BLG 336E: Analysis of Algorithms II, Spring 2017
Project #1

Lecturers: Zehra Çataltepe, Sanem Sariel
TA: Gönül Uludağ

Total Worth: 10% of your grade

Handed Out: 27.02.2017 - Monday

Due: 14.03.2017 - Tuesday- 23:00 PM

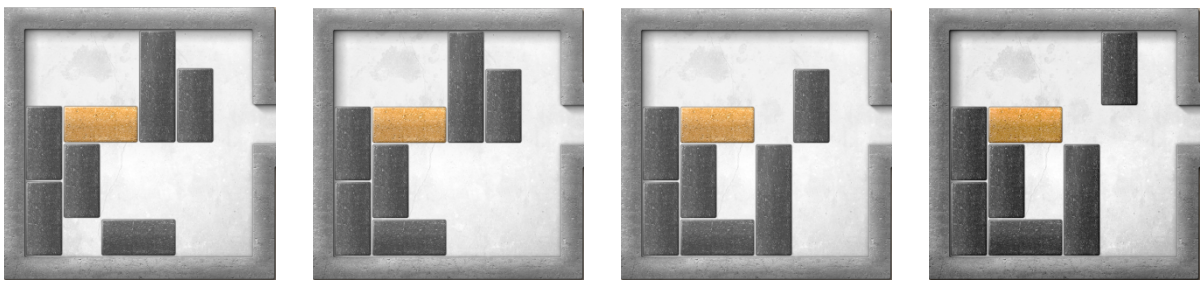
Overview

Breadth First Search (BFS) and Depth First Search (DFS) are well-known graph traverse algorithms.

In this project, you will need to implement the following search algorithms to solve the sliding blocks problem:

1. **Breadth-First Search (BFS)**
2. **Depth-First Search (DFS)**

In this problem, the objective is moving a target block out of the environment from an exit gate. The environment also contains different sized blocks as obstacles. Environment can be represented by a 6 x 6 matrix and the exit gate is on location [3,6]. To clarify the problem, an example illustration scenario of the game is given in the following figure.



(a) (b) (c) (d)

There are some constraints on movement of blocks;

- Blocks are either ***1 x n*** or ***n x 1*** sized rectangles. Here, **n** defines the length of the block and it should be greater than 1 (cannot be 1).
- ***1 x n*** sized blocks can move along right/left directions.
- ***n x 1*** sized rectangles can move along up/down directions.

(a) Formulize this problem in a well-defined form and present your state and action representations in detail.
(b) Run two search versions of both BFS and DFS and analyze the results in terms of:

- the number of nodes generated
- the maximum number of nodes kept in the memory
- the running time.

If any of the algorithms does not last, please specify the reason.

The problem description will be given in a text file (**blocks.txt**) including the properties of the blocks. In this file, each row represents a block. In each row, the coordinates of the bottom left corner of the block, the length of the block and its direction is given (h for horizontal, v for vertical), respectively. The first row always presents the properties of the target block which is needed to be moved out of the environment.

Code (60 points)

You have an input file (blocks.txt) contains the properties of the blocks.

- ✓ [20 Points] Implement BFS.
- ✓ [10 Points] Check the cycle by using BFS.
- ✓ [20 Points] Implement DFS
- ✓ [10 Points] Check the cycle by using DFS.

Your program should compile and run using the following commands:

```
g++ yourStudentID.cpp –o project1
```

```
./project1 bfs input.txt output.txt

./project1 dfs input.txt output.txt
```

Besides the detailed analysis, your programs should present the solution as a sequence of environment states in a file (You can see an example of state format in the input file

(blocks.txt) provided with the homework for your output file). A Matlab code will also be given for you to visualize your solution as a simulation. There is also an online code repository at http://aima.cs.berkeley.edu/code.html which you can use.

Report (40 points)

- ✓ [20 Points] Your program
  - How does your algorithm work? Write your pseudo-code.
  - Explain your classes and your methods. What are their purposes?
- ✓ [20 Points] Show complexity of your algorithm on pseudo-code.
  - [10 Points] What is the extra complexity that is caused by the cycle search?
  - [10 Points] If you use adjacency list representation, how does the complexity of your algorithm change?

Submission:

You should be aware that the Ninova e-Learning System clock may not be synchronized with your computer, watch, or cell phone. Do not e-mail the teaching assistant or the instructors your submission after the Ninova site submission has closed. If you have submitted to Ninova once and want to make any changes to your report, you should do it before the Ninova submission system closes. Your changes will not be accepted by e-mail. Connectivity problems to the Internet or to Ninova in the last few minutes are not valid excuses for being unable to submit. You should not risk leaving your submission to the last few minutes. After uploading to Ninova, check to make sure that your project appears there.

Policy: You may discuss the problem addressed by the project at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. **You should submit your own, individual project.** Plagiarism and any other forms of cheating will have serious consequences, including failing the course.

Submission Instructions: Please submit your homework through Ninova e-Learning System. Please zip and upload all your files using filename HW1 studentID.zip. In the archived file, you must include your completed Report studentId file and all your program and header files.

All your code must be written in C++, and we must be able to compile and run on it on ITU's Linux Server (you can access it through SSH) using g++. You should supply one yourStudentID.cpp file that calls necessary routines for all questions (Multiple files are acceptable, as long as you state the compilation instructions in your report).

When you write your code, try to follow an object-oriented methodology with well-chosen variable, method, and class names and comments where necessary. **Your code must compile without any errors; otherwise, you may get a grade of zero on the assignment.**

If a question is not clear, please let the teaching assistant know by e-mail **uludagg@itu.edu.tr**