2016115572 권현수 fts.pdf

-입력처리

구조체 Vertex를 선언하고, 안의 내용으로는 좌표, 다녀간 도시의 수, 다음 갈 수 있는 Vertex의 포인터, 이전 다녀간 Vertex가 있습니다. 100개의 Vertex의 배열에 총 도시 구조체를 저장합니다.

-함수 구조

모든 Vertex로부터 갈 수 있는 Vertex를 연결리스트로 만들고 하나씩 경로를 계산합니다. Vertex가 하나씩 추가될수록 Vertex 속성 중 distance부분이 증가하게 되고 최종 100,100도시에 도착했을 때 정해진 distance가 fuel보다 크지 않을 때 그 vertex를 result로 저장합니다. 그 result의 prev속성을 따라가면서 x와 y값을 저장하고 이것을 역방향으로 출력하면 지나간 경로가 나오게 됩니다.

-구조 및 함수설명

void CreateVertex(Vertex* arr, int x, int y) - Vertex를 만듭니다. vertex 내부 속성들을 초기화 해줍니다.

addEdge(Vertex* a, Vertex* b)- 모든 vertex는 0,0 vertex로부터 시작하므로 0,0과 다른 모든 vertex를 연결시켜주는 역할을 합니다.

insertNode(Vertex* arr, Vertex* IndexNode, int index, int fuel, Vertex* result_vertex) - Vertex를 집어넣습니다. 어떤 한 지점에서 다른 지점으로 갈 수 있는 모든 경우의 수를 계산하여 재귀적으로 수행됩니다. 수행중에 distance 가 fuel을 넘어서면 수행시간 단축을 위해 Vertex를 추가하지 않습니다.

simulate_overwrite_distance(Vertex* arr, int fuel) - 기름이 주어지고, 구조체 배열이 주어지면 이것으로 시뮬레이션을 합니다. addEdge를 수행하고 insertNode를 수행합니다. 마지막 Vertex부분은 result_vertex에 저장 후 result_vertex의 이전 경로를 나타내는 vertex의 x, y값을 차례대로 result_coor문자열변수에 끝에서부터 저장합니다. 그렇게 해야 역방향으로 출력이 되어 x값이 작은 값부터 출력이 되기 때문입니다. 300과 700, 1100, 1500일때의 시뮬레이션 총 4번을 합니다.

void QuickSort(Vertex* arr, int l, int r) , int Partition(Vertex* arr, int l, int r)

- 퀵 정렬입니다. 과제의 문제는 x축으로 직진만이 가능하기 때문에 x를 기준으로 정렬을 하면 Vertex별로 다음 갈 수 있는 경로를 지정하기가 훨씬 수월해집니다. 그렇기에 정렬알고리즘을 사용했습니다.

double dist(Vertex a, Vertex b) - Vertex a와 b의 x,y 속성에 접근하여 두 점 사이의 거리를 찾아냅니다. sqrt와 pow함수가 사용되었습니다.