COMP 7038
NON-LINEAR DATA STRUCTURES & ALGORITHMS

Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

ASSIGNMENT 2 – Backtracking search and dynamic programming

SUBMISSION & DUE DATE

This assignment should be submitted to Canvas before 11:59pm on **Friday 11/4/2025**. The usual late submission penalties apply in accordance with MTU's marks and standards policy.

Please submit a single ZIP file with your student number and name in the filename. Your submission should contain **exactly 3 files**:

- *HDDAllocation.java*
- *RodCutting.java*
- *LongestCommonSubsequence.java*

Please do NOT include any other files in your submission.

Make sure that you document your code and include sufficient comments where appropriate. Explain exactly what you are implementing and provide arguments for the correctness of your solution.

EVALUATION PROCEDURE

You can achieve a total of 25 points for the submission as indicated in the tasks. For each subtask you are given full marks for correct answers in your submission, 70% for minor mistakes, and 35%, 15%, or 0% for major mistakes or omissions depending on severity.

To mitigate against any form of plagiarism, including but not limited to the use of generative AI tools, **you MUST also present and explain your solution and confidently answer questions** about your submission to be graded. This oral assessment will take place during the scheduled labs after the submission deadline (or the week after, should we run out of time during these sessions).
If you do not show up your submission will not be graded unless there are approved IECs!

Your ability to fully explain your own submission will be crucial. You will only receive full marks if you are able to confidently present and explain all your work. Your assignment grade will be capped at 70% or 50% in case of minor issues with your explanations, and at 35%, 15%, or 0% in case of major problems with your explanations during the oral assessment.

The oral assessment will cover everything you uploaded onto Canvas, so be very careful what you include into your submission and make sure that you **only submit what you fully understand** to avoid your final marks being capped; in particular, do NOT include anything, including snippets taken from labs or lectures, unless you are able to explain it in detail or risk the whole rest of the assignment being marked down.

TASK 1 (HDD allocation, 7 points)

In this task you will implement a backtracking search procedure to allocate files onto hard drives. Given a fixed number of hard drives with different overall storage capacity your task is to distribute a set of files of different sizes onto these.

Download the file *HDDAllocation.java* from Canvas and implement the following method where indicated (you can add additional methods and fields to the class if required):

```
public int[] generate_allocation()
```

It should return an `int[]` of length equal to the number of files containing for each file the number of the hard drive the file should be stored on. Implement a backtracking search algorithm to decide for each file which hard drive it should be allocated to and make sure at each decision point that the storage capacities are not exceeded.

The constructor of the class

```
HDDAllocation(int[] hdds, int[] files)
```

has two parameters:

- the maximum capacities of all the hard drives (`int[] hdds`)
- and the sizes of all the files (`int[] files`).

You can use the main method in this class to test your code. The output for the example should look similar to:

```
File 0 has size 300MB and goes on HDD0.
File 1 has size 200MB and goes on HDD0.
File 2 has size 300MB and goes on HDD1.
File 3 has size 1200MB and goes on HDD2.
File 4 has size 400MB and goes on HDD0.
File 5 has size 700MB and goes on HDD1.
File 6 has size 700MB and goes on HDD2.
HDD0 space used 900MB / 1000MB.
HDD1 space used 1000MB / 1000MB.
HDD2 space used 1900MB / 2000MB.
```

Please note that the allocation for the example is not unique, so it is possible that you get a different solution.

TASK 2 (rod cutting, 8 points)

In this task you will implement a dynamic programming algorithm to determine what lengths to cut a piece of material (e.g. a metal rod, timber, paper, fabric, etc.) into so that the overall price is maximised.

Download the file *RodCutting.java* from Canvas and implement the following method where indicated (you can add additional methods and fields to the class if required):

```
public LinkedList<Integer> best_cuts()
```

It should return a `LinkedList<Integer>` containing the different lengths of the pieces to cut. Make sure you use an iterative dynamic programming approach in this task (if you use backtracking search with or without memoization the task will be graded at 0 points).

The constructor of the class

```
public RodCutting(int[] prices)
```

has the price list (`int[] prices`) as parameter containing the different prices for all possible lengths. The length of the list is the length of the original piece, and each position corresponds to the price of a smaller piece (i.e. `prices[0]` contains the price of a cut if length 1, `prices[1]` contains the price of a cut of length 2, etc.).

You can use the main method in this class to test your code. The output for the example should look like this:

```
The best cuts for a rod of length 10m are
 - 2m selling at €5
 - 2m selling at €5
 - 3m selling at €8
 - 3m selling at €8
The overall price is €26.
```

TASK 3 (longest common subsequence, 10 points)

In this task you will implement a dynamic programming algorithm to determine the longest common subsequence of two strings.

Download the file *LongestCommonSubsequence.java* from Canvas and implement the following method where indicated (you can add additional methods and fields to the class if required):

```
public String compare()
```

It should return a `String` containing the longest common subsequence of the two inputs. Make sure you use an iterative dynamic programming approach in this task (if you use backtracking search with or without memoization the task will be graded at 0 points).

The constructor of the class

```
public LongestCommonSubsequence(String X, String Y)
```

takes the two input strings (`X` and `Y`) as parameter.

You can use the main method in this class to test your code. The output for the example should look like this:

```
The longest common subsequence of 'ABCBDAB' and 'BDCABA' is 'BCBA'.
```