인공지능 과제 보고서

2012003909 김승현

과제목표

이번 과제의 목표는 수업에서 배운 여러가지 경로를 찾는 알고리즘을 이용하여 각 층마다 최선의 알고리즘과 그를 구현하는 것입니다. 수업에서 배운 경로를 찾는 알고리즘은 다음과 같습니다

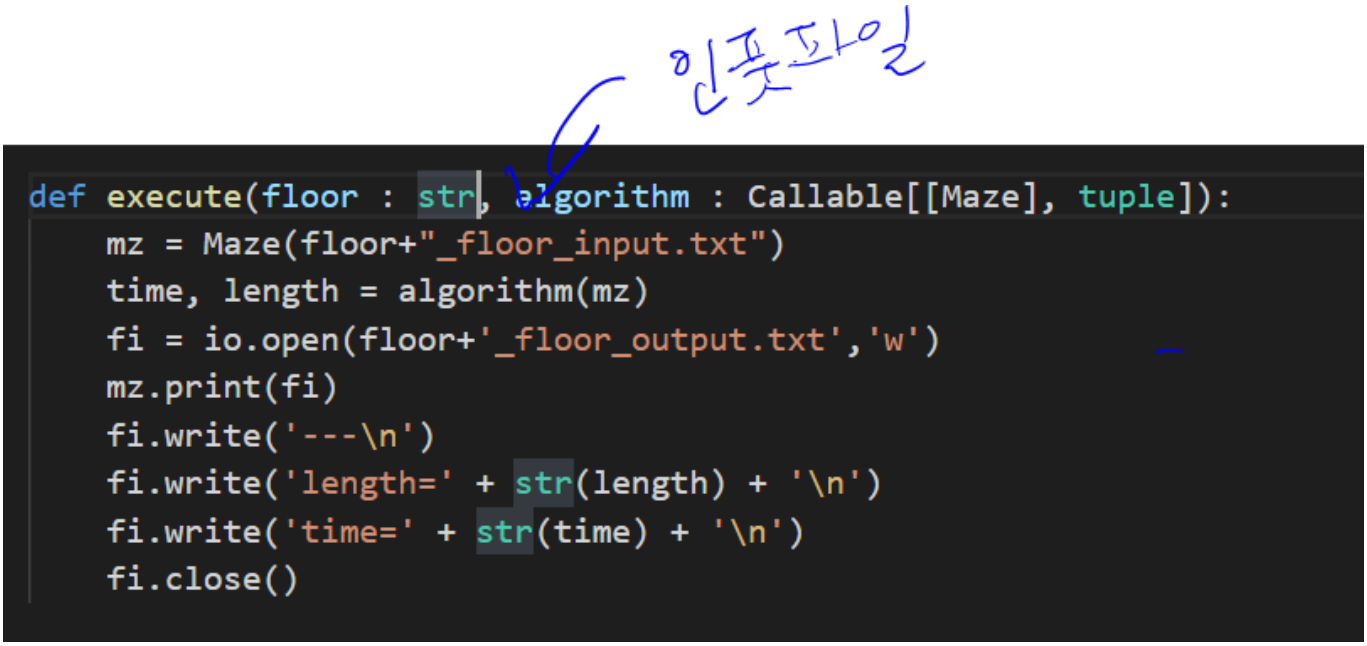
- Depth First Search (DFS)
- Breath First Search (BFS)
- Iterative Deepening Search (IDS)
- Greedy Best First Search
- A* Search
- Local Search

들이 있습니다. 저는 여기서 **Optimality**가 확실하게 보장되지 않는 **Depth First Search**, **Greedy Best First Search**와 **Local Search**는 제외하였습니다.

소스 코드 설명

소스 코드에는 Maze 클래스와 필요한 메서드들이 선언 되어 있습니다.

코드의 재사용성과 추후의 수정을 고려하여 모든 층의 로직은 execute 함수로 동작하도록 설계하였습니다.



execute 함수는 첫번째 인자에 맞춰 input과 을 읽어 Maze 인스턴스를 만든 후,두번째 인자의 함수를 실행시켜 결과를 얻어 output 파일에 출력하는 함수 입니다. Maze 클래스에는 A*, BFS, IDS 알고리즘이 구현되어 있습니다.

다음은 Maze 클래스 변수별 설명입니다.

