컴퓨터 전공

2015058222 전하영

인공지능 과제1

**변형 미로 경로탐색**

1. **서론**

과제목표

다양한 탐색 알고리즘을 이용한 변형 미로 경로 탐색 수행

사용언어

C++

사용버전

1. 우분투 16.04
2. Mingw 크로스 컴파일러

컴파일 방법

1. Makefile 파일이 있는 폴더로 이동하여 make 명령어를 이용하여 빌드한다.
2. bin폴더와 exe 파일이 생성된다.

사용 탐색 알고리즘

1. Iterative Deepening Search
2. Greedy Best-first Search
3. A\* Search

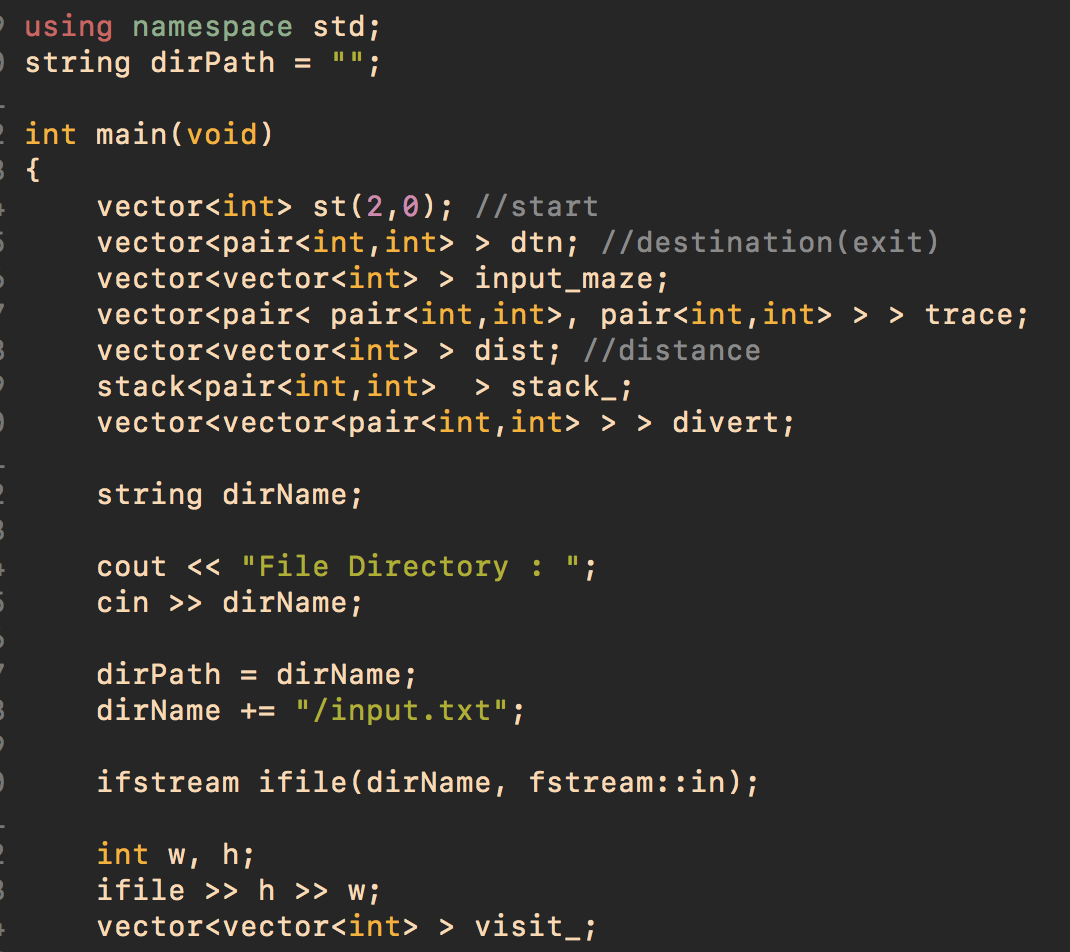
파일명

1. assignment1\_2015058222\_IDS.cc
2. assignment1\_2015058222\_GBS.cc
3. assignment1\_2015058222\_ASS.cc

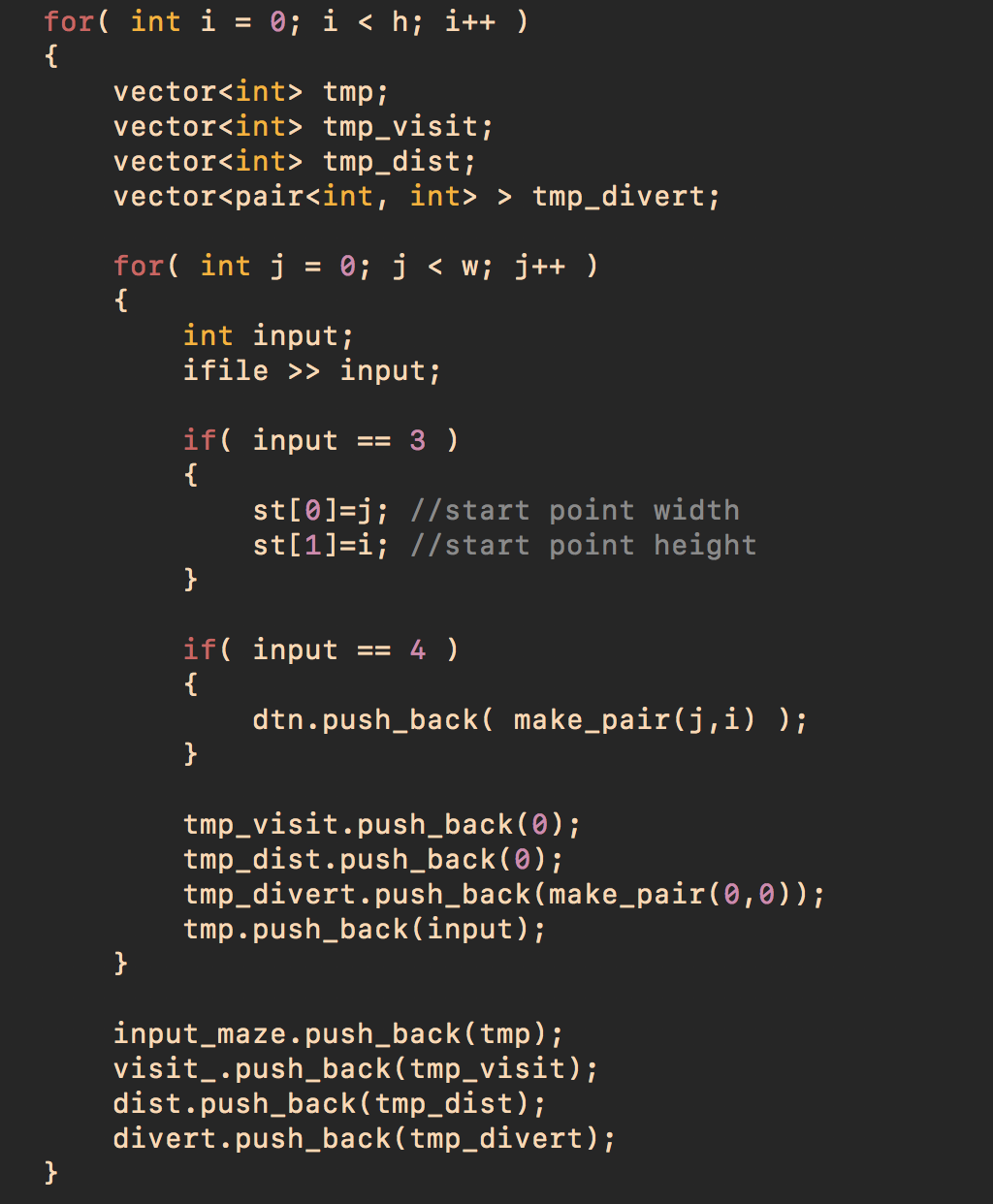
사용방법

1. 실행파일을 실행한 후 input.txt 파일이 있는 경로를 입력한다.
2. 입력한 경로에 output.txt 파일이 생성된다.
3. **본론**

