보고서

학과 : 컴퓨터공학과
학번: 21720903
이름 : 조태식

목차

1.개요

수업시간에 디스크 스케쥴러에 대해서 배워보았다. FCFS부터 SSTF그리고 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 스케쥴러 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
에센바흐 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 트렉의 위치
int count = 0;  //무작의 트렉의 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 스케쥴링 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("\n디시크 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 선택햇으므로 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 표쥰편차는 %lf입니다.", standard_deviation);
        }
        else if (which == 2) {//SSTF
                printf("SSTF기법을 선택하셨습니다.");
                printf("\n디시크 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 선택햇으므로 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 햇던 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 표쥰편차는 %lf입니다.", standard_deviation);
        }
        else if (which == 3) {//LOOK
                printf("LOOK기법을 선택하셨습니다.");
                printf("\n디시크 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 표쥰편차는 %lf입니다.", standard_deviation);
        }
        else if (which == 4) {//섹터큐잉
                printf("섹터큐잉 기법을 선택하셨습니다.");
                printf("\n디시크 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 선택하셧습니다.하지만 sector의
개수를 4개로 정하셧으니, 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 분류햇으니 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 선택하셧습니다.하지만 sector의
개수를 8개로 정하셧으니,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 분류햇으니 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 선택하셧습니다.하지만 sector의
개수를 10개로 정하셧으니, 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 분류햇으니 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 표쥰편차는 %lf입니다.", standard_deviation);
                printf("\n\n");
        }
        else if (which == 5) {//시기의기법
                printf("시기의기법을 선택하셨습니다.교수님의 수업을 듣던중 P-SCAN과
SATF기법은 있지만 우선순위와 데이터 전송시간을 동시에 고려한 기법은 아직
없는것같앗다.\n시기의기법: 아무리 우선순위가 높아도 데이터전송시간이 ");
                printf("느려지면 뒤에 있는 것들이 늦게걸리므로 우선순위와 데이터크기를
```

```
조화롭게 처리할수잇는 방법을 선택해보았다. ");
                printf("\n시기의기법을 선택하셨습니다.");
                printf("\n디시크 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;
				}
			}
		}
```

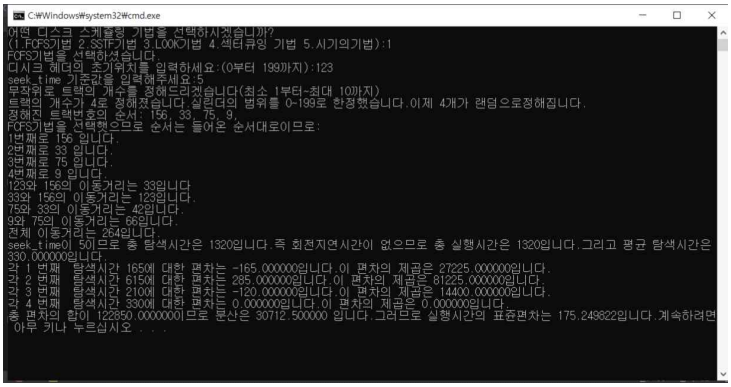
```
printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n우선순위와
데이터의 양에 따른 순서를 정하면:\n");
		for (int i = 0; i < count; i++) {
			printf("%d번째로는 결론이 %d인 %d가 시작된다.\n",i+1, a[i][3],
a[i][0]);
		}
		for (int i = 0; i < count; i++) {
			if (i == 0) {
				seek_distance[i] = abs(headerposition - 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");
```

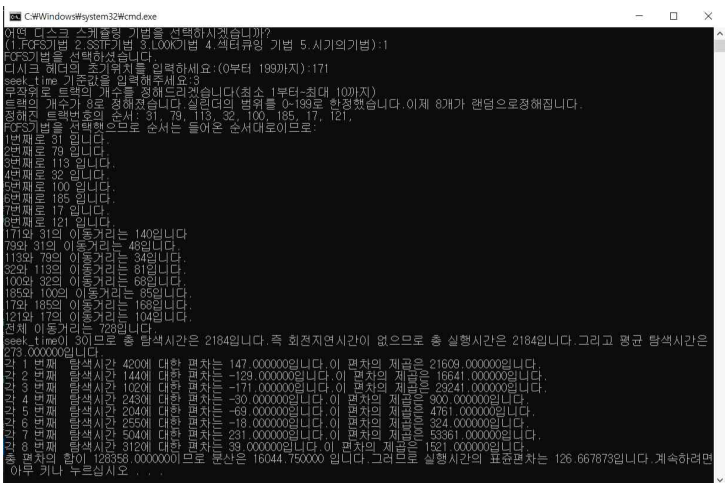
```
                for (int i = 0; i < count; i++) {
                        if (i == 0) {
                                response_time[i] = taesik_seek_time[i] +
transport_time[i];
                        }
                        else {
                                response_time[i] = taesik_seek_time[i] +
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 표쥰편차는 %lf입니다.", standard_deviation);
        }
        else {
                printf("당신이 선택한 기법은 없는 디스크 스케쥴링 기법입니다.");
        }
        return 0;
}
```

3.실험결과 분석

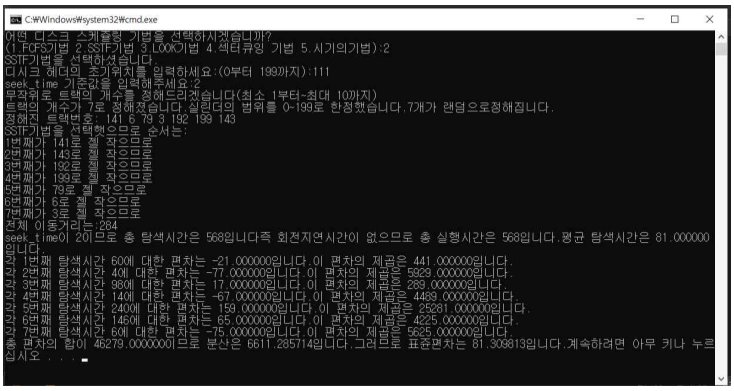
아직은 코드를 짜는것에 대해 미흡하고 솔직히 이 디스크 스케쥴링 기법에 있어서 정확히 이해를 하고있는 것이 아닌것같다. 그래도 열심히 해서 코드를 작성하고 돌려보았고 곧 다가올 여름방학때 좀더 공부를 할생각이다.

