ENGR151 — Accelerated Introduction to Computer and Programming

Lab 3

Manuel — UM-JI (Fall 2021)

Goals of the lab

- Arrays and Matrices
- Function
- Recursion

1 Background

Last weekend Haruka, Kana, and Chiaki went to an amusement park. They really enjoyed it, but what they preferred were all the applications they could find to their computer and programming course. And best of the best, they could practice recursion!

2 Tasks

The three favorite attractions they they wanted to test above all were (i) the maze, (ii) the escape room, and (iii) the toy grabbing machines. The first place they spot on their way is the maze...

Mandatory part

2.1 The maze

At the entrance of the maze, a sign was announcing that k treasures had been hidden in it, and the first person to find them all would get the entrance ticket to the amusement park reimbursed.

However, Haruka somehow figured out that up to now nobody had ever won the prize. Hence she started to question how real this treasure hunt was, some treasures maybe being inaccessible. In order to find out the truth, the three sisters decided to have a ride on the big wheel and take pictures of the maze from atop. Unfortunately it was far too large to be processed manually, and they could not figure out the number of treasures that could be found if the search starts from the entrance. After ensuring it was not Jane's nap time they phoned her to get her advice.

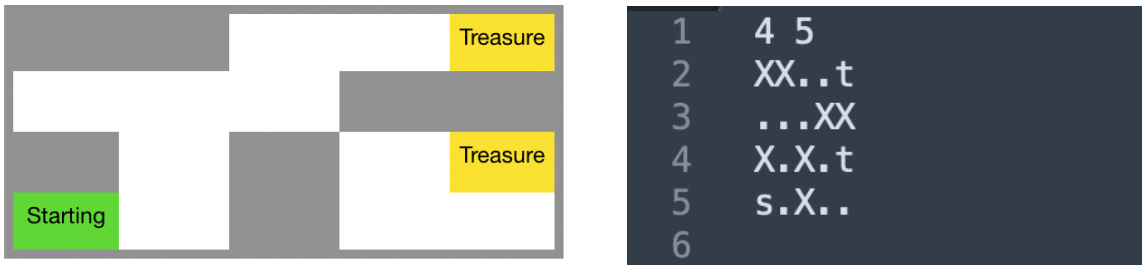
Jane's idea is to proceed in two steps: (i) based on the photos they have taken from the wheel, build a map of the maze; and (ii) use Depth First Search (DFS) algorithm to search all accessible locations inside the maze.

2.1.1 Map of the maze

The map is written in a simple text file composed of a first line showing the size of the maze x and y. The rest of the file contains x lines each with y columns. Those remaining lines can only feature four types of characters:

- . to show a path, i.e. can walk there;
- X to represent a wall, i.e. the path is blocked;
- s to specify the starting point;
- t to locate a treasure;

In the three sisters' model a player can only walk following the four directions, up, down, left, and right. For example, Figure 1 shows a maze and its corresponding map file. The grey blocks indicate walls that block the way, and in particular we see that one of the two treasures is inaccessible.



(a) Maze map. (b) Input file.

Figure 1: The input of the maze

Help the three sisters to

- Design an algorithm to read a map file; (3 min – 3 students)
- Write the corresponding function to read a map file; (5 min – 2 students)

Present and demonstrate to the three sisters the result of your work. (2 groups – 5 min each)

2.1.2 Searching treasures

Now that they can parse a map file the sisters need to tackle the search problem. Following Jane's advice the look into the DFS algorithm. They expect to output how many treasures can be accessed when starting from the entrance of the maze.

When they realise DFS is a recursive algorithm they remember what Jane taught them a few days before.

When to use recursion?

Sometimes both iterative and recursive algorithms can be applied and it might be hard to choose which to implement. While there is no definite answer, a good start is to evaluate the following three points.

