|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2주차 - B 문제해결기법 보고서 | | | | | |
| 문제 | 식당 | | | | |
| 분반 | 002 | 학번 | 12141540 | 이름 | 박영창 |
| 알고리즘 개요 | * Testcase의 개수 T를 입력 받는다. (0<=T<=1000) * 최소 가격이 들어갈 변수인 result를 800001로 초기화한다. (최대 가격이 8000원, 최대 일수 100일이므로 나올 수 있는 최대 가격인 800000보다 큰 값으로 초기화) * 식당의 가격이 들어갈 2차원 배열 price[100][100]과 각 날짜의 최소 가격이 들어갈 restaurant[100][100]을 선언한다. * 식당 수 n과 날짜 수 m을 입력 받는다. (2<=n<=100, 1<=m<=100) * 입력 받은 n과 m만큼 식당과 날짜에 맞는 가격을 price 배열에 입력하고, 동시에 restaurant 배열의 같은 위치에 800001의 값을 넣어 준다. (2000<=가격<=8000) * restaurant 배열은 price[i][j]에 위치하는 식당에 갔을 때의 최소 가격을 의미한다. 따라서 restaurant의 첫 열은 price의 첫 열과 같을 때 최소이므로 반복문을 이용해 첫 열을 같게 만들어 준다. * 같은 방법으로 다음 열도 가장 최소값을 restaurant 배열에 넣어 주어야 하므로 앞 열의 restaurant 배열 내의 값 중 최소값에 해당 위치 인덱스의 price를 더한 값을 restaurant의 같은 인덱스에 넣어 준다. 이 때, 같은 식당을 두 번 연속으로 가면 안되니 이 조건을 넣기 위해서 총 3개의 for 반복문을 사용한다.   restaurant[j][k] = min(restaurant[j][k], restaurant[l][k-1] + price[j][k]); (l != j라는 조건)   * 같은 방법으로 반복해 restaurant 배열을 restaurant[n-1][m-1]까지 채워준 뒤, restaurant 배열의 마지막 열중 가장 작은 값을 result에 넣어 준다. 그 다음 result를 출력한다. * T를 입력 받는 것을 제외한 위 과정을 T-1번 반복한다. | | | | |
| 성능분석 | 식당 수 : N, 날짜 수 : M  시간 복잡도 : O(N^2\*M) | | | | |
| 실험결과 | 소요시간 : 0.95s  사용된 메모리 공간 : 262.144KB | | | | |
| 비고 |  | | | | |