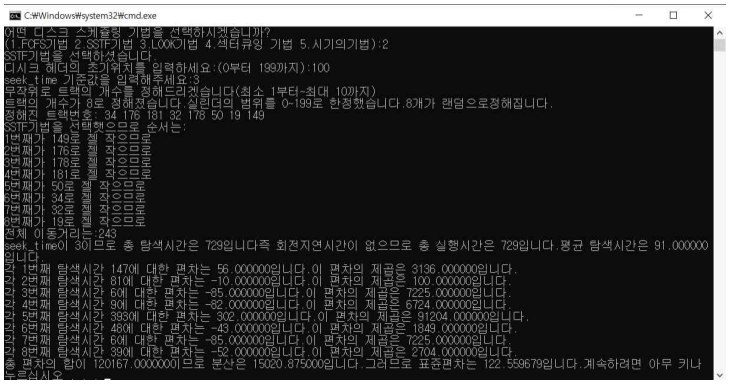
1. FCFS기법





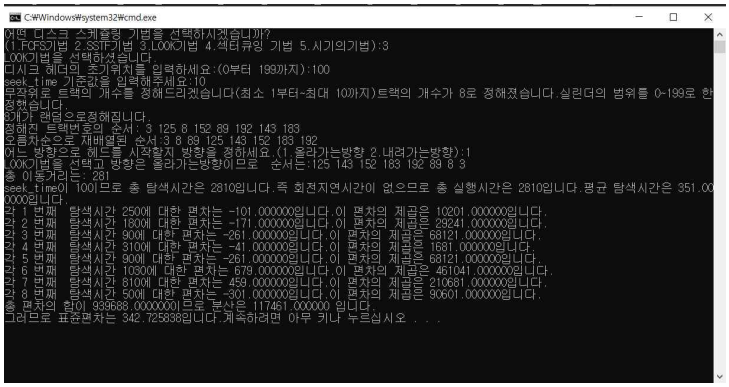
2. SSTF기법

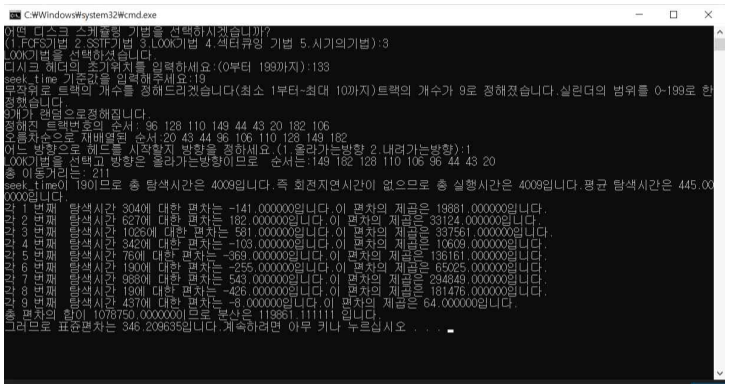




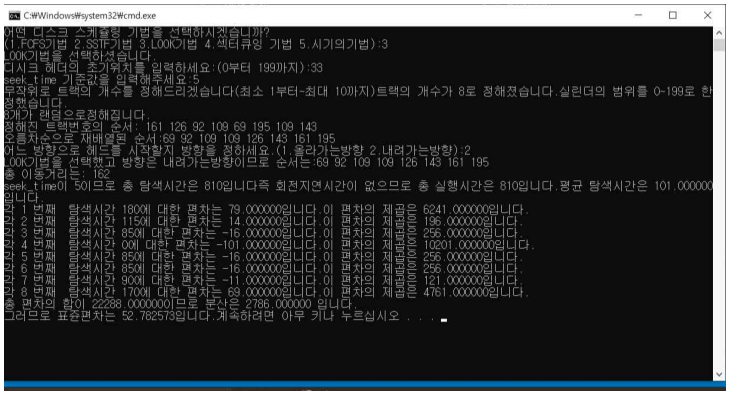
3. LOOK기법

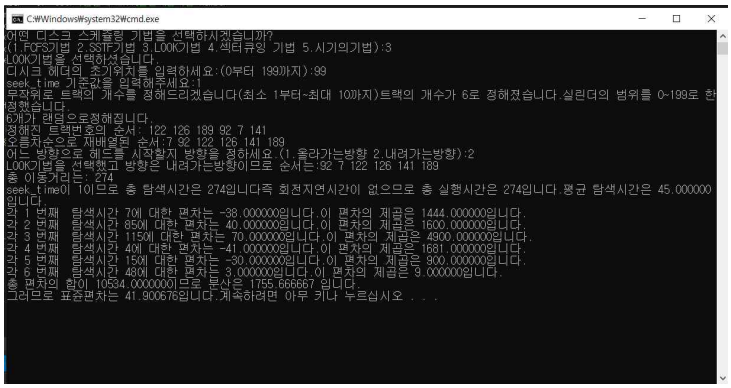
(1)올라가는 방향





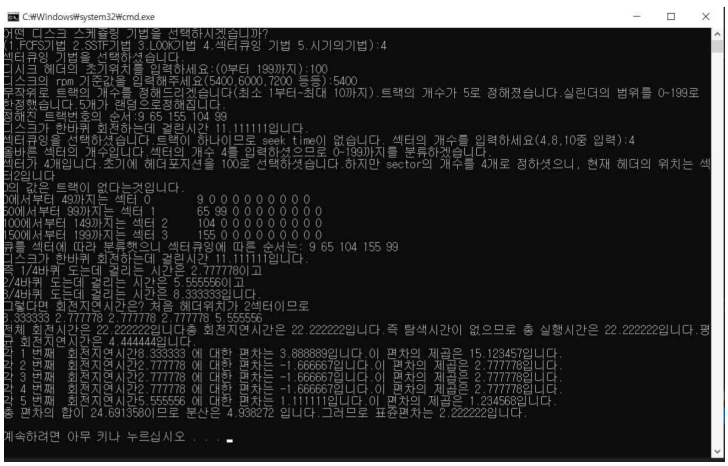
(2)내려가는 방향

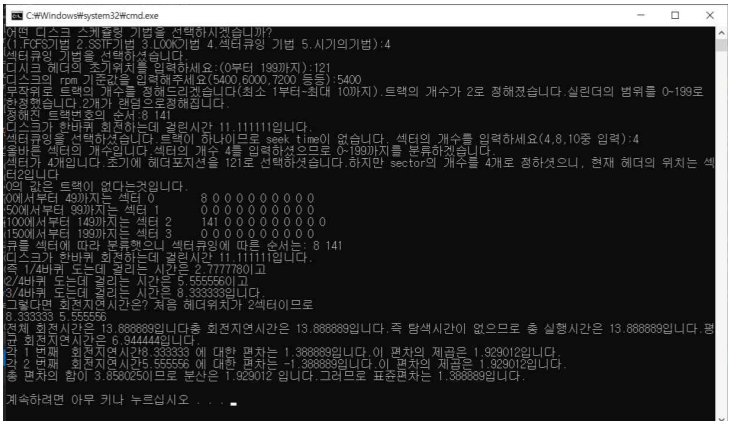




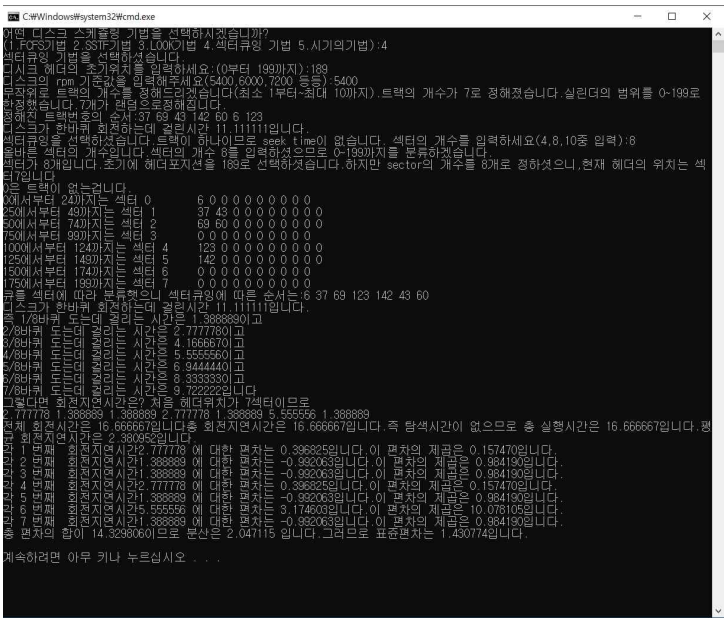
4.섹터큐잉 기법

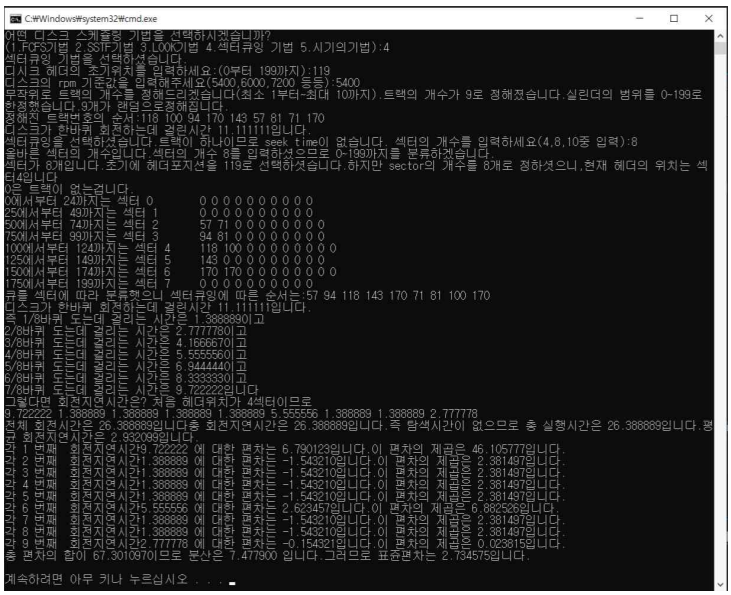
(1) 5400 RPM 섹터 4개



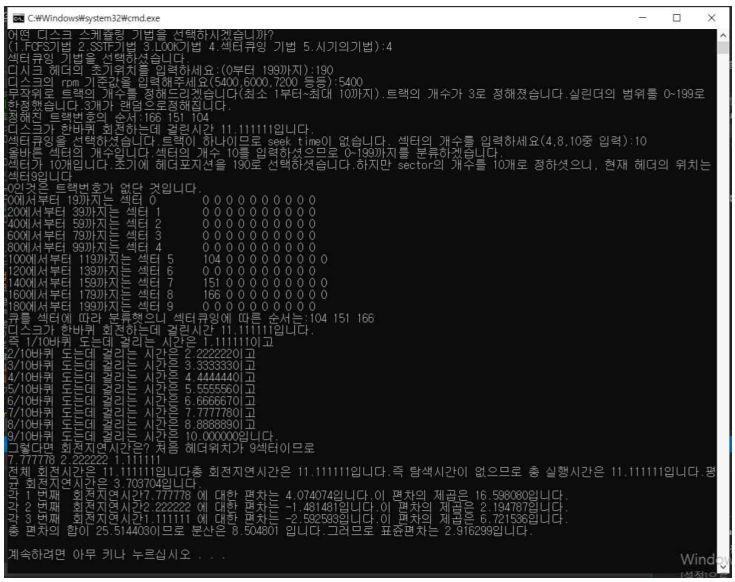


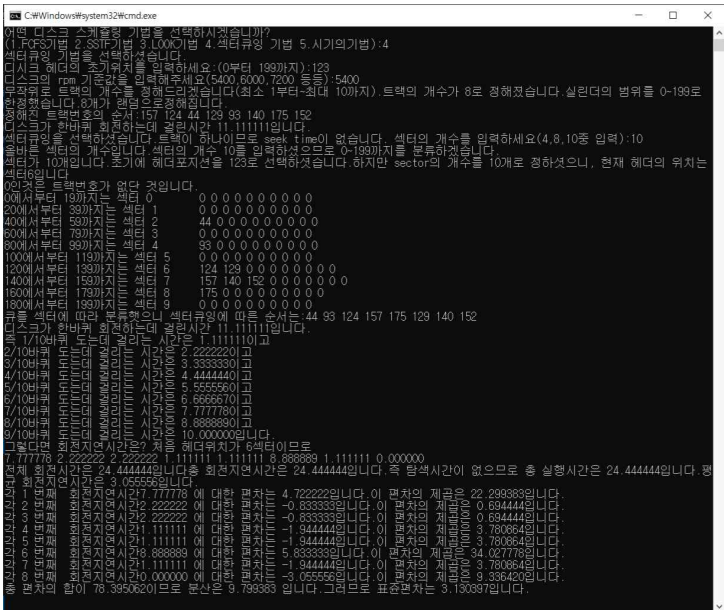
(2) 5400 RPM 섹터 8개



