컴퓨터학부 20192403 박상철

**개요**

링크드 리스트를 통해 데이터를 관리하고 파일과 디렉토리 관련 시스템콜과 라이브러리 함수들을 사용하는 법을 익히며 쓰레드의 사용을 익히는 것을 목표로 학생들이 제출한 답안과 정답 파일을 비교히여 답안 파일을 생성하는 프로그램이다.

**기능**

1. ssu\_score : 학생들의 답안 파일과 정답 파일의 경로를 입력받아 답안과 정답을 비교하여 채점을 수행하고 결과를 csv파일로 저장

-n : 들어 온 인자의 파일명으로 점수 테이블 생성

-m : 점수 테이블 파일 내의 특정 문제 번호의 배점 수정

-c : 가변인자로 학생들의 학번을 입력받아 해당하는 학번의 점수와 총점을 출력

-p : 링크드리스트로 학생들의 틀린 문제를 관리하여 채점이 끝나면 가변인자로 입력받은 학생들의 틀린 문제 목록을 출력

-t : 가변 인자로 입력받은 c파일 컴파일 시 lpthread 옵션을 추가하여 채점

-s : (학번 순 | 점수 순), (오름차순 | 내림차순)을 입력받아 정렬하여 파일에 저장

-e : 에러 디렉토리를 생성하여 c파일에서 발생한 에러를 학번으로 생성

-h : 사용법 출력

**상세 설계**

ssu\_score.c에서는 아래 함수들이 사용된다 중요한 함수를 위주로 프로그램을 설명하자면 먼저 ssu\_score함수가 실행된다.

ssu\_score함수에서는 인자로 받은 학생 디렉토리 경로와 정답 디렉토리 경로를 통해 set\_idTable함수로 학생 테이블과 set\_scoreTable함수로 점수 테이블을 생성하고 옵션을 처리한 후 score\_students 함수로 채점을 진행한다. score\_students에서는 학생들의 점수를 기록 할 csv파일을 생성하고 score\_student 함수를 호출해 학생의 점수를 계산한 후 기록한다.

score\_student 함수에서는 빈칸 문제인지 c 프로그램인지를 판별하여 각각 score\_blank와 score\_program 함수를 호출하여 채점을 진행하고 채점 진행 상황을 화면에 출력한다.

score\_blank 함수에서는 blank.c에 정의 된 함수들로 학생들의 답안을 토큰으로 분리하여 make\_tree 함수로 이진 트리로 변환하여 정답 파일을 이진트리화 한것돠 비교하여 정답 여부를 판정한다.

score\_program에서는 compile\_program 함수로 컴파일을 진행하고 excute\_program 함수로 실행 파일을 실행하여 정답 여부를 판정한다. 위 과정을 진행하는 동안 -n 옵션을 이용하면 newScoreFile 변수에 새로운 파일명을 기록하여 원하는 이름으로 정답 파일을 생성 할 수 있다.

-m 옵션을 이용하면 채점을 진행하기 전에 do\_mOption함수를 호출하여 수정할 문제와 점수를 입력받고 score\_table.csv에서 해당 문제의 배점을 수정하여준다, 이때 -m 옵션 사용 했을 때 시작시에 score\_table이 존재하지 않았다면 생성하지 않고 종료한다.

다음으로 -c와 -p 옵션을 사용하면 getopt 함수 내에서 retfirstTarget와 retStudentList함수를 통해 각각 첫번째 인자와 전체 인자 리스트를 가져온다.

첫번째 인자는 c와 p 옵션이 개별적으로 가지며 전체 인자 리스트는 공유한다.

두 옵션이 같이 쓰이는 경우 둘 중 하나만 인자를 가질 수 있으므로 첫번째 인자가 둘 다 NULL이 아니라면 에러를 출력하고 종료하며 인자가 5개가 넘으면 에러 메세지와 함께 채점에 반영하지 않도록 구현하였다.

c 옵션을 사용하면 채점이 종료 될 때 사용자 학 번 뒤에 점수가 출력되며 p 옵션을 사용하면 틀린 문제들을 링크드 리스트로 관리하여 채점이 끝날 때 마다 출력을 진행한다.

만약 옵션이 선언되었는데 인자가 없다면 모든 학생에 대하여 옵션을 적용한다.

-t 옵션을 사용하면 threadFiles에 인자를 받아 해당 인자에 속한 파일들에 대해 컴파일시 -lpthread 옵션을 적용하여 컴파일을 진행한다.

이 또한 인자가 주어지지 않으면 모든 문제에 대해 처리하고 인자가 5개가 넘으면 에러문구와 함께 6번째 부터는 적용하지 않는다.

다음 -s옵션을 사용하면 <stdid(학번 순) | score(성적 순)>과 <1(오름차순) | -1(내림차순)> 을 입력받아 해당 하는 순서로 정렬하여 점수파일에 저장한다.

