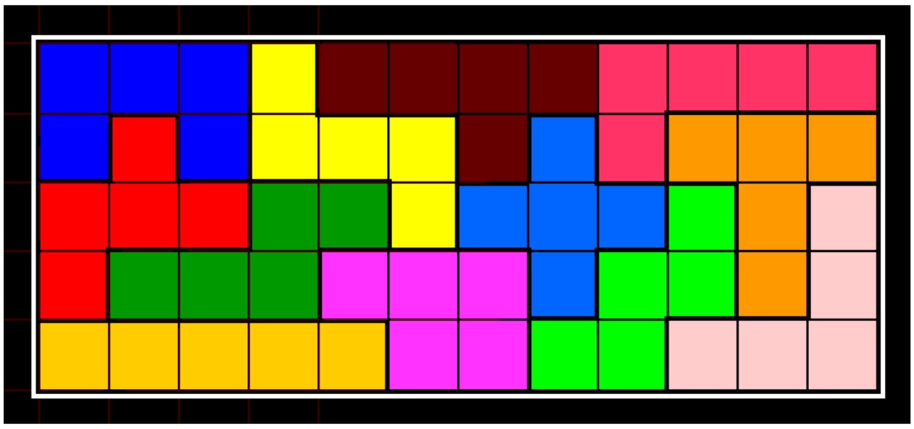
Project Manual Bachelor Year 1
Project 1.1
Pentominoes

Tom Bitterman, Martijn Boussé, Remzi Celebi, Adriana Iamnitchi,
Ashish Sai, Barbara Franci, Enrique Hortal



Period 1.1, 1.2, and 1.3
Academic year 2023-2024
Computer Science
Faculty of Science and Engineering
Maastricht University

Courses:

Introduction to Computer Science (BCS1110)
Procedural Programming (BCS1120)
Discrete Mathematics (BCS1130)
Objects in Programming (BCS1220)
Calculus (BCS1440)
Logic (BCS1530)

1 Project description

The central topic of Project 1-1 is to create a computer application with a user-friendly interface for solving the three-dimensional knapsack problem.

1.1 Background

Pentominoes are the planar structures that can be obtained by attaching 5 squares of size 1x1 to each other. There are precisely 12 pentominoes and the following picture shows a representation of these.

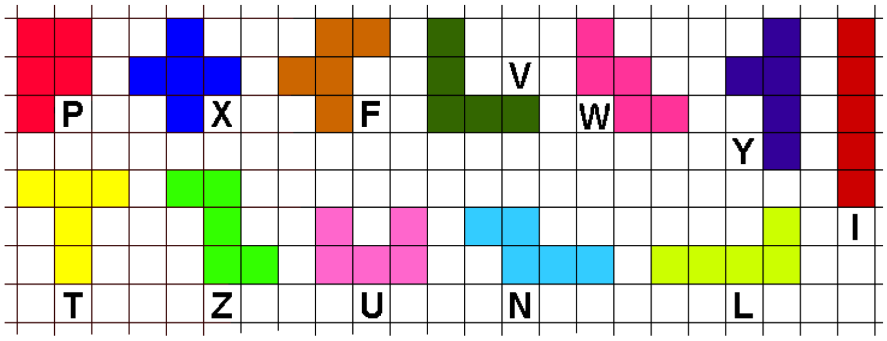


Figure 1: Visualization of all 12 pentomino shapes and their associated letter.

As is clear from the picture, the pentominoes can be identified by characters in a natural way.

This structures give us the opportunity to deal with "packing" problems in computer science in a more "imaginative" manner. These are very natural problems that occur when you want to make the most economically efficient use of a given space, subject to conditions on the total size of the space, the shapes of the items you are packing, and their value. It probably won't surprise you to learn that good algorithms for packing problems are important in, for example, shipping container loading. The knapsack problem is a typical example of a combinatorial problem. Combinatorial problems are very common in operations research, applied mathematics, and theoretical computer science. In combinatorial problems exhaustive search is generally not tractable, and even clever ideas to find optimal solutions on large instances can remain intractable. So, approximate (near-optimal) solutions are often a suitable answer, for example, obtained by either heuristic or provably approximate approaches. Other common problems involving combinatorial optimization are the travelling salesman problem or the minimum spanning tree problem.

1.2 Objective

Within the scope of this project, your main objective is the development and implementation of a computer program using Java. The program's core functionality involves populating a predefined rectangular space with specific objects known as pentominoes. The initial phase of the project aims to create efficient algorithms that can arrange a given set of pentominoes to maximize space utilization.

In the second phase, you will implement a Tetris-style game. In this game, the falling game pieces will be the abovementioned pentominoes. This modification of this classic game not only adds an element of fun but also provides an opportunity to test your programming skills in a real-time scenario.