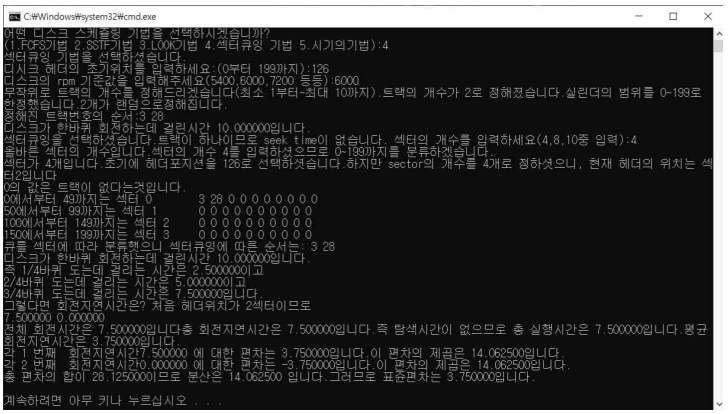


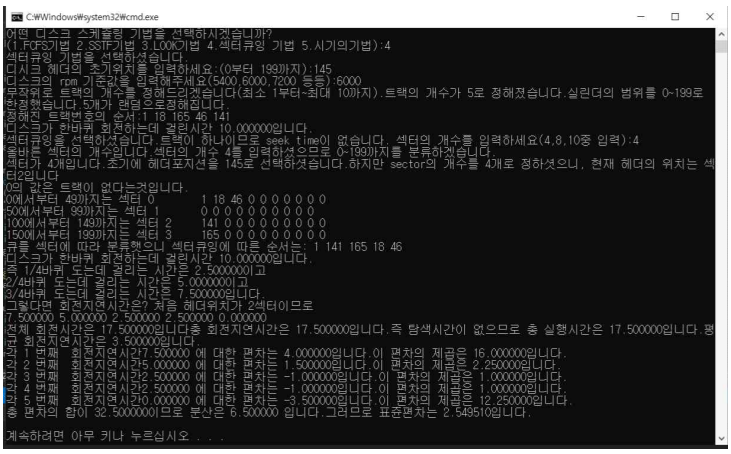
(3) 5400 RPM 섹터 10개



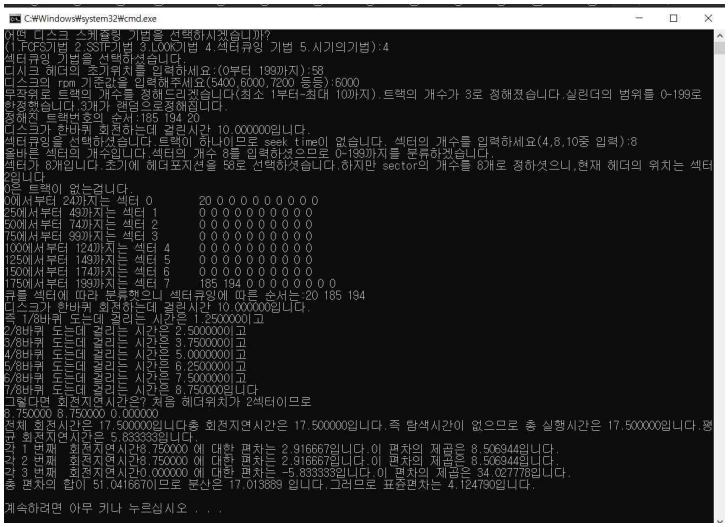


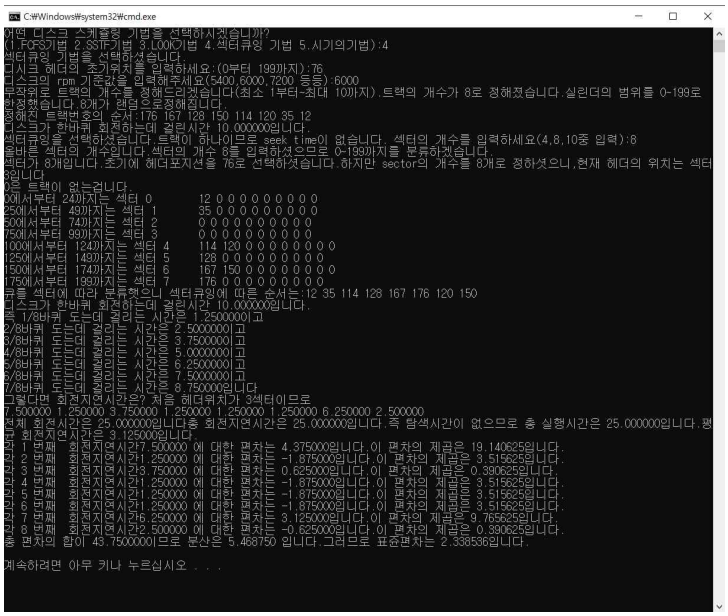
(4) 6000 RPM 섹터 4개



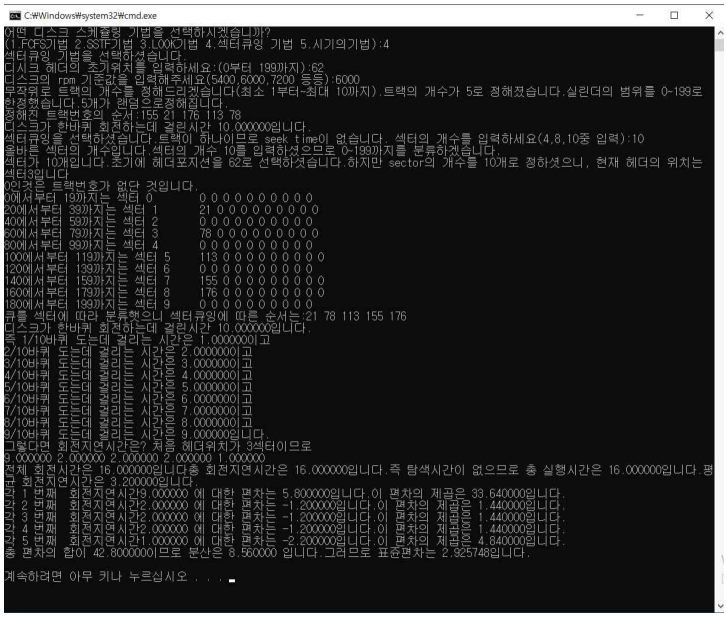


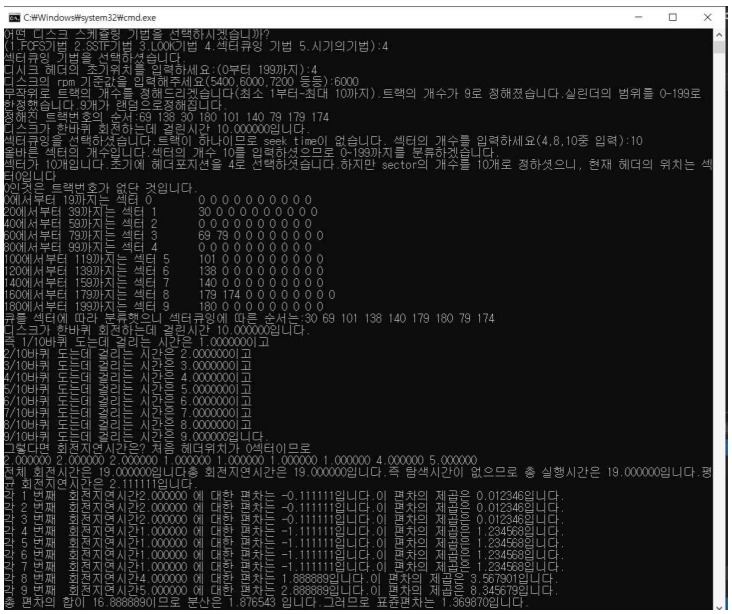
(5) 6000 RPM 섹터 8개



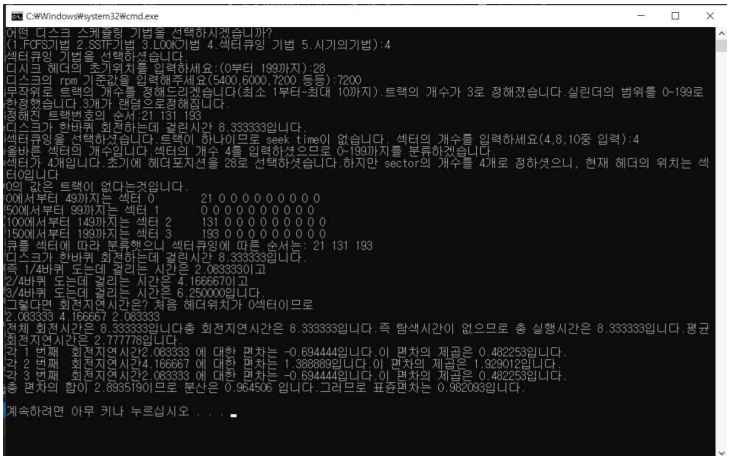


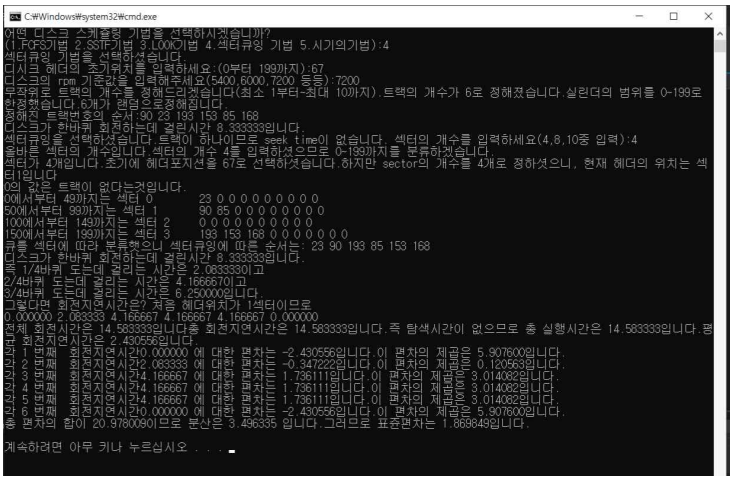
(6) 6000 RPM 섹터 10개



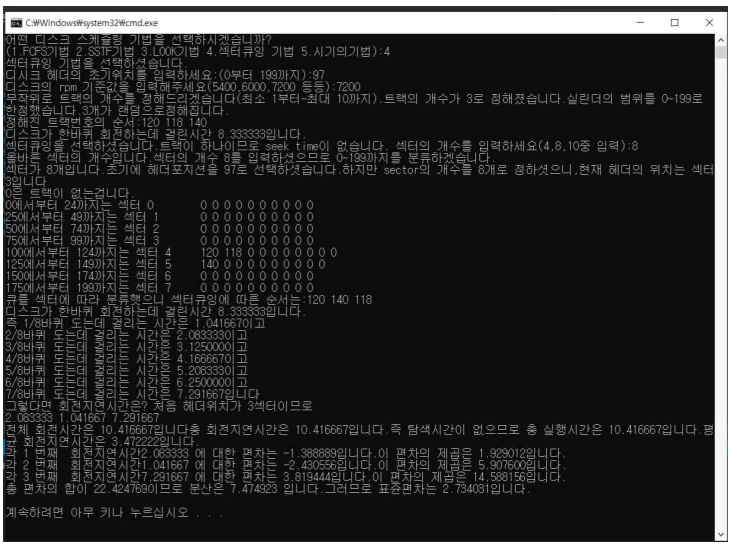


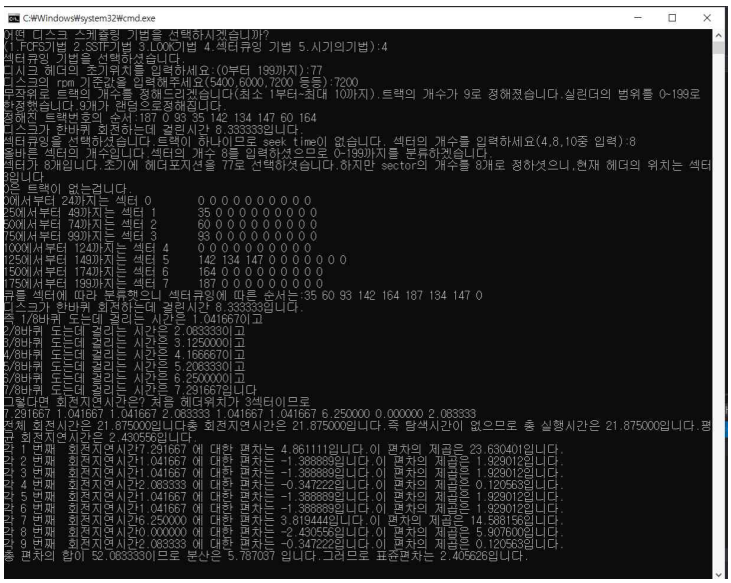
(7) 7200 RPM 섹터 4개



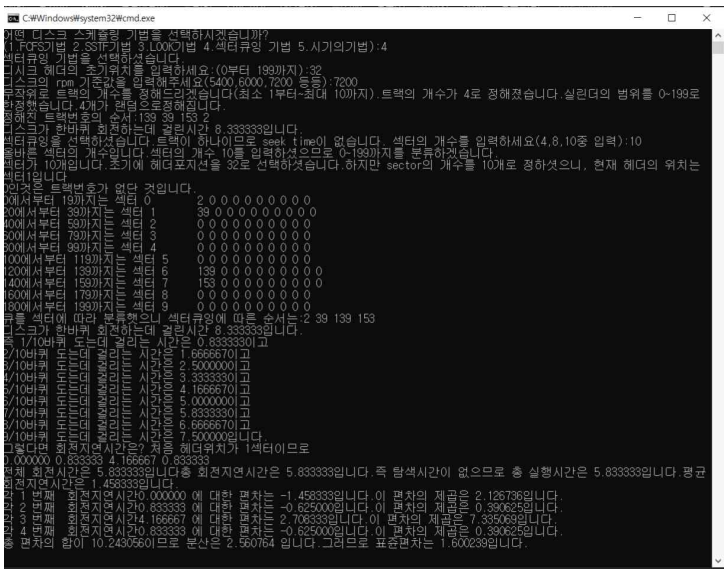


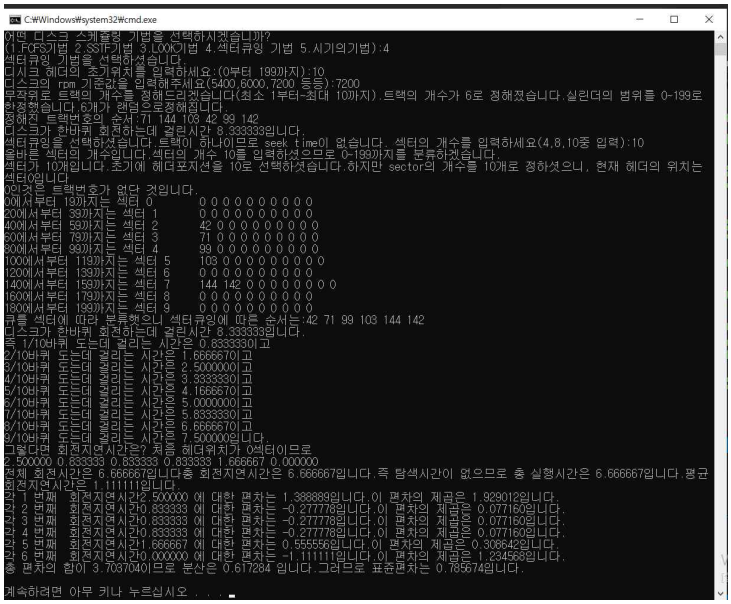
