|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 8주차 - A 문제해결기법 보고서 | | | | | |
| 문제 | 토익 성적 순서 찾기 | | | | |
| 분반 | 002 | 학번 | 12141540 | 이름 | 박영창 |
| 알고리즘 개요 | * Testcase의 개수 T를 입력 받는다. (T<=10000) * 전체 학생의 수 n, 비교 결과의 수 m을 입력한다. (5<n<=50, 0<m<=500) * Student[51][51] 배열의 모든 값을 1000으로 초기화한다. 1000으로 초기화하는 이유는 배열에 들어갈 값이 아무리 커도 1000보다 작기 때문이다. * Std1, std2라는 변수에 비교 결과를 입력 받고, student[std1][std2]에 1을 저장해 준다. * 그 다음, 입력된 비교 결과를 통해 알 수 있는 비교 결과 또한 배열에 채워줘야 한다, k, I, j를 이용해 3중 반복문을 사용한다. K는 몇 번째 학생에 대해 봐주는지를 나타내고, I, j는 배열의 값을 확인하기 위해 사용된다. 이를 이용해 student[i][j] > student[i][k] + student[k][j]이면, student[i][j]에 studeng[i][k] + student[k][j]를 저장한다. 예를 들어 1>2, 2>3이라는 조건이 있을 때, student[1][3]은 1000이 아니라면 1>3이라는 것을 나타낸다. 초기값으로 1000이 저장되어 있지만, student[1][3] ( 1000 ) > student[1][2] ( 1 ) + student[2][3] ( 1 )이 만족하므로 student[1][3]에는 2가 들어가게 된다. 모순되는 조건을 잡는 방법은 student[1][3]에 1000이 아닌 값이 들어 있다면, student[3][1]에는 1000이 아닌 값이 들어가 있어서는 안된다. 왜냐하면 1>3, 3>1을 동시에 만족시켜선 안 되기 때문이다. 이를 값의 변경과 함께 확인하면서 bool 변수 valid를 설정해 준다. * 모든 반복문을 마치고 난 뒤에, i번 학생보다 성적이 높은 학생의 수는 student[j][i] (가로)에 존재하는 1000이 아닌 값의 수와 같고, 성적이 낮은 학생 수는 student[i][j] (세로)에 존재하는 1000이 아닌 값의 수와 같다. 이런 식으로 better, worse 값을 구해준 뒤 출력한다. 만약 valid가 false라면 -1을 출력한다. * T를 입력 받는 것을 제외한 위 과정을 t-1번 반복한다. | | | | |
| 성능분석 | 학생의 수 : N  시간 복잡도 : O(N^3) | | | | |
| 실험결과 | 소요시간 : 1.14s  사용된 메모리 공간 : 131.072KB | | | | |
| 비고 |  | | | | |