코드설명

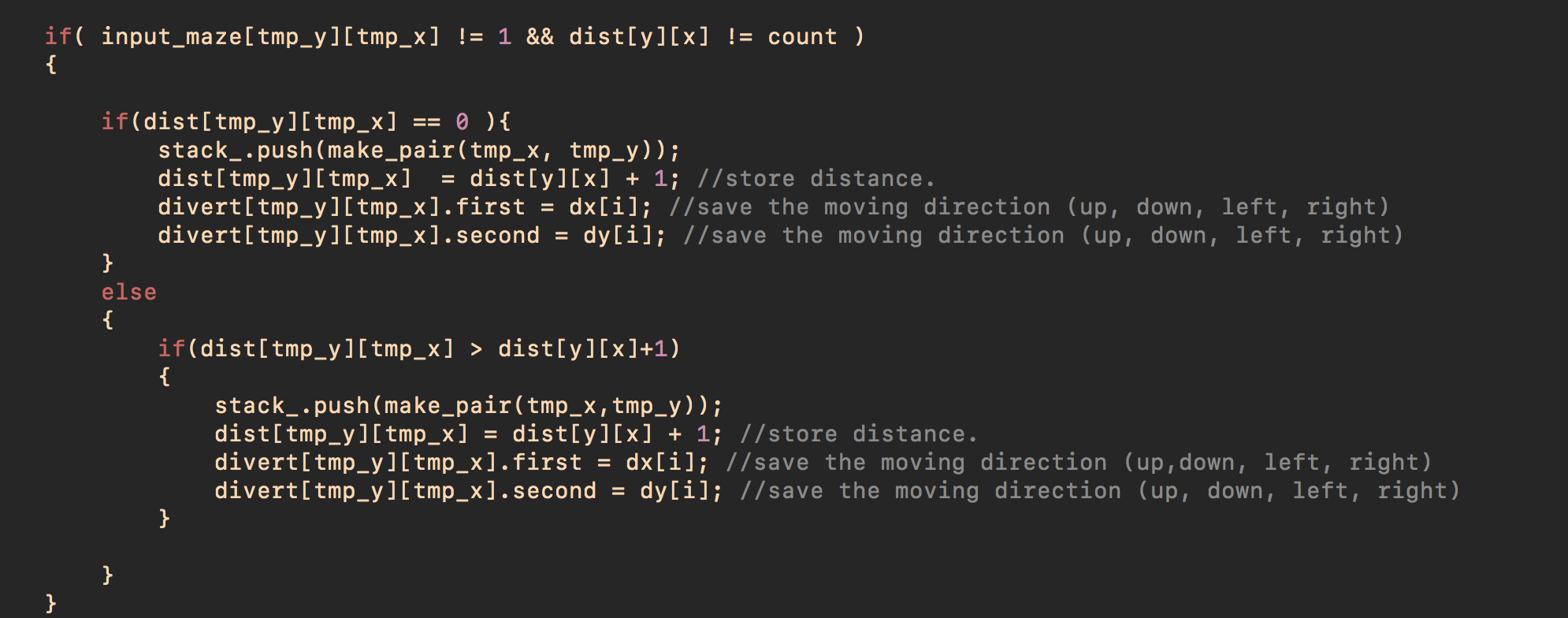
1. Iterative Deepening Search ( 파일명 : assignment1\_2015058222\_IDS.cc )

dirPath 변수를 이용하여 input.txt 의 경로를 저장하고 output.txt를 저장할 경로로 사용한다.

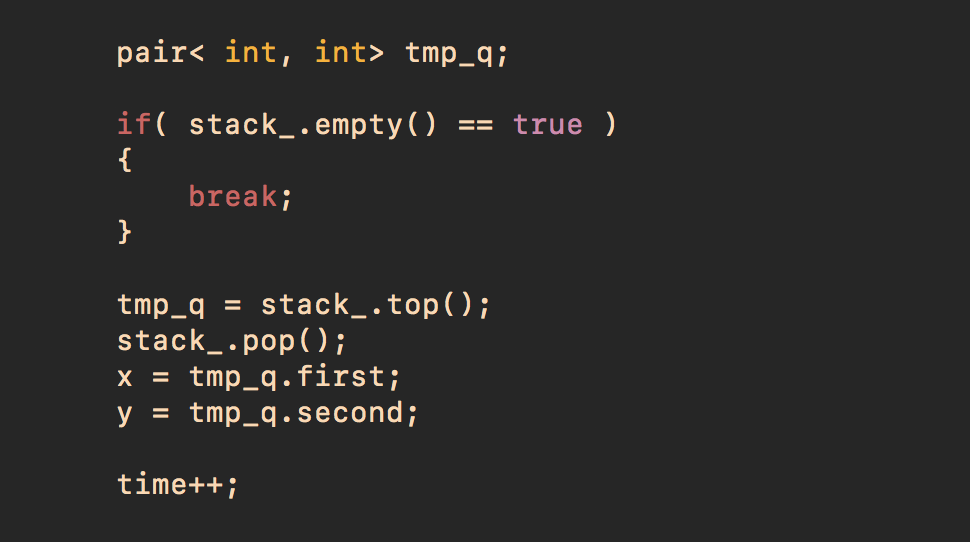


미로(input\_maze)를 input.txt 파일로 입력 받는다.

이동방향을 저장할 divert 벡터, 방문을 했는지 체크하는 visit\_ 벡터, 출발점으로 부터 이동 거리를 계산하는 dist 백터를 0으로 초기화 한다.

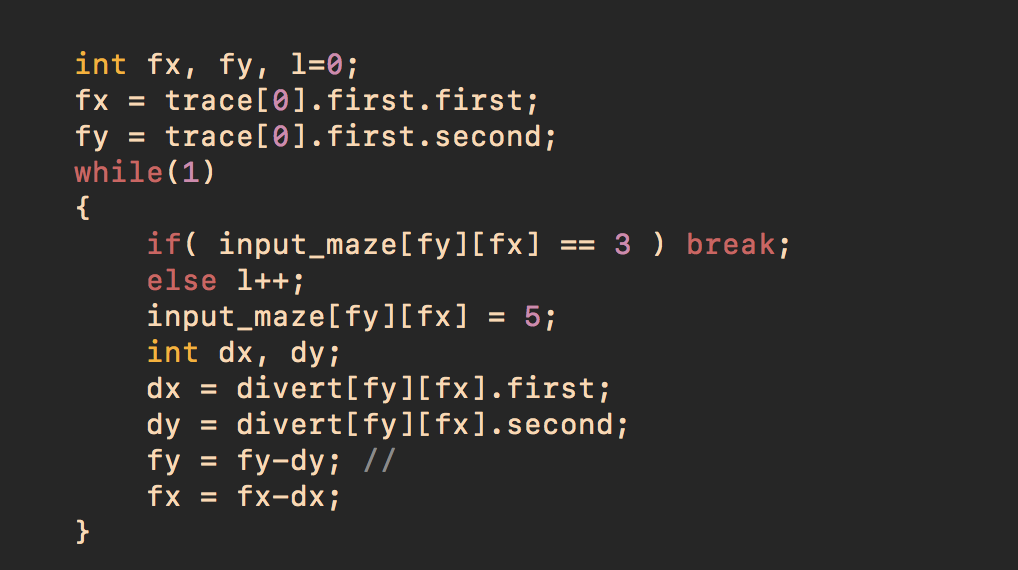
도착점에 도달하면 trace벡터에 현재 위치와 이동한 방향(상,하,좌우) 을 기록한다.