(8) 7200 RPM 섹터 8개





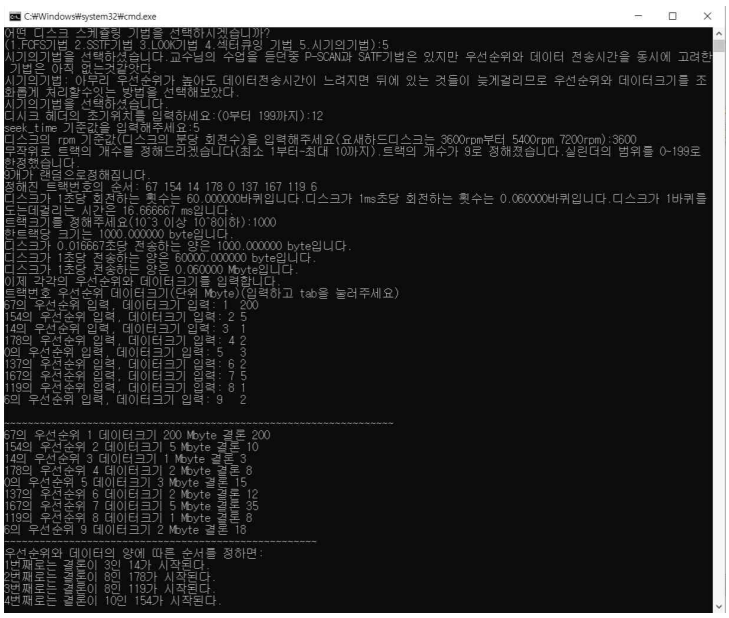
(9) 7200 RPM 섹터 10개

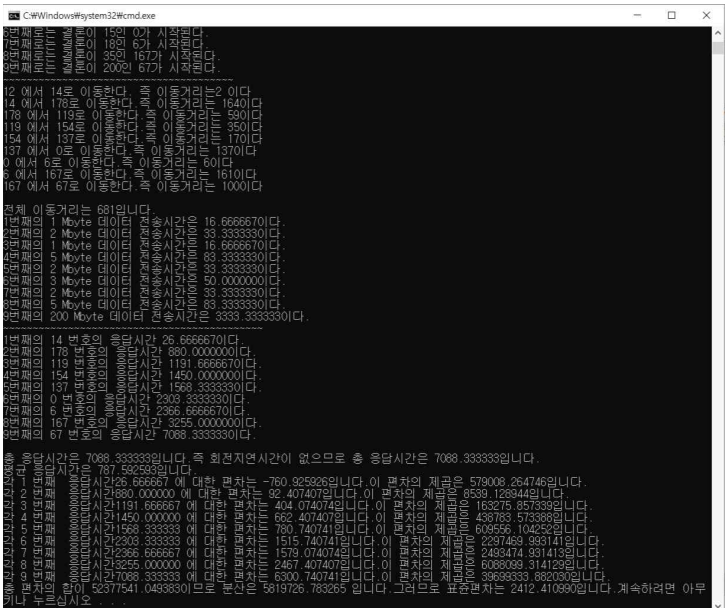




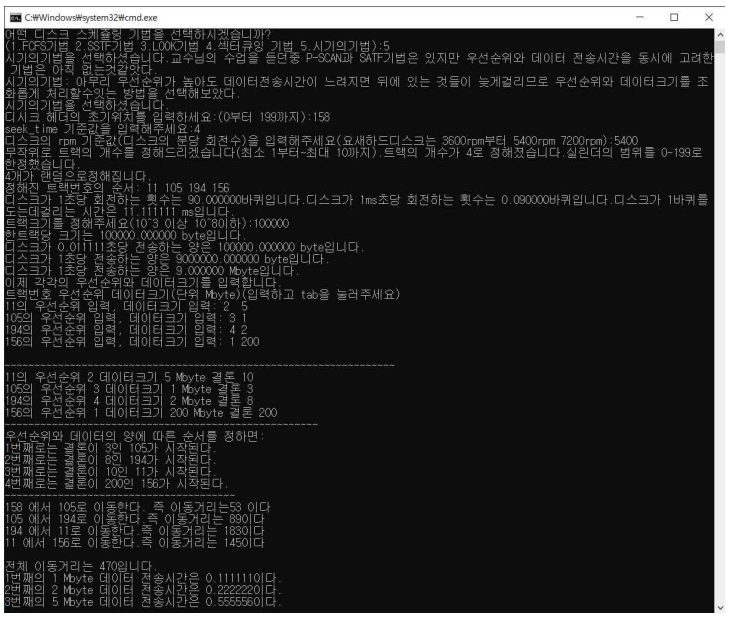
5. 시기의 기법

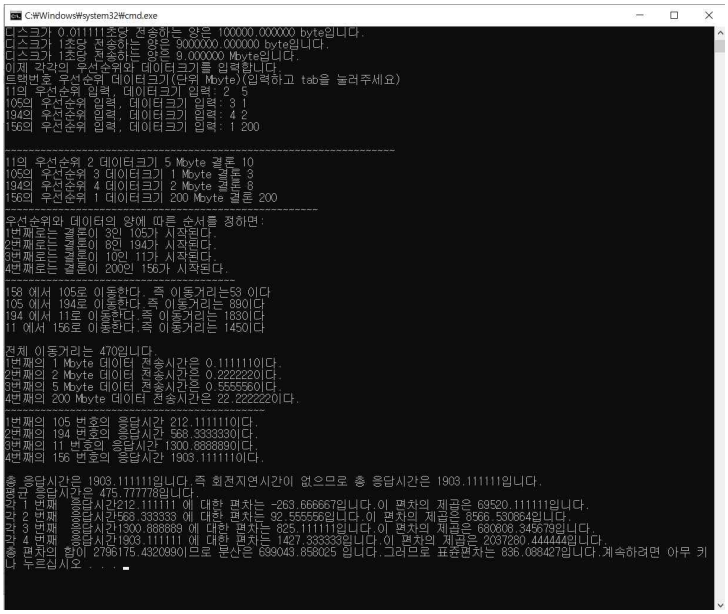
(1)3600RPM



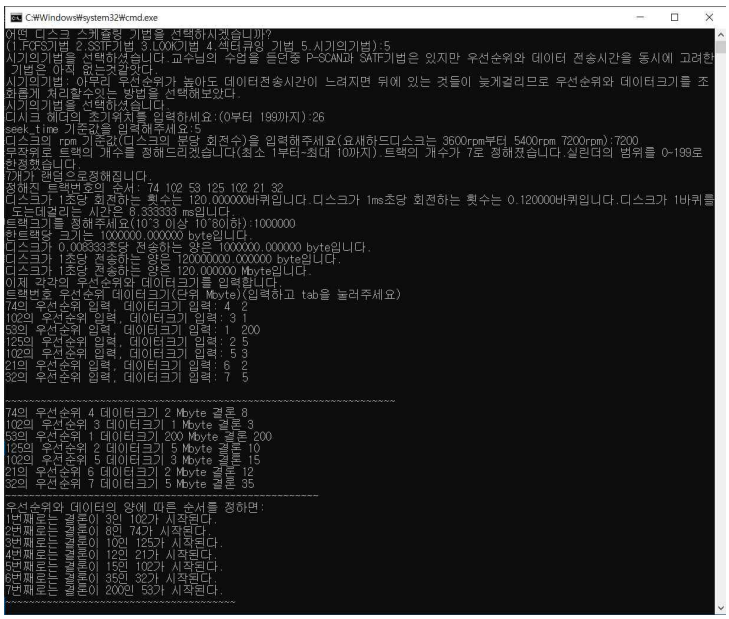

C:₩Windows₩system32₩cmd.exe
2번째로는 결론이 15인 0가 시작된다.
7번째로는 결론이 18인 6가 시작된다.
8번째로는 결론이 35인 167가 시작된다.
9번째로는 결론이 200인 67가 시작된다.
12 에서 14로 이동한다. 즉 이동거리는2 이다
14 에서 178로 이동한다.즉 이동거리는 164이다
178 에서 119로 이동한다.즉 이동거리는 59이다
119 에서 154로 이동한다.즉 이동거리는 35이다
154 에서 137로 이동한다.즉 이동거리는 17이다
137 에서 0로 이동한다.즉 이동거리는 137이다
