# Algorithms I

## Semestral Project Assignment 2024/2025

Jiří Dvorský

April 9, 2025

**Revision history**

April 9, 2025 Initial revision

## Contents



Submit your project solutions by May 18, 2025 23:59 to the following URL

`https://www.dropbox.com/request/ZMYbfJbg12Q3yCxWaDo6`

## General guidelines

- Each student is assigned one assignment. The distribution of assignments among students is included at the end of this document.
- The dates of the defense will be announced in the Edison system.
- The deadline is final and will not be further postponed. Projects submitted after this date will not be considered. The project can of course be submitted and a defense date can be arranged earlier.
- As a project solution, you will submit source code files, header files, project files, etc., in other words everything that is needed for a smooth compilation of the submitted project and nothing extra.
- The source code of your program will include programmer's documentation in the form of documentation comments, which can be processed by Doxygen, see `www.doxygen.org`. It is not necessary to submit the generated documentation. It is sufficient if the submitted archive would contains the `doxyfile` configuration file.
- The assignments might seem hard at first, but don't worry. Any assignment should take no more than one evening for an experienced programmer to solve. First and second year students may take more time, but it should not take tens of man-hours. The same is true for the length of the code produced. If you've already written thousands of lines of code and you still haven't finished, that's not normal. If you find yourself in this situation, it's time to discard that source code and start fresh. The key is to use a pencil and paper and think things through. Try to think through the solution first, drawing various diagrams, sketches of data structures, function calls, and so on. Look through the literature. Then, start writing code.

## Evaluation criteria

The criteria for the evaluation of project solutions are as follows:

1. **Correctness of the solution** The correctness of the solution is a necessary condition. An application that does not provide correct results will automatically be scored 0 points regardless of other criteria. Test data and results are available for each assignment, so the correctness of the solution can be verified.
2. **Choosing appropriate data structures** This criterion evaluates, depending on the specific task, the choice of the appropriate data structure and algorithms for manipulating the chosen data structure.
3. **Decomposition of the problem into smaller units**
   - In the Algorithms I procedural decomposition, i.e. decomposition of solutions into functions, is required. The use of object-oriented programming is optional in this course.
   - The Algorithms II requires an object-oriented design of the solution and the corresponding implementation.
4. **Implementation** This criterion evaluates the separation of declaration and definition of functions or classes in h and cpp files, the use of constants instead of directly written values, the use of function parameters and function results instead of the use of side effects based on global variables. This also includes the level of source code style, for example, indentation of nested constructs, appropriate naming of variables, functions, classes, following the chosen

naming convention[1] of variables, functions, and classes, following the chosen convention for writing code blocks[2], and so on.

5. **Efficiency of the implemented algorithm** The purpose of this criterion is not to force you to implement the best known algorithm in the best possible way. By using this criterion, teachers leave themselves room to possibly lower the score for using a completely inappropriate, nonsensical, confusing algorithm.

6. **Project documentation** Each function, class, class attribute must have at least a short documentation comment in a format that can be processed by Doxygen. Any of the formats supported by Doxygen can be used to write documentation comments. It is not necessary to submit the generated documentation, but it is necessary to submit the `doxyfile` configuration file in order to generate the documentation without problems.

   There is extensive documentation for Doxygen freely available at the URL https://www.doxygen.nl/manual/index.html. For documentation of the semester project, it is advisable to read the following parts of the documentation:

   - "Getting started", https://www.doxygen.nl/manual/starting.html,
   - "Documenting the code", https://www.doxygen.nl/manual/docblocks.html
   - "Doxygen usage", https://www.doxygen.nl/manual/doxygen_usage.html
   - "Doxywizard usage", https://www.doxygen.nl/manual/doxywizard_usage.html
     The Doxywizard program is used to conveniently generate the `doxyfile` configuration file, so you don't have to create it manually.

7. **Citation of resources** No program is built from scratch, not even semester projects. You can use literature, textbooks, source code examples from other courses, and Internet resources to solve projects. In this case, you must indicate that "I took this algorithm from...", "I took this piece of code from...". These possible citations must be placed in the documentation comments. It is probably not necessary to cite Levitin's book, that I "read something about graph depth-first traversal in Levitin's book", or that "we solved this in the seminar". But other sources should be cited.

# 1 Kangaroo

## Problem

Imagine a simple single-player game, which we can call, for example, "Kangaroo". The basis of this game is a sequence $P = p_0 p_1 \dots p_{n-1}$, which contains $n$ digits 0 to 9, the digits can be repeated. Mathematically speaking, $p_i \in \{0, 1, \dots, 9\}$ for $i = 0, 1, \dots, n-1$.

A kangaroo jumps on the sequence $P$ and suppose that the kangaroo is at position $i$. The kangaroo can make the following jumps:

1. it can jump to the position on the left, i.e. to position $i - 1$, if this position exists,

2. it can jump to the position on the right, i.e. to position $i + 1$ if this position exists, or

3. it can jump to any position $j$ such that $p_i = p_j$ for $i \neq j$. In other words, the kangaroo can jump to any other position in the sequence $P$ with the same digit as the one it is currently standing on.

