

Algorithm

Project REPORT



과목명 | 알고리즘

담당교수 | 최준수 교수

학과 | 소프트웨어학부

팀원 | 권우철(20152875)

맹산하(20171619)

원정희(20165155)

정준권(20152857)

제출일 | 2019.12.08

1. 프로젝트 주제 및 목표

- 목표: C++ 프로그래밍 언어로 ncurses library 를 사용하여 push box game 을 구현한다.
단, 자동으로 수행될 수 있도록 알고리즘을 구현해야 한다.
- 1 단계: ncurses library 함수들을 사용하여 push box map 을 game 화면으로 표시하는 프로그램을 완성한다.
- 2 단계: 1 단계 맵 위의 캐릭터를 표시하고 화살표를 입력 받아 캐릭터가 움직이도록 프로그램을 완성한다. 프로그램은 캐릭터의 이동방향에 벽이 있거나 상자가 두 개 이상 연속으로 있으면 캐릭터가 움직이지 않도록 해야 하고, 이동 방향에 상자가 한 개 있으면서 그 상자 반대편이 또 다른 상자나 벽으로 막혀있지 않으면 상자도 캐릭터와 함께 이동 방향으로 한 칸 이동해야 한다.
- 3 단계: 2 단계 프로그램에서 step 횟수(캐릭터가 이동한 횟수)와 push 횟수(상자가 움직인 횟수)를 화면에 보여주고 모든 상자가 목적지에 도달하면 게임을 끝내고 다음 map 으로 넘어가서 다시 게임을 시작하는 프로그램을 완성한다.

그 외의 기능: 중도 게임 중지(q 버튼 활성화), 게임 리셋 기능(r 버튼 활성화), 게임 자동 해결 기능(a 버튼 활성화)
- 4 단계: 위와 같은 작업을 자동으로 수행할 수 있도록 구현한다.

(3.1) Box 가 1 개인 경우를 해결할 수 있는 수준
(3.2) Box 가 2 개인 경우를 해결할 수 있는 수준
(3.3) Box 가 3 개 이상인 경우를 해결할 수 있는 수준

(제한조건) 게임의 크기가 최대 8x8 으로 간주해도 됨.

2. 프로그램 설명

■ ./pushbox 실행

(1-3 단계)

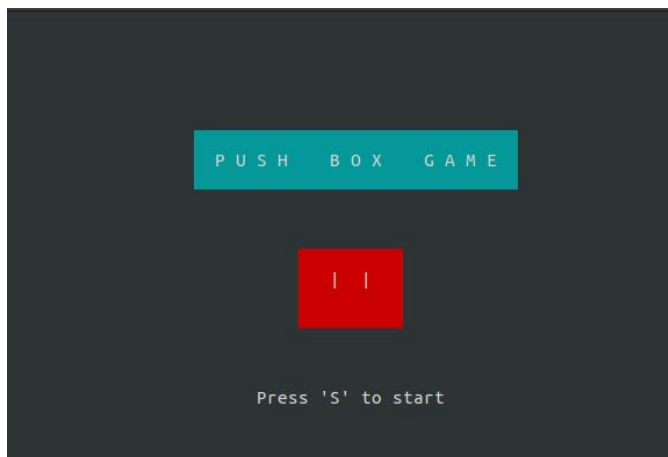
- User 는 키보드 방향키를 이용하여 캐릭터를 움직일 수 있고, 상자를 상자가 도착해야 할 목적지에 이동시키면 맵을 통과할 수 있다.
- 캐릭터 움직이는 방법: 화살표 방향키
- 캐릭터가 움직이는 방향으로 상자는 같이 움직인다. 단 상자 2 개가 겹쳐있거나, 벽에 막혀있을 경우 움직이지 않고, 캐릭터 또한 움직이지 않는다.
- 6 단계를 모두 클리어 할 경우 게임이 종료된다.
- s 버튼을 누르면 게임 1 단계 시작되고, r 버튼을 누르면 게임이 리셋되며, q 버튼을 누르면 게임이 종료된다.