- The target is clearly defined;
- The number of steps necessary to complete the work is unknown, i.e. loops would be hard to write in a clean way;
- At each step, the process to run is almost the same;

Help the three sister to

- Verify that the task to complete falls under the case "When to use recursion?" as described by Jane; (3 min – 4 students)
- Find and read online English resources clearly describing DFS; (5 min – 3 students)
- Implement DFS and return the expected output; (10 min – 2 students)

Present and demonstrate to the three sisters the result of your work. (2 groups – 5 min each)

Optional part

Haruka, who loves when a work is fully completed, suggests to write a function to generate a random map files. Her goal would be to generates maps where it is ensured that all treasures are accessible.

Help the three sisters to

- Write a function to generate a map where the size and location of the various elements are random;
- Write to test whether all treasures can be found in a given map;
- Write a function which write a map into a text file;
- Think of a solution where the accessibility of the treasures is tested and ensured while building the map;

As Haruka s still busy completing her perfect maze program, Chiaki can not wait anymore: she wants to move to her favorite attraction, the escape game.

2.2 Escape room

A few minutes later, Haruka joins Kana and Chiaki in the waiting line. They are very excited, hoping for a difficult challenges. Unfortunately the first few rooms are to easy for them as within a few minutes they have already reached the last one. As they enter the last room its door automatically locks behind them. After a quick search they found a few clues hinting for a password of length n to unlock the door. A bit more thinking lead them to the conclusion that at most m characters are likely to appear in this pin code.

As Kana takes her phone to called Jane to the rescue she realises that there must a mobile phone jammer in the escape room. Indeed, she does not have any signal. The three sisters are on their own and they must solve that problem themselves without Jane's help!

2.2.1 Preparation

As Haruka has no idea how to proceed, she offers to help with the input output format. She already worked on it for the maze, so it should not be too hard...

Help Haruka to

- Write a function which reads a file composed of two positive integers n and m on the first line, representing the length of the password and the number of characters that can appear in the password, respectively; The second line is composed of a string of m characters long, listing all the possible symbols for the pin code;
- Write a function which takes a list of strings as input and dumps them into a text file;

2.2.2 Permutations

While Haruka works on the input and output file formats, Kana and Chiaki figure out that there is a total of $\mathcal{P}_m^n = \frac{m!}{(m-n)!}$ possible passwords. For example, if they looked at all the permutations of length three, composed of the four characters "UMJI":

- JMU
- MUJ
- MIU
- JUM
- UMJ
- UIM
- MJU
- IMU
- MUI
- UJM
- IUM
- UMI

- IJU
- JUI
- UIJ
- IJM
- JMI
- MIJ
- JIU
- IUJ
- UJI
- JIM
- IMJ
- MJI

When Haruka has completed her part the two other sisters are still struggling to solve the problem. If at least the could call Jane... Haruka tells them not to despair and asks if they tried to a more simple problem. For instance if they could list all the permutations of length m, maybe afterwards it would possible to adjust their idea to only list the ones of length n?

Their initial idea is rather simple: if they already have a permutation P of length $n-1$, then they can easily insert an n^{th} element (i) between any two other elements of P, (ii) at the very beginning or (iii) very end of P. They test their idea on a toy example, generating all the permutations of length three.

$$\mathbf{1} \Longrightarrow \begin{matrix} 1\mathbf{2} \\ \mathbf{2}1 \end{matrix} \Longrightarrow \begin{matrix} 12\mathbf{3} & 21\mathbf{3} \\ 1\mathbf{3}2 & 2\mathbf{3}1 \\ \mathbf{3}12 & \mathbf{3}21 \end{matrix}$$

Here, their idea consists in solving the problem using *Divide and Conquer*. In fact, they face a problem of large size that is impossible to solve directly. However, they realise that they can divide the problem into several subproblems that are similar to the original problem, but of smaller size. In this way, they can keep "dividing" the problem again and again into smaller and smaller subproblems that are easier and easier. When the last one is very simple to solve, then it is possible to eventually solve the original one.

Kana notes that it should be possible to write this algorithm in an iterative way, i.e. using loops instead of recursion. She argues that it should be possible to implement that idea without recursion since in that case they exactly know how many steps are necessary. To confirm her idea she recall Jane's lesson on "When to use recursion".

