2012학년도 대구과학고등학교
2차 사이언키피아 연구보고서

연구 제목

탐구과목	정보과학

2012. 11. 15.

학년	반	번호	학생이름
2	1	6	박형기
2	3	12	허성연
2	3	15	황석하

1. 연구동기 및 목적

가. 연구동기

1) 근래에 IT 기기는 임베디드 시스템을 이용하는 추세이다. 대표적인 예로 로봇청소기가 있다. 로봇청소기가 자동으로 청소하기 위해서는 경로를 탐색하는 알고리즘이 필요하다. 그런데 지금의 로봇청소기에는 충돌, 경로 이탈, 비효율적인 경로선택 등의 문제가 존재한다. 특히 비효율적인 경로선택 문제는 청소시간의 증가, 전력낭비

등의 원인이 된다. 본 연구에서는 이러한 문제를 일반화하여 다양한 로봇에서 쓰일 수 있는 효율적인 경로선택 알고리즘에 대해 연구하였다.

나. 연구목적

1) 로봇에 쓰일 수 있는 효율적인 경로선택 알고리즘을 알아보고, 이를 컴퓨터 시뮬레이션을 통해 구현한다.

2) 구현한 경로선택 알고리즘에서, 평가함수를 바꾸어 실험하여 각각의 함수들의 성능을 비교하여 어떤 평가함수를 사용하는 것이 최적의 경로를 잘 구할 수 있는 지 알아본다.

2. 이론적 배경

가. 깊이 우선 탐색법(Depth First Search)

1) 깊이 우선 탐색(depth first search)은 탐색방법의 하나로 탐색트리의 최근에 첨가된 노드를 선택하고, 이 노드에 적용 가능한 방법 중 하나를 적용하여 그래프에 다음 수준(level)의 한 개의 자식노드를 첨가하며, 첨가된 자식 노드가 목표노드일 때까지 앞의 자식 노드의 첨가 과정을 반복해 가는 방식이다. 이러한 방식으로 원하는 노드를 찾게 되면, 검색을 끝내고 값을 반환하는 방식의 탐색법이다.

2)장점 및 단점

A)장점

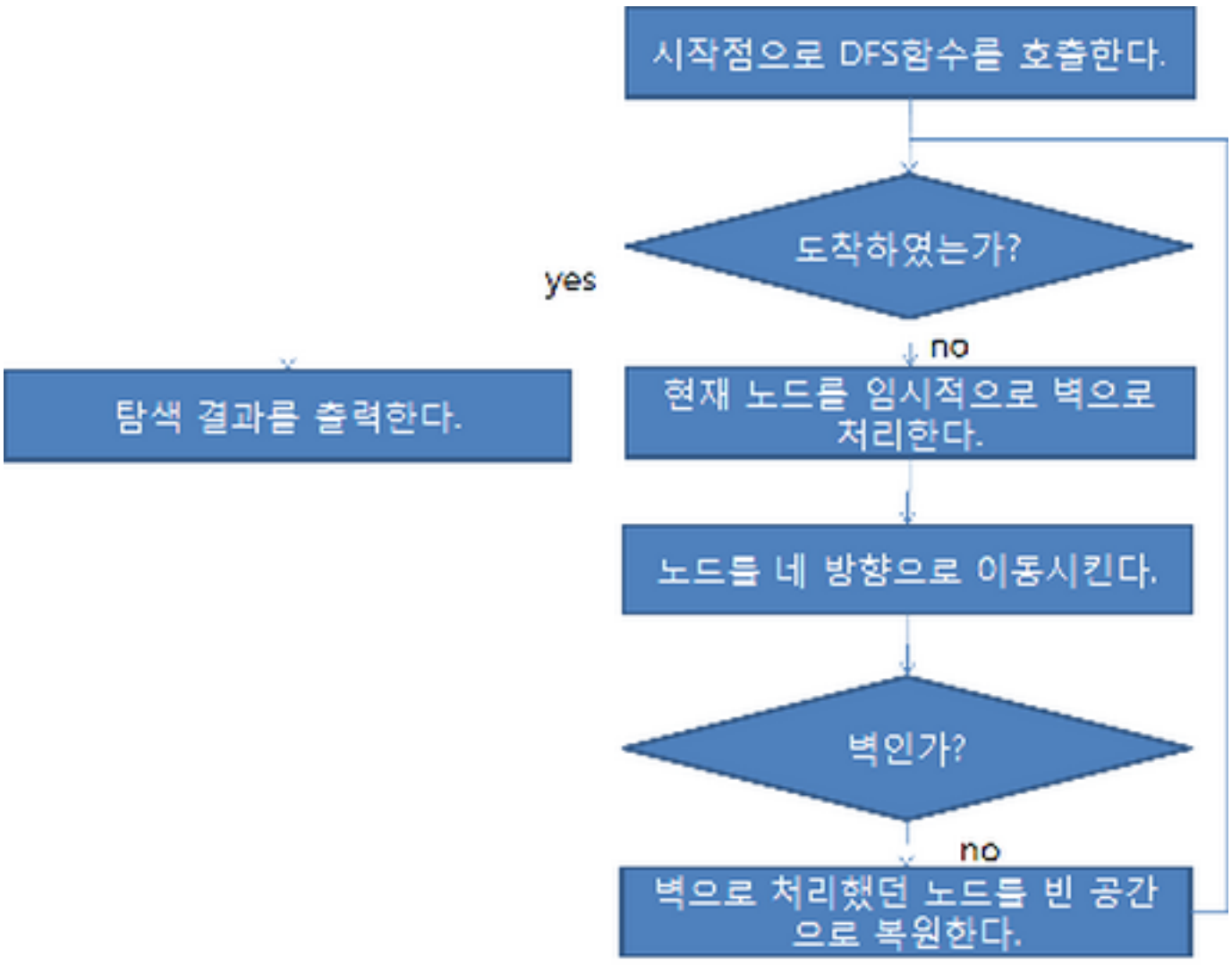
a) 단지 현 경로상의 노드들만을 기억하면 되므로 저장공간의 수요가 비교적 적다.

