Assignment 1

2015004448 김정욱

# 코드설명+함수설명

미로를 입력받고 시작점에서 열쇠위치까지 길을 찾은 후 열쇠에서 부터 도착점까지 다시 한번 길을 찾아준다.

길을 찾을 때 이동하는 방향을 저장 해둔 다음 찾고나면 되돌아 오며 역추적을 해준다.

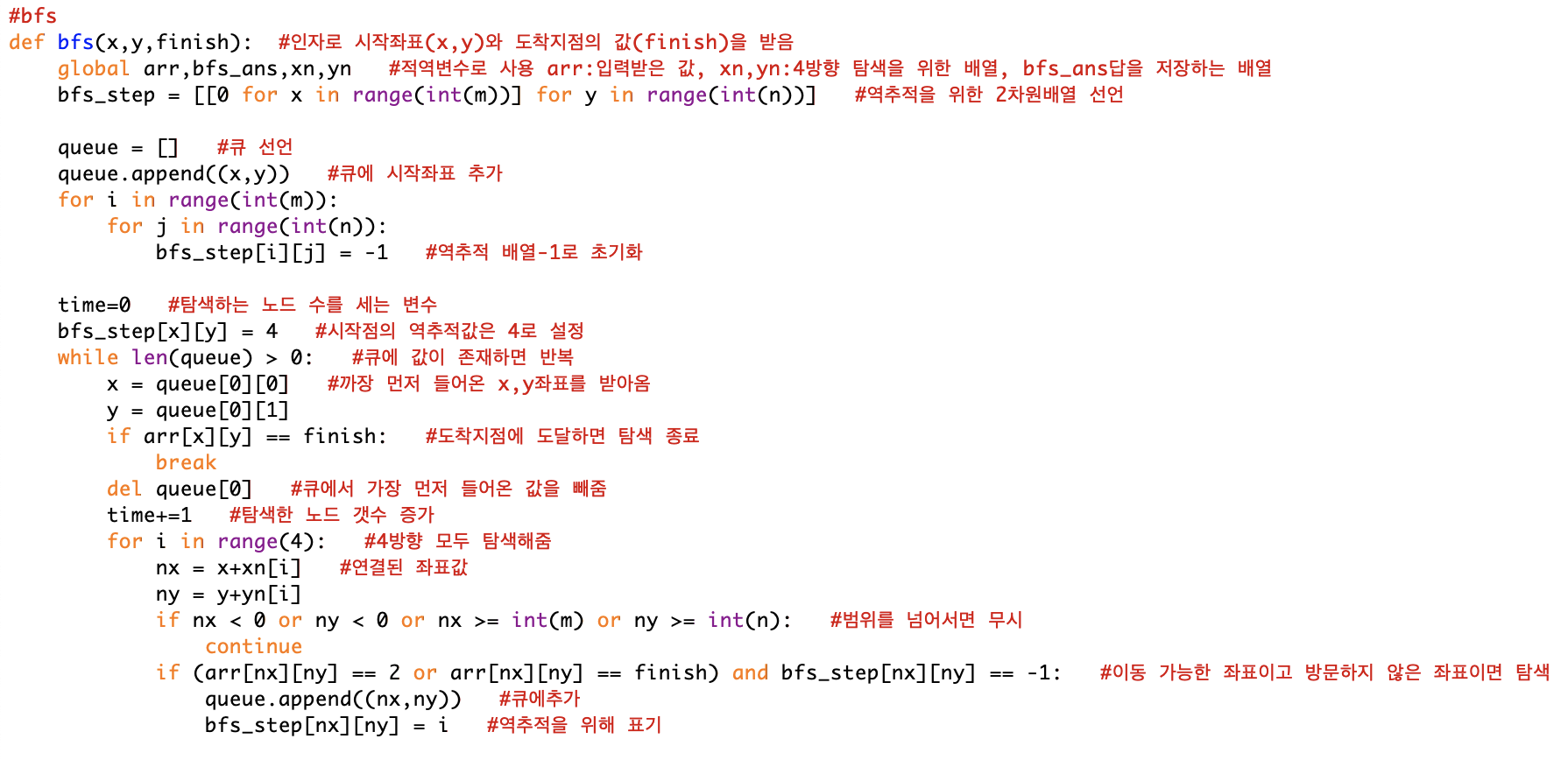
1) BFS : 별도의 큐를 구현하지 않고 파이썬의 list를 활용하여 큐 처럼 사용

1. 시작점을 찾아서 큐에 넣음.