Stack에 이동할 위치를 push 한다. 상하좌우로 이동할 방향을 divert 벡터에 기록하고, 이동할 위치의 distance는 현재 위치의 distance 에 1을 더한 값을 저장한다. 현재 위치의 distance + 1 보다 큰 값이 저장되어 있으면 비교하여 작은 값으로 갱신한다.

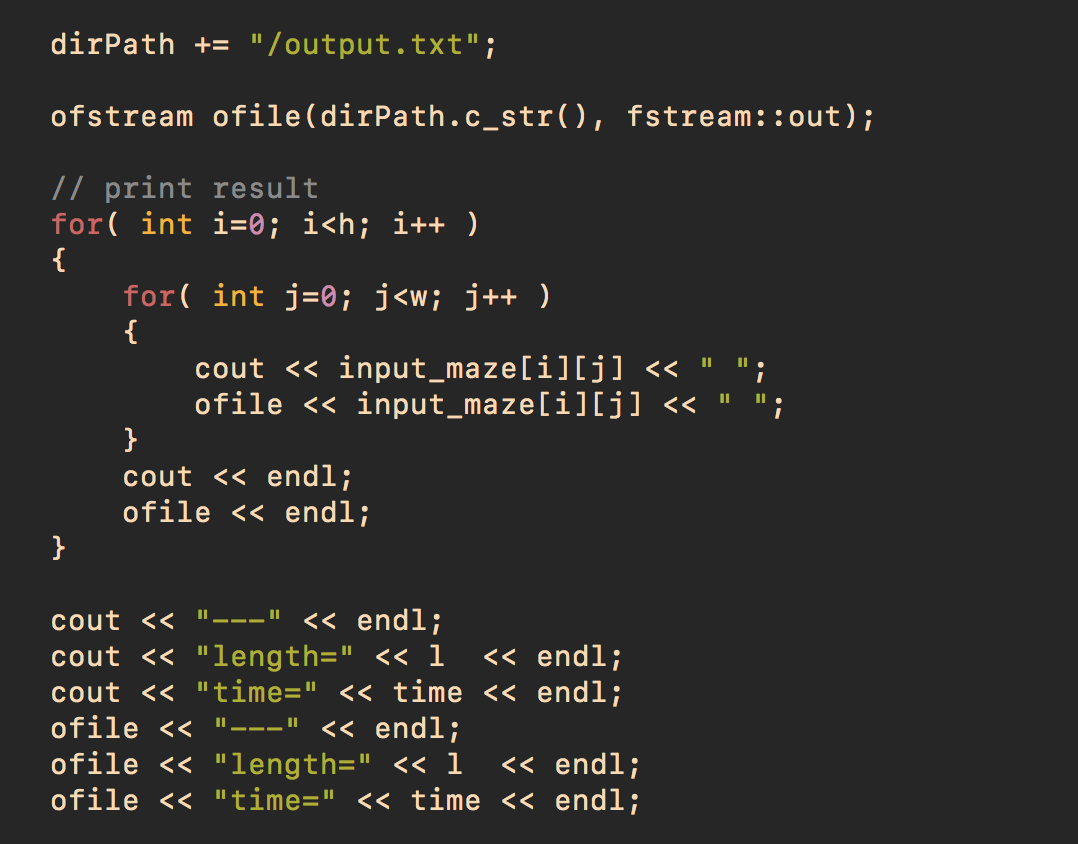


Stack 의 top(), pop()을 사용하여 다음 이동할 위치를 set한다.

탐색 횟수를 1씩 증가시킨다.



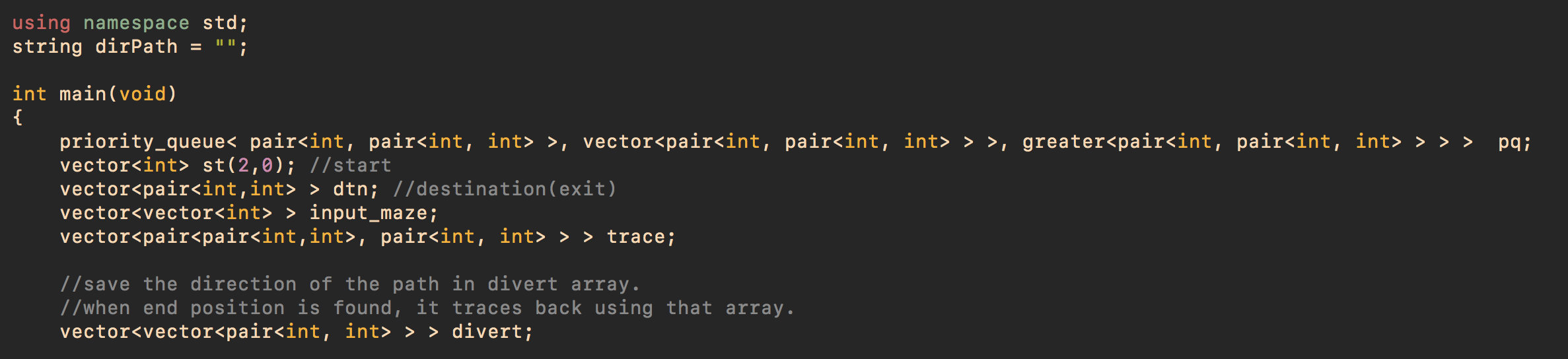
Trace 벡터에 저장해 놓은 마지막 위치와 이동방향을 기록해 놓은 divert 벡터를 이용하여 이동경로를 역추적한다. 이동경로를 5로 표시한다,



