위를 데이터 크기를 조
로 정해서 정렬하십시오.
시기의 기법을 선택하십시오.
디스크 헤더의 초기 위치를 입력하십시오. (0부터 199까지) :12
seek time 기본값을 입력하십시오 :5
디스크의 rpm 기본값 (디스크의 회전속도)을 입력하십시오 (요새 하드디스크는 3600rpm부터 5400rpm 7200rpm) :3600
무작위로 트랙의 개수를 정하십시오 (최소 1부터 최대 10까지) 트랙의 개수가 9로 정해졌습니다. 실린더의 범위를 0-199로
정확화합니다.
정해진 트랙번호의 순서: 67 154 14 178 0 137 167 119 6
디스크가 1초당 회전하는 횟수는 60.000000바퀴입니다. 디스크가 1ms동안 회전하는 횟수는 0.060000바퀴입니다. 디스크가 1바퀴를
도는 데 걸린 시간은 16.666667 ms입니다.
트랙의 개수를 정하십시오 (10 이상 10 이하) :1000
하루 동안의 회전수는 1000 000000 byte입니다.
디스크가 0.016667초당 전송하는 양은 1000.000000 byte입니다.
디스크가 1초당 전송하는 양은 60000.000000 byte입니다.
디스크가 1초당 전송하는 양은 0.060000 Mbyte입니다.
이제 각 트랙의 우선순위를 데이터 크기를 입력하십시오.
트랙번호 우선순위 데이터 크기 (단위 Mbyte) (입력하고 tab을 눌러주세요)
57의 우선순위 입력: 데이터 크기 입력: 1 200
154의 우선순위 입력: 데이터 크기 입력: 2 5
14의 우선순위 입력: 데이터 크기 입력: 3 1
178의 우선순위 입력: 데이터 크기 입력: 4 2
137의 우선순위 입력: 데이터 크기 입력: 5 3
167의 우선순위 입력: 데이터 크기 입력: 6 5
119의 우선순위 입력: 데이터 크기 입력: 7 5
8의 우선순위 입력: 데이터 크기 입력: 8 1
6의 우선순위 입력: 데이터 크기 입력: 9 2
-----
57의 우선순위 1 데이터 크기 200 Mbyte 결과 200
154의 우선순위 2 데이터 크기 5 Mbyte 결과 10
14의 우선순위 3 데이터 크기 1 Mbyte 결과 3
178의 우선순위 4 데이터 크기 2 Mbyte 결과 8
137의 우선순위 5 데이터 크기 3 Mbyte 결과 15
167의 우선순위 6 데이터 크기 5 Mbyte 결과 30
119의 우선순위 7 데이터 크기 5 Mbyte 결과 35
8의 우선순위 8 데이터 크기 1 Mbyte 결과 8
6의 우선순위 9 데이터 크기 2 Mbyte 결과 18
-----
우선순위와 데이터의 양에 따른 순서를 정하면:
1번 데이터가 32와 14가 시작된다.
2번 데이터가 88와 178가 시작된다.
3번 데이터가 88와 119가 시작된다.
4번 데이터가 10와 154가 시작된다.
```





```
C:\Windows\system32\cmd.exe
18번이 4가 시작된다.
102번이 12가 시작된다.
125번이 21가 시작된다.
150번이 102가 시작된다.
350번이 32가 시작된다.
200번이 53가 시작된다.
-----
6에서 102번이 도착한다. 즉 이동거리는 76이다
8에서 74번이 도착한다. 즉 이동거리는 28이다
4에서 125번이 도착한다. 즉 이동거리는 101이다
25에서 21번이 도착한다. 즉 이동거리는 1040이다
1에서 102번이 도착한다. 즉 이동거리는 81이다
8에서 32번이 도착한다. 즉 이동거리는 700이다
2에서 53번이 도착한다. 즉 이동거리는 210이다
-----
현재 이동거리가 431입니다.
1번의 데이터 전송시간은 0.0083333이다.
2 Mbyte 데이터 전송시간은 0.016667이다.
5 Mbyte 데이터 전송시간은 0.041667이다.
2 Mbyte 데이터 전송시간은 0.016667이다.
5 Mbyte 데이터 전송시간은 0.0250000이다.
200 Mbyte 데이터 전송시간은 1.66667이다.
-----
102번 중의 응답시간 380.0083333이다.
74번 중의 응답시간 520.0250000이다.
125번 중의 응답시간 775.066667이다.
21번 중의 응답시간 1235.083333이다.
102번 중의 응답시간 1700.1083333이다.
53번 중의 응답시간 2050.1500000이다.
-----
현재의 총 응답시간은 2156.816667입니다. 즉 회전 지연 시간이 없으므로 총 응답시간은 2156.816667입니다.
-----
1번의 데이터 전송시간은 308.116667입니다.
2 Mbyte 데이터 전송시간은 380.0083333에 대한 편차는 71.891667입니다. 이 편차의 제곱은 5168.411735입니다.
5 Mbyte 데이터 전송시간은 520.0250000에 대한 편차는 211.908333입니다. 이 편차의 제곱은 44905.141735입니다.
2 Mbyte 데이터 전송시간은 775.066667에 대한 편차는 466.950000입니다. 이 편차의 제곱은 218042.302500입니다.
5 Mbyte 데이터 전송시간은 1235.083333에 대한 편차는 986.966667입니다. 이 편차의 제곱은 974103.201111입니다.
2 Mbyte 데이터 전송시간은 1700.108333에 대한 편차는 1391.891667입니다. 이 편차의 제곱은 1937640.800089입니다.
5 Mbyte 데이터 전송시간은 2050.150000에 대한 편차는 1742.033333입니다. 이 편차의 제곱은 3034680.134444입니다.
-----
현재의 편차 제곱의 합이 9632231.68158701으로 분산은 1376033.097371입니다. 그러므로 표준 편차는 1173.044371입니다. 계속하려면 아무
키를 눌러주세요
```