2. 큐에서 가장 앞에있는 노드(가장 먼저 큐에 추가된 노드)를 추출

3. 추출한 노드에 연결된 노드들을 탐색 하여 이동 가능한 노드들을 큐에 추가

4. 도착지점을 찾을때 까지 2,3과정 반복



2) IDS : 별도의 큐를 구현하지 않고 파이썬의 list를 활용하여 스택 처럼 사용

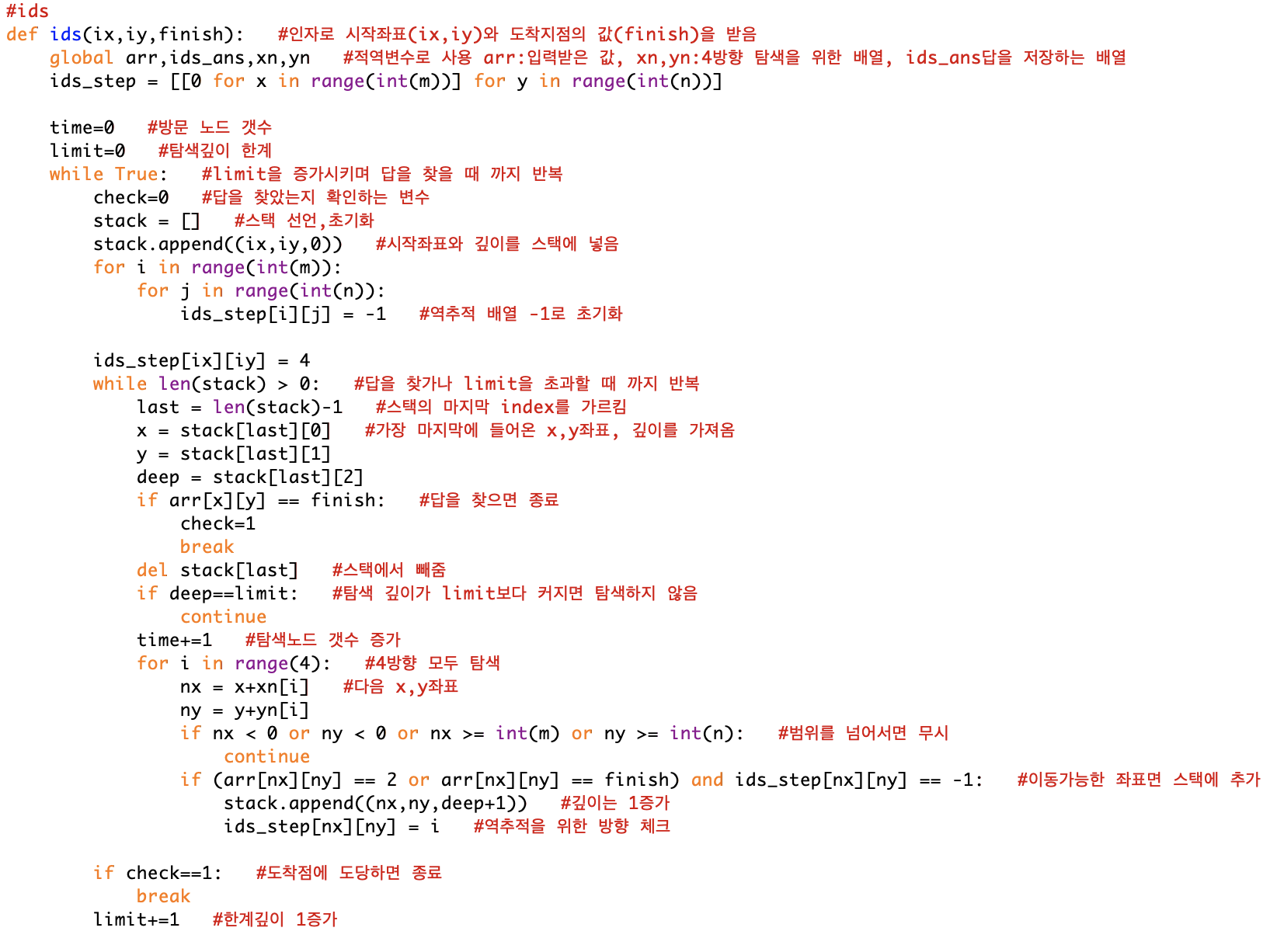
1. 시작점을 찾아서 스택에 넣음, limit설정(깊이를 몇까지 확인 할 것인지)