(4 단계)

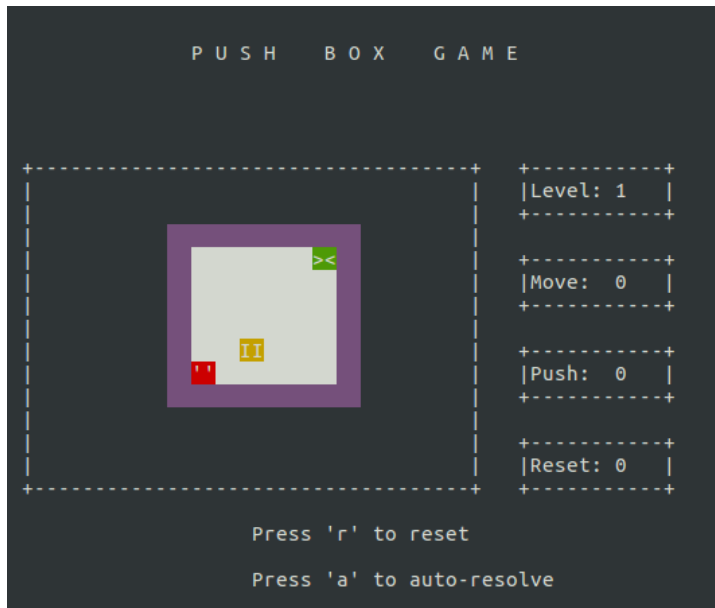
- a 버튼을 누르면 게임 자동 해결 알고리즘이 실행된다.

■ 게임화면

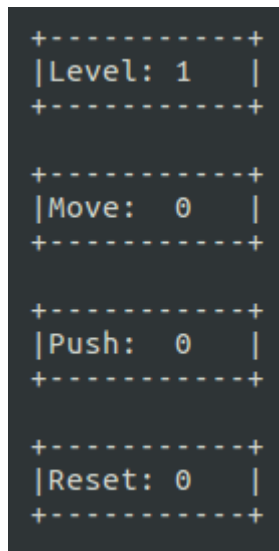
- 실행화면(1)




- 실행화면(2)




- 사용자 인터페이스



: Push(박스가 움직인 횟수), Move(User 가 움직인 횟수), Level(단계)