0 에서 6로 이동한다.즉 이동거리는 6이다
6 에서 167로 이동한다.즉 이동거리는 161이다.
167 에서 67로 이동한다.즉 이동거리는 100이다
전체 이동거리는 681입니다.
1번째의 1 Mbyte 데이터 전송시간은 16.6666670이다.
2번째의 2 Mbyte 데이터 전송시간은 33.3333330이다.
3번째의 1 Mbyte 데이터 전송시간은 16.6666670이다.
4번째의 5 Mbyte 데이터 전송시간은 83.3333330이다.
5번째의 2 Mbyte 데이터 전송시간은 33.3333330이다.
6번째의 3 Mbyte 데이터 전송시간은 50.0000000이다.
7번째의 2 Mbyte 데이터 전송시간은 33.3333330이다.
8번째의 5 Mbyte 데이터 전송시간은 83.3333330이다.
9번째의 200 Mbyte 데이터 전송시간은 3333.3333330이다.
1번째의 14 번호의 응답시간 26.6666670이다.
2번째의 178 번호의 응답시간 880.0000000이다.
3번째의 119 번호의 응답시간 1191.6666670이다.
4번째의 154 번호의 응답시간 1450.0000000이다.
5번째의 137 번호의 응답시간 1568.3333330이다.
6번째의 0 번호의 응답시간 2303.3333330이다.
7번째의 6 번호의 응답시간 2366.6666670이다.
8번째의 167 번호의 응답시간 3255.0000000이다.
9번째의 67 번호의 응답시간 7088.3333330이다.
총 응답시간은 7088.333333입니다.즉 회전지연시간이 없으므로 총 응답시간은 7088.333333입니다.
평균 응답시간은 787.592593입니다.
각 1 번째 응답시간126.666667 에 대한 편차는 -760.925926입니다.이 편차의 제곱은 579008.264746입니다.
각 2 번째 응답시간880.000000 에 대한 편차는 92.407407입니다.이 편차의 제곱은 8539.128944입니다.
각 3 번째 응답시간1191.666667 에 대한 편차는 404.074074입니다.이 편차의 제곱은 163275.857339입니다.
각 4 번째 응답시간1450.000000 에 대한 편차는 662.407407입니다.이 편차의 제곱은 438783.573388입니다.
각 5 번째 응답시간1568.333333 에 대한 편차는 780.740741입니다.이 편차의 제곱은 609556.104252입니다.