2. 큐에서 가장 앞에있는 노드(가장 나중에 스택에 추가된 노드)를 추출

3. 추출한 노드에 연결된 노드들을 탐색 하여 이동 가능한 노드들을 큐에 추가

4. 도착지점을 찾거나 limit을 초과할 때 까지 2,3과정 반복

5. 도착지점을 찾지 못햇으면 limit을 증가시키고 1~4과정 반복



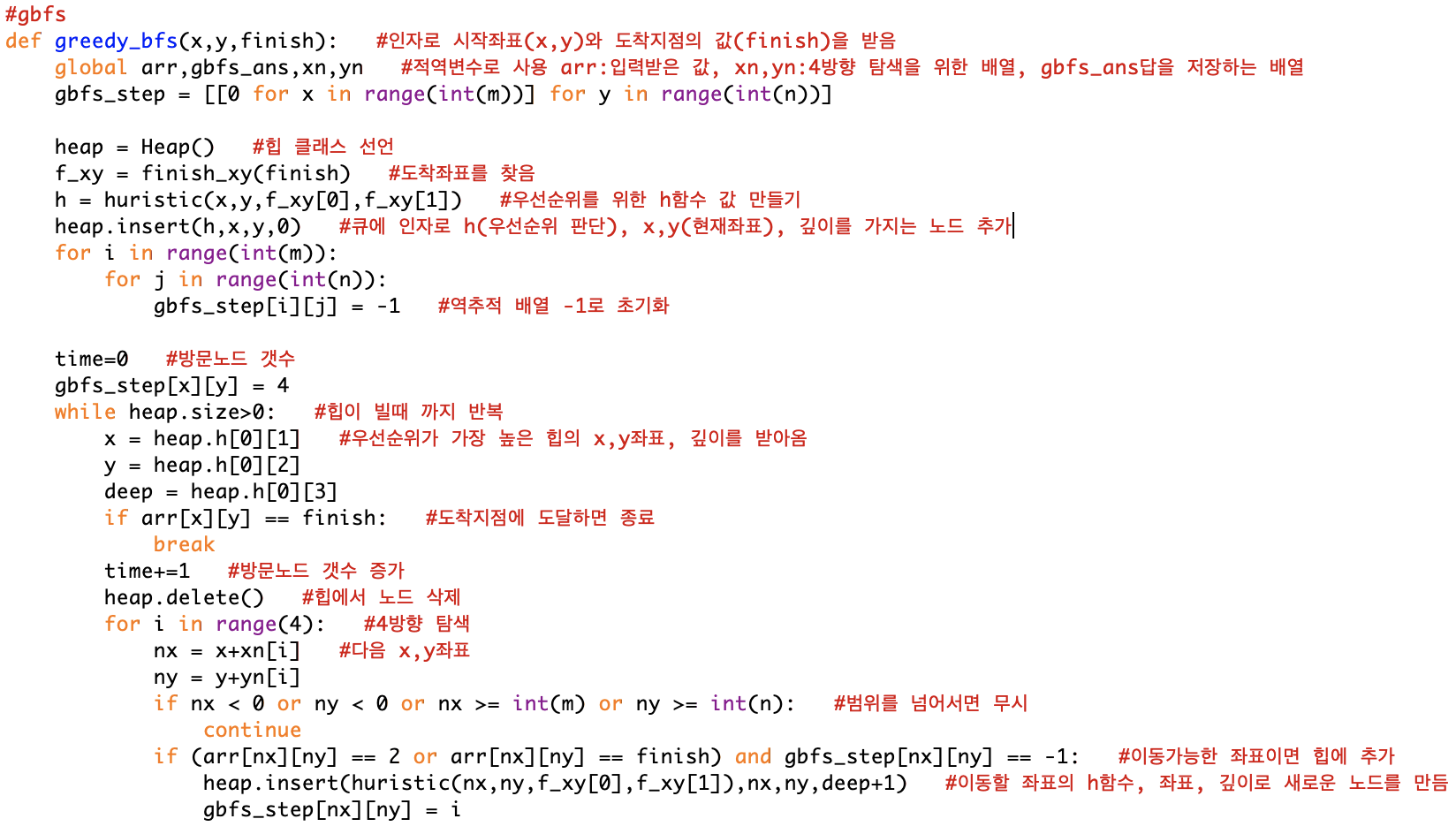
3) GBFS : 힙 class구현(우선순위 : 현재지점에서 도착지점까지의 벽을 무시한 거리)

1. 시작점을 찾아서 큐에 넣음.