이때 링크드 리스트로 채점이 끝날때 마다 학번과 점수를 저장하는 구조체로 링크드 리스트를 만들었고 프로그램이 끝나면 링크드 리스트를 입력받은 조건으로 정렬하여 채점 결과 파일의 내용을 줄단위로 입력받아 배열에 저장하고 링크드리스트 순서에 맞게 정렬하여 다시 작성하는 방법을 사용했다.

-e 옵션은 에러 디렉토리 경로를 인자로 받아서 해당 인자에 디렉토리를 생성하고 학생별 디렉토리를 만들어 에러가 발생한 파일들의 에러문구를 저장한다.

-h 옵션을 프로그램 가장 시작 부분에서 실행하며 존재 할시 사용법을 출력하고 프로그램을 종료한다.

<ssu\_score에서 사용되는 함수>

void ssu\_score(int argc, char \*argv[]);

int check\_option(int argc, char \*argv[]);

void print\_usage();

void score\_students();

double score\_student(int fd, char \*id);

void write\_first\_row(int fd);

char \*get\_answer(int fd, char \*result);

int score\_blank(char \*id, char \*filename);

double score\_program(char \*id, char \*filename);

double compile\_program(char \*id, char \*filename);

int execute\_program(char \*id, char \*filname);

pid\_t inBackground(char \*name);

double check\_error\_warning(char \*filename);

int compare\_resultfile(char \*file1, char \*file2);

void readScoreFile(char scoreArr[][BUFLEN]);

void writeSortedFile(char scoreArr[][BUFLEN]);

void bubbleSortScore();

void reverseBubbleSortScore();

void bubbleSortStdid();

void reverseBubbleSortStdid();

void do\_sOption();

void do\_mOption(char \*ansDir);

int is\_exist(char (\*src)[FILELEN], char \*target);

int is\_thread(char \*qname);

int get\_file\_type(char \*filename);

void rmdirs(const char \*path);

void to\_lower\_case(char \*c);

void set\_scoreTable(char \*ansDir);

void read\_scoreTable(char \*path);

void make\_scoreTable(char \*ansDir);

void write\_scoreTable(char \*filename);

void set\_idTable(char \*stuDir);

int get\_create\_type();

void sort\_idTable(int size);

void sort\_scoreTable(int size);

void get\_qname\_number(char \*qname, int \*num1, int \*num2);

int get\_s\_mode(char \*mode);

int get\_s\_mode2(int argc, char \*argv[]);

char \*getExt(char\* filename);

void appendWrongList(wrongList \*\*wrongRoot, char \*fileName, double score);

void printWrongList(wrongList \*wrongRoot);

void FreeWrongList(wrongList \*wrongRoot);

void appendScoreList(char \*studentName, double totalScore);

void freeScoreList();