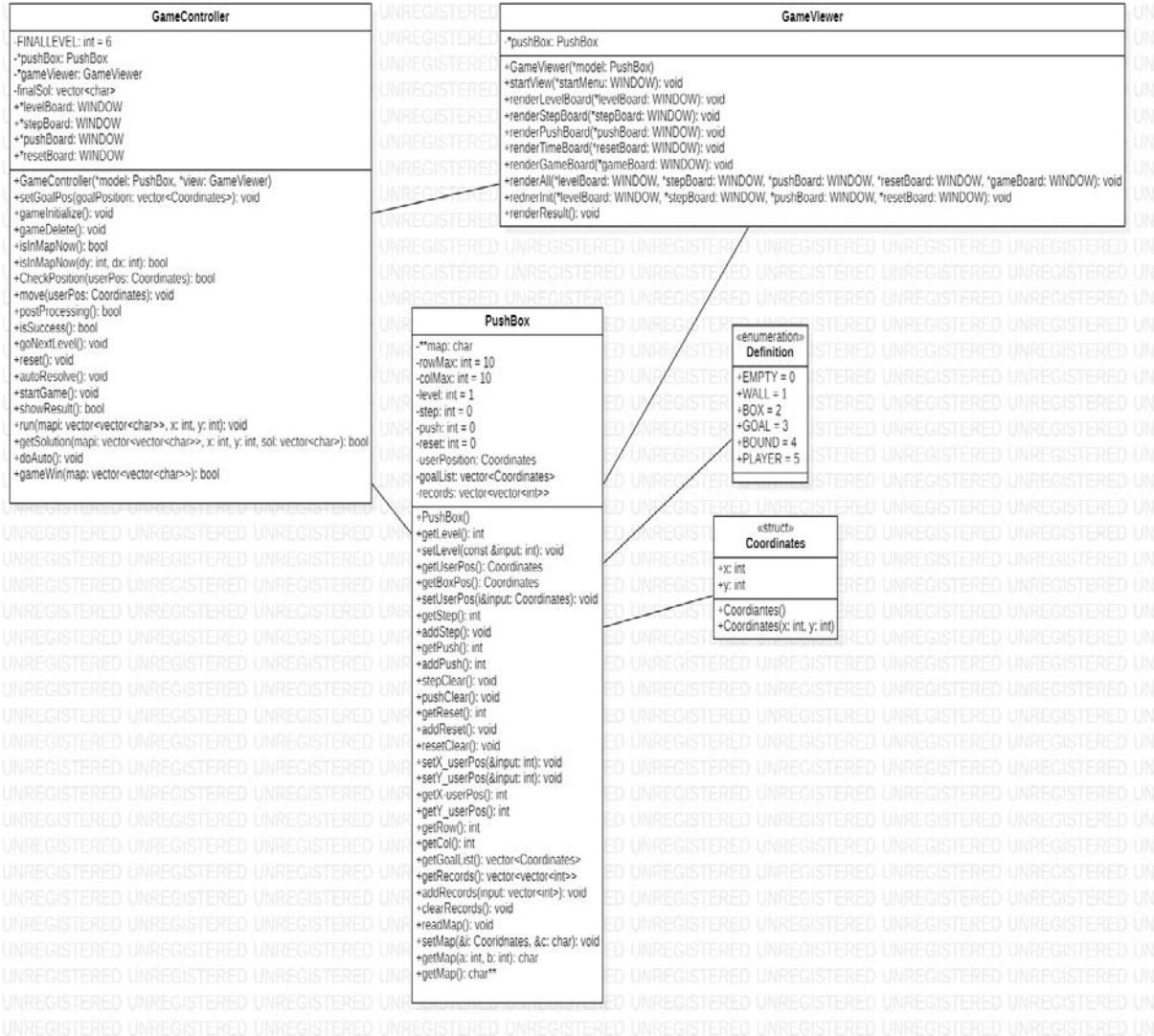
 : 상자가 도착해야 할 목적지

 : 상자

 : 캐릭터

 : 벽

3. 프로그램 내 파일, 클래스, 함수

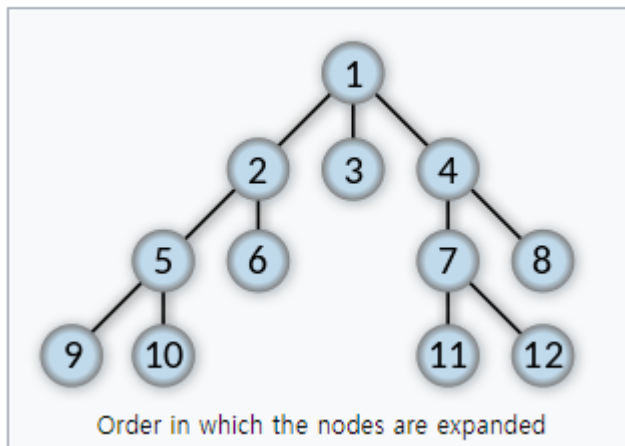


4. 알고리즘

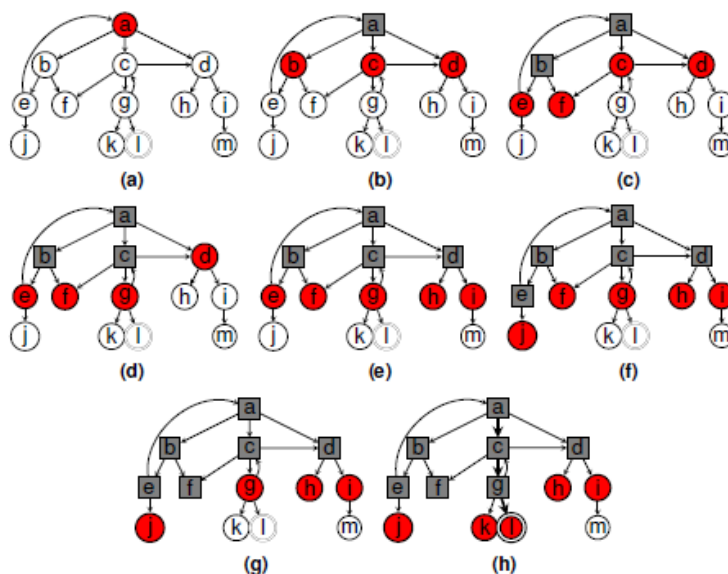
■ BFS

- 맹목적 탐색방법의 하나로 시작 정점을 방문한 후 시작 정점에 인접한 모든 정점들을 우선 방문하는 방법이다. 더 이상 방문하지 않은 정점이 없을 때까지 방문하지 않은 모든 정점들에 대해서도 너비 우선 탐색을 적용한다.