2. 큐에서 가장 앞에있는 노드(큐에 있는 노드들 중 가장 우선순위가 높은 노드)를 추출

3. 추출한 노드에 연결된 노드들을 탐색 하여 이동 가능한 노드들을 큐에 추가

4. 도착지점을 찾을때 까지 2,3과정 반복



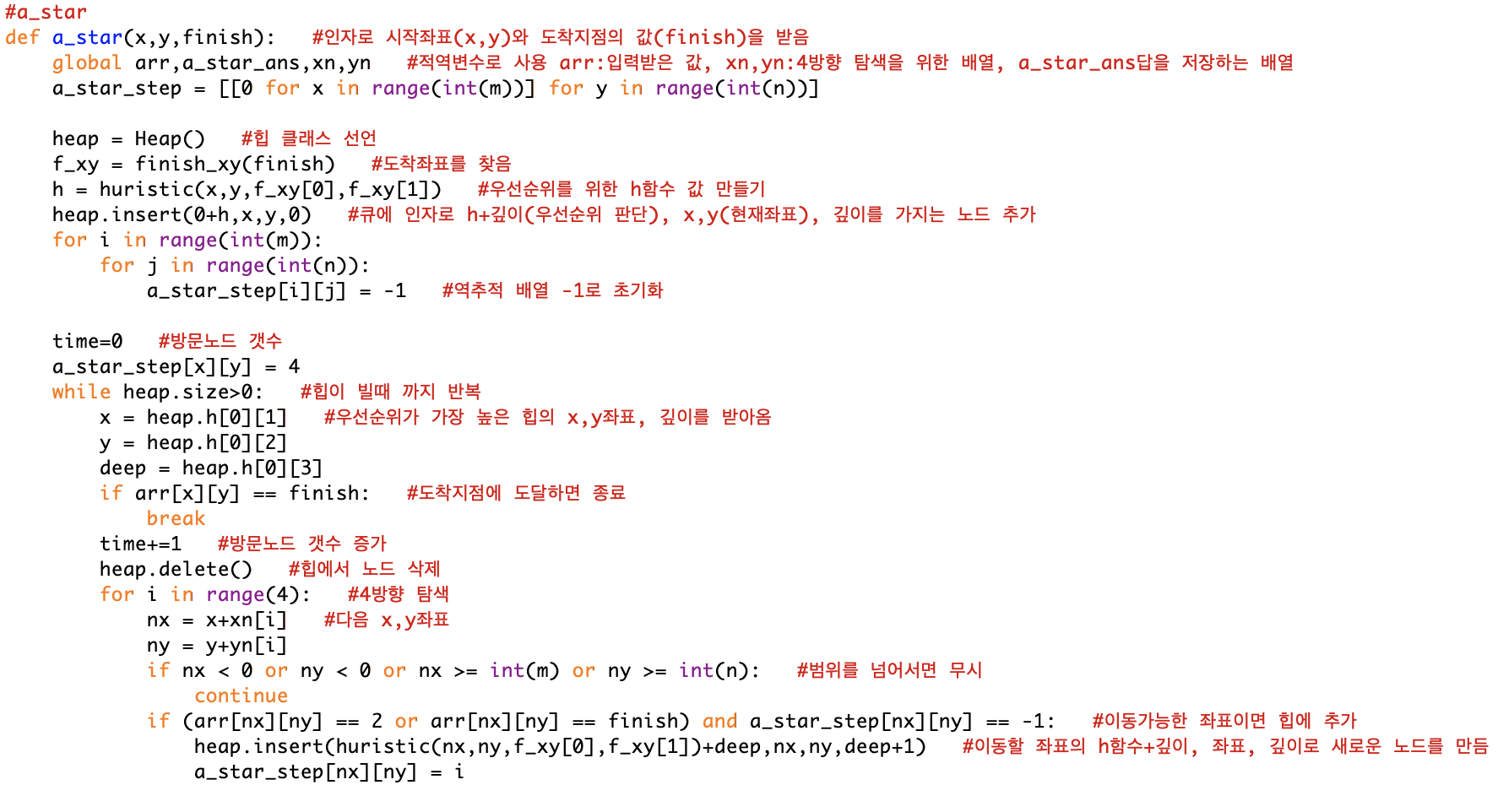
4 ) A\_star : 힙 class구현(우선순위 : 현재지점에서 도착지점까지의 벽을 무시한 거리 + 현재까지 온 깊이(길이))