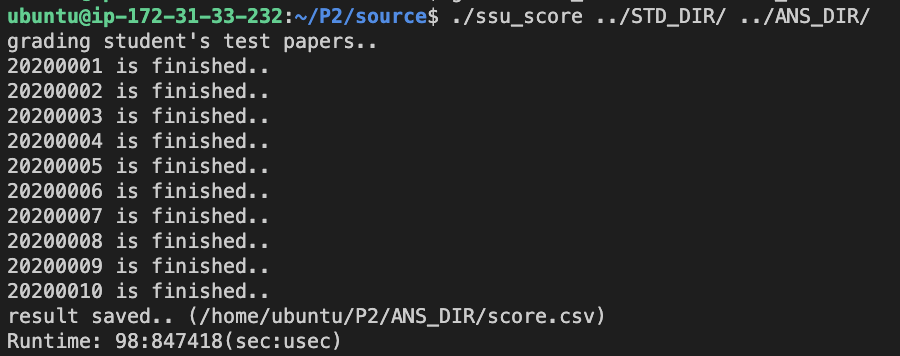
**프로그램 설계도**

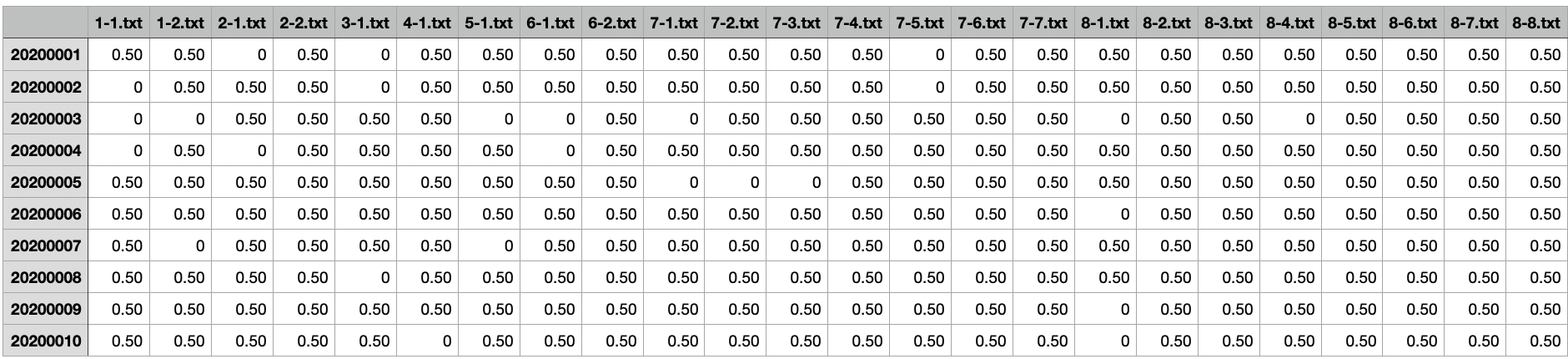
도표이(가) 표시된 사진

자동 생성된 설명

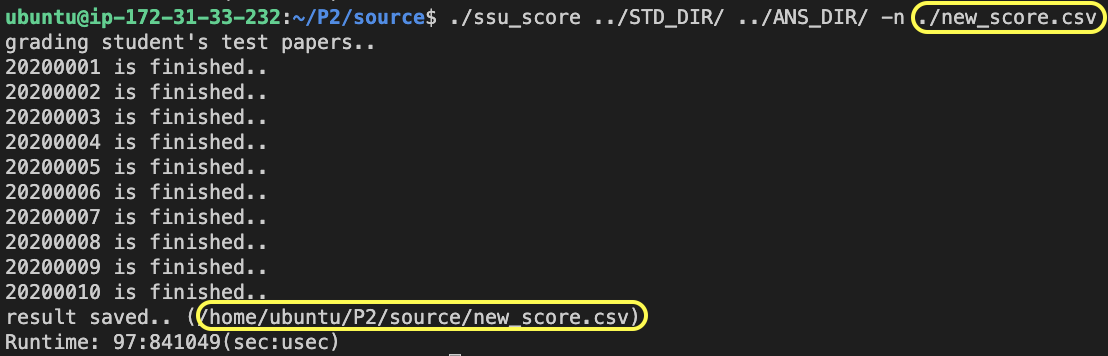
**실행 결과**

**기본 채점 실행 결과)**

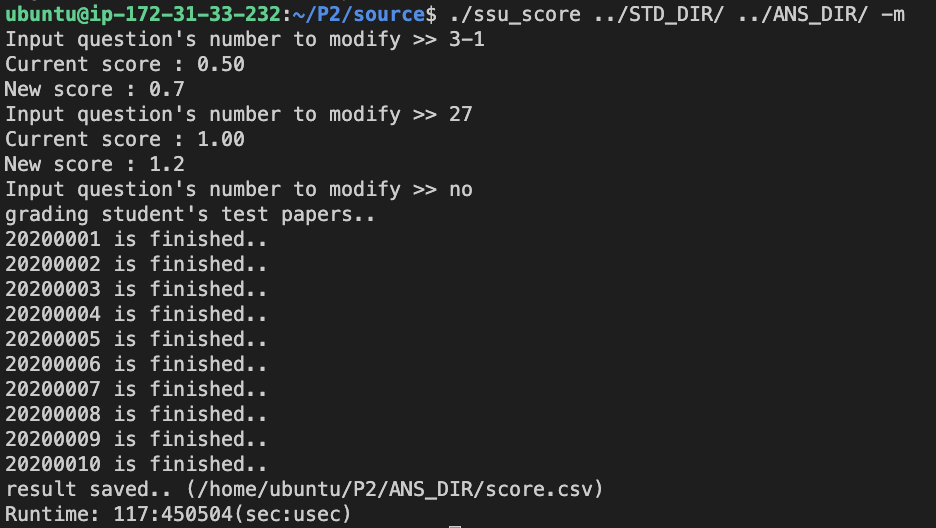




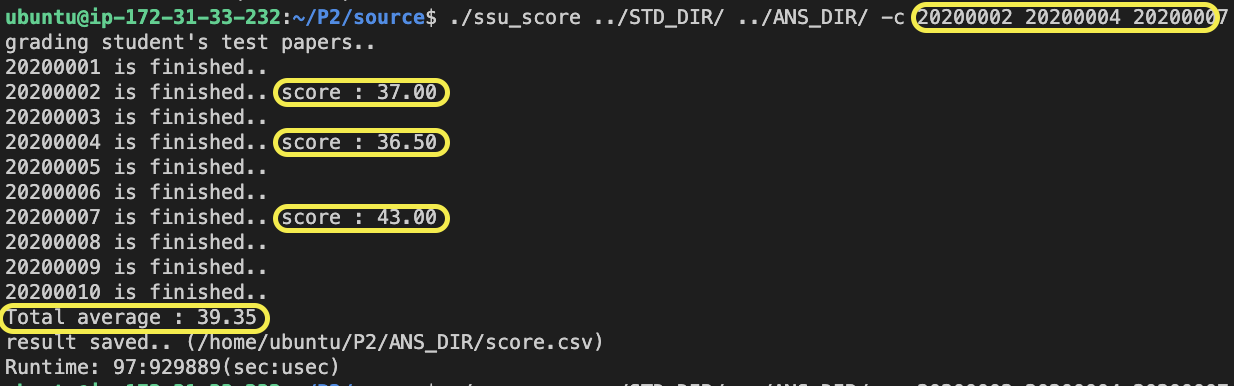