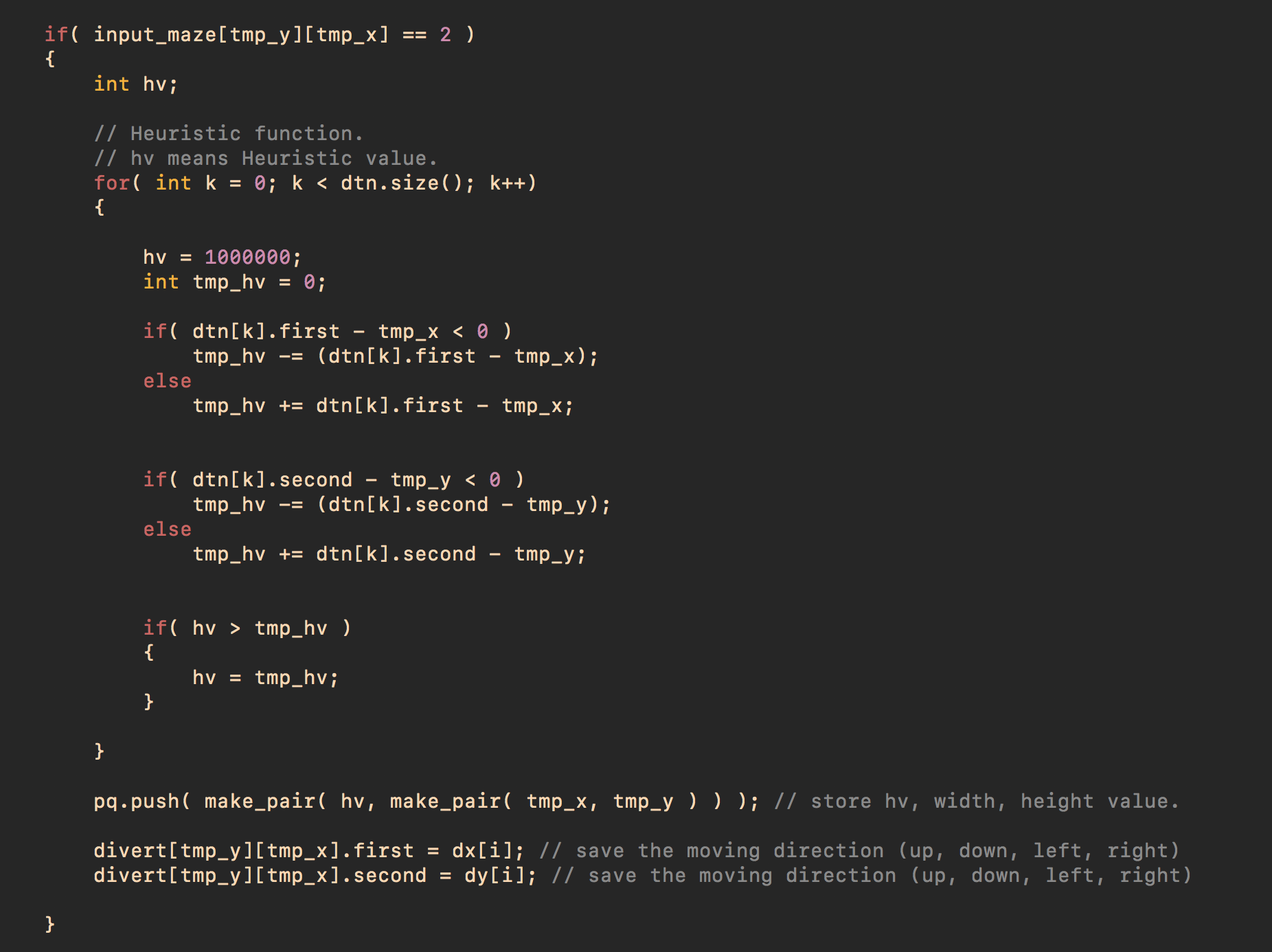
실행파일을 실행한 후 입력하였던 경로로 탐색 경로 결과에 대한 output.txt 파일을 생성한다,

이동경로 Length와 탐색 횟수 time을 표시한다.

1. Greedy Best-first Search ( 파일명 : assignment1\_2015058222\_GBS.cc )

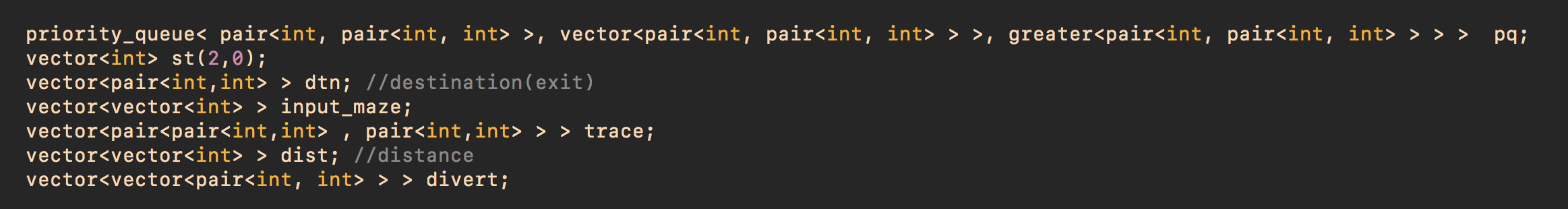


Priority\_queue를 사용한다. Heuristic function을 통해 heuristic value가 작은 값이 우선순위가 된다.

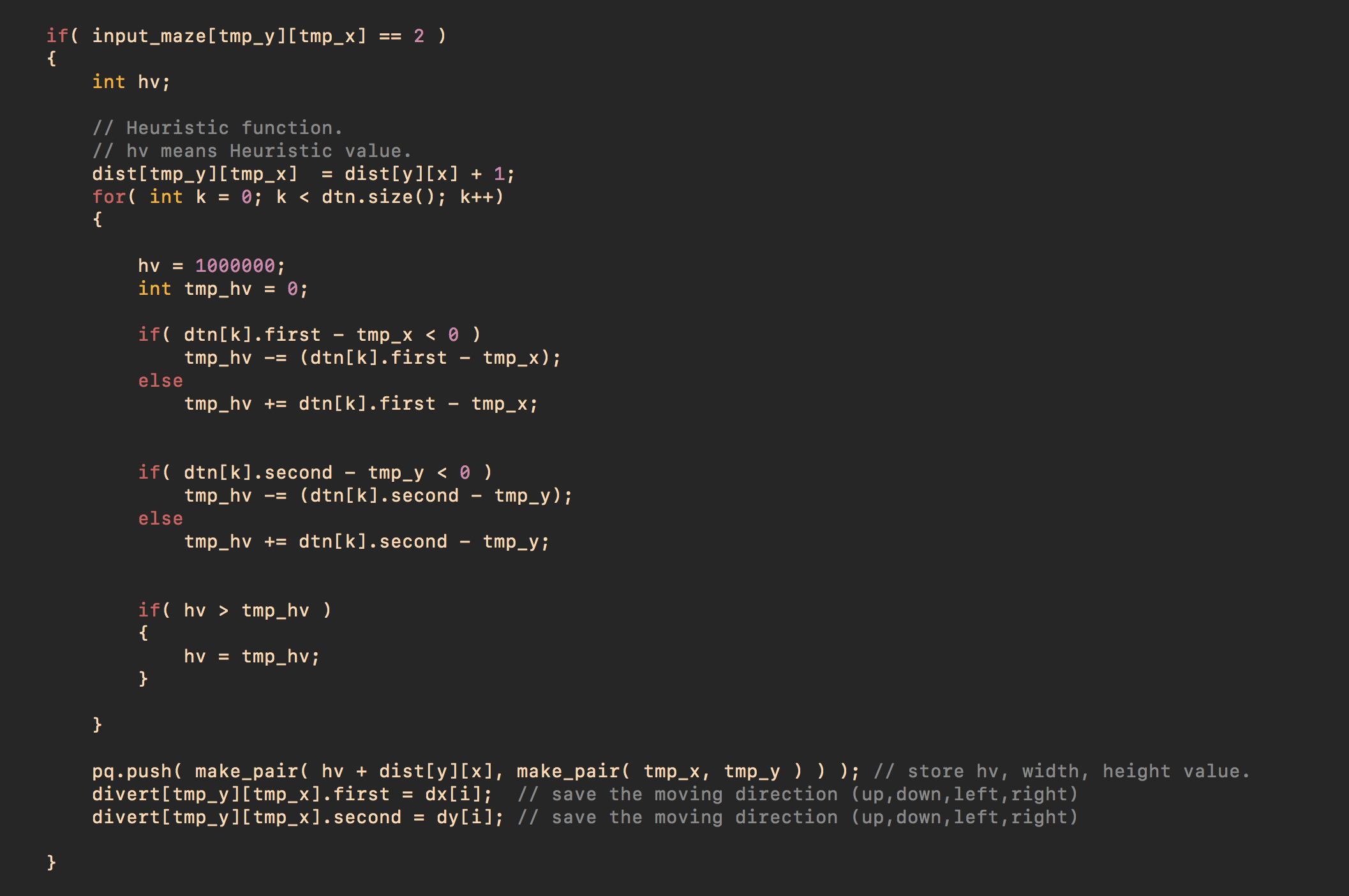


도착지점이 여러 개인 경우 모든 경우에 대해 heuristic value를 계산하여 그중 가장 작은 값을 저장한다. 이 때 heuristic value는 Manhattan distance 로 계산한다.

