인공지능 과제 1

IT/BT 탈출하기

2013012278 하경모

1. 목적

- 미로의 입구에서 출발해 열쇠를 획득한 후 탈출구까지 가는 최단경로와 탐색된 총 노드의 수를 구하기

2. 디자인

- 비효율적일 것으로 예상되지만 가장 단순한 알고리즘인 bfs를 큐를 이용해 먼저 구현
  - dfs는 최단경로를 찾지 못할 가능성 때문에 보류
- 좀 더 효율적인 알고리즘을 구현하여 기존의 알고리즘들과 비교해가며 최선의 결과를 내는 알고리즘을 채택

3. 구현

- bfs (def bfs(maze))
  - 큐를 이용해 목적지가 나올 때까지 계속 탐색 (import queue)
- A* (def heuristic(maze))
  - 힙을 이용해 구현 (import heapq)

- 현재 위치에 도달하기까지 거쳐온 노드의 개수와 탈출구까지의 거리 합으로 우선순위 부여
- 우선순위가 가장 높은 (합이 가장 작은) 노드부터 우선 탐색

- greedy (def heuristic_greed(maze, filename))
  - 힙을 이용해 구현 (import heapq)
  - 기본적으로 A* 알고리즘과 동일하지만 우선순위 부여시 단순히 탈출구까지의 거리만으로 계산

- heuristic function을 이용한 알고리즘은 최단경로를 찾는다고 확실히 보장할 수 없기에 bfs를 먼저 사용해 최단거리를 확실시하고 사용

- 최단경로 표시를 위해 트리 클래스 구현 (class Tree, class Node)
  - 3 branch factors
  - 탈출구 노드부터 시작까지 반대로 탐색하기 위해 node.parent 추가

4. 구현 및 실행 결과

- readmazeinfo(maze)
  - 미로의 총 크기, 즉 행과 열을 읽어오는 메소드

- find(maze)
  - heuristic function을 위해 입구, 열쇠, 출구의 위치를 찾는 메소드

- **모든 층에서 greedy 알고리즘이 가장 좋은 결과를 도출했으므로 모든 층에 greedy 알고리즘 선택**

- distance_key_greed & distance_exit_greed
  - greedy 알고리즘에 쓰일 heuristic function
  - 현재 위치에서 원하는 목적지까지의 거리를 단순히 행과 열의 차이의 합으로 반환

- **heuristic_greed**(maze, filename)

  - 입력 파일의 미로 전체가 담긴 maze와 각 층의 출력 파일의 이름이 담긴 filename
  - 성공시 0 반환, 실패시 -1 반환
  - distance_* 메소드의 반환값을 우선순위로 힙을 이용해 가장 목적지와 가까운 노드부터 탐색
  - 탐색 중인 노드 기준으로 동서남북의 노드들 중 탐색 가능한 (이전에 탐색하기 않았고 벽이 아닌) 노드를 힙에 넣고 가장 우선순위가 높은 노드를 선택해 탐색 반복
  - 선택 시마다 totalnode 변수를 증가시켜 총 탐색노드 계산
  - 먼저 열쇠를 찾고 열쇠를 찾았다면 힙을 초기화해 다시 탐색 시작
    - 즉, 열쇠의 위치를 출발지로 놓고 탈출지점을 탐색
  - 힙에 노드를 추가할 때마다 트리에도 노드를 추가
    - 후에 최단경로를 출력할 때 이용

- printmaze(nodes, maze, length, time, filename)
  - 실행결과를 파일에 출력하는 메소드
  - 트리에서 탈출구의 위치 값을 저장한 노드부터 자신의 부모노드를 계속 검색해 나가면서 최단경로를 출력

- ***_floor**()
  - 모든 층의 알고리즘이 같기에 통합해서 설명
    - 예로, 첫 번째 층의 미로를 탐색하는 메소드의 이름은 first_floor()
    - first_floor.txt 파일을 열어서 미로를 읽어들임
  - find와 heuristic_greed 메소드 수행, 실패하면 프로그램이 정지
    - 입구, 열쇠, 출구가 없거나 탈출할 수 없는 미로일 경우

- class Node
  - 트리에 추가될 노드 클래스
  - 자신이 가질 데이터 하나와 자식 노드로 향하는 가지 3, 부모노드로 향하는 가지 1개 보유

- class Tree
  - 삼진 트리
  - 부모와 자식노드 모두 인자로 받아 서로간을 연결
- main()
  - first_floor()부터 fifth_floor()까지 모두 실행
- **실행결과**
  - 1층 length : 3850, time : 5815
  - 2층 length : 758, time : 994
  - 3층 length : 554, time : 655
  - 4층 length : 334, time : 432
  - 5층 length : 106, time : 122