

Big Homework 2

Data Structures and Algorithms

Graphs & Trees

General mentions

- For this project you will work either in **teams of 2 people or alone**.
 - The homework will be uploaded on the **Moodle** platform (fils.curs.pub.ro), for Hw1 assignment. If you encounter problems with the platform, contact by email your lab assistant.
 - The homework must be submitted by **15.05.2023, at 8:00**. No late submissions will be accepted.
 - You will be asked details about your solutions during the following lab. Projects **not presented at the following lab won't be graded!**
 - The final submission will contain an archive named Student1FamilyName_Student1Name_Student2FamilyName_Student2Name_HW1 with:
 - **the source files** of your project (.cpp and .h), **grouped in separate folders for each exercise** (and not contain the object files (.o), the codeblocks files (.cbp) or executables (.exe))
 - **a README file** in which you will briefly specify all the functional sections of the project, together with instructions for the user; additionally, if you have parts of the homework that don't work, you may offer solution ideas for a partial score on these sections. You will have to explain your propositions at the presentation lab.
 - For all questions regarding the project, communicate on Microsoft teams with your lab assistant (recommended to create a new post on the Teams group so that everyone can see the question & the answer). Don't forget to mention @General (to notify everyone) or @TeacherName.
 - Warning: we will use plagiarism detection software on your submissions (Stanford's tool Moss). Copied homework will be marked with **0 points**.
- ! Observation: You can use the graphs and trees that we used in class (they will be in headers (.h) and generic classes (template)). Alternatively, you can use standard C++ implementations, BUT not other custom implementations from the internet.

Tasks:

1. Graphs (4p) - Ancient alphabet decoder

Sydney Fox (world famous teacher of ancient history) and her assistant, Nigel Bailey, found a book written in an unknown language, but using the same letters as the English alphabet - no uppercase, only lowercase (example a, b, c). The book contains a brief index, but the order of the index words is different from the English alphabet. The treasure hunters attempted to use this index to determine the order of characters in the unknown language but failed. Write a program to help treasure hunters to determine the **order of characters** in the unknown language and display this order in a file. You must use topological sorting with **graphs**. Display the built graph used for solving the problem.

Input data: index.in

In the file "index.in" there are at least 1 and at most 50 words followed by a line containing the character "." (point). Each word has a maximum of 10 characters. The words of an index contain only the lowercase characters of the English alphabet and appear in ascending order of the unknown alphabet. Not necessarily all lowercase letters appear in the index, but an index will result in a unique sequence of characters that appear.

Output data: index.out

In the file "index.out" we write the unique sequence of characters that appear in "index.in", in the correct order of the unknown alphabet.

Example:

index.in	graphe	index.out
ion ana adonia doina doinn ddan ddao .	<pre> graph TD i((i)) --> a((a)) a --> d((d)) a --> n((n)) d --> o((o)) n --> o </pre>	ianod
b .	<pre> graph TD b((b)) </pre>	b
xwy zx zxy zxw ywwx .	<pre> graph LR x((x)) --> z((z)) z --> y((y)) y --> w((w)) </pre>	xzyw

Points:

0.5p handling input

1p building the graph correctly + displaying it

2p topological sort algorithm

0.5p correct output + validations

Useful references:

fclose: <http://www.cplusplus.com/reference/cstdio/fclose/>

fopen: <http://www.cplusplus.com/reference/cstdio/fopen/>
 fprintf: <http://www.cplusplus.com/reference/cstdio/fprintf/>
 fscanf: <http://www.cplusplus.com/reference/cstdio/fscanf/>
<http://www.cplusplus.com/doc/tutorial/files/>
<http://www.cs.washington.edu/education/courses/cse373/02au/lectures/lecture191.pdf>

2. Quad Trees (6p)

A Quad tree is used frequently in image processing. Each image can be divided into four quadrants. Each quadrant can also be divided into other four quadrants, etc. In a Quad tree, the image is represented as a parent node, while the quadrants are represented as four child nodes. If the whole image has only one colour, then the quad tree contains only one node. A quadrant has to be divided only if it has pixels of various colours. This means that the tree might not have a balanced form.

A graphical designer works with images of $32 * 32$ units, this means 1024 pixels/ image. One of the operations he has to make is overlapping two images, with the purpose of obtaining a new black-and-white one. In the new image, one pixel is black if it is black in at least one of the two initial images, otherwise it is white. He wants to know how many black pixels he will have in the new image, before doing the overlapping operation, as images with too many black pixels are expensive to be printed. You have to help him by writing a program which calculates how many black pixels the final image will have. Unfortunately, he does not trust anybody, so he does not want to give you the initial pictures, only the Quad representations of them.

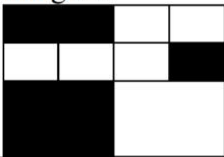

Input for your program: a pair of strings representing the two images. Thus, an image is represented by the preorder traversal of its Quad tree: letter „p” means a parent-node, letter „b” means a child-node for a black quadrant, letter „w” means a child-node for a white quadrant. The pair of strings can be read from the console or from a file.

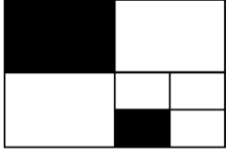
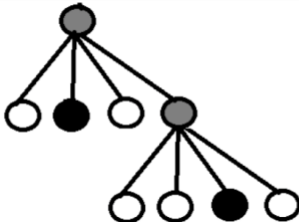
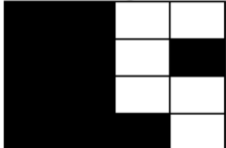

Output for your program: preorder traversal of the Quad tree of the final image and the number of black pixels contained by it, based on the black pixels contained by each initial image. Pay attention to overlapping black pixels of the initial images.

Explanations:

Order of representing quadrants:

2	1
3	4

Image 1: 	Quad tree for Image 1: 	Preorder representation of image 1: ppwwwbpbbwwbw	Nr of pixels in image 1: 448
Image 2:	Quad tree for Image 2:	Preorder representation of	Nr of pixels in image 2:

		image 2: pwbwpwwbw	320
Final image: 	Quad tree for final image: 	Preorder representation final image: ppwwwbbbppwwbw	Nr of pixels in final image: 640

Tasks:

- (1p) Find a suitable representation for a Quad tree.
- (1p) Write a function which transforms the preorder traversal you give as an input into the representation you have chosen for Quad trees.
- (1.5p) Write a function which receives as arguments two Quad trees and transforms them into another Quad tree, for the final image, based on the algorithm described in the problem.
- (1p) Write a function which calculates the pixels for a given Quad tree representing an image.
- (1p) Test your program with various inputs. Display the final Quad tree representation and the number of pixels.
- (0.5) Create an interaction menu which allows the user to test the above functionalities (a-e). The program should allow testing multiple functionalities in the same execution.