[1] Typically camelCase, PascalCase, less appropriate is the Hungarian notation.
[2] Typically – a left brace after a statement or on a new line.

The player's task is to get the kangaroo from the beginning of the sequence $p_0$ to the end of the sequence $p_{n-1}$ using the three types of jumps mentioned above, and using the *minimum number* of jumps. Your task is to implement a function that, for a given sequence $P$, solves this game.

### Example

- $P = 123415$
  The kangaroo made two jumps – from position 0 he jumped to position 4 (using a jump to a position with the same digit) and then straight to position 5 (jumping to the adjacent position). The kangaroo visited positions 0, 4 and 5.

- $P = 123451$
  The kangaroo made a single jump straight to the end of the sequence. The kangaroo used a jump to a position with the same digit, in this case digit 1. The kangaroo visited positions 0 and 5.

- $P = 12345$
  In this case, the kangaroo made four jumps, all leading to the adjacent position. There is no other option in this case. The kangaroo visited positions 0, 1, 2, 3 and 4.

- $P = 180557994366325331615166132279$
  For a sequence $P$ of length $n = 30$, the kangaroo made a total of six jumps – visiting positions 0, 17, 16, 9, 8, 7, and 29.

Now, it's important to remember that some assignments may have more than one solution with the same minimum number of jumps. It is enough to find one of these solutions to use for the project.

### Remarks

1. Test the implemented function on sample sequences.
2. Sample sequences can be specified as constants directly in the source code, there is no need to enter them from standard input or read them from a file.
3. Print the number of jumps and the positions visited by the kangaroo to the standard output in a suitable form.

## 2 World Map

### Problem

You have been given a map of the world, which is made up of a square grid, where the green squares represent land and the blue squares represent the sea. A sample world map is shown in 1. A contiguous region of land squares that are adjacent to at least one edge forms *continent*. Implement a function or class method that counts all the continents on the given world map.

### Example

The world map in 1 contains a total of 6 continents. But there can also be a world map without a single continent, image 2a, or a world map with a single continent, image 2b.

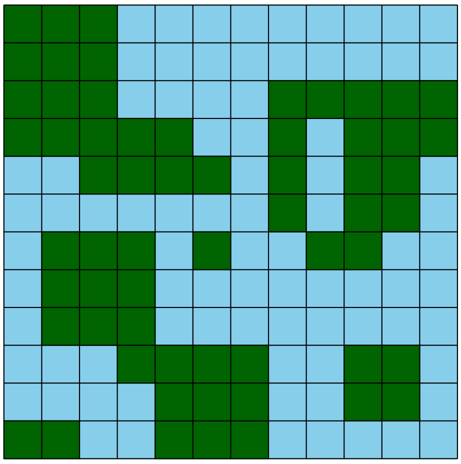
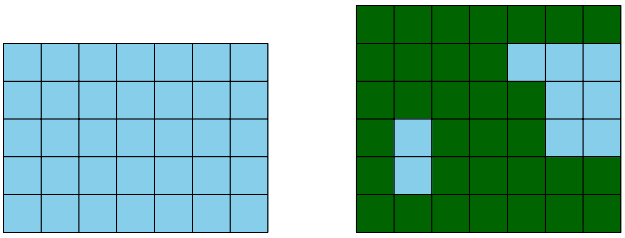


Figure 1: World map



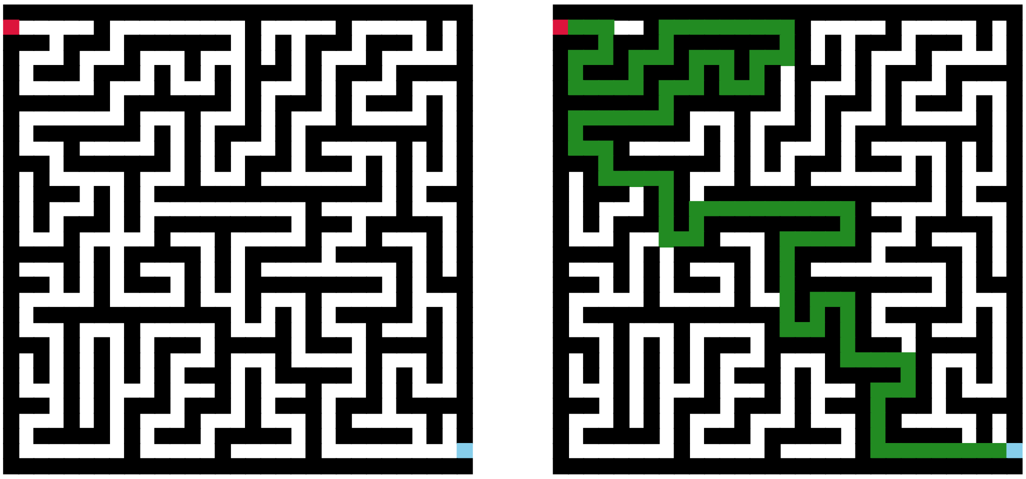
(a) water world (b) single continent

Figure 2: Another sample maps

## Remarks

1. The world map is stored in a text file in the following format:
   - on the first line is the number of lines of the world map $r$,
   - on the second line is the number of columns of the world map $c$, and
   - on the next $r$ lines is a string consisting of $c$ ones or zeros, where one corresponds to land and zero to sea.
2. The world map from the image 1 will be saved as follows:

```
12
12
111000000000
111000000000
111000011111
111110010111
001111010110
000000010110
011101001100
011100000000
011100000000
000111100110
000011100110
110011100000
```



(a) sample maze (b) path through maze

Figure 3: Searching for a path through maze

3. Test the function that was implemented on all three sample maps shown here.

## 3 Maze

### Problem

Imagine you are given a maze. There are two passages to the maze from the outside. One will be considered the entrance and the other the exit. Implement a function that finds the path through the maze from the entrance to the exit.

