State Final Examination (Sample Questions)

Spring 2023

1 Sorting (3 points)

1. Describe an algorithm that sorts (i.e., permutes to a non-decreasing order) a sequence $x_1, \dots, x_n$ of pairwise distinct items using binary search trees.
2. What changes if we allow multiple occurrences of the same item?
3. What conclusion can we draw about the minimum possible time complexity of operations with a binary search tree? You can assume that keys in the tree can only be compared.
4. What is the time complexity of your algorithm from subtask 1 if we use an AVL-tree and items will be strings of length L?

2 Object-oriented design (3 points)

Assume we are implementing a bitmap image processing application. The user should be able to have multiple images open at once. The user opens a new image in the application's user interface (UI) by selecting its file name – the application decides on image format based on the file extension. With an open image, the user can gradually select some of the filters offered – in the application UI, the user always selects a filter and its specific settings, and it is then saved in a list of selected filters. At the end, the user will have the option to apply the selected filters together. Each open file can be saved by the user at any time either in the original or another supported bitmap format. For sake of simplicity, assume that one image format corresponds to exactly one file extension (i.e. for JPEG format only the extension `.jpg` is valid, and additional `.jpeg` is not), and that all filters can freely change the image data, but the pixel dimensions of the image are always preserved.

We have prepared a skeleton of the main classes of our program in C# (see below), where the `Image` class represents the loaded image data (we will remember other metadata like the original file name etc. elsewhere; note: `[,]` in C# means a two-dimensional array), and the `Program.Main` method represents a simple scenario of using the other classes:

```
struct Color { public byte R, G, B; }

static class Filters {
        public static void ApplySharpness(Image image, float amount, float radius) { ... }
        public static void ApplyBrightness(Image image, float amount) { ... }
}

abstract class Image {
        public Color[,] bitmap = null;
        public abstract void Load(string fileName);
        public abstract void Save(string fileName);
}

class PngImage : Image {
        public override void Load(string fileName) { ... }
        public override void Save(string fileName) { ... }
}

class JpegImage : Image {
        public override void Load(string fileName) { ... }
        public override void Save(string fileName) { ... }
}
```

```
static class ImageLoader {
        public static Image Load(string fileName) {
                Image image;
                if (fileName.EndsWith(".png")) {
                        image = new PngImage();
                } else if (fileName.EndsWith(".jpg")) {
                        image = new JpegImage();
                } else {
                        throw new ArgumentException();
                }
                image.Load(fileName);
                return image;
        }
}

class Program {
        public static void Main() {
                var fileName = "a.jpg";
                var image = ImageLoader.Load(fileName);
                Filters.ApplySharpness(image, 9.0f, 1.3f);
                Filters.ApplyBrightness(image, 1.7f);
                image.Save(fileName);
        }
}
```

Reimplement the entire object design to better support implementing, maintaining, and further extending the application described above. Write the new code skeleton in C#, C++, or Java. Focus on the design of types and their relationships (write everything in the syntax of the chosen language), declaration of methods, and only key properties and fields. You do not have to provide an implementation for most methods – just edit the `Program.Main` implementation to make an appropriate use of your OO design. In your solution, please also implement the code that will represent the sequence of steps in your design from having the image file name to selecting the file format and loading the image in that format.

In your proposal, consider not only the easy possibility of implementing all aspects of the application usage scenario outlined above, but also the fact that in the future we want to add a large number of different filters – but we always plan to add the implementation of a new filter to the main source code of the application, we do not want to ever support new filters delivered as plugins. Therefore, in your solution, consider only the batch application of selected filters; you do not have to deal with any infrastructure for general filter description.

In contrast, we want to add support for new image formats using plugins. Assume that we will implement each newly supported image format in the form of a new plugin to our application (which we'll load as a dynamically linked library at runtime) – so we want the process of loading images from a file to be flexible enough to allow us to provide a reference to a new file format that a plugin will provide (do not deal with actual plugin loading, or support for loading dynamically linked libraries, or directly searching for a description of the format in the plugin).

3 Linear maps (3 points)

1. Define the term *matrix of a linear map with respect to given bases.*
2. Formulate the theorem on the matrix of the composition of linear maps.
3. Consider the linear map $g\colon \mathbb{Z}_5^2 \to \mathbb{Z}_5^2$ such that $f \circ g \circ h = id$ and we know about linear maps f and g that

$$f((1,0)^T) = h((0,1)^T) = (3,4)^T,$$
$$f((0,1)^T) = h((1,0)^T) = (4,1)^T.$$

   Find the matrix of g with respect to the canonical basis.

4 Planar graphs (3 points)

1. State Euler's formula for planar graphs (about the number of vertices, edges, and faces).
2. Decide whether the following planar graph G can exists: G has 40 vertices, 80 edges, 5 components, and moreover, G contains no C_3 as a subgraph.

