방향성을 고려한 길 탐색 알고리즘

김일주, 빙영민, 이송근
전주 대학교

Street Search Algorithm That Consider Direction

Il-ju Kim, Kyu-hwa Lee, Song-keun Lee
Jeonju University

Abstract - 최단 경로란 유향(有向) 또는 무향(無向) 그래프에서 어떤 두 점 사이를 맺는 유향 또는 무향 경로 중 가장 짧은 것, 즉 가지의 길이 합을 최소로 하는 것을 구하는 문제. 그래프가 평면 접속인 경우에는 쌍대(雙對) 그래프의 최대 흐름을 구하는 문제와 등가이다. 본 논문에서는 최단 경로 문제를 풀기 위하여 Dijkstra의 장점은 살리고 단점을 보완하는 방향성을 가지는 Dijkstra 알고리즘을 제안하였다. 사례연구를 통하여 제안한 알고리즘은 출발점에서 도착점까지 최단 경로를 빠른 시간에 찾아가는 것을 보였다.

1. 서 론

현재까지의 최단경로 탐색의 알고리즘인 Dijkstra, DFS, BFS, A*알고리즘 등의 많은 종류의 알고리즘들이 나와 있으며 각각의 알고리즘들은 각기 다른 장점과 단점을 보유하고 있다.[1]

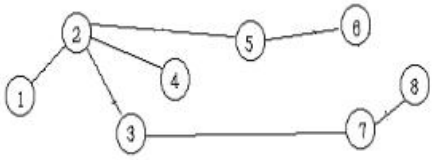
본 논문에서는 Dijkstra 알고리즘의 장점인 빠른 계산시간을 적용하였으며 오차범위를 줄여줌으로써 방향성을 가진 최단경로 탐색 프로그램을 구현하였고 최단경로 탐색을 위하여 가상 맵을 작성하고 가상 맵을 토대로 최단 경로를 찾아내는 프로그램을 작성하였다. 경로 프로그램은 MATLAB을 이용하였다.

2. 본 론

2.1 경로탐색

2.1.1 DFS 알고리즘

DFS(Depth First Search)란 깊이우선탐색이다. 시작 노드를 방문함으로서 시작된다. 방문한 정점에 인접하면서 아직 방문하지 않은 정점을 선택하여 그 정점을 시작점으로 하는 깊이 우선 탐색을 다시 시작한다. 이 정점들은 스택에 넣어 기억을 시키게 되고 결국에 가서 방문하지 않은 정점이 없는 정점에 도달하게 되면 스택에서 하나의 정점을 꺼내어 이 정점에서 계속 순회를 합니다. 이미 방문된 정점은 무시가 된다. 스택이 비거나 찾고자 하는 곳이 탐색이 되면 탐색이 종료가 된다. 그림 1은 탐색지도의 예를 보여주고 있다.



<그림 1> 탐색지도의 예

2.2 BFS 알고리즘

BFS(Breadth First Search)란 폭(너비)우선탐색이다. 루트의 탐색을 한다면 먼저 루트의 자식을 차례로 방문한다. 다음으로 루트 자식의 자식, 즉 루트에서 두 개의 간선을 거쳐서 도달할 수 있는 정점을 방문한다. 다음으로 루트로부터 세 개의 간선을 거쳐서 도달하는 정점들 순으로 루트에서의 거리 순으로 방문한다. BFS 알고리즘은 루트의 탐색을 한다면 먼저 루트의 자식을 차례로 방문한다. 다음으로 루트 자식의 자식, 즉 루트에서 두 개의 간선을 거쳐서 도달할 수 있는 정점을 방문한다. 다음으로 루트로부터 세 개의 간선을 거쳐서 도달하는 정점들 순으로 루트에서의 거리 순으로 방문한다.

2.2.1 A*알고리즘

A*알고리즘은 게임에서 가장 많이 사용되는 대표적인 알고리즘이다. A*알고리즘은 기본적으로 트리구조에서 많이 쓰이고 BFS의 방법을 쓰기 때문에 효율성도 갖추면서 코드도 깔끔히 만들 수 있는 장점이 있으며 시작 노드에서 목표 노드까지의 최단 경로를 효율적으로 구해준다. 하지만 맵이 매우 클 경우에 효율적인 검색을 하지 못하며, 맵의 장애물을 개산할 때에 움직이는 객체가 길을 막고 있다면 경로가 없다고 판단을 내리고 길 찾기를 종료한다.