1. 시작점을 찾아서 큐에 넣음.

2. 큐에서 가장 앞에있는 노드(큐에 있는 노드들 중 가장 우선순위가 높은 노드)를 추출

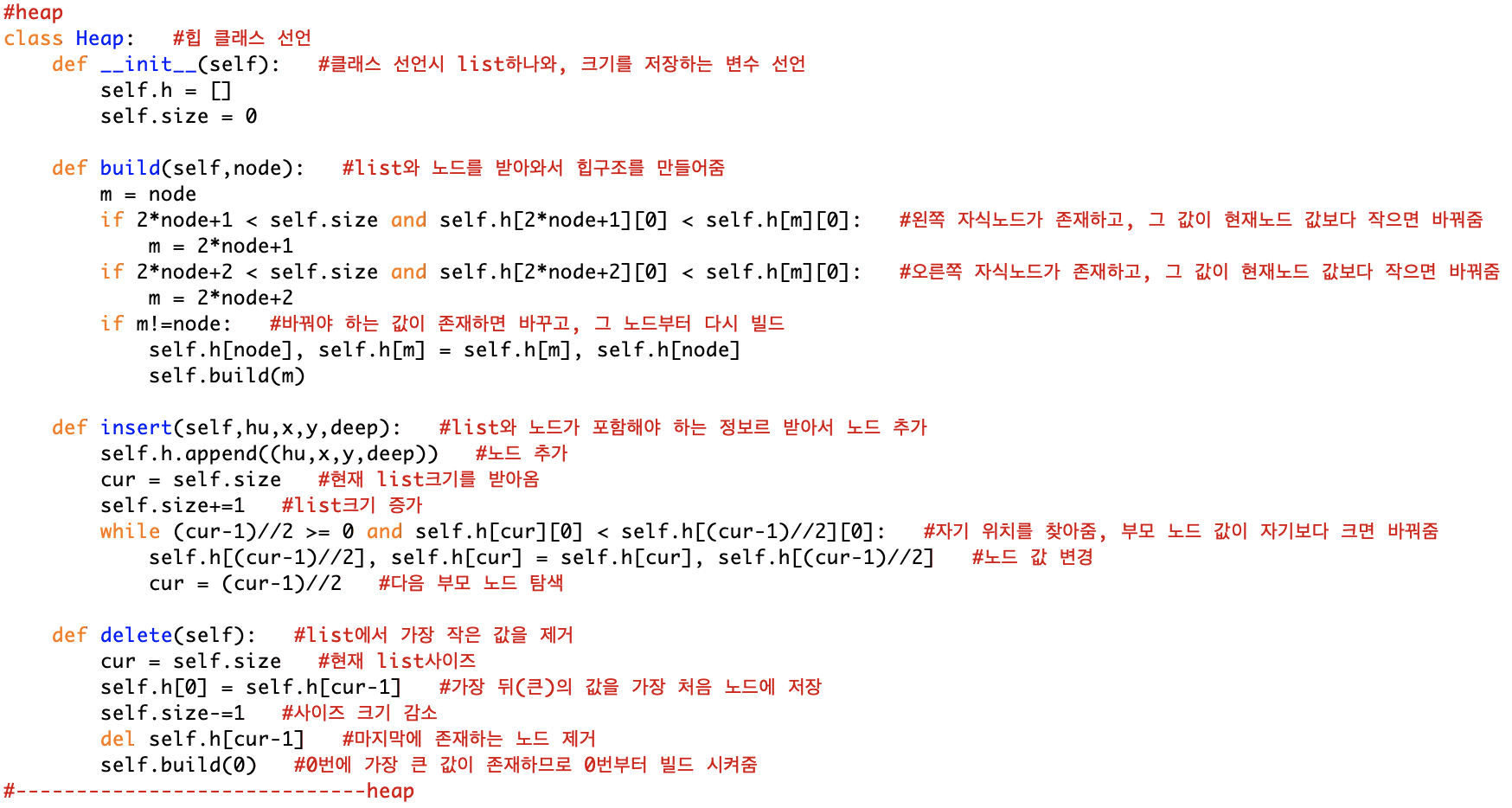
3. 추출한 노드에 연결된 노드들을 탐색 하여 이동 가능한 노드들을 큐에 추가