b) 목표노드가 깊은 단계에 있을 경우 해를 빨리 구할 수 있다.

B) 단점

해가 없는 경로에 깊이 빠질 가능성이 있다. 따라서 실제의 경우 미리 지정한 임의의 깊이까지만 탐색하고 목표노드를 발견하지 못하면 다음의 경로를 따라 탐색하는 방법이 유용할 수 있다.

순서도로 나타내면 다음과 같다.



나. 너비 우선 탐색(Breadth First Search)

1) 너비 우선 탐색(Breadth first search, BFS)은 탐색방법의 하나로 시작 정점을 방문한 후 시작 정점에 인접한 모든 정점들을 우선 방문하는 방법이다. 더 이상 방문하지 않은 정점이 없을 때까지 방문하지 않은 모든 정점들에 대해서도 넓이 우선 검색을 적용한다. 인접한 정점들을 검색하는 도중 원하는 정점이 발견되면 모든 검색을 멈추고 해를 반환한다.

2) 장점 및 단점

A) 장점

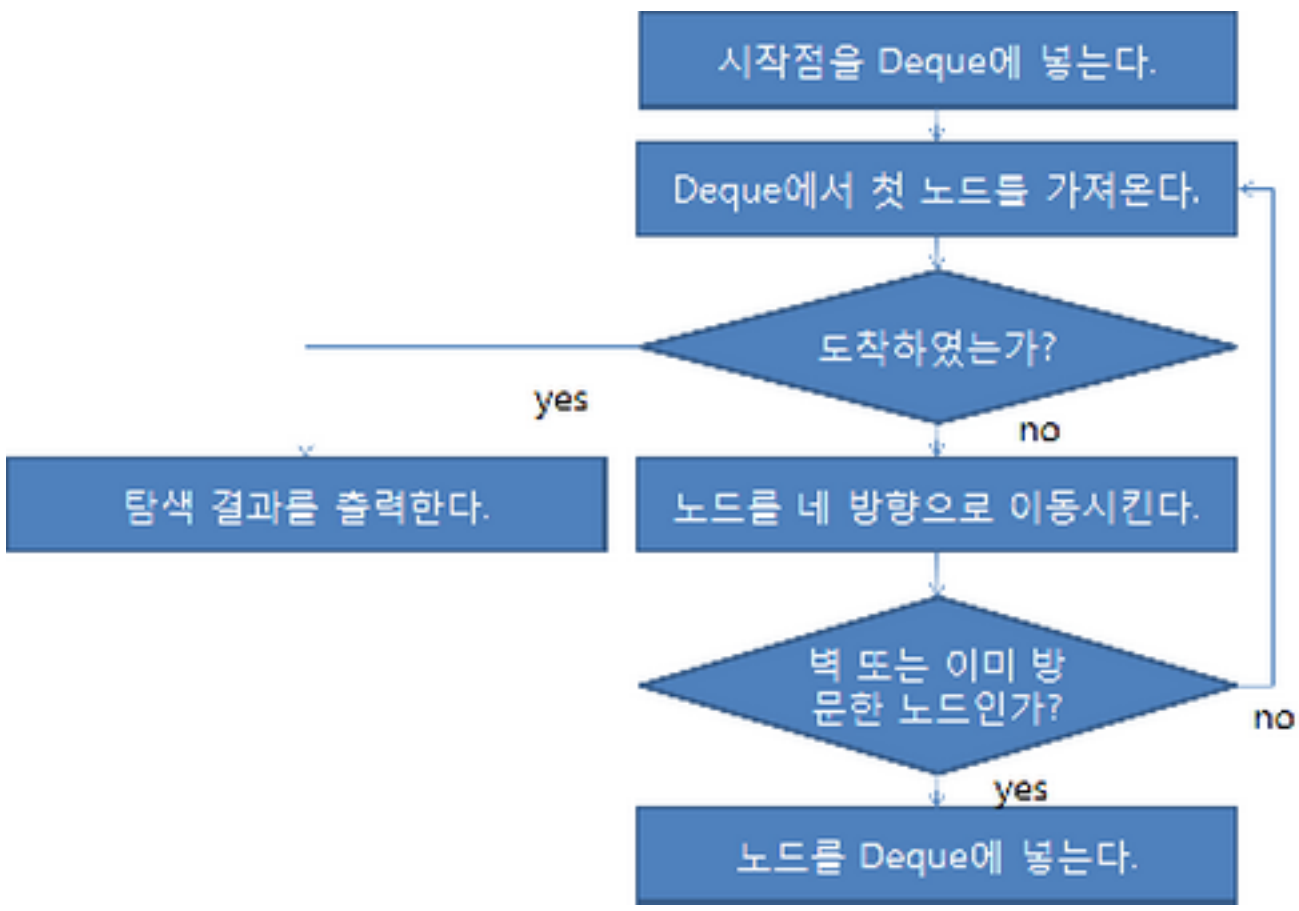
a) 출발노드에서 목표노드까지의 최단 길이 경로를 보장한다.

B) 단점

a) 해가 존재하지 않는다면 유한 그래프(finite graph)의 경우에는 모든 그래프를 탐색한 후에 실패로 끝난다.