Variable	type	Description
start	tuple	미로의 시작지점 위치를 나타냅니다
key	tuple	열쇠가 있는 지점을 나타냅니다
map	tuple	미로 배열입니다.
height	int	미로의 높이를 나타냅니다
width	int	미로의 너비를 나타냅니다

다음은 Maze 클래스 메서드별 설명입니다.

Method	Description
copy	미로 인스턴스를 복제하여 반환합니다
checkPos	지정좌표 y,x 가 막힌 길인지 체크합니다
move	지정한 좌표 y,x에서 지정 방향으로 움직일 수 있는지 여부를 체크합니다
createPathTrack	경로를 역추적하기 위한 배열을 생성합니다
applyPath	역추적하기 위해 기록한 배열을 이용해 경로를 추적합니다
dfs_sub	step만큼 DFS를 사용해 경로를 탐색합니다
ids_sub	IDS를 이용해 경로를 탐색합니다
ids	ids_sub를 이용해 키를 찾고 도착지점까지 경로를 탐색합니다
astar_sub	a* 알고리즘으로 경로를 탐색합니다

Method	Description
astar	a* 알고리즘으로 키를 찾고 도착지점까지 경로를 탐색합니다 astar_sub를 이용
bfs	BFS 알고리즘으로 키를 찾고 도착지점까지 경로를 탐색합니다 astar_sub의 heuristic 함수를 0으로 지정해 사용합니다
print	미로를 출력합니다

1층

- 사용한 알고리즘 : **A* Search**
- 탐색한 노드 개수 (time) : **6611**
- 최단 경로의 길이 (length) : **3850**
- 최단 경로 여부 : **True**

Optimality를 보장하지 않는 알고리즘들을 제외하면 Iterative Deepening Search, Breath First Search, A* Search가 있습니다.

1층은 101*101으로써 상대적으로 갈 경로가 많은 미로라서 일반적으로 탐색이 빠른 A* 알고리즘이 다른 두 알고리즘에 비해 빠를 것이라고 생각 했습니다.

측정한 결과 역시 A*가 제일 빠르게 나왔습니다 아래는 3개의 알고리즘을 비교한 사진입니다

```
===first floor===
A*; time=6611 length=3850
BFS; time=6712 length=3850
IDS; Recursion Overflow
-------------
```

2층

- 사용한 알고리즘 : **A* Search**
- 탐색한 노드 개수 (time) : **1618**
- 최단 경로의 길이 (length) : **758**
- 최단 경로 여부 : **True**

2층은 51x51으로써 1층 보다 작은 미로이지만 여전히 A* 알고리즘이 다른 두 알고리즘에 비해 빠를 것이라 생각했습니다.