4. 도착지점을 찾을때 까지 2,3과정 반복

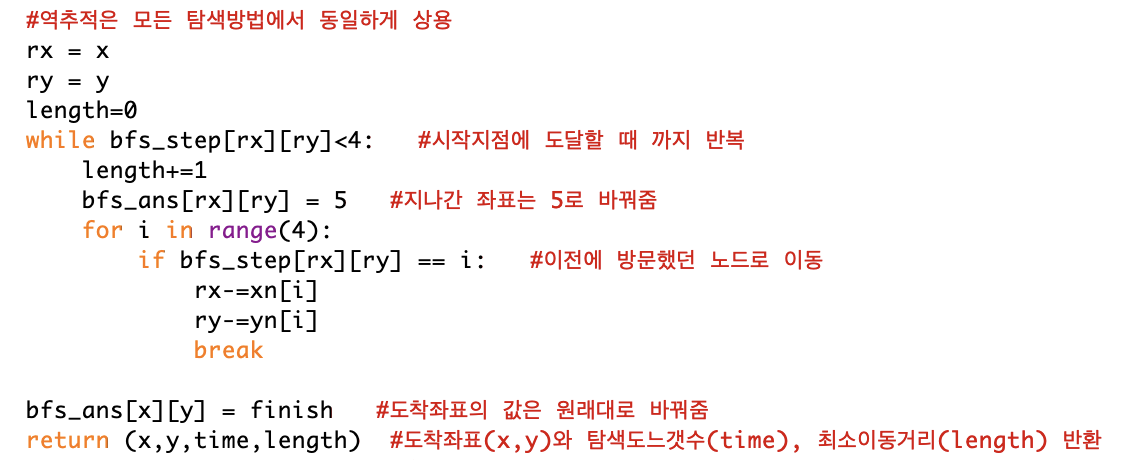


5) 힙 : 클래스 안에 노드추가를 위한 insert, 제거를 위한 delete, 제거 후 다시 힙으로 바꿔주는 build를 구현 하였고, 선언시 힙으로 사용할 리스트 하나와 리스트 크기를 가진다.

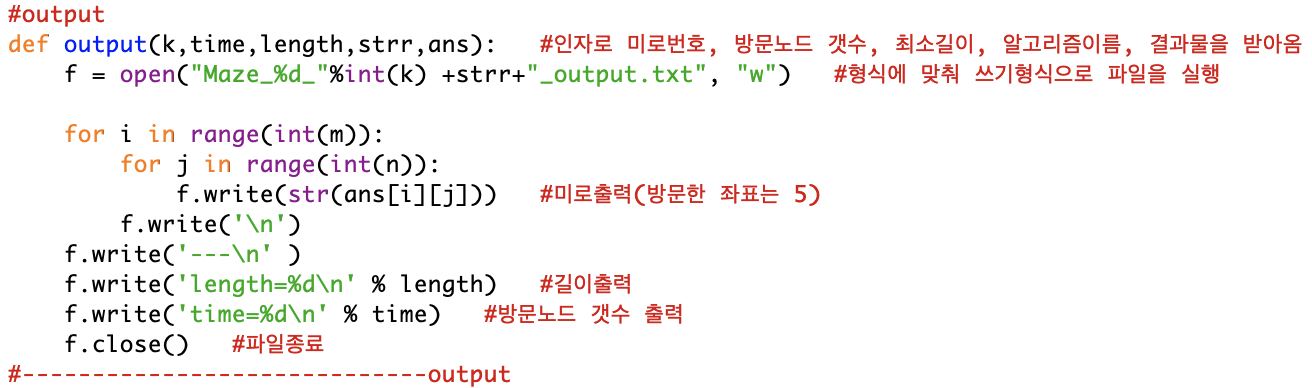
하나의 노드는 우선순위를 위한(hu), 좌표(x,y), 깊이(deep)를 가진다.