b) 무한 그래프(infinite graph)의 경우에는 결코 해를 찾지도 못하고, 끝내지도 못한다.

순서도로 나타내면 다음과 같다.



다. 우선순위 큐(Priority Queue)

우선순위 큐는 우선순위의 개념을 큐에 도입한 자료 구조이다. 보통의 큐는 선입선출(FIFO)의 원칙에 의하여 먼저 들어온 데이터가 먼저 나가게 된다. 그러나 우선순위 큐(Priority Queue) 에서는 데이터들이 우선순위를 가지고 있고 우선순위가 높은 데이터가 먼저 나오게 된다. 우선순위 큐를 스택이나 큐와 비교해보면, 스택에 들어있는 데이터들은 우선순위가 없다. 단지 먼저 들어간 데이터가 가장 늦게 나오게 되고, 큐는 가장 먼저 들어간 데이터가 가장 먼저 나오게 된다. 반면에 우선순위 큐 에서는 들어있는 데이터 중 우선순위가 가장 높은 데이터가 먼저 나오게 된다.

3. 연구 내용

가. A* 알고리즘

1) A* 알고리즘이란 너비 우선 탐색에 깊이 우선 탐색을 조합하여 휴리스틱한 기법을 사용한 알고리즘이다. A* 알고리즘이 다른 그래프 탐색 알고리즘과 다른 점은 목표에 얼마나 근접한 것인지를 평가하는데 휴리스틱 함수를 사용한다는 것이다.