Help the three sisters to

- Develop a clear recursive algorithm to generate all the permutations of a list of m elements;
- Develop a clear iterative algorithm to generate all the permutations of a list of m elements;
- Implement both algorithms;
- Compare the running time and memory usage for both implementations;

2.2.3 Subset of permutations

The three sisters now hurry up due to the time constrain of the game, but they are almost done. They now only need to adjust the recursive algorithm to generate permutations such that it "skips" some insertions, directly "jumping" to the next one.

Help the three sisters to

- Take a simple toy example to understand how to skip characters in the recursive process;
- Implement the final version to generate all the permutations of length n from a string with m characters;

The three sisters have never lost in any escape game, ensure they do not fail this time!

2.3 Toy grabbing machine

After their tight and stressful escape in the previous game the sisters need a bit of rest. While Haruka and Chiaki eat a candy floss, Kana releases the pressure on a toy grabbing machine. And the is very, very good at it...

In just a few minutes she managed to grab n toys. Not knowing what to do with them she decides to offer them to all her friends. As each toy comes with a tag price she decides to sort them from most expensive to cheapest. She plans to give the most expensive to her best friend, the second most expensive to her second best friend, and so on.

2.3.1 Input and output

Hakura immediately notice the need for some input and output file to store the initial and the sorted lists, respectively. As she has performed such a task several times already during they stay at the park, she is now a real expert at it.

Help Haruka to

- Write a function to read a file where the first line is a positive integer n representing the number of toys grabbed by Kana, and the second contains a space separated list of n prices;
- Write a function to write a file composed of a single line containing a list of space separated numbers;

2.3.2 Using Quick Sort algorithm to solve the problem

Based on what the three sisters have learnt, there exist many ways to sort a list. However they remember that one day Jane mentioned that quicksort was among the best ones. And as it is a recursive algorithm this is exactly what they need!

After a quick search they understand how quicksort works:

(i) Pick an element from the unsorted array, and call it the *pivot*. Any element can be taken as the pivot.

(ii) Rearrange the array such that all the elements with value larger than the pivot pushed behind it, and all the elements with value smaller than it, before. As a result the pivot quickly takes its final position.

(iii) As the array is now partitioned into two unsorted subarray, recursion can easily be applied by repeating steps (i) and (ii). Eventually the whole array will be sorted.

Help the three sisters to

- Search online for the most famous sorting algorithms;
- Make sense of Figure 2 that they found on Wikipedia; In other words, based on this toy example, explain how quicksort works;

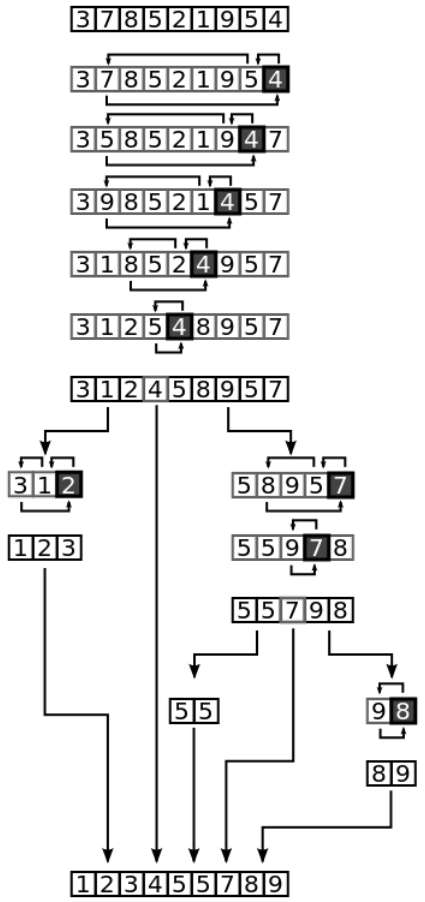


Figure 2: Quick Sort Process

- Write a clear algorithm for quicksort;
- Implement quicksort and connect it to the input/output functions written earlier by Haruka;

After this busy weekend the three sisters were very happy to enjoy a good night rest before a new intensive week in their university. Jane received the best toy, and you what did you get? If you did not get anything, do not worry, a good grade in VG101 is always better than a toy!