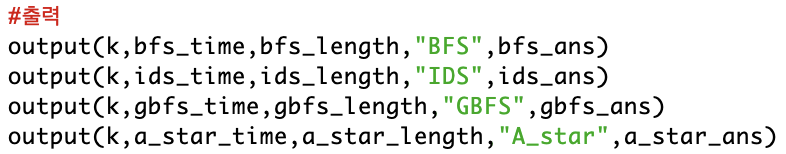
6) 역추적 : 모든 알고리즘에서 동일하게 사용되며 step이란 배열에 이동해온 위치를 저장해 둔 뒤 그것을 확인하여 도착지점부터 거꾸로 가며 이동한 곳은 5로 바꿔준다.



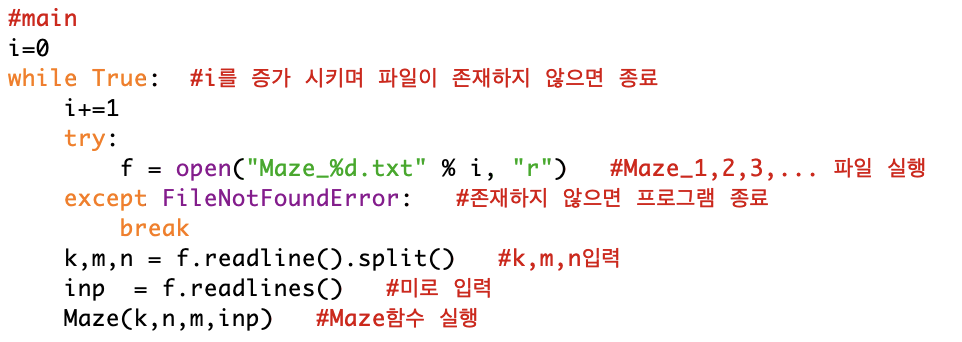
7) 출력 : 인자로 미로번호(k), 방문횟수(time), 최소길이(length), 탐색알고리즘(str), 정답배열(ans)를 받아와서 형식에 맞춰 출력한다.