**-n 옵션 적용 결과)**



**-m 옵션 적용 결과)**



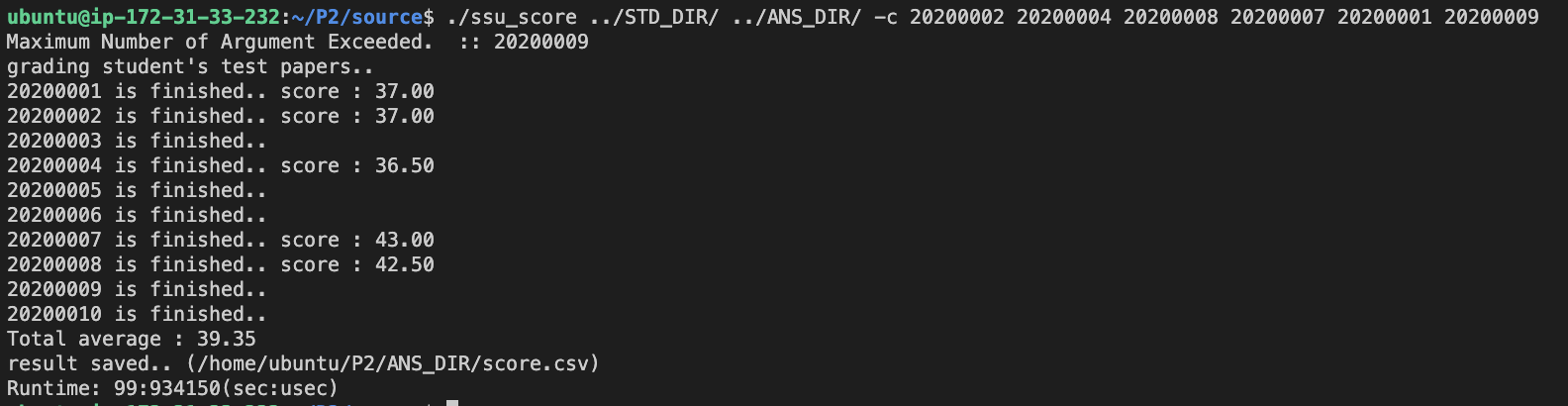
**-c 옵션 적용 결과)**



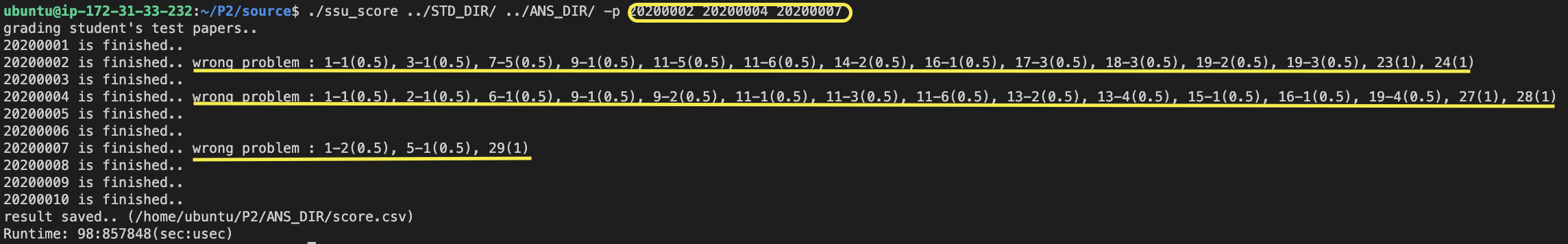
-c 없는 학생일 때)



-c 인자 5개를 넘을 때



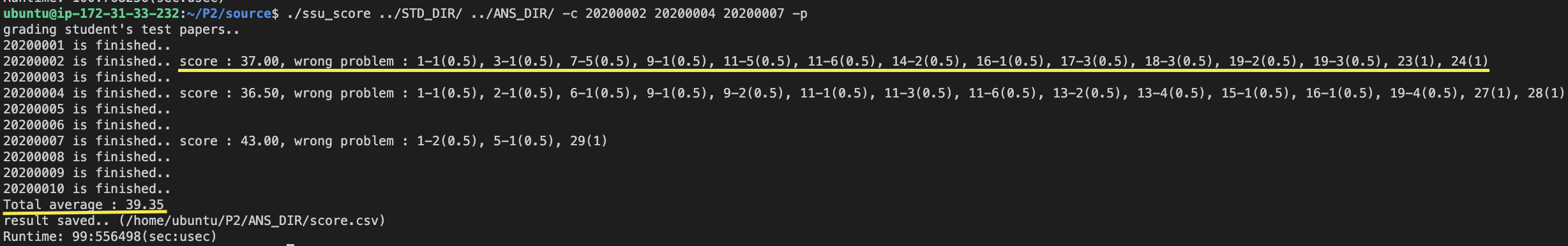
**-p 옵션 적용 결과)**



-p 없는 학생일 때)