2) 알고리즘 순서

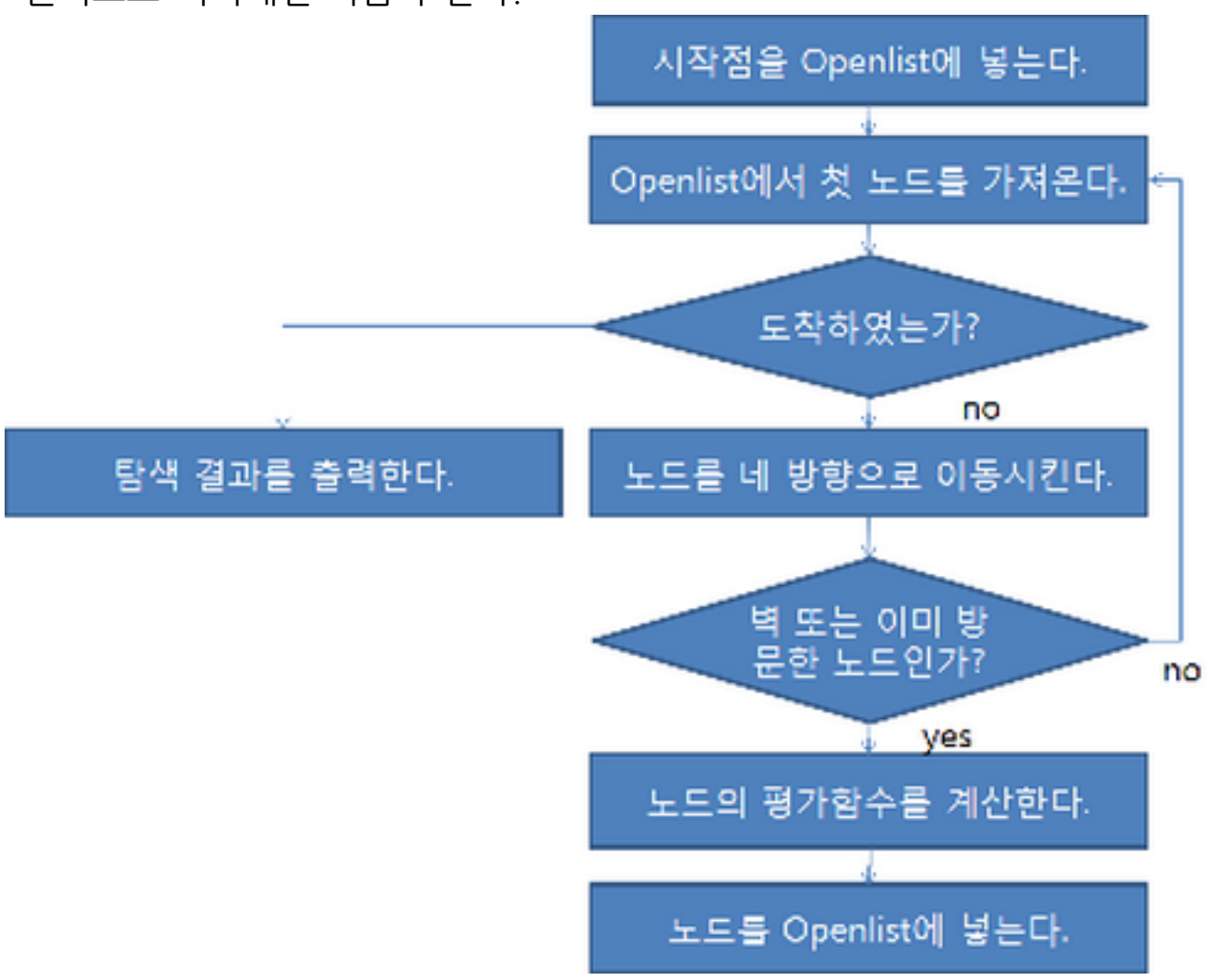
(1) 먼저 Openlist라는 것을 정의한다. Openlist는 방문할 예정인 정점들을 저장해둔 우선순위 큐 배열이다. 우선순위 큐를 사용함으로써 Openlist의 가장 첫 번째 Index는 최솟값을 가지는 노드를 저장하게 된다. 그리고 시작지점을 Openlist에 넣는다.

