|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 12주차 - B 문제해결기법 보고서 | | | | | |
| 문제 | 반지원정대 B | | | | |
| 분반 | 002 | 학번 | 12141540 | 이름 | 박영창 |
| 알고리즘 개요 | * Testcase의 개수 T를 입력 받는다. (T<=20) * 1에서부터의 최단경로의 길이를 나타내는 route 배열을 큰 수로 초기화하고 node를 확인했는지 여부를 나타내는 visit 배열과 최단경로를 표시하는 FRoute 배열을 초기화한다. * 마을의 수 n (5<=n<=1000)과 경로의 수 m(5<=m<=500000)을 입력 받는다. * a, b, t를 m번 입력 받아 a, b중 작은 수가 앞으로 오도록 node를 만들어 VRR 벡터에 추가한다. * 무향 그래프이므로 VRR벡터에 들어가 있는 모든 노드를 VR 2차원 벡터에 추가한다. VR[i][j]는 i에서 j로 이어지는 길을 의미한다. * 다익스트라 알고리즘을 사용해 route 배열을 갱신한다. Route 배열이 갱신됨에 따라서 FRoute 배열도 갱신해 최단경로의 길이와 최단경로를 구한다. * FRoute 배열에 있는 최단경로를 VRR 벡터 내에서 찾아 fastRoute bool 변수를 true로 바꿔준다 * fastR 변수에 route[n]을 저장한다. 이는 1부터 n까지의 최단경로의 길이이다. * A번 문제처럼 m번 반복문을 돌지만 최단경로 외의 선을 지우면 결국 최단경로 그대로의 결과가 나오므로 최단경로 내에 있는 길만 지운 값을 확인한다. 이 경우들을 확인하고 최대값을 maxR에 저장한다. * maxR이 초기값이라면 갈수 없는 경우가 발생하는 것이므로 -1을 출력하고 그렇지 않다면 maxR – fastR을 출력한다. * T를 입력 받는 것을 제외한 위 과정을 t-1번 반복한다. | | | | |
| 성능분석 | N : 마을의 수, M : 길의 수  시간 복잡도 : O(M \* N^2) | | | | |
| 실험결과 | 소요시간 : 0.3s  사용된 메모리 공간 : 131.072KB | | | | |
| 비고 | A번과 시간 복잡도가 같지만 해결된 이유는 최단경로에 있는 길을 지운 부분만 탐색하는 최적화 때문인 것 같습니다. | | | | |