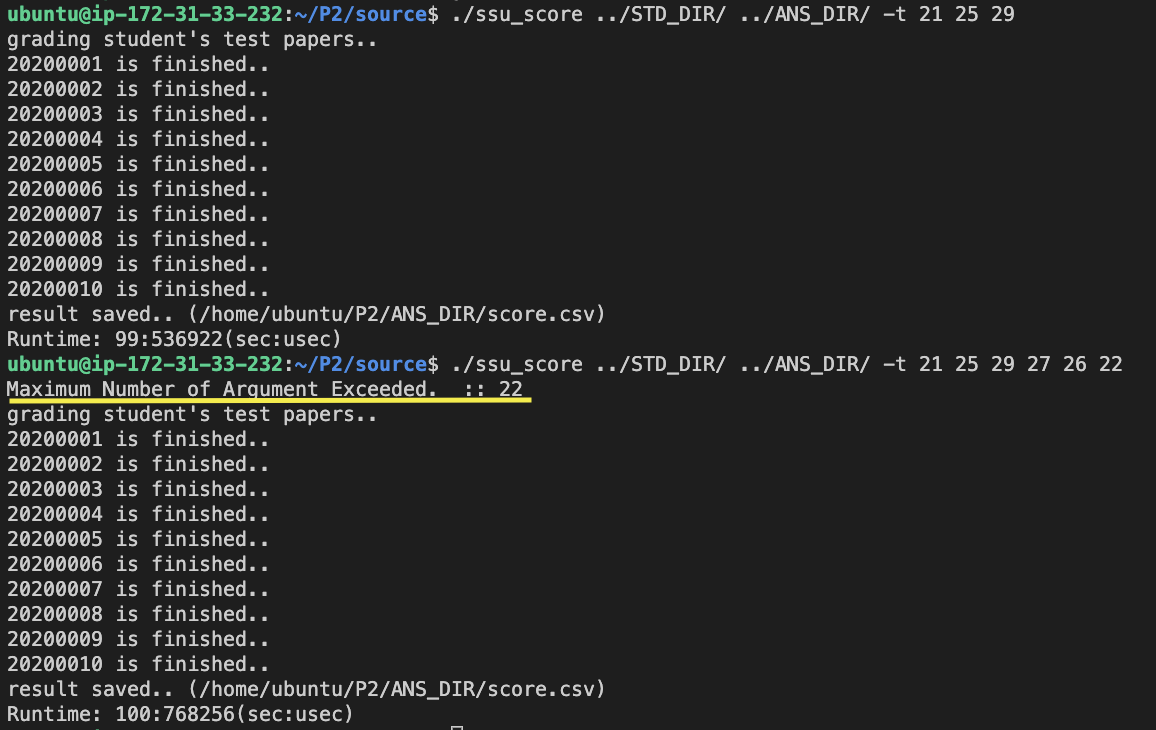
**-c와 -p 이중 적용 결과)**



-c, -p 옵션이 이중으로 들어올 때)