(2) Openlist에 있는 노드 중 1개를 빼서 4 방향 주변노드를 탐색한다. 만약 인접한 노드가 방문을 하지 않았던 노드였다면 그 노드에서의 평가함수 F(x,y)를 계산한다. F(x,y)는 다음과 같이 정의된다.

$$F(x,y) = G(x,y) + H(x,y)$$

G : 시작노드에서 현재노드까지의 경로의 값
H : 현재노드N에서 목표노드까지의 최단경로의 값이다. 이것은 휴리스틱한 요소

로써 이 함수를 어떻게 변환시키는지에 따라 A* 알고리즘의 성능에 큰 영향을 미친다.
따라서 평가함수 F(x,y)라는 것은 시작노드에서 목표노드까지 현재노드를 통해 갈 수 있는 모든 가능한 경로 중 최단경로의 값을 가지게 된다.
(3) (2)에서 Openlist에서 뺀 노드를 Closelist에 삽입하여 더 이상 방문하지 않게 만든다.
(4) (2)에서 탐색한 노드들을 Openlist에 삽입한다. Openlist는 가장 평가함수 값에 따라 오름차순으로 정렬하기 때문에 가장 낮은 평가함수 값을 가진 노드를 찾을 수 있다.
(5) 만약 Openlist에서 꺼내온 노드가 도착점이라면, 탐색을 끝내고 경로를 반환한다. 만약 Openlist가 비었다면 탐색을 실패한 것이므로 이 경우에도 탐색을 끝내고 실패한 경로를 반환한다.
(6) (2)에서 (5)까지 계속 반복한다.
순서도로 나타내면 다음과 같다.



나. A* 알고리즘 성능 향상

1) 본 연구에서는 F(x,y)를 조율하는 것 외에 A* 알고리즘의 성능을 향상시킬 수 있는 방법에 대해 연구하였다. 그 방법으로는 출발점과 도착점에서 동시에 경로 탐색을 하는 방법, 맵의 특성에 따라 함수를 달리하는 방법, 연산속도에서 우수한 성능을 보이는 함수를 이용하여 실시간으로 새로운 경로를 탐색하는 것으로 점점 더 최적화된 길을 찾아내는 방법 등이 있을 수 있다. 본 연구에서는 실시간으로 경로를 탐색하는 방법에 대해 연구하였다.

다. 평가함수 F(x,y)

1) 평가함수 F(x,y)는 휴리스틱한 함수이다. F(x,y)에 따라 A*의 성능이 결정되며 맵의 특성에 따라 적합한 F(x,y)가 다를 수 있다. 앞서 설명하였듯이 F(x,y) = G(x,y) + H(x,y)가 되는데, G(x,y)의 경우에는 지금까지 지나온 거리의 값이므로 어떤 경로

를 따라 왔다면 항상 일정한 값이고, 조정할 수 있는 값은 H(x,y)이다. H(x,y)를 계산하는 가장 간단한 방법으로는 먼저 도착점까지의 거리를 생각할 수 있다. 기본적으로 2차원 공간에서 두 점의 거리는 sqrt {(x-x _{0}) ^{2} +(y-y _{0}) ^{2}}가 된다. 이동 방향이 네 방향일 경우 (x-x _{0})+(y-y _{0})가 더 현실적인 값일 수 있다. H(x,y)의 값이 클수록 G(x,y)의 비중이 줄어들어 도착점에 도달하려는 성향이 더 커지고 값이 작을수록 BFS와 가까워진다. 그러므로 H(x,y) 값의 크기 정도를 일반화하면 ((x-x _{0}) ^{n} +(y-y _{0}) ^{n}) ^{m}을 생각할 수 있다. 이외에 교점 개수를 이용한 방법, 교점들에 외접하는 사각형을 이용하는 방법 등이 있을 수 있다.

라. 연산시간과 경로선택 효율의 관계

1) 평가함수는 일반적으로 연산시간이 길수록 짧은 경로를 선택하고 연산 시간이 짧을수록 긴 경로를 선택하는 성향이 있다. 본 연구에서 추구하는 알고리즘은 짧은 시간 내에 최대한 짧은 경로를 찾는 것으로 연산시간과 경로선택 효율이 모두 어느 수준 이상으로 우수한 성능을 보장하는 평가함수를 찾아낼 필요가 있다. 고로 알고리즘의 성능을 평가할 수 있는 지표에 대해 연구하였다.

4. 연구 방법

가. A* 알고리즘의 구현

1) visual studio 2008 을 이용하여, C언어를 이용한 개발환경을 설정한다.

2) A* 알고리즘 중 길을 탐색하는 함수를 구현한다.