Finally, the third part of this project involves tackling a complex computational problem: solving three-dimensional knapsack problems. This phase demands the implementation of more sophisticated algorithms capable of optimizing the selection of items to fit into a knapsack with volume constraints.

1.3 Software

The programming language for this project is Java. The use of external libraries is not allowed unless explicitly mentioned. It is your responsibility to show (or prove to) the examiners that your code works.

2 Project tasks

The project consists of several design and developing tasks, organized in three main phases. The main phases are defined:

2.1 First phase: Pentominoes

In the first phase, we want you to find out if a given set of pentominoes completely covers a rectangle of a given size. If this is possible we ask you to find (and show) one solution. Note that in a solution for a rectangle of a given size, each type of pentomino selected as input occurs at most once.

All 12 pentominoes together (i.e., input set P, X, F, V, W, Y, T, Z, U, N, L, I) fit in a rectangle of size 5x12. Actually, there are many ways to make them fit. The next picture shows one example how to make them fit.

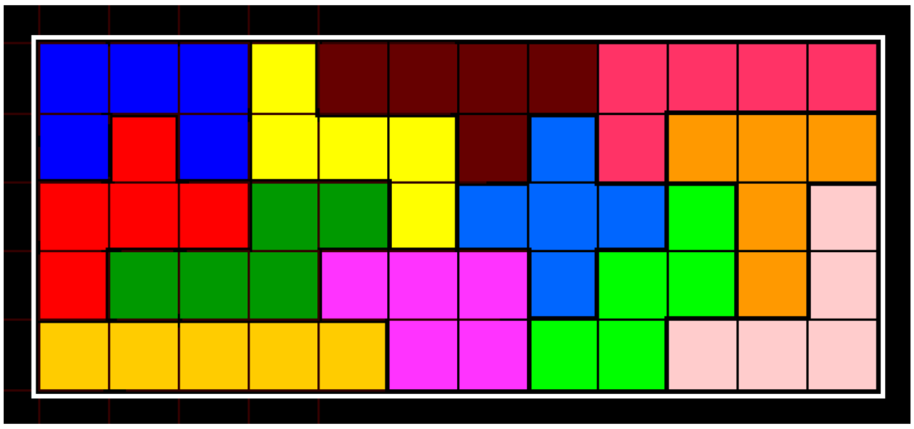


Figure 2: All pentominoes fit in a 5×12 rectangle.

To get you started, the following picture shows you one solution with half of the pentominoes (input set T, W, Z, L, I, Y) on a board of size 5 by 6.

While using any pentomino, it may be rotated or flipped over any way you like.

At the beginning of the project week, we will provide you with a template code of a basic search algorithm. We will provide the code via Canvas. Please be aware that this code is incomplete. Some parts are missing and you are expected to complete those in order to obtain a working code. Comments in the code will show which components you still need to implement.

During the project (each day at 9:00 and 13:00), we will provide you via Canvas with some periodic hints on how to proceed further: in particular, how to implement algorithms that run more quickly than the basic search approach. The "hint"-schedule is also uploaded to Canvas. Please do not see those hints as the only way to proceed. They should help

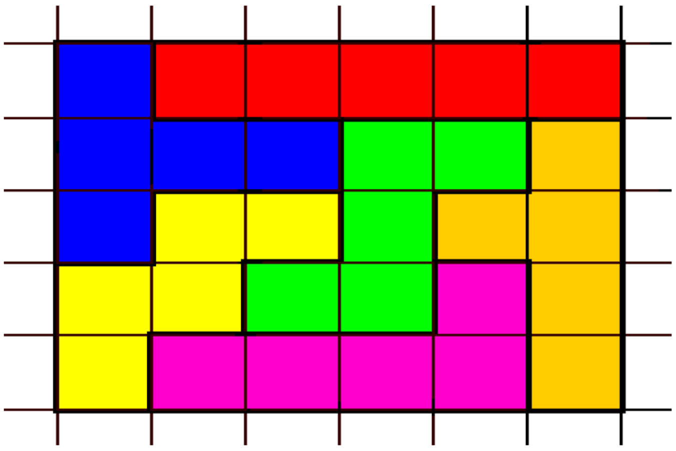


Figure 3: Half of the pentominoes fit in a 5 × 6 rectangle.

you to structure the rest of your project week. If you have not succeeded in finishing open tasks, do not feel rushed and focus on completing those before moving on to the next task. The criteria at the end of this page will show the least we expect from you in order to pass the first phase of the project.

It is very important that you test each algorithm with several inputs (i.e. different sizes of rectangles, and different sets of pentominoes). The two pictures above show two examples of rectangle sizes and pentomino inputs. The code provided on the first day already gives a visualization of the pentominoes. This will help you to test your algorithms.