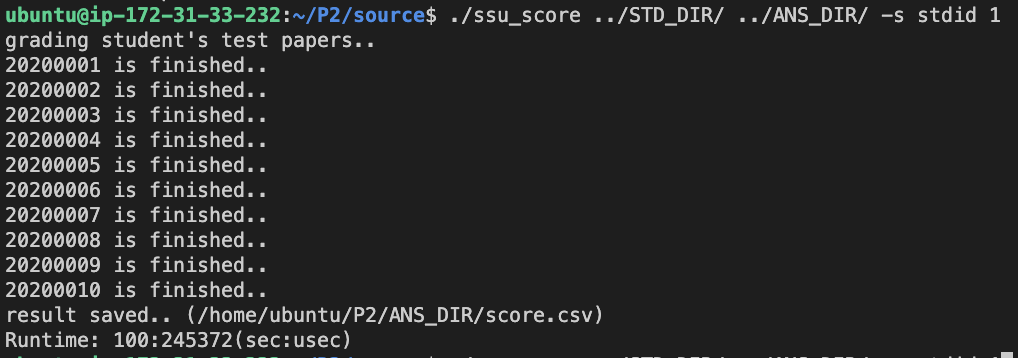


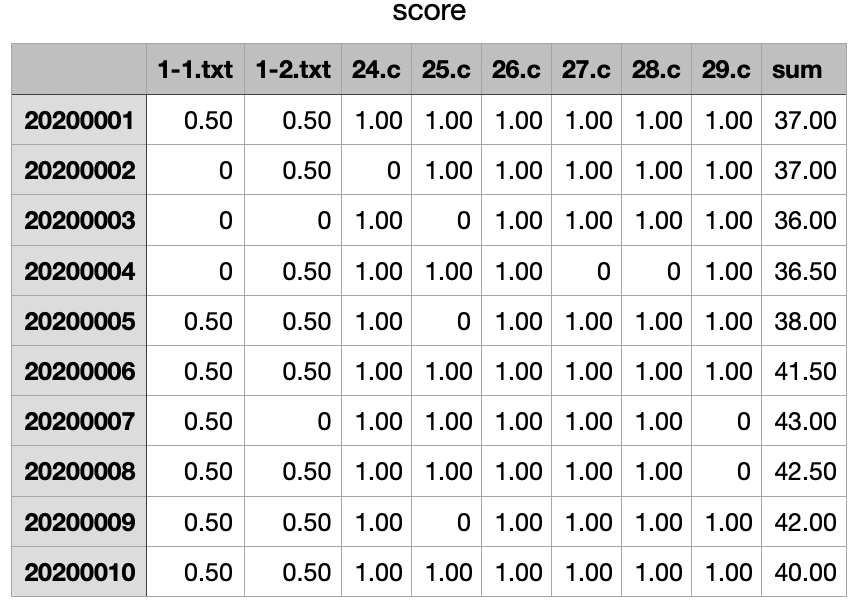
**-t 옵션 적용 결과 및 인자 5개를 넘을 때)**



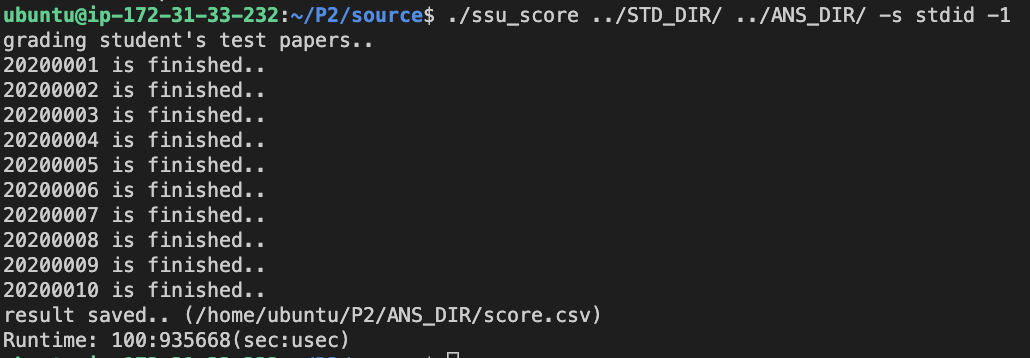
**-s 옵션 적용 결과)**

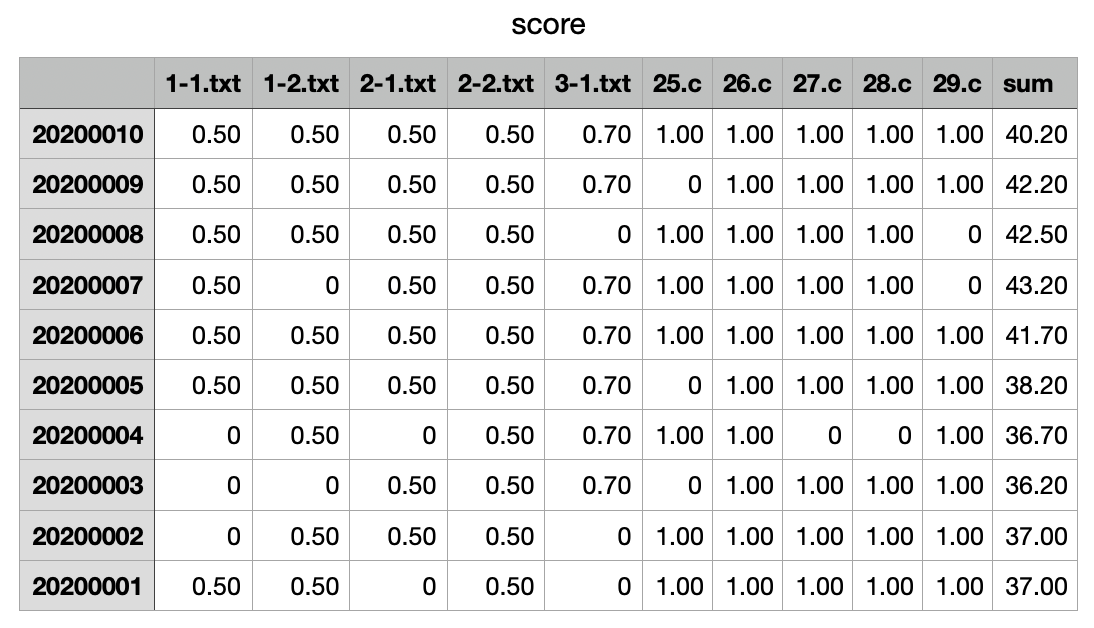
학번 오름차순)