### Example

The Figure 3a shows a sample maze. The maze forms a square grid. Black squares represent walls, white squares represent paths, red squares represent the entrance, and blue squares represent the exit. The path found, marked in green, in the sample maze is shown in the Figure 3b.

### Remarks

1. The maze is, as already mentioned, created in a square grid and is stored in a text file in the following format:
   - on the first line is the number of lines of the square grid of the maze $r$,
   - on the second line gives the number of columns of the square grid maze $c$, and
   - on the next $r$ rows gives a string $c$ consisting of the digits 0, 1, 2 or 3, where

     0 corresponds to the path,
     1 corresponds to a wall,
     2 corresponds to the entrance to the maze and
     3 corresponds to the exit of the maze.
2. The maze from the image 3a will be stored as follows:

```
31
31
1111111111111111111111111111111
```

```
2000001000000000100000100000101
1110111011111110101011101110101
1000100010001000101010001000001
1011101110101010111010111011111
1000001000100010100010100010001
1111111011111110101110101110101
1000000010001000101000100000101
1011111110101011101011111111101
1000100000101010001010000010101
1110111111101010111011101010101
1010000010001000100000001010001
1011101010111111111111111010111
1010001010000000000010000010001
1010111010111111111011101111101
1000100000100010000010001000101
1111101011111010111110111110101
1000101010001010100000000010101
1110101011101010111111111010111
1000001000001000100010000010001
1011111111111110101010111110101
1000100010000010001010001000101
1110101010111011111011101111101
1010101010100010001000001000101
1010101010101110101111101010101
1000101000100010100010001010001
1110101011101110101110111011111
1000101010001000100010100000101
1011111010111011111010111110101
1000000010000010000010000000003
1111111111111111111111111111111
```

3. Print the solution to the problem, that is, the maze with the path found, in the same format as the input. Denote the found path with the number 4.

## Distribution of assignments among students

| | Student Id | Name | Assignment |
|---|---|---|---|
| 1 | AIT0008 | Ait Massaoud Issam | 1 |
| 2 | BAK0053 | Bakkou Adnane | 2 |
| 3 | BEN0354 | Benyahya Ghita | 3 |
| 4 | BOU0108 | Bouchama Nizar | 1 |
| 5 | BOU0113 | Boussouf Marwane | 2 |
| 6 | DAS0037 | da Silva Figueira Gonçalves Diogo Manuel | 3 |
| 7 | DEH0014 | Dehbi Aya | 1 |

continued on next page...

| | Student Id | Name | Assignment |
|---|---|---|---|
| 8 | EKA0004 | Ekambie Souamy Yohann | 2 |
| 9 | ELA0014 | El Aslaoui Mohamed Rabi | 3 |
| 10 | ELM0018 | El Mhassani Safae | 1 |
| 11 | FER0174 | Ferreira Pereira Guilherme | 2 |
| 12 | GAR0171 | García-Brabo Góngora Alvaro | 3 |
| 13 | GRI0077 | Grigoropoulos Ioannis | 1 |
| 14 | JEO0019 | Jeong Wooseong | 2 |
| 15 | KAI0031 | Kaisbayev Almas | 3 |
| 16 | KIM0094 | Kim Hyunjeong | 1 |
| 17 | KIS0073 | Kisu Yukari | 2 |
| 18 | LAC0099 | Lach-Hab Rayane | 3 |
| 19 | LAC0100 | Lach-Hab Mohammed | 1 |
| 20 | LUQ0006 | Luque Núñez Alejandro | 2 |
| 21 | MOR0288 | Morchid Saad | 3 |
| 22 | MYA0009 | Myaris Stylianos | 1 |
| 23 | NTS0006 | Ntsouori Woue Merlycia Estyna | 2 |
| 24 | PAK0023 | Pakingan Mark Daniel Gantiles | 3 |
| 25 | RED0024 | Bouhraoua Reda | 1 |
| 26 | SAL0205 | Salai Meryem | 2 |
| 27 | SHA0082 | Shahraeini Seyed Iman | 3 |
| 28 | SHA0083 | Shalhawi Sedra | 1 |
| 29 | TAH0014 | Tahir Chaimae | 2 |
| 30 | TBA0007 | Tbal Mohamed Lamine | 3 |
| 31 | THA0034 | Thai Thi Thuy An | 1 |
| 32 | TCH0010 | Tchaiwou Tchemtchoua Winnie Lena | 2 |