**PtyQueue Class**

자료구조와실습  
- 기말 시험 과제 –

동국대학교 컴퓨터공학과  
2016112122 이한빈

BFS를 사용하기 위해선 Queue가 필요하다. 상자를 민 횟수와 걸은 횟수가 적을수록 우선순위를 부여하고, 우선 순위가 높은 값이 먼저 Dequeue 될 수 있어야 하므로 Linked List로 구현된 Queue 대신에, heap 구조를 기반으로 구현된 우선순위 큐를 사용하였다.

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function name** | **Description** |
| void | enqueue | Queue의 마지막 노드 뒤에 data가 삽입된 후, Reheapup 과정이 일어난다. 이를 통해 힙속성을 회복하고 우선순위가 높은 data는 먼저 dequeue 될 수 있게 된다. |
| DT(template) | dequeue | Queue의 top 노드가 출력된다. Queue의 마지막 노드와 top노드가 교체된 후, root부터 reheapdown 과정을 통해 힙속성을 회복한다. |

**Point Class**

그래프에서 좌표 값을 표시하기 위한 class이다. copy constructor, operator overloading을 사용하여 프로그래밍을 편히 할 수 있게 했다.

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function name** | **Description** |
|  | Point(const Point& ref) | 복사 생성자. row, col 값을 깊은 복사한다. |
| void | set(int x, int y) | Maze Class에서 사용자의 입력을 받은 후 box, user, target의 좌표가 설정되므로 이를 위해 좌표를 조정할 수 잇는 set()함수를 구현. |
| bool | operator==(Point& ref); | Point 간 == 연산자 지원 |
| bool | bool operator!=(Point& ref); | Point 간 != 연산자 지원 |
| Point | operator+(const Point& ref); | Point 간 + 연산자 지원 |
| Point | operator-(const Point& ref); | Point 간 - 연산자 지원 |
| Point | operator\*(const int num); | Point 간 \* 연산자 지원 |

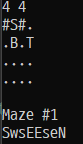
**MazeVertex Class**

일반적인 Graph의 Vertex이다. box, user, 상자를 밀은 횟수, user가 상자까지 걸어간 횟수, 현 위치까지 user의 동작에 대한 정보를 담고 있다. 초기에는 인접행렬을 vertex로 사용하여 Queue에 담았었지만 box, user의 상대위치 또한 저장하기 위하여 MazeVertex Class를 구현하고 Vertex로 사용하였다.

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Description** |
|  | MazeVertex(const MazeVertex& ref) | 복사 생성자 |
| int | pty() | 우선순위 큐에서 enqueue, dequeue 후 힙속성을 회복할 때 사용하는 우선순위인 상자를 push한 횟수이다. |
| int | pty2() | 상자를 push한 횟수가 같을 때, 사용자가 걸은 횟수로 우선순위를 비교하기 위한 함수이다. |

**Maze Class**

문제를 해결하는 class이다. BFS, 인접행렬, 방문여부 확인 등 모든 기능 및 필요한 데이터를 저장하는 가장 중요한 클래스이다.