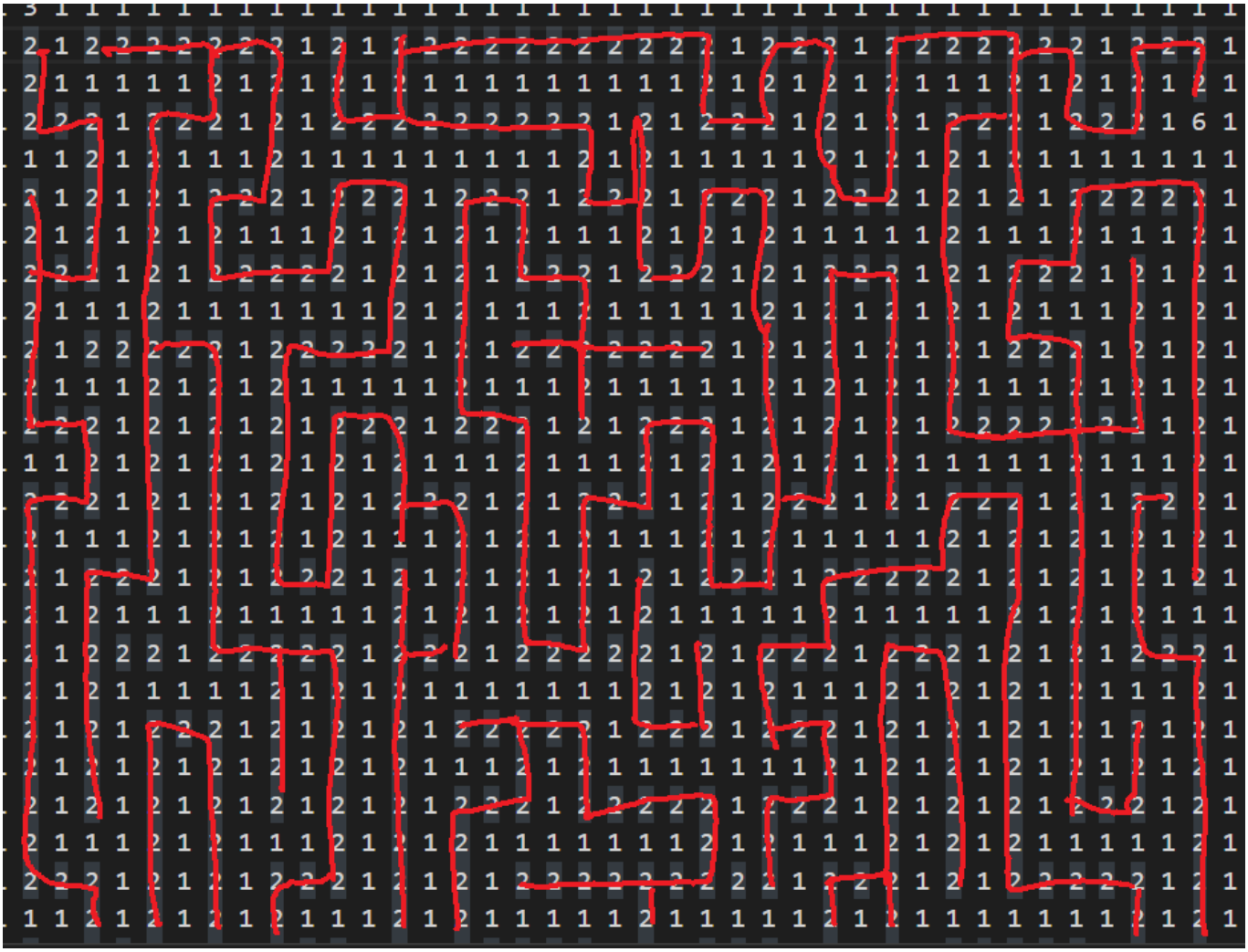
측정한 결과 역시 A*가 제일 빠르게 나왔습니다.

```
===second floor===
A*; time=1618 length=758
BFS; time=1680 length=758
IDS; time=308413 length=758
-------------
```

3층

- 사용한 알고리즘 : **A* Search**
- 탐색한 노드 개수 (time) : **832**
- 최단 경로의 길이 (length) : **554**
- 최단 경로 여부 : **True**

3층은 41x41 으로써 미로의 열린 길은 다음과 같습니다



길을 살펴보면 다른 경로로 가는 갈림길이 많은 것을 알 수 있습니다. 이런 경우 IDS나 BFS로 탐색 했을 때는 더 많은 시간이 소요될 것이라 생각했습니다.