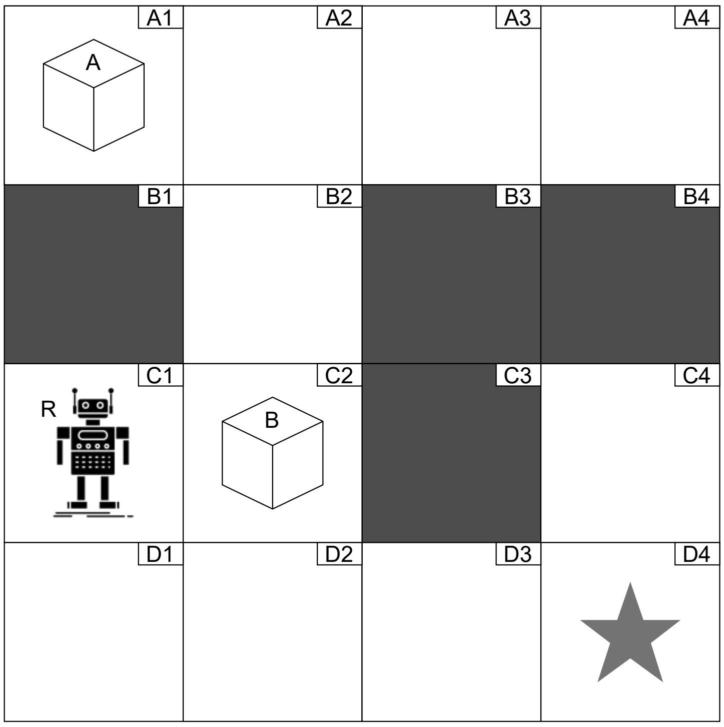
3. Determine the maximum nuber of edges of a planar graph with 40 vertices, 5 components and containing no C_3 as a subgraph.

5 Planning (3 points)

Assume a robot that can execute the following actions:

- Move – move to a neighboring square if there is not a wall, if the robot carries a box, it additionally cannot move to any square that contains a box
- Pick Up – pick up the box at the robot location, the robot cannot pick up a box if it is already carrying a box
- Put Down – put down the carried box at the robot location

The initial state is given in the image below. The robot is denoted by the letter R, two boxes are denoted by A and B. The dark squares are walls, the light squares are free. The goal is to move the box A to the square denoted by the star. You can assume that the state is described by predicates `neighbor`(X, Y) expressing that squares X and Y are neighbors and there is no wall on Y (i.e. it is possible to move from X to Y), `at`(X, B) expressing there is a box B at square X, and `robot`(X) expressing that X is the current position of the robot. The signs in the top right corner of every square give its name, which you can use to refer to that square in your solution.



1. Formalize the problem given above as a planning problem. Use the predicates given above and potentially other predicates you define to describe the initial state, goal state, and the actions. You do not have to explicitly mention the predicates that do not change (e.g. `neighbor`) in the definition of the initial state.

   Hint: You will have to define two different move actions – one for moving while carrying a box, the other for moving empty-handed.
2. Describe forward and backward planning.
3. Find a plan that solves the problem above.

6 Ensembles (3 points)

1. Explain the bagging and boosting techniques.
2. Describe the random forest training algorithm.
3. Give an example of a boosting technique. Briefly explain its main idea.

7 Binary classification (3 points)

Assume we have a binary classifier with sensitivity 0.8 and specificity 0.9. The classifier was applied to 10000 data instances out of which 500 belong to the positive class.

Note: Sensitivity is also called *recall* or *true positive rate* and specificity is also called *selectivity* or *true negative rate.*

1. Compute the full confusion matrix of the model.
2. Assume that the classifier predicts for a specific instance that it is positive. What is the probability that it is indeed positive?
3. What is the accuracy of the classifier?

8 Linear regression (3 points)

Assume we have a linear model $m(x) = 2x_1 - x_2 + 3$ and data given in the table below.

x_1	x_2	y
1	1	6
2	1	3
1	4	-1
3	2	6

1. Compute the mean squared error of the model.
2. Describe the stochastic gradient descent method.
3. Make one step of the stochastic gradient descent method (starting with the model defined above) with batch size 4 (i.e. all the data are in a single batch) and learning rate $\alpha = 0.1$. Our goal is to minimize the MSE of the model. What is the new model after this single step?

9 Ensembles (specialization question – 3 points)

1. Explain the bagging and boosting techniques.
2. Describe the random forest training algorithm.
3. Give an example of a boosting technique. Briefly explain its main idea.

10 Binary classification (specialization question – 3 points)

Assume we have a binary classifier with sensitivity 0.8 and specificity 0.9. The classifier was applied to 10000 data instances out of which 500 belong to the positive class.

Note: Sensitivity is also called *recall* or *true positive rate* and specificity is also called *selectivity* or *true negative rate.*

1. Compute the full confusion matrix of the model.
2. Assume that the classifier predicts for a specific instance that it is positive. What is the probability that it is indeed positive?
3. What is the accuracy of the classifier?

11 Noisy channel model in NLP (specialization question – 3 points)

Describe the noisy channel model as used in NLP, and show how Bayes rule is used in it. Explain the main benefit of applying Bayes rule (hint: argmax). Show how the noisy channel is used in at least two NLP applications.