void BFS()  
BFS 알고리즘을 활용하여 push, walk가 최소가 될 수 있는 경로를 탐색하고 이를 출력한다. 경로를 찾지 못하면 impossible을 출력한다. 초기에는 Box와 User 각 각에 대하여 BFS를 실행하였다. Box부터 Target까지 BFS를 실행하여 상자의 최단경로를 탐색하고, 상자가 이동하기 전 상자의 뒤에 user가 올 수 있도록 상자 뒤까지 user를 BFS를 통해 최단경로로 이동하고 상자를 Push하게 하였다. 그러나 위와 같이 구현할 경우 위와 같은 배치에서 Impossible이 출력되었다. User이 Box를 밀어서 위치를 조정한 후, Target까지 도착할 수 있지만 Start 위치에 상관없이 Box는 Target까지 BFS를 통해 최단경로를 탐색해 EEE로 이동하여야 했기 때문이다. 위와 같은 문제를 해결하기 위해, mark를 4차원으로 확장하여 box, user의 상대 위치에 따라 방문 여부를 표하게 했다. 또한 box, user, 상자를 민 횟수, 걸은 횟수, 지금까지의 경로에 대한 정보를 저장하는 MazeVertex class를 생성하였고 이를 우선순위 큐에 Enqueue하여 우선순위에 따라 Dequeue하고 box, user의 위치를 업데이트 해가며 target까지의 경로를 탐색하게 하였다. 우선순위 큐에서 MazeVertex가 dequeue되면 해당 user의 위치와 박스의 위치에 대한 사방탐색을 진행하여 각각 다음 위치를 계산하고 다음 위치가 미로의 범위 안에 있고, 방문하지 않았으며 돌이 아닐 때, 상자의 다음 경로가 벽이 아니고 user가 상자의 진행방향과 일치하게 인접하면 push하고, 인접하지 않으면 상자까지 walk 할 수 있도록 Queue에 다음 경로에 대한 MazeVertex를 삽입하였다. 조건에 만족하지 않을시 Queue에 삽입하지 않는다.

bool\*\*\*\* mark  
방문 여부를 표시하는데 나타내는 4차원 배열이다. 초기에는 user의 위치만 저장하여 인접행렬과 차원이 일치하는 2차원으로 구현하고 BFS를 진행하였다. 그러나 특이케이스를 해결할 수 없어 user와 상자의 상대 위치에 따른 방문여부를 저장하는 4차원 배열을 구현하였다.

|  |  |  |
| --- | --- | --- |
| **Type** | **Variable Name** | **Description** |
| char\*\* | adjMatrix | 미로에 대한 정보를 저장하는 2차원 배열이다. 돌은‘#’, user는 ‘S’, box는 ‘B’, target은 ‘T’, 이동할 수 있는 길은 ‘.’로 저장한다. |
| bool\*\*\*\* | mark | box, user의 상대위치에 따른 방문여부를 확인하는 4차원 배열이다. |
| PtyQueue<MazeVertex>\* | q | BFS에서 사용되는 우선순위 큐이다. 힙을 기반으로 구현되었다. heap 속성을 회복하는데 push, walk 횟수를 사용한다. |
| Point[] | direct | 2차원 배열 상에서 동서남북 방향으로 이동하기 위해 움직여야 할 x, y 값을 저장한 Point 배열이다. 미로는 수학에서의 좌표가 아닌, array로 구현 되어있다. 그러므로 x는 행, y는 row를 나타내므로 (x+1, y)은 남쪽 이동, (x-1, y)은 북쪽 이동, (x, y+1)은 동쪽이동, (x, y-1)은 서쪽이동이다. |
| string | push\_path | 상자를 미는 방향을 저장하는 변수. |
| string | walk\_path | 사용자가 걷는 방향을 저장하는 변수. |

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Description** |
| void | create\_maze() | 사용자에게 입력을 받고, 입력에 따라 미로를 생성하는 함수이다. |
| void | init\_mark() | box, user의 상대위치에 따른 방문여부를 false로 초기화 하는 함수이다. |
| void | bfs() | BFS를 통해 push, walk가 최소가 될 수 있는 경로를 탐색하고 이를 출력한다. 경로를 찾지 못하면 impossible을 출력한다. |
| bool | check(Point pt) | 해당 좌표가 미로 내에 존재하는 좌표인지 확인한다. |
| bool | is\_path(Point pt) | 해당 좌표가 이동할 수 있는 길인지 확인한다. 돌(#)일 경우 이동할 수 없으니 false를 반환한다. |
| bool | is\_visit(Point, Point) | box, user의 상대위치에 따른 방문여부를 반환한다. |
| bool | can\_push(Point, Point) | box와 user의 다음 위치가 일치하는지 반환한다. 즉, user가 상자의 한 칸 뒤에 위치하여 인접한지 확인한다. |
| void | visit(Point, Point) | box, user의 상대위치에 따른 방문여부(mark)를 true로 바꿔준다. |

**결과**