- 트리나 그래프에서 사용하는 탐색 알고리즘이다. 트리 루트에서 인접 노드를 검색하는데, 다음 깊이 전 단계의 모든 노드를 검색한다. 최대한 멀리 검색하는 DFS와 반대의 전략이다.

Breadth-first search

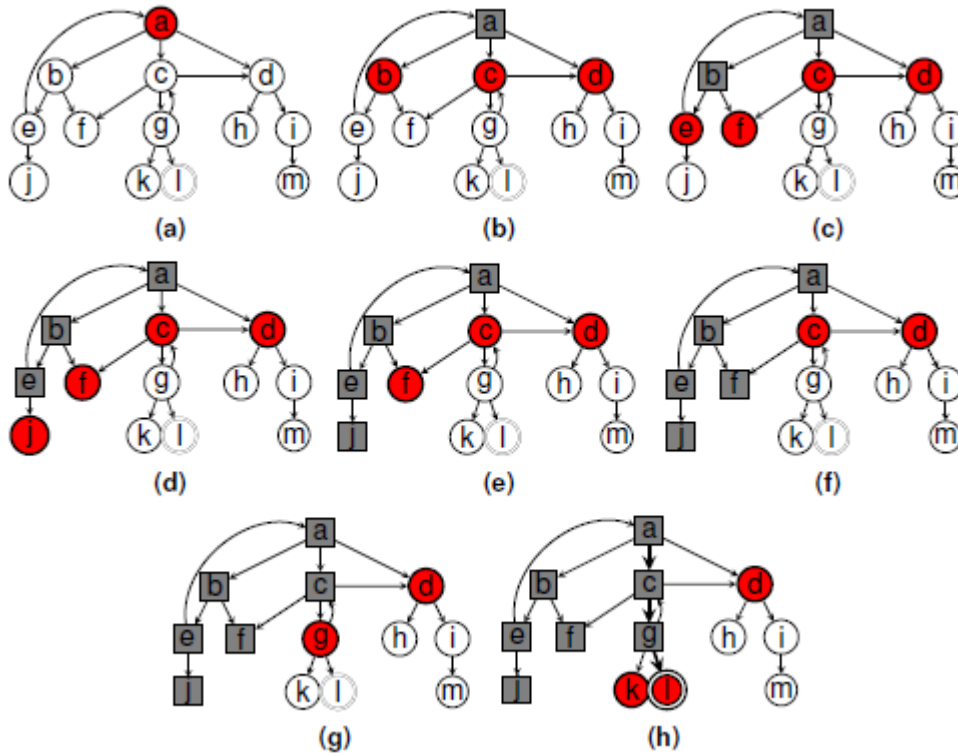


- 그래프내에서 루트 정점부터 탐색을 시작한 후, 같은 레벨에 있는 모든 노드를 방문하기 때문에, 나중에 탐색할 모든 루트들은 메모리에 저장되어야 한다.