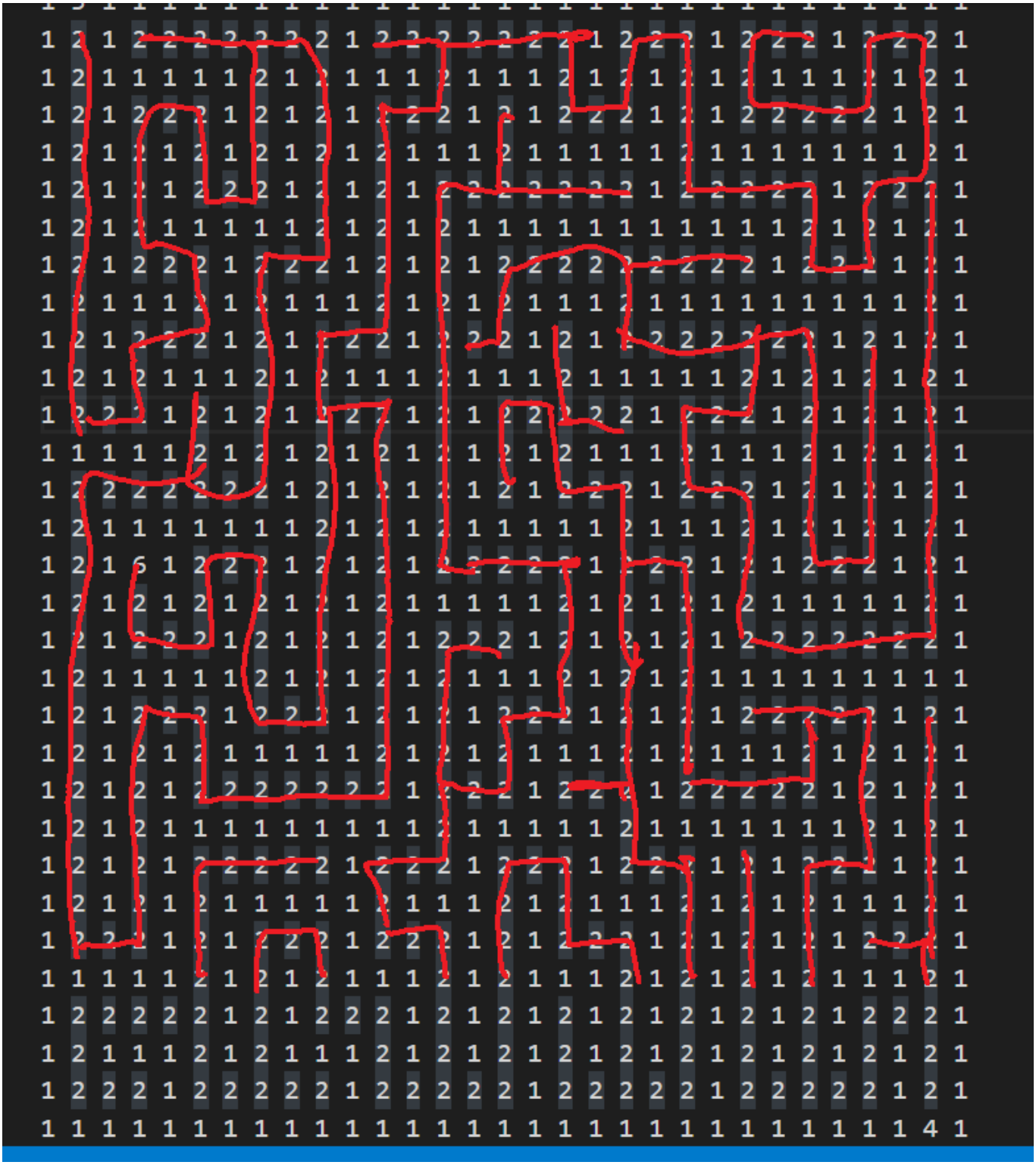
측정한 결과 역시 IDS나 BFS가 상대적으로 느리다는 것을 알 수 있습니다

```
===third floor===
A*; time=832 length=554
BFS; time=999 length=554
IDS; time=144584 length=554
-------------
```

4층

- 사용한 알고리즘 : **A* Search**
- 탐색한 노드 개수 (time) : **566**
- 최단 경로의 길이 (length) : **334**
- 최단 경로 여부 : **True**

4층은 31x31 으로써 미로의 열린 길은 다음과 같습니다



길을 살펴보면 다른 경로로 가는 갈림길이 그렇게 많지 않은 것을 알 수 있습니다. 이런 경우 BFS로 탐색했을 때 A*와 별 차이가 없거나 더 빠를 수 있다고 생각했습니다.

