BLG336E - Analysis of Algorithms II 27.02.2019
2018–2019 Spring, Project 1

Submission Deadline: 17.03.2019 23:00

Overview

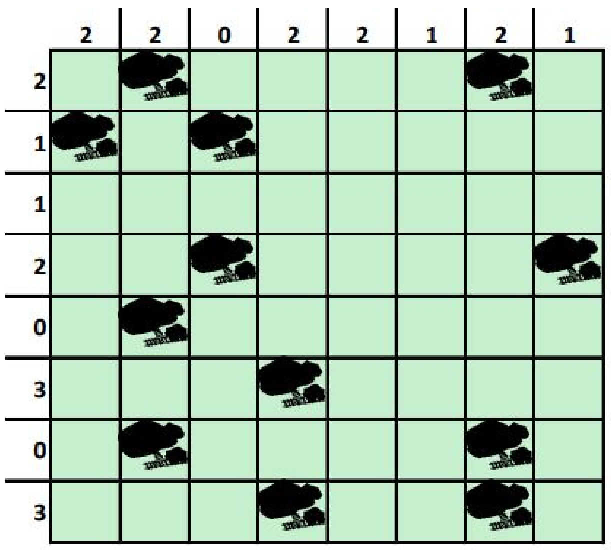
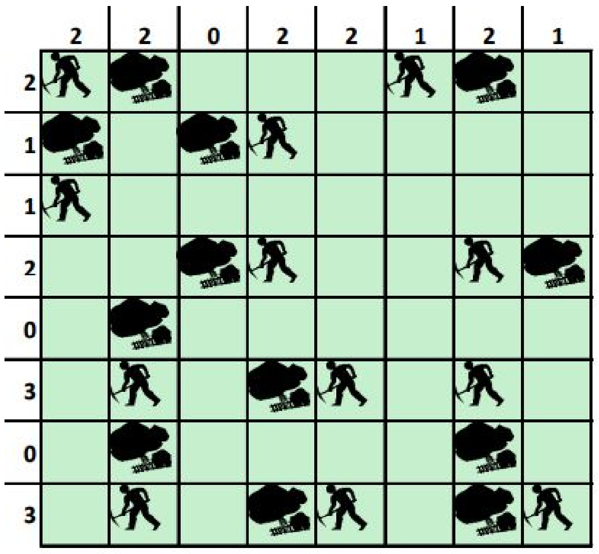
Breadth First Search (BFS) and Depth First Search (DFS) are well-known graph traverse algorithms. In this project, you will need to implement BFS and DFS algorithms in order to solve the given problem.

Gold Miners

In this problem, your job is to place miners next to mining sites.

- Each miner can only be attached to **a single mining site**, there are as many miners as there are mining sites.
- **The numbers across the top and down the side** tell you **how many miner** are in the respective row or column.
- A miner can only be placed **horizontally** or **vertically** adjacent to a mining site.
- Miners are never adjacent to each other, **neither vertically, horizontally, nor diagonally.**
- A mining site might be next to two miners, but is **only one miner can work on a single mining site**.

An example problem and the solution can be seen below.

The problem given in the sample input file.	The solution given in the sample output file.
	

A) Implementation

1) Formalize this problem in a well-defined graph form and present your state and action representations in detail.
   a) Each state can be represented as a node, and placement of a new miner may lead to a new node.
   b) You are not required to optimize the search procedure. Do **not** implement any forward checking mechanisms.
   c) **Upon visiting a node**, **the constraints of the node must be checked**. If the node does not meet the constraints, the node will **not** be expanded.

2) Run both BFS and DFS algorithms **(graph traversal version)**, and print the following information and analyze the results in terms of:
   - the number of the visited nodes
   - the maximum number of nodes kept in the memory
   - the running time

3) The algorithm (BFS or DFS) to be used for search, the input file name and the output file name should be chosen by a command line argument as in the given example. Sample input files and output files are given. When the solution is found, you should write the results to the specified output file in the same format as the input file.

4) You must use a graph representation for the model and solve it using DFS or BFS. Any other method will not be accepted.

5) All your code must be written in C++, and should be compiled and run on ITU's Linux Server (you can access it through SSH) using g++. Otherwise your code will not be evaluated.

Your program should compile and run using the following commands:
(if you have other commands for compilation, you should state them as a comment in your code)

```
g++ sourcecode.cpp –o project1

./project <dfs or bfs> <input-file-name> <output-file-name>

./project1 bfs input.txt output.txt

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

Sample Input File: (input.txt)

- The number of columns and the number of rows are given in the first row. Your code might be tested with the different number of rows or columns
- The numbers on the second line represents the number of miners that can be placed on the column.
- The numbers on the first column represents the number of miners that can be placed on the row
- "." represents an empty location, "s" represents a mining site and "m" represents a miner.
- The columns are separated by tab(\t) characters, and the rows are separated by line feed(\n) characters.

```
8       8
        2       2       0       2       2       1       2       1
2       .       s       .       .       .       .       s       .
1       s       .       s       .       .       .       .       .
1       .       .       .       .       .       .       .       .
2       .       .       s       .       .       .       .       s
0       .       s       .       .       .       .       .       .
3       .       .       .       s       .       .       .       .
0       .       s       .       .       .       .       s       .
3       .       .       .       s       .       .       s       .
```

Sample Output File: (output.txt)

```
8       8
        2       2       0       2       2       1       2       1
2       m       s       .       .       .       m       s       .
1       s       .       s       m       .       .       .       .
1       m       .       .       .       .       .       .       .
2       .       .       s       m       .       .       m       s
0       .       s       .       .       .       .       .       .
3       .       m       .       s       m       .       m       .
0       .       s       .       .       .       .       s       .
3       .       m       .       s       m       .       s       m
```

Sample Output: The numbers may not reflect the real results.

```
> ./project1 dfs input.txt output.txt
Algorithm: DFS
Number of the visited nodes: 21
Maximum number of nodes kept in the memory: 9
Running time: 0.34 seconds
Solution is written to the file.
```

B) Report

1) Present your problem formulation,
    a) Describe state and action representations in detail.
    b) Write your pseudo-code.
    c) Show complexity of your algorithm on the pseudo-code.

2) Analyze and compare the algorithm results in terms of:
    - the number of visited nodes
    - the maximum number of nodes kept in the memory
    - the running time.

3) Why should we maintain a list of discovered nodes? How this affects the outcome of the algorithms?

Note: If you have any questions, please feel free to contact Res. Asst. Çağatay Koç via e-mail (kocca@itu.edu.tr).

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.
- **Academic dishonesty including but not limited to cheating, plagiarism, collaboration is unacceptable and subject to disciplinary actions.**

Submission Instructions:

- Please submit your homework through Ninova e-Learning System.
- You must submit **all your source code in a single cpp file** and **a softcopy report (PDF).** You can define multiple classes in a single cpp file.
- All your code must be written in C++, and should be compiled and run on ITU's Linux Server (you can access it through SSH) using g++. Your code must compile without any errors; otherwise, you may get a grade of zero on the assignment.
- When you write your code, try to follow an object-oriented methodology with well-chosen variable, method, and class names and comments where necessary.
- 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.