1. A\* Search ( 파일명 : assignment1\_2015058222\_ASS.cc )



Priority\_queue를 사용한다. Heuristic function을 통해 heuristic value가 작은 값이 우선순위가 된다.

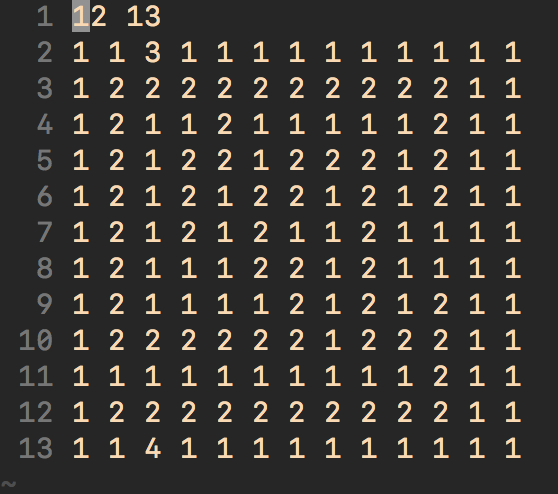
출발점으로 부터 이동한 거리를 dist 벡터에 저장한다.

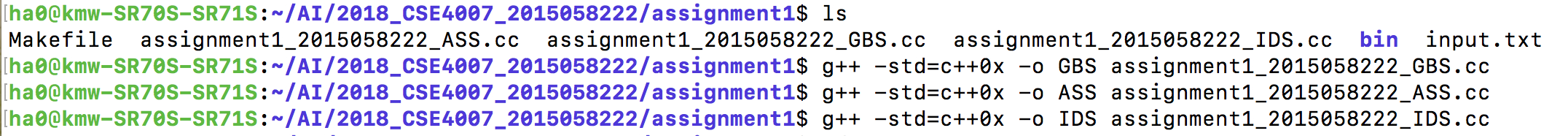
도착지점이 여러 개인 경우 모든 경우에 대해 heuristic value를 계산하여 그중 가장 작은 값을 저장한다. 이 때 heuristic value는 Manhattan distance 와 출발점 으로부터 이동한 거리 수의 합으로 계산한다.