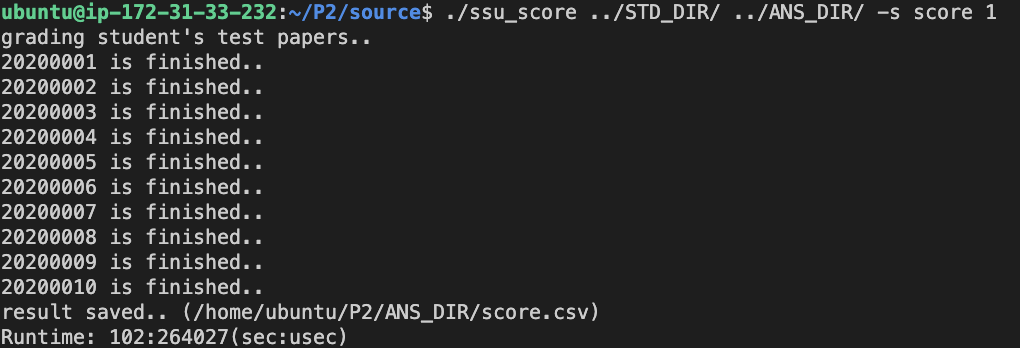


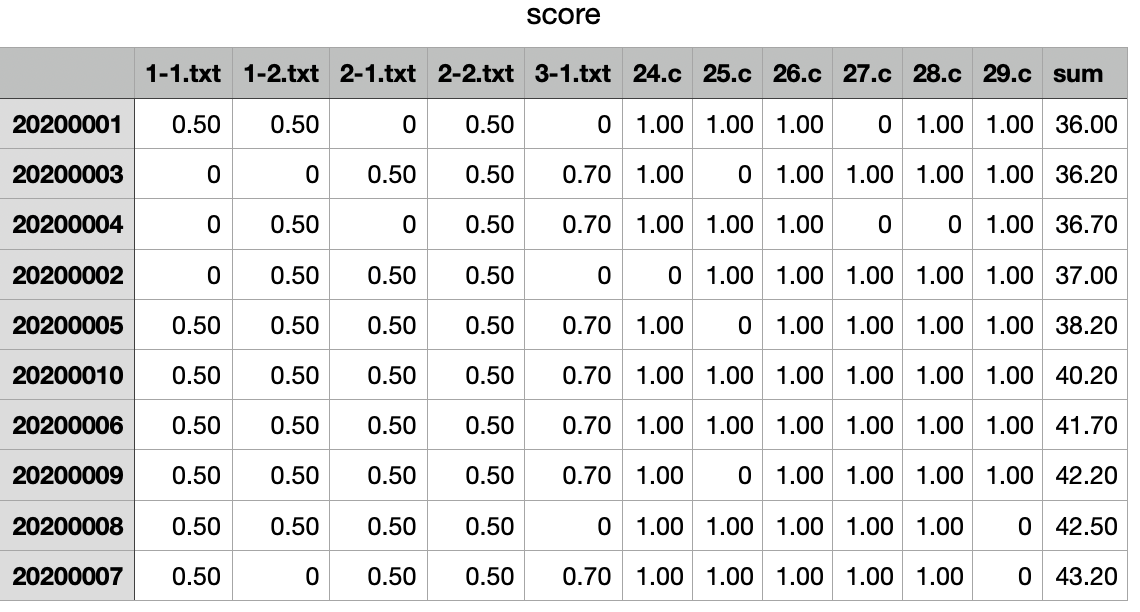
학번 내림차순



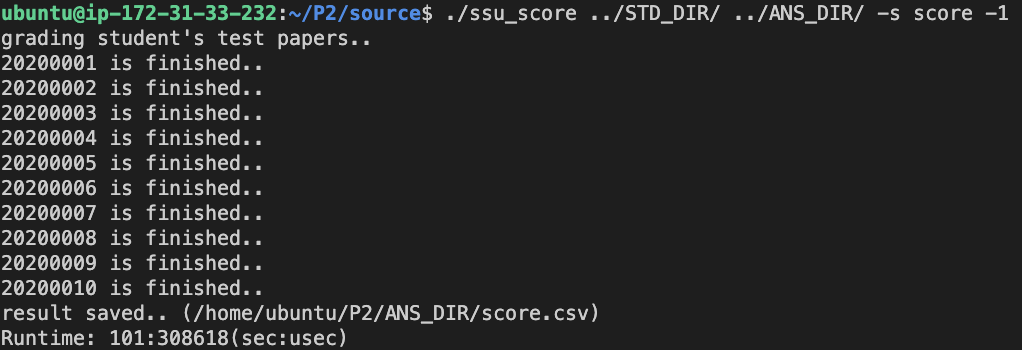


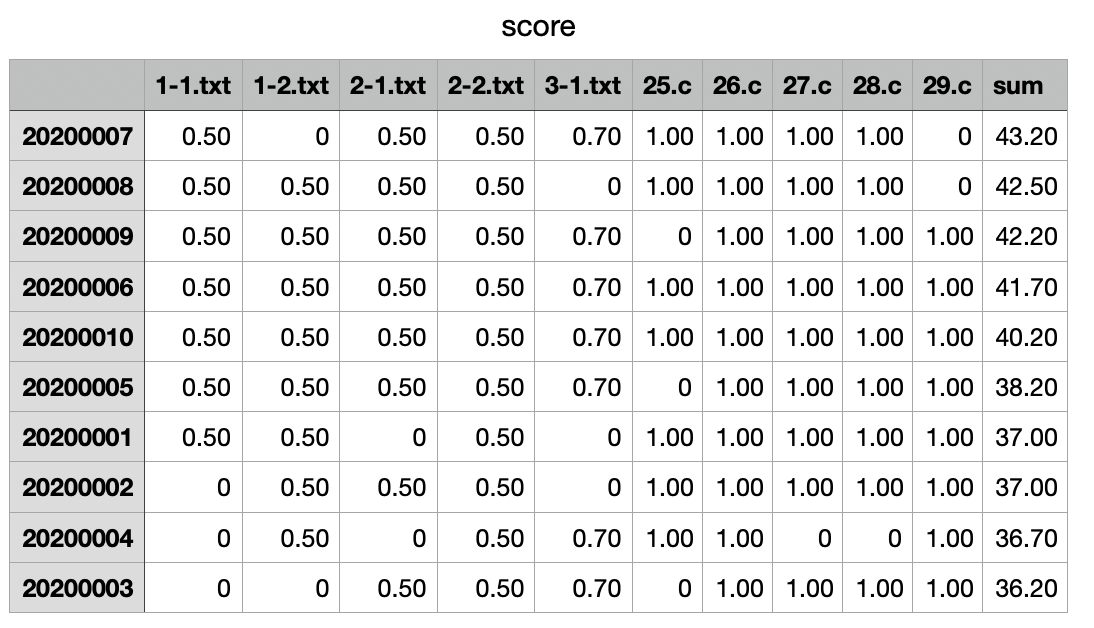
성적 오름차순