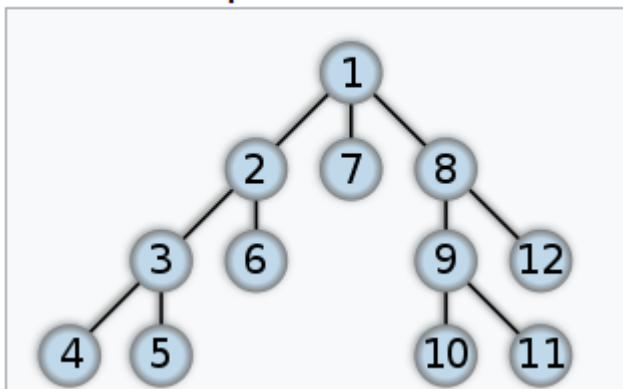
■ DFS

- 데이터 병목 현상을 방지함. 루트로 이동하기 전에 특정 깊이의 모든 정점을 검색하는 대신(BFS 대신) 더 높은 깊이로 노드를 검색한다. 검색할 수 없는 노드까지, 검색을 진행할 수 없는 경우, 아직 탐색되지 않은 노드의 루트로 이동해야한다.



- 탐색 과정이 시작 노드에서 한없이 깊이 진행되는 것을 막기 위해 깊이 제한을 사용한다. 목표노드가 발견되지 않아 부모노드로 되돌아가는 것을 backtracking 이라 한다

Depth-first search



5. 코드

코드가 길고, 파일이 많아 github 사이트로 대체합니다.

Github Homepage: <https://github.com/joshua-dev/2019-2-algorithm-term-project>

6. 실행예시(영상)

오픈소스 디자인 참고(<https://opensrcdesign.com/designDetail/3431>)

7. 소감

- 권우철: 다른 학우들은 이미 알고 지나간 부분을 거듭 물어보고, 인터넷으로 찾아보고, 이해하기까지 많은 어려움이 있었고, 수시로 팀원들에게 도움을 요청하고 물어봤는데, 친절히 답해주고 이끌어줘서 감사한 마음이 컸다. 이번 프로젝트를 하면서 GitHub 사용법에 대해서 상세하게 알게 되었고(여전히 부족하지만), 알고리즘 구현 능력이 다른 사람들에 비해 현저히 낮아 더 보완해야 한다고 생각했다.
- 맹산하: 한 번도 풀어보지 않은 종류의 문제를 해결해봐서 재밌었다. GitHub 를 사용해서 팀원들과 협업하는 법도 배우고 실력도 많이 는 것 같다. 장애물이나, 처음부터 풀 수 없는 게임 등에 대한 예외 처리에 좀 더 신경 썼다면 더 좋은 프로그램을 만들 수 있었을 텐데, 이 점이 살짝 아쉽다.
- 원정희: push box 게임을 푸는 알고리즘 구현 방법에 대해 생각하면서 인간의 머리로 푸는 방법을 컴퓨터로 구현하는 것, 알고리즘을 생각하면서 모든 맵에 예외처리를 생각을 해보았지만, 이 생각들을 프로그래밍으로 옮기지 못한 것이 아쉬웠다. 수업에서 배운 내용을 실제로 해보니 나에게 부족한 점이 무엇인지 알 수 있었다.
- 정준권: 알고리즘 수업을 듣고, 수업에서 배운 알고리즘들을 토대로 자동화 알고리즘을 구현하였다. 구현을 통해서 알고리즘이 어떻게 적용되는지에 대해서 알게 되어서 좋았다. 하지만, 구현한 알고리즘 이외에도 다른 알고리즘에 대한 접근이 미흡했던 것 같다.

8. 개발환경 & 참고문헌

- 개발환경: Linux(C++)
- 참고문헌
 - https://en.wikipedia.org/wiki/Breadth-first_search
 - https://en.wikipedia.org/wiki/Depth-first_search
 - Nils Froleyks, (2016), Using an Algorithm Portfolio to Solve Sokoban, Institute of Theoretical Informatics, Algorithmics Department of Informatics Karlsruhe Institute of Technology