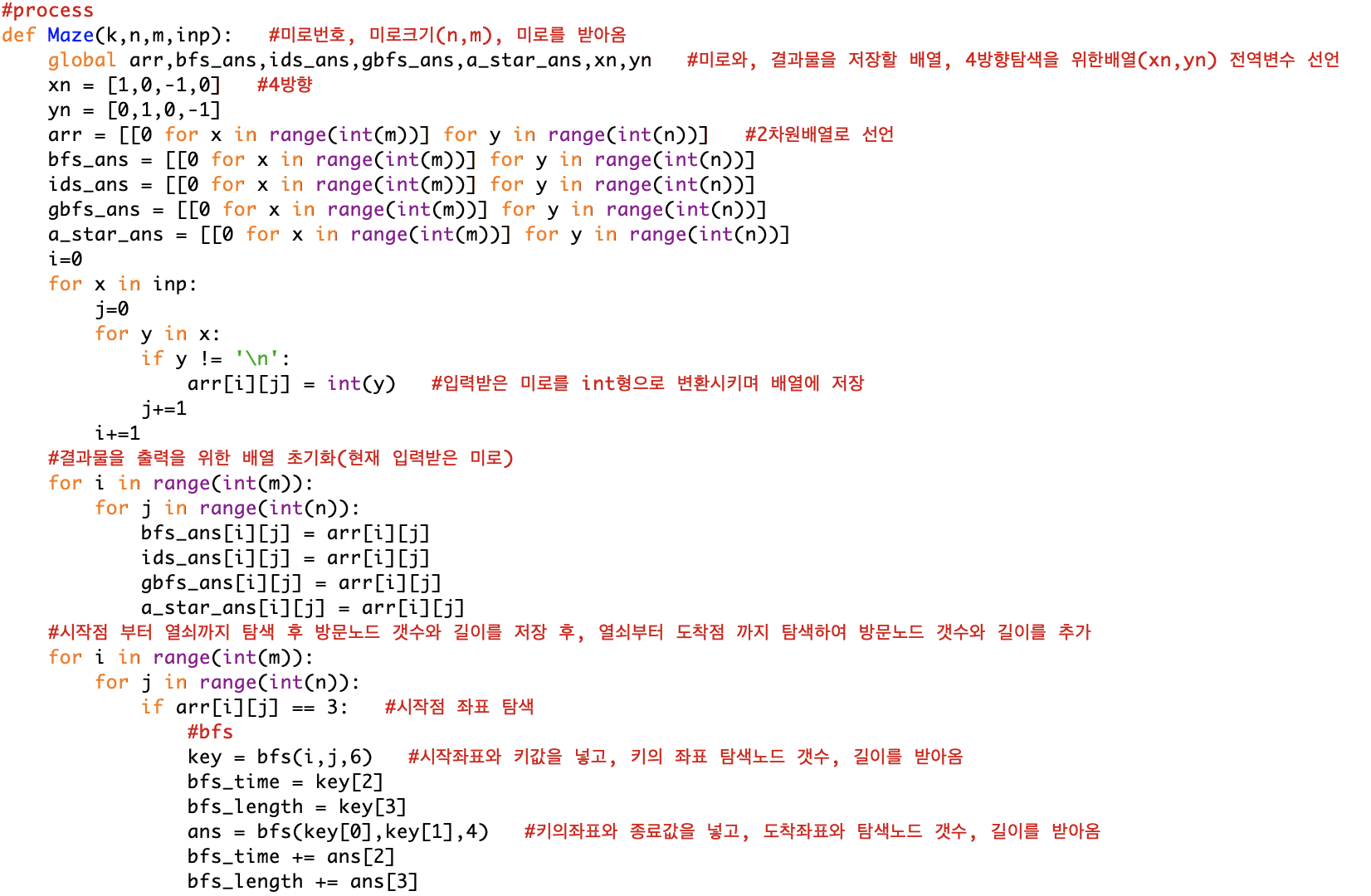
Maze 함수에서 출력을 위해 넘겨주는 부분



8) 메인함수 : Maze\_i라는 형식의 파일을 1부터 순서대로 연다. 실패 시 종료한다. 첫줄을 읽어온 후 split을 사용하여 k,m,n을 입력받고, readlines()를 사용하여. Inp라는 리스트에 모두 가져 온 후 Maze함수로 넘겨준다.

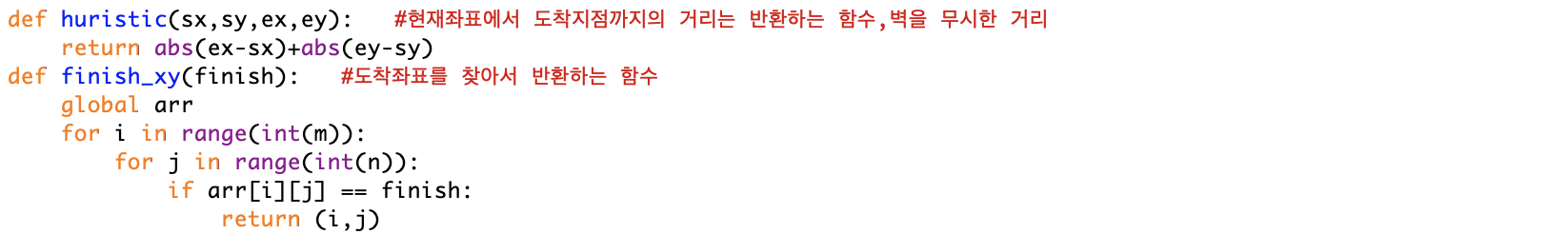


9) 미로처리를 위한 함수 :



bfs, ids, gbfs, a\_star 같은방법으로 반복해준 후 출력함수로 넘겨준다.

10) 기타함수 : 도착해야할 값을 알 때 그 위치를 찾아서 반환해주는 finish\_xy함수, 힙 에서 사용하는 우선순위를 위해 사용하는 함수 huristic이다. Huristic함수는 벽이나 장애물을 생각하지 않고 현재 위치에서 도착 위치까지의 거리를 반환 해준다.



# 실험결과

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | BFS | IDS | GBFS | A\_STAR |
| Maze\_1 | Length | 2782  6830 | 2782  3935 | 2782  4184992 | 2782  6376 |
| Time |
| Maze\_2 | Length | 878  1830 | 878  1604 | 878  368367 | 878  1732 |
| Time |
| Maze\_3 | Length | 265  397 | 265  335 | 265  25337 | 265  387 |
| Time |