In order to obtain a pass for the first phase of the project, we expect you to fulfill the following requirements: you must have at least a working basic search algorithm (think about a "brute force" algorithm) and at least one working branching algorithm. Further improvements on the code (e.g. proper code structure, JavaDoc documentation, etc.) or the implementation of additional algorithms will help increase the grade.

2.2 Second phase: Tetris

In the second phase, we want you to build a computer application with a user-friendly interface to play the Tetris-style game using the 12 pentomino-shapes that appear for the player in random order, each with the same probability. For the structure of the field, your game should use a board of width 5 and height 15.

Here is some information on the Tetris game: random pentominoes appear, one by one, starting from the top (line 15) and gradually fall down to the bottom-(line 1), or to the lowest line where they are supported by (i.e. touch) other pentominoes. During its fall, a pentomino may be rotated or shifted left or right (please note, flipping pieces is no longer allowed). By pushing "space bar" the pentomino drops down all the way. At the very moment a pentomino stops falling down something may happen: If, at that very moment a horizontal line of the board is completely filled with pentomino parts, then this line disappears and the remaining parts from higher lines fall down until they hit the next line. As long as these remaining parts are connected, they fall down together as one piece, and this piece is hindered in its fall as soon as one of its parts is hindered. Then it stops falling down. If all pieces are settled, a new random pentomino appears in the top line.

The player receives credits (1 point) for each horizontal line they have been able to delete (to make disappear).

The application to be built keeps track of a high score list.

In addition, each project team should decide which order of the 12 different pentominoes would give the highest score if one could choose the order and if the game had only these 12 pieces falling down one by one.

Finally, you implement a bot playing the Tetris game with pentominoes. You might find it helpful to make use of the algorithms implemented in Phase 1 for this part of the project.

At the end of this phase, you MUST have implemented all 3 components:

- A playable game, with GUI and high-score table, that fulfills the described rules
- Determine an (ideally) optimal ordering of the 12 pieces
- Building a bot

To get a 6 or higher, you must have implemented all 3 components, at least at a basic level. In means that examiners expect, at least:

- A basic GUI: e.g. functional, but with a basic GUI and only basic functionalities.
- A basic ordering: e.g. produces some 'good' ordering but not necessarily the optimal one
- A basic bot: e.g. greedily sticks the next piece at a sensible place, without taking into account longer-term effects of this short-term decision.

2.3 Third phase: Knapsack

In the third phase, we want you to build a computer application with a user-friendly interface that can be used for solving the so-called three-dimensional knapsack problems:

Assume that your company owns trucks with a cargo-space of 16.5 m long, 2.5 m wide and 4.0 m high. Assume that your company transports parcels of three different types: A, B and C. The sizes of the types are:

A: 1.0 x 1.0 x 2.0
B: 1.0 x 1.5 x 2.0
C: 1.5 x 1.5 x 1.5

A parcel of a given type also has a certain value. Denote these values by vA, vB and vC for types A, B and C respectively. Now, make a computer application that computes, for a given set of parcels (that may or may not fit into a truck), a packing that maximizes the total value.

The application does not necessarily have to find the best answer in all cases, but it should be able to find a good approximation. The application should also make a 3D-visualization of its answer – from different perspectives.

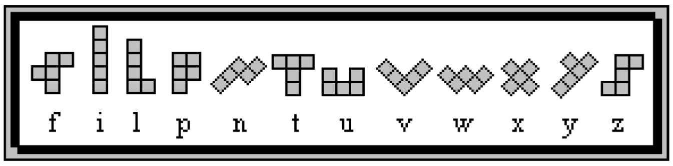
Use your application to answer the following questions:

A Is it possible to fill the complete cargo space with A, B and/or C parcels, without having any gaps?

B If parcels of type A, B and C represent values of 3, 4 and 5 units respectively, then what is the maximum value that you can store in your cargo-space?

Once you have answered these questions, assume that your company now transports pentomino shaped parcels of types **L**, **P** and **T**, where each of these pentominoes consists of 5 cubes of size 0.5 x 0.5 x 0.5.

Adapt your application from Phase 1 to answer the following questions:



C Is it possible to fill the complete cargo space with L, P and/or T parcels, without having any gaps?

D If parcels of type L, P and T represent values of 3, 4 and 5 units respectively, then what is the maximum value that you can store in your cargo-space?

To pass this phase, the group must be able to answer these questions and provide 1) code to solve the given problems and 2) a (3D)visualization of the solutions calculated by it.

References

[1] Morrison, D. R., Jacobson, S. H., Sauppe, J. J., & Sewell, E. C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization, 19, 79-102.

[2] Knuth, D. E. (2000). Dancing links. arXiv preprint cs/0011047.

[3] Procedural Programming course, Module 6, Department of Advanced Computing Sciences, Maastricht University. PrP Module 6.