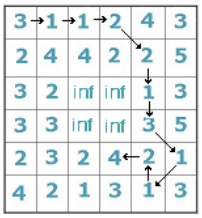
2.2.1 Dijkstra 알고리즘

Dijkstra 알고리즘은 최단거리를 구하는 방법으로 유명한 알고리즘이다. Dijkstra 알고리즘은 1959년에 만들어진 최소의 거리를 가지는 최단경로 탐색을 구하는 알고리즘 중에서 한가지 이며, 적용분야는 최소의 경로를 요구하는 철도 건설 및 통신네트워크 경로 설계 등에 주로 사용 한다. Dijkstra 알고리즘의 특징은 현 위치에서 갈 수 있는 주위의 노드만을 검색하여 그중에서 가장 작은 거리로 이동하면서 나아가는 방법이다. 이 Dijkstra 알고리즘은 계산시간이 짧다는 강한 장점을 가지고 있다. 작은 맵이나 갈 수 있는 노드의 숫자가 작은 경우에는 유리하다. 하지만 맵의 크기가 커지게 되면 최적의 최단경로를 찾을 수 없을 뿐만 아니라 무한 루프에도 빠지기 쉽다는 단점을 가지고 있다.

3	1	1	2	4	3
2	4	4	2	2	5
3	2	inf	inf	1	3
3	3	inf	inf	3	5
2	3	2	4	2	1
4	2	1	3	1	3

<그림 2> Dijkstra프로그램의 가상 맵

그림 2는 Dijkstra의 가상 맵이고 그림 3은 MATLAB에서 Dijkstra 프로그램의 간단한 예이다. 시작점을 3이라하고 도착점을 4이라고 했을 때 현 위치에서 작은 거리로 검색해 나가면서 경로를 탐색해 나가는 것을 볼 수 있다.



<그림 3> Dijkstra프로그램 결과

3. 실험결과

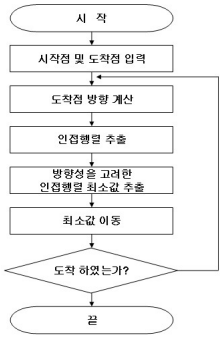
3.1 가상맵

맵 제작과 최단경로 탐색 프로그램은 MATLAB으로 가상의 맵을 작성하였으며 가상 맵은 행렬을 이용하여 각 노드의 거리를 정해주어

최단경로 탐색에 문제없이 작동하도록 작성하였다. 최단경로 탐색은 작성된 가상맵을 이용하여 출발 지점과 목적지를 입력해주면 프로그램이 각 각의 노드를 검색하여 최단경로를 찾아가도록 작성하였다.

3.2 순서도

그림 4는 최단경로 탐색의 순서도 이다. 최단경로 탐색의 과정은 우선 시작점과 도착점을 입력하게 된다. 그 후 입력받은 데이터 값을 이용하여 시작점에서의 도착점 방향을 계산한다. 그 후 현 위치에서 인접한 위치의 노드거리 중 방향성을 고려하여 최단경로에 적합한 노드를 선택한다. 선택한 노드거리로 이동한 후 그 노드거리가 도착점이 되면 최단경로 탐색이 끝나게 되고 아직 도착하지 않았으면 위의 과정을 반복하게 된다.

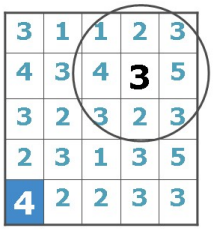


<그림 4> 순서도

3.3 개선된 최단경로 알고리즘 방향성 결정

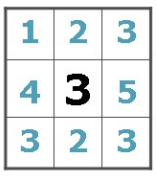
기존의 Dijkstra 알고리즘의 경로 오차가 많다는 점을 보안하기 위하여 본 논문에서는 목적지까지의 방향성을 고려하여 최단경로의 노드를 결정 하도록 하였다. 개선된 알고리즘의 방향성을 고려하여 최단경로 결정방법은 기존 알고리즘의 특성인 인접노드에서 최소 거리로 이동하는 데는 변함이 없다. 하지만 그 최소 거리를 결정하는데 있어서 목적지까지의 방향성을 미리 계산하여 인접행렬이 아무리 최소 거리라도 목적지까지 다시 최소 거리를 구하도록 하였다. 하단에 나오는 그림 5, 6, 7은 개선된 알고리즘의 계산 과정을 보여주고 있다.