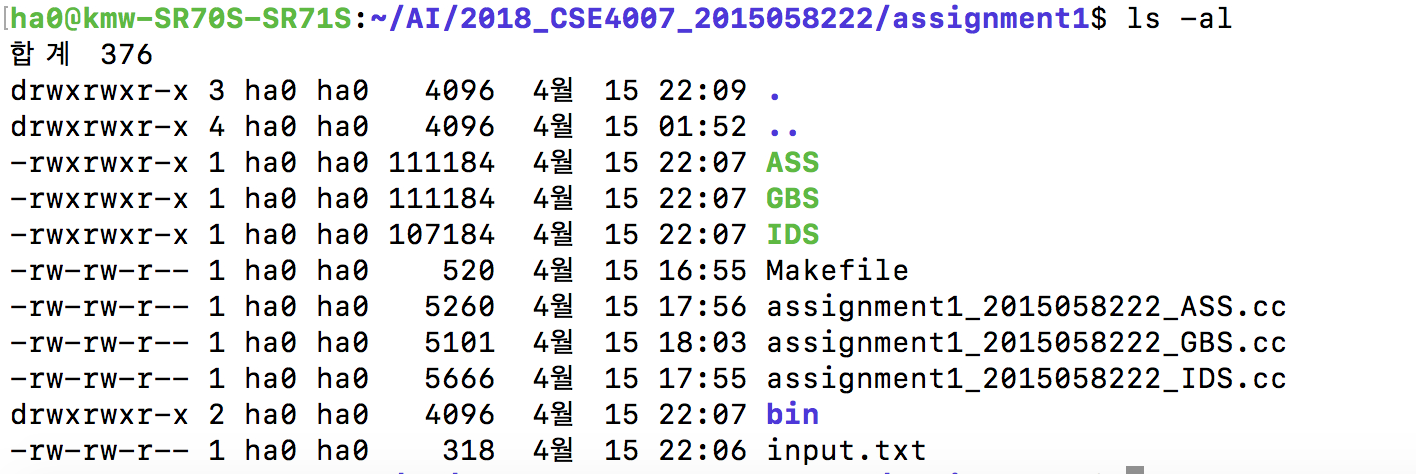
1. **결론**

실험 결과

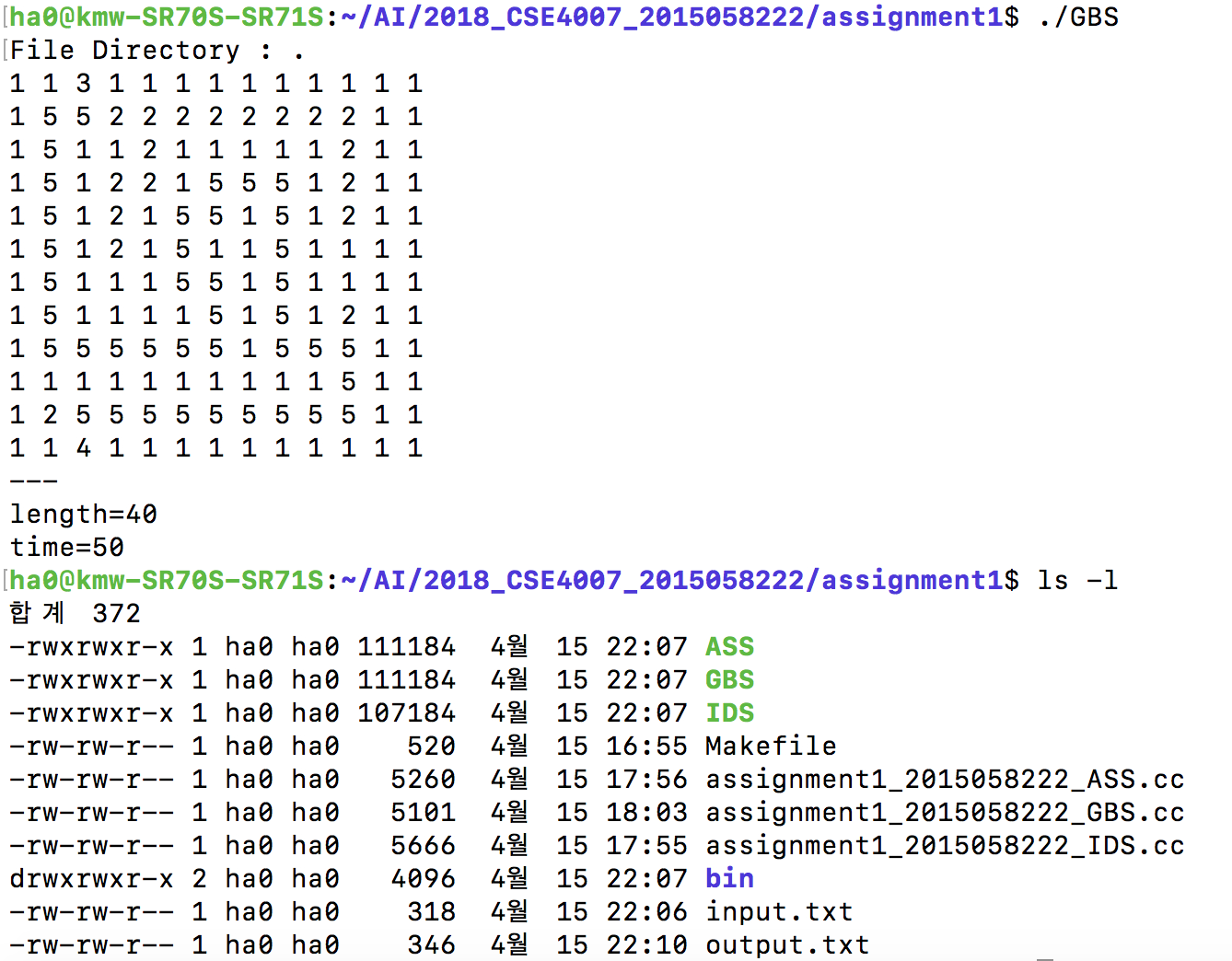
1. 사용한 input.txt



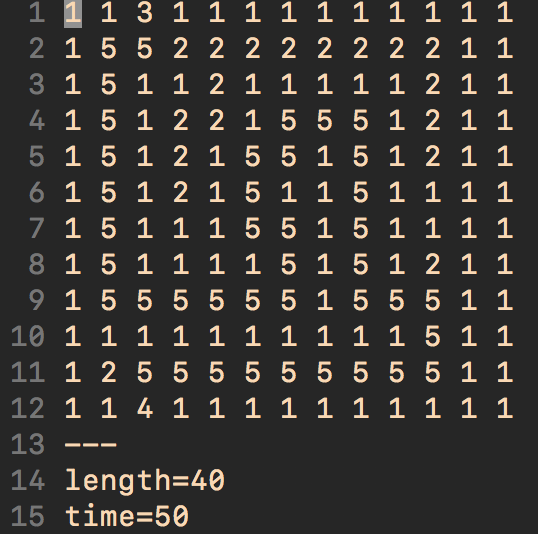


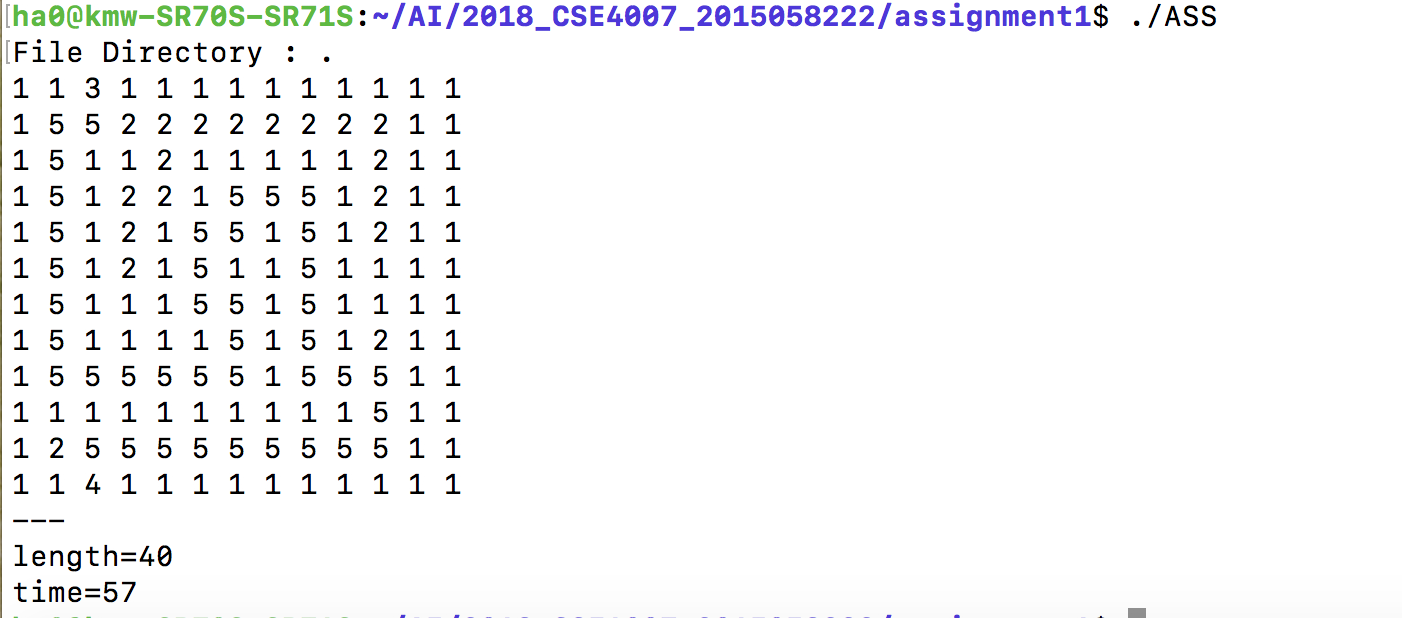


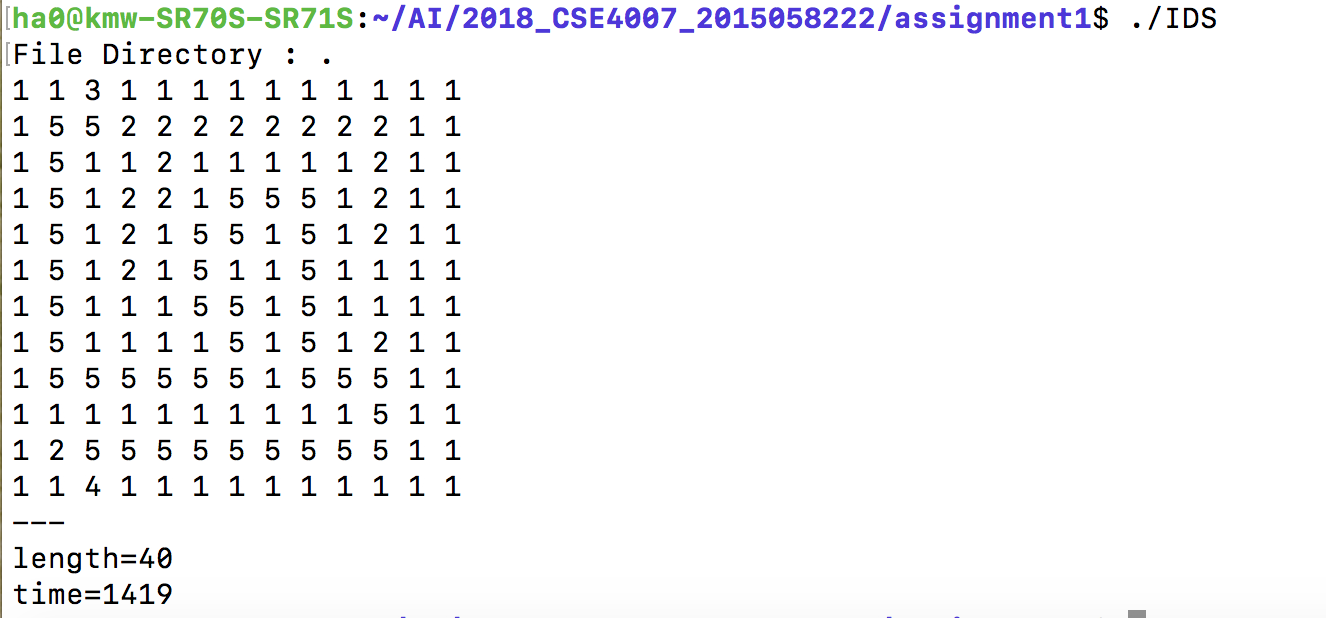
빌드한 후 input.txt 경로를 입력하여 실행한다.