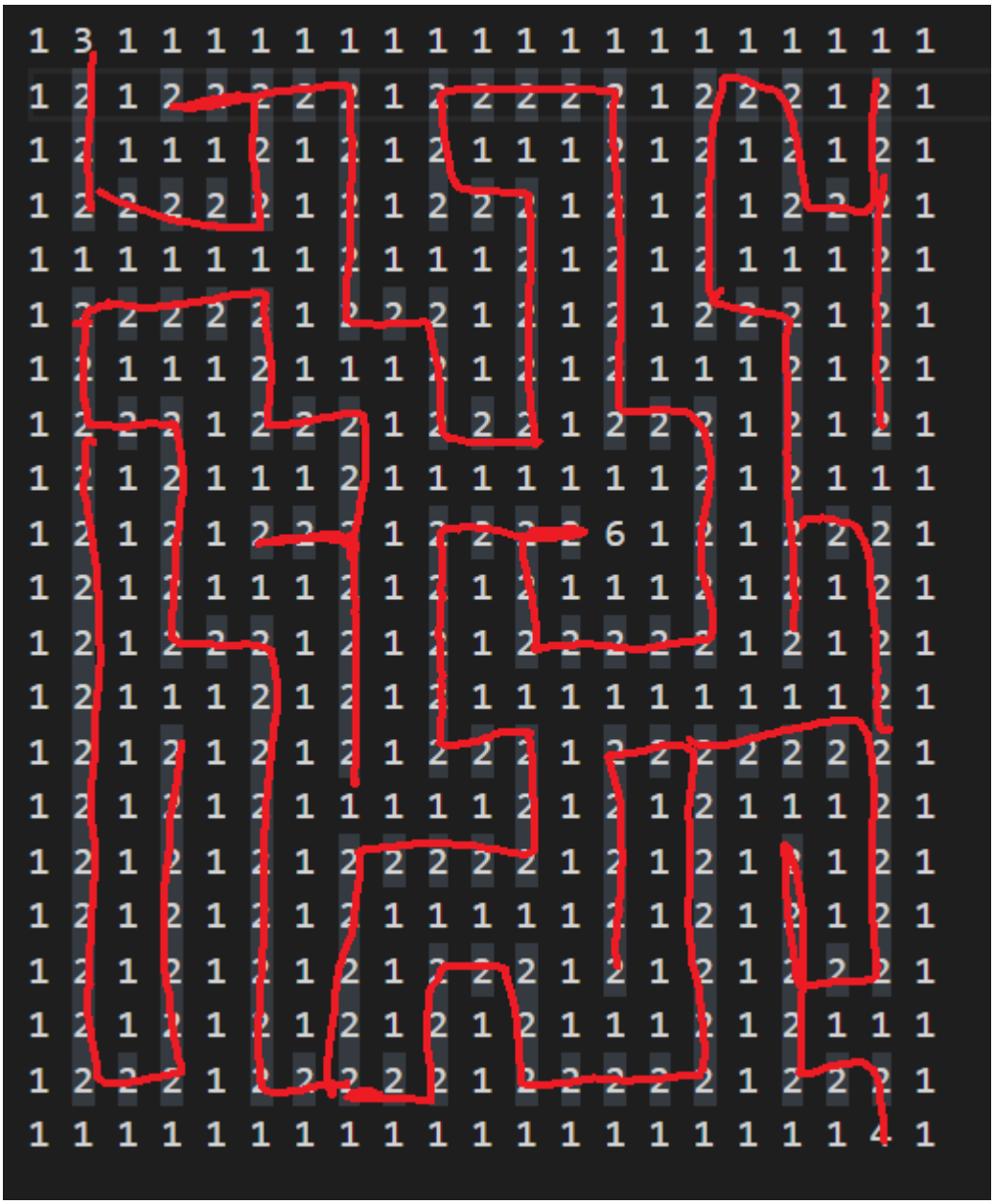
측정한 결과 제 예상과는 다르게 비슷비슷 하지만 아직 A* 알고리즘에 비해 느리다는 것을 알 수 있었습니다

```
===fourth floor===
A*; time=566 length=334
BFS; time=593 length=334
IDS; time=55452 length=334
-------------
```

5층

- 사용한 알고리즘 : **A* Search**
- 탐색한 노드 개수 (time) : **160**
- 최단 경로의 길이 (length) : **106**
- 최단 경로 여부 : **True**

5층은 21x21 으로써 미로의 열린 길은 다음과 같습니다



길을 살펴보면 미로의 크기도 적고 다른 경로로 가는 갈림길이 그렇게 많지 않다고 생각하여 BFS로 탐색했을 때 A*와 별 차이가 없거나 더 빠를 수 있다고 생각했습니다.

측정한 결과 제 예상과는 다르게 비슷 하지만 아직 A* 알고리즘에 비해 많이 느리다는 것을 알 수 있었습니다.

```
===fifth floor===
A*; time=160 length=106
BFS; time=202 length=106
IDS; time=5171 length=106
-------------
```

결론

소스코드가 잘못 되었는지 계속 살펴보았지만 놀랍게도 여전히 5개층 **모두 A* 알고리즘이 제일 빠른 것**으로 측정되었습니다. 이 과제로써 저는 A* 알고리즘을 이용하면 heuristic 함수만 정의를 잘 한다면 일반적으로 BFS, IDS 보다는 빠른 결과를 얻을 수 있다는 것을 알 수 있었습니다.