그림 5은 개선된 알고리즘의 계산과정 중 첫 번째 단계로서 출발점과 도착점을 입력했을 때 출발점에서의 인접노드를 표시한 그림이다. 출발점은 숫자 3이고 도착점은 칸이 칠해진 4이다.

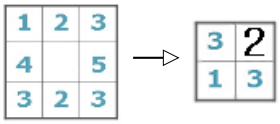


<그림 5> 개선된 알고리즘 1단계

출발점에서의 위, 아래, 양옆의 행렬을 추출한다. 그림 6은 출발점에서의 인접노드를 출력한 행렬이다. 기존 알고리즘이라면 이 행렬에서 바로 1이라는 노드를 선택 하였을 것이다. 하지만 개선된 알고리즘을 사용하게 되면 하단의 그림 7과 같이 방향성을 계산하여 값을 산출한 후 최소값을 결정한다.

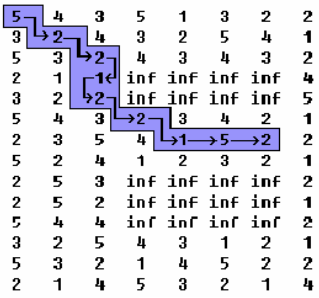


<그림 6> 개선된 알고리즘 2단계



<그림 7> 개선된 알고리즘 3단계

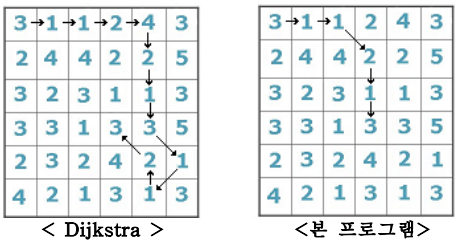
위와 같은 계산을 반복해 나감으로써 방향성을 고려하여 최단경로를 탐색할 수 있게 된다. 개선된 최단경로 알고리즘 프로그램의 동작방법은 아래에서 보여주고 있다. 가상 맵에서 시작점인 행과 열을 작성하고, 도착점인 행과 열을 작성하면 그림 8과 같이 시작점에서 도착점까지의 행과 열의 진행 방행이 나타난다. 표 1은 Dijkstra의 최단거리 탐색 경과 시간은 0.9739초가 걸리고 본 프로그램에서의 최단거리 탐색 경과 시간은 0.1092초가 걸리게 된다. 위의 Dijkstra의 계산 속도보다 0.8647초 빠른 것을 볼 수 있다. MATLAB에서 속도측정 프로그램은 tic과 toc을 사용하였다.[2]



<그림 8> 경로 탐색 결과

<표 1> 최단경로 계산 시간

프로그램 종류	경과 시간
Dijkstra	0.9739초
본 프로그램	0.1092초



< Dijkstra > <본 프로그램>

<그림 9> 두 프로그램의 비교

4. 결 론

본 논문에서는 현재 나와 있는 최단경로 탐색 알고리즘 중에서 계산 속도는 빠르지만 최적의 경로를 찾기 힘들다는 단점을 가진 Dijkstra 알고리즘의 계산시간이 빠르다는 장점을 이용하여 최단경로 결정시 방향성을 가진 최단경로의 검색을 하게 하여 새로이 개선된 알고리즘을 작성 하였다. 개선된 알고리즘은 기존의 Dijkstra 알고리즘에서 다음노드를 갈 때 최소거리를 따라가는 방식을 개선하여 방향성을 주면서 최소거리를 구하도록 함으로써 최단경로를 검색할 경우 계산시간이 빠르다는 장점을 보존하면서 오차범위를 많이 줄였다.

방향성을 가진 최단경로 프로그램이 더 정확하고 빠른 경로 탐색하는 것을 볼 수 있다. 표 1은 MALTA 에서 Dijkstra와 방향성을 가진 최단거리 프로그램인 두 개의 프로그램을 실행 하여 속도를 측정한 값이고 방향성을 가진 최단거리 프로그램이 더욱 빠른 계산 속도를 볼 수 있다.

[참 고 문 헌]

[1] 문병로. “쉽게 배우는 알고리즘” 2007
[2] 김상운, “Matlab으로 배우는 패턴인식 및 학습” 2003