각 6 번째 응답시간2303.333333 에 대한 편차는 1515.740741입니다.이 편차의 제곱은 2297469.993141입니다.
각 7 번째 응답시간2366.666667 에 대한 편차는 1579.074074입니다.이 편차의 제곱은 2493474.931419입니다.
각 8 번째 응답시간3255.000000 에 대한 편차는 2467.407407입니다.이 편차의 제곱은 6088089.314129입니다.
각 9 번째 응답시간7088.333333 에 대한 편차는 6300.740741입니다.이 편차의 제곱은 39699333.882030입니다.
총 편차의 합이 52377541.0493830이므로 분산은 5819726.783265 입니다.그러므로 표준편차는 2412.410990입니다.계속하려면 아무
키나 누르십시오 . . .

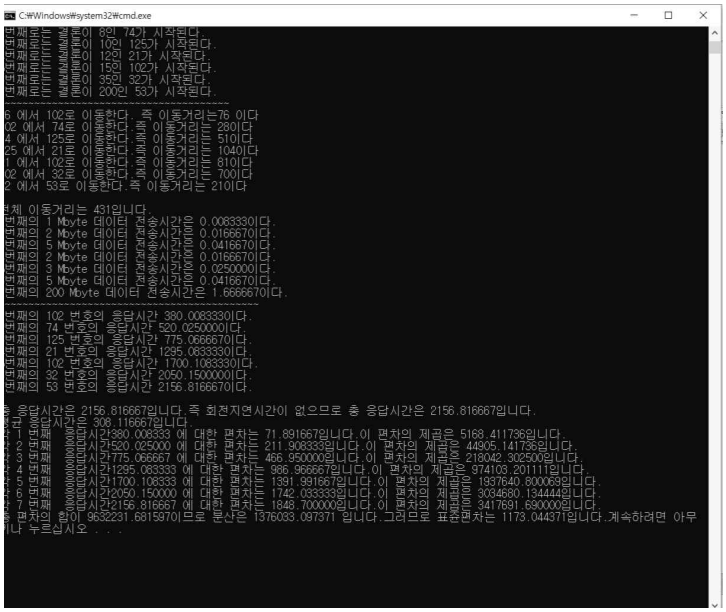
(2)5400RPM


C:₩Windows₩system32₩cmd.exe
어떤 디스크 스케줄링 기법을 선택하시겠습니까?
(1.FCFS기법 2.SSTF기법 3.LOOK기법 4.섹터큐잉 기법 5.시기의기법):5
시기의기법을 선택하셨습니다.교수님의 수업을 듣던중 P-SCAN과 SATF기법은 있지만 우선순위와 데이터 전송시간을 동시에 고려한
기법은 아직 없는거같았다.
시기의기법: 아무리 우선순위가 높아도 데이터전송시간이 느려지면 뒤에 있는 것들이 늦게걸리므로 우선순위와 데이터크기를 조
화롭게 처리할수있는 방법을 선택해보았다.
시기의기법을 선택하셨습니다.
디시크 헤더의 초기위치를 입력하세요:(0부터 199까지):158
seek time 기준값을 입력해주세요:4
디스크의 rpm 기준값(디스크의 분당 회전수)을 입력해주세요(요새하드디스크는 3600rpm부터 5400rpm 7200rpm):5400
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터~최대 10까지).트랙의 개수가 4로 정해졌습니다.실린더의 범위를 0~199로
한정했습니다.
4개가 랜덤으로정해집니다.
정해진 트랙번호의 순서: 11 105 194 156
디스크가 1초당 회전하는 횟수는 90.000000바퀴입니다.디스크가 1ms초당 회전하는 횟수는 0.090000바퀴입니다.디스크가 1바퀴를
도는데걸리는 시간은 11.111111 ms입니다.
트랙크기를 정해주세요(10^3 이상 10^8이하):100000
한트랙당 크기는 100000.000000 byte입니다.
디스크가 0.011111초당 전송하는 양은 100000.000000 byte입니다.
디스크가 1초당 전송하는 양은 9000000.000000 byte입니다.
디스크가 1초당 전송하는 양은 9.000000 Mbyte입니다.
이제 각각의 우선순위와 데이터크기를 입력합니다.
트랙번호 우선순위 데이터크기(단위 Mbyte)(입력하고 tab을 눌러주세요)
11의 우선순위 입력, 데이터크기 입력: 2 5
105의 우선순위 입력, 데이터크기 입력: 3 1
194의 우선순위 입력, 데이터크기 입력: 4 2
156의 우선순위 입력, 데이터크기 입력: 1 200
11의 우선순위 2 데이터크기 5 Mbyte 결론 10
105의 우선순위 3 데이터크기 1 Mbyte 결론 3
194의 우선순위 4 데이터크기 2 Mbyte 결론 6
156의 우선순위 1 데이터크기 200 Mbyte 결론 200
우선순위와 데이터의 양에 따른 순서를 정하면:
1번째로는 결론이 3인 105가 시작된다.
2번째로는 결론이 8인 194가 시작된다.
3번째로는 결론이 10인 11가 시작된다.
4번째로는 결론이 200인 156가 시작된다.
158 에서 105로 이동한다. 즉 이동거리는53 이다
105 에서 194로 이동한다. 즉 이동거리는 88이다
194 에서 11로 이동한다.즉 이동거리는 183이다
11 에서 156로 이동한다.즉 이동거리는 145이다
전체 이동거리는 470입니다.
1번째의 1 Mbyte 데이터 전송시간은 0.1111110이다.
2번째의 2 Mbyte 데이터 전송시간은 0.2222220이다.
3번째의 5 Mbyte 데이터 전송시간은 0.5555560이다.


C:₩Windows₩system32₩cmd.exe
디스크가 0.011111초당 전송하는 양은 100000.000000 byte입니다.
디스크가 1초당 전송하는 양은 9000000.000000 byte입니다.
디스크가 1초당 전송하는 양은 9.000000 Mbyte입니다.
이제 각각의 우선순위와 데이터크기를 입력합니다.
트랙번호 우선순위 데이터크기(단위 Mbyte)(입력하고 tab을 눌러주세요)
11의 우선순위 입력, 데이터크기 입력: 2 5
105의 우선순위 입력, 데이터크기 입력: 3 1
194의 우선순위 입력, 데이터크기 입력: 4 2
156의 우선순위 입력, 데이터크기 입력: 1 200
11의 우선순위 2 데이터크기 5 Mbyte 결론 10
105의 우선순위 3 데이터크기 1 Mbyte 결론 3
194의 우선순위 4 데이터크기 2 Mbyte 결론 8
156의 우선순위 1 데이터크기 200 Mbyte 결론 200
우선순위와 데이터의 양에 따른 순서를 정하면:
1번째로는 결론이 3인 105가 시작된다.
2번째로는 결론이 8인 194가 시작된다.
3번째로는 결론이 10인 11가 시작된다.
4번째로는 결론이 200인 156가 시작된다.
158 에서 105로 이동한다. 즉 이동거리는53 이다
105 에서 194로 이동한다. 즉 이동거리는 89이다
194 에서 11로 이동한다.즉 이동거리는 183이다
11 에서 156로 이동한다.즉 이동거리는 145이다
전체 이동거리는 470입니다.
1번째의 1 Mbyte 데이터 전송시간은 0.111111이다.
2번째의 2 Mbyte 데이터 전송시간은 0.222222이다.
3번째의 5 Mbyte 데이터 전송시간은 0.555556이다.
4번째의 200 Mbyte 데이터 전송시간은 22.222220이다.
1번째의 105 번호의 응답시간 212.111111이다.
2번째의 194 번호의 응답시간 568.333333이다.
3번째의 11 번호의 응답시간 1300.888889이다.