실행한 후 결과 모습과 생성된 output.txt 파일을 확인한다.

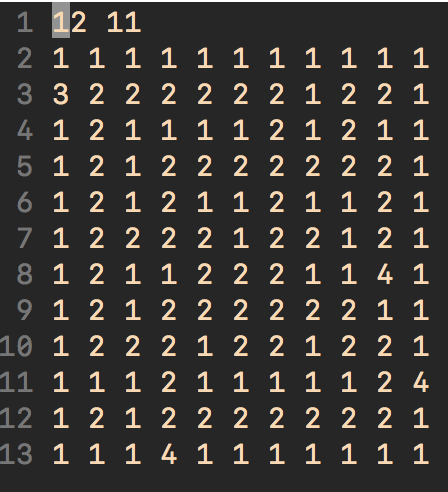


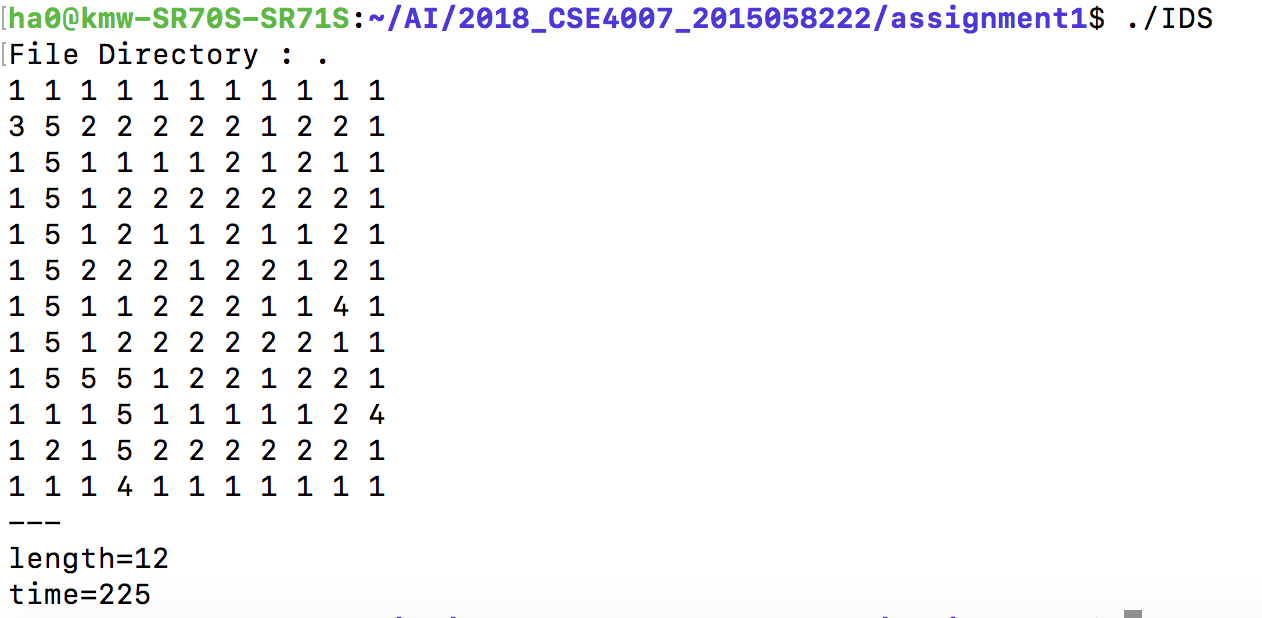


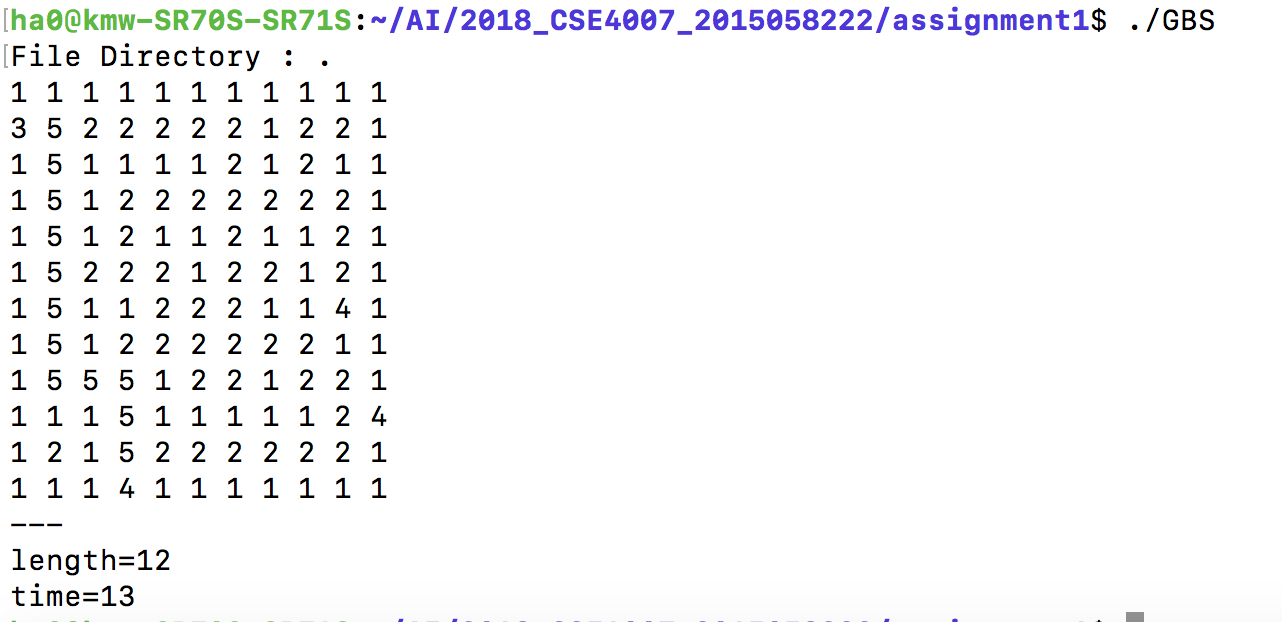


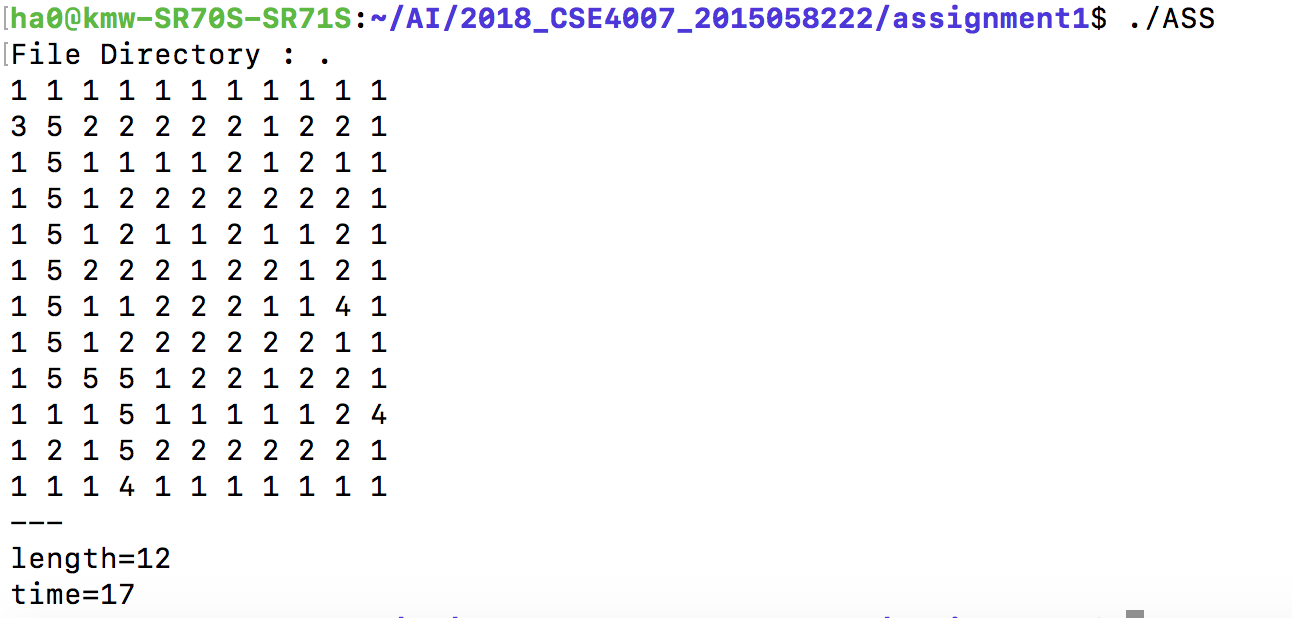
|  |  |  |  |
| --- | --- | --- | --- |
| 미로1 | IDS | GBS | ASS |
| length | 40 | 40 | 40 |
| time | 1419 | 50 | 57 |

1. 사용한 input.txt



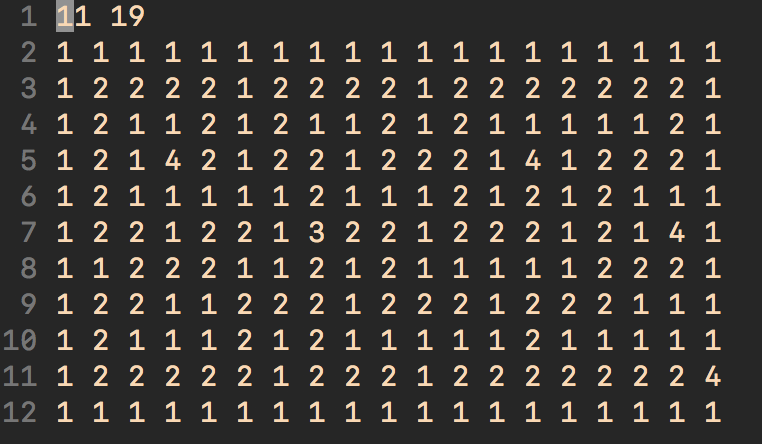


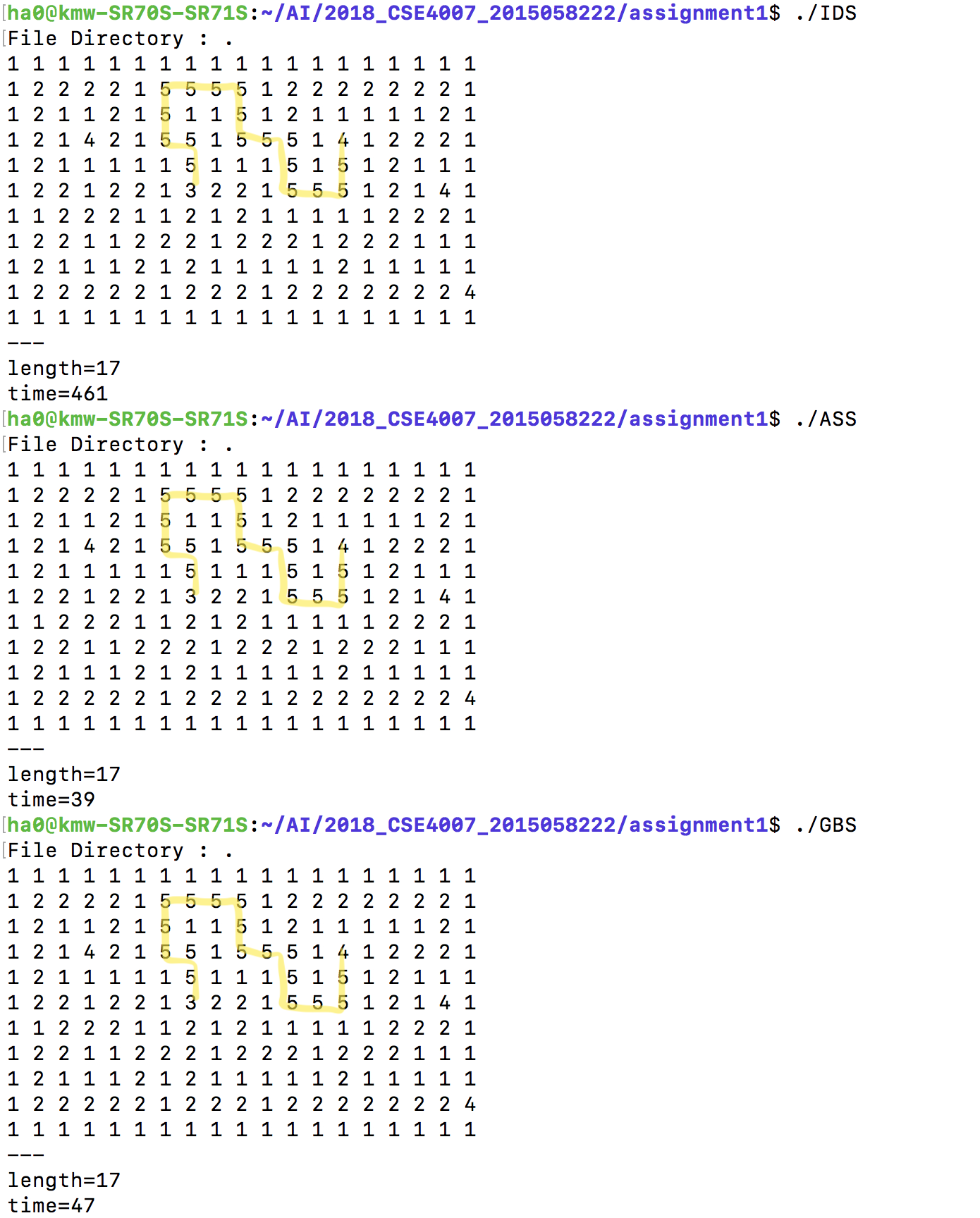




|  |  |  |  |
| --- | --- | --- | --- |
| 미로2 | IDS | GBS | ASS |
| length | 12 | 12 | 12 |
| time | 225 | 13 | 17 |

1. 사용한 input.txt





|  |  |  |  |
| --- | --- | --- | --- |
| 미로3 | IDS | GBS | ASS |
| length | 17 | 17 | 17 |
| time | 461 | 47 | 39 |

3개의 예시를 통한 위의 결과에서 이동 경로와 length는 3개의 알고리즘이 동일하나 time 은 다 다르다. Heuristic function을 사용하지 않는 Iterative Deepening Search가 time이 가장 크며 heuristic function을 사용하는 Greedy Best-first Search 와 A\* Search 두 알고리즘은 time 이 비교적 비슷하다.