3) 탐색을 위한 맵을 만드는 함수를 구현한다.

4) 맵을 이용해 탐색하여 경로를 알아내는 함수를 구현한다.

5) 만들어진 프로그램의 성능을 분석하는 프로그램을 구현한다.

6) 가시화된 출력이 필요하므로, 객체의 이동상황이 실시간으로 화면상에서 출력되도록 한다.

나. A* 알고리즘의 성능 향상

1) 일정 시간 또는 일정 거리마다 새로운 경로 탐색 결과를 준다.

2) 이동거리의 단축을 확인한다.

3) 여러 가지 맵에 대해 실험을 반복한다.

다. 평가함수 F(x,y)

1) 평가함수 F(x,y) = ((x-x _{0}) ^{n} +(y-y _{0}) ^{n}) ^{m} 꼴로 놓는다.

2) n과 m 의 값을 변화시키면서 경로탐색에 필요한 시간과, 얼마만큼의 이동거리를 거쳐 도착점에 도달했는지 기록한다.

3) n, m에 따른 경로 탐색 시간 그래프와, n, m에 따른 이동거리 그래프를 그려 어느 n과 m 값에서 최소시간과 최소 이동거리를 가지는지 구한다.

4) 여러 가지 맵에 대해 실험을 반복한다.

라. 연산시간과 경로선택 효율의 관계

1) 새로운 함수 K(t,l)를 K(t,l)=At+Bl 로 정의한다.

2) 연산시간과 경로 중 어떤 것에 가중을 둘 건지에 따라 A와 B 값을 결정한다.

3) 결정된 K(t,l) 함수에 (다)에서 나온 결과를 대입하고, n, m에 대한 K(t,l) 그래프를 그린다.

4) 어떤 n, m 값에서 K(t,l) 값이 최소가 되는지 구한다.

5) 여러 가지 맵에 대해 실험을 반복한다.

5. 결과 및 활용방안

가. A*를 이용한 경로 탐색 결과

1) A* 알고리즘을 이용한 경로 탐색이 가능하였고, (이미지 첨부) .....

나. A* 알고리즘의 통계분석 결과

1) 시간

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
1	1334	1334	1334	1334	1334	1334	1334	1334	1334	1350	1566
2	1334	1334	1334	1334	1334	1526	1572	1572	1572	1572	1572
3	1334	1334	1334	1446	1568	1572	1572	1572	1572	1572	1572
4	1334	1334	1432	1562	1568	1572	1572	1572	1572	1572	1572
5	1334	1334	1560	1568	1568	1572	1572	1572	1572	1572	1572
6	1334	1436	1574	1582	1582	1586	1586	1586	1586	1586	1586

2) 거리

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
1	124	125	109	110	124	110	109	125	124	172	109
2	109	125	109	125	125	94	93	94	93	94	78
3	109	125	124	110	93	94	93	94	109	94	93
4	125	125	109	109	94	94	109	93	110	93	94
5	125	140	94	109	93	94	109	94	109	94	109
6	125	124	110	93	109	94	109	94	109	94	109

3) 우리가 만든 지표

(이 때, 지표를 거리 + 10*시간 으로 두었다.)

2574	2584	2424	2434	2574	2434	2424	2584	2574	3070	2656
2424	2584	2424	2584	2584	2466	2502	2512	2502	2512	2352
2424	2584	2574	2546	2498	2512	2502	2512	2662	2512	2502
2584	2584	2522	2652	2508	2512	2662	2502	2672	2502	2512
2584	2734	2500	2658	2498	2512	2662	2512	2662	2512	2662
2584	2676	2674	2512	2672	2526	2676	2526	2676	2526	2676

위 결과, 가장 작은 지표인 n=1, m=0.4일때가 최소임을 알 수 있었다.

6. 반성 및 전망

1) 우리는 처음부터 끝까지의 완벽한 계획을 세워 연구를 진행하지 않았고, 어느 정도의 계획만 세워 연구를 진행했다. 그렇기 때문에 연구의 방향이 중간에 바뀌어 효율적이지 못한 부분이 있었다. 그 부분이 아쉽다.

2) A*알고리즘을 변형하면 경로탐색 이외의 인공지능에서도 충분히 응용될 수 있을 것 같다. 빠른 연산을 위한 기법으로서 다양한 방면에 활용될 수 있을 것이다.