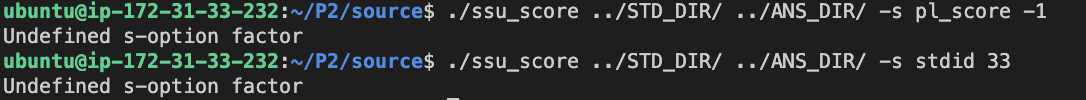


성적 내림차순)

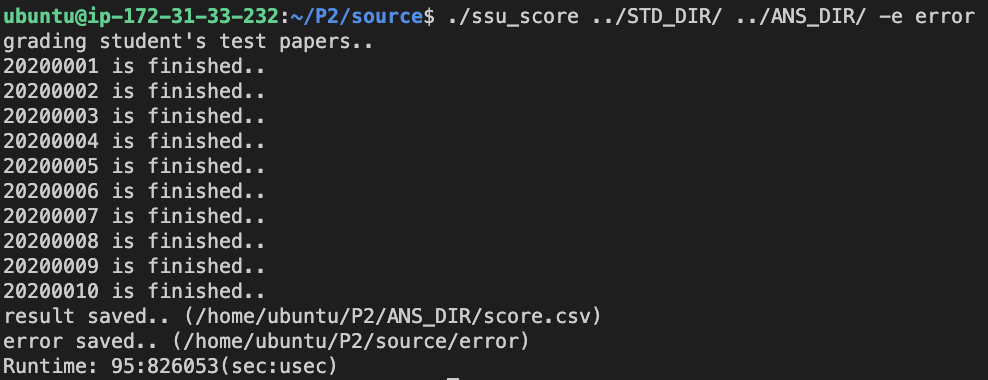




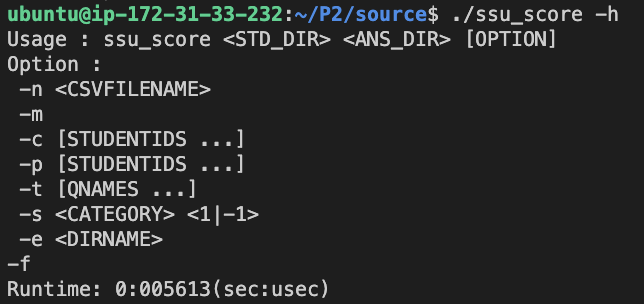
-s 잘못된 인자)



-e 옵션 적용 결과)



-h 옵션 적용 결과)



여러 가지 옵션 동시 적용 결과)