4번째의 156 번호의 응답시간 1903.111111이다.
총 응답시간은 1903.111111입니다.즉 회전지연시간이 없으므로 총 응답시간은 1903.111111입니다.
평균 응답시간은 475.777778입니다
각 1 번째 응답시간212.111111 에 대한 편차는 -263.666667입니다.이 편차의 제곱은 69520.111111입니다.
각 2 번째 응답시간568.333333 에 대한 편차는 92.555556입니다.이 편차의 제곱은 8566.530864입니다.
각 3 번째 응답시간1300.888889 에 대한 편차는 825.111111입니다.이 편차의 제곱은 680808.345679입니다.
각 4 번째 응답시간1903.111111 에 대한 편차는 1427.333333입니다.이 편차의 제곱은 2037280.444444입니다.
총 편차의 합이 2796175.4320990이므로 분산은 699043.858025 입니다.그러므로 표준편차는 836.088427입니다.계속하려면 아무 키나 누르십시오 . . .

(3)7200RPM


C:₩Windows₩system32₩cmd.exe
어떤 디스크 스케쥴링 기법을 선택하시겠습니까?
(1.FCFS기법 2.SSTF기법 3.LOOK기법 4.섹터큐잉 기법 5.시기의기법):5
시기의기법을 선택하셨습니다.교수님의 수업을 듣던중 P-SCAN과 SATF기법은 있지만 우선순위와 데이터 전송시간을 동시에 고려한 기법은 아직 없는것같아다.
시기의기법: 아무리 우선순위가 높아도 데이터전송시간이 느려지면 뒤에 있는 것들이 늦게걸리므로 우선순위와 데이터크기를 조화롭게 처리할수있는 방법을 선택해보았다.
시기의기법을 선택하셨습니다.
디시크 헤더의 초기위치를 입력하세요:(0부터 199까지):26
seek_time 기준값을 입력해주세요:5
디스크의 rpm 기준값(디스크의 분당 회전수)을 입력해주세요(요새하드디스크는 3600rpm부터 5400rpm 7200rpm):7200
무작위로 트랙의 개수를 정해드리겠습니다(최소 1부터-최대 10까지) 트랙의 개수가 7로 정해졌습니다.실린더의 범위를 0~199로 한정했습니다.
7개가 랜덤으로정해집니다.
정해진 트랙번호의 순서: 74 102 53 125 102 21 32
디스크가 1초당 회전하는 횟수는 120.000000바퀴입니다.디스크가 1ms초당 회전하는 횟수는 0.120000바퀴입니다.디스크가 1바퀴를 도는데걸리는 시간은 8.333333 ms입니다.
트랙크기를 정해주세요(10~3 이상 10~50이하):1000000
한트랙당 크기는 1000000.000000 byte입니다.
디스크가 0.008333초당 전송하는 양은 1000000.000000 byte입니다.
디스크가 1초당 전송하는 양은 120000000.000000 byte입니다.
디스크가 1초당 전송하는 양은 120.000000 Mbyte입니다.
이제 각각의 우선순위와 데이터크기를 입력합니다.
트랙번호 우선순위 데이터크기(단위 Mbyte)(입력하고 tab을 눌러주세요)
74의 우선순위 입력, 데이터크기 입력: 4 2
102의 우선순위 입력, 데이터크기 입력: 3 1
53의 우선순위 입력, 데이터크기 입력: 1 200
125의 우선순위 입력, 데이터크기 입력: 2 5
102의 우선순위 입력, 데이터크기 입력: 5 3
21의 우선순위 입력, 데이터크기 입력: 6 2
32의 우선순위 입력, 데이터크기 입력: 7 5
74의 우선순위 4 데이터크기 2 Mbyte 결론 8
102의 우선순위 3 데이터크기 1 Mbyte 결론 3
53의 우선순위 1 데이터크기 200 Mbyte 결론 200
125의 우선순위 2 데이터크기 5 Mbyte 결론 10
102의 우선순위 5 데이터크기 3 Mbyte 결론 15
21의 우선순위 6 데이터크기 2 Mbyte 결론 12
32의 우선순위 7 데이터크기 5 Mbyte 결론 35
우선순위와 데이터의 양에 따른 순서를 정하면:
1번째로는 결론이 3인 102가 시작된다.
2번째로는 결론이 8인 74가 시작된다.
3번째로는 결론이 10인 125가 시작된다.
4번째로는 결론이 12인 21가 시작된다.
5번째로는 결론이 15인 102가 시작된다.
6번째로는 결론이 35인 32가 시작된다.
7번째로는 결론이 200인 53가 시작된다.


C:₩Windows₩system32₩cmd.exe
번째로는 셀룰이 8인 74가 시작된다.
번째로는 셀룰이 10인 125가 시작된다.
번째로는 셀룰이 12인 21가 시작된다.
번째로는 셀룰이 15인 102가 시작된다.
번째로는 셀룰이 35인 32가 시작된다.
번째로는 셀룰이 200인 53가 시작된다.
6 에서 102로 이동한다. 즉 이동거리는76 이다
02 에서 74로 이동한다.즉 이동거리는 280이다
4 에서 125로 이동한다.즉 이동거리는 510이다.
25 에서 21로 이동한다.즉 이동거리는 1040이다
1 에서 102로 이동한다.즉 이동거리는 810이다
02 에서 32로 이동한다.즉 이동거리는 700이다
2 에서 53로 이동한다.즉 이동거리는 210이다
체 이동거리는 431입니다.
번째의 1 Mbyte 데이터 전송시간은 0.0083330이다.
번째의 2 Mbyte 데이터 전송시간은 0.0166670이다.
번째의 5 Mbyte 데이터 전송시간은 0.0416670이다.
번째의 2 Mbyte 데이터 전송시간은 0.0166670이다.
번째의 3 Mbyte 데이터 전송시간은 0.0250000이다.
번째의 5 Mbyte 데이터 전송시간은 0.0416670이다.
번째의 200 Mbyte 데이터 전송시간은 1.6666670이다.
번째의 102 번호의 응답시간 380.0083330이다.
번째의 74 번호의 응답시간 520.0250000이다.
번째의 125 번호의 응답시간 775.0666670이다.
번째의 21 번호의 응답시간 1295.0833330이다.
번째의 102 번호의 응답시간 1700.1083330이다.
번째의 32 번호의 응답시간 2050.1500000이다.
번째의 53 번호의 응답시간 2156.8166670이다.
응답시간은 2156.816667입니다.즉 회전지연시간이 없으므로 총 응답시간은 2156.816667입니다.
균 응답시간은 308.116667입니다.
각 1 번째 응답시간380.008333 에 대한 편차는 71.891667입니다.이 편차의 제곱은 5168.411736입니다.
각 2 번째 응답시간520.025000 에 대한 편차는 211.908333입니다.이 편차의 제곱은 44905.141736입니다.
각 3 번째 응답시간775.066667 에 대한 편차는 466.950000입니다.이 편차의 제곱은 218042.302500입니다.
각 4 번째 응답시간1295.083333 에 대한 편차는 986.966667입니다.이 편차의 제곱은 974103.201111입니다.
각 5 번째 응답시간1700.108333 에 대한 편차는 1391.991667입니다.이 편차의 제곱은 1937640.800069입니다.
각 6 번째 응답시간2050.150000 에 대한 편차는 1742.033333입니다.이 편차의 제곱은 3034680.134444입니다.
각 7 번째 응답시간2156.816667 에 대한 편차는 1848.700000입니다.이 편차의 제곱은 3417691.690000입니다.
편차의 합이 9632231.6815970이므로 분산은 1376033.097371 입니다.그러므로 표준편차는 1173.044371입니다.계속하려면 아무
나 누르십시오 . . .