

CENG 315

Algorithms

Fall 2025-2026

Take-Home Exam 1

Due date: 26 October 2025, Sunday, 23.59

1 Problem Definition

A Young tableau is an fixed-size $n \times m$ matrix where each row and each column are sorted in non-decreasing order. Some of the entries in a Young tableau may be ∞ , which we treat as nonexistent elements. Thus, a Young tableau can be used to hold $r \leq mn$ in partial sorted order.

Formally, a matrix Y is a Young tableau if:

$$Y[n][m] \leq Y[n][m+1] \text{ and } Y[n][m] \leq Y[n+1][m]$$

$$\begin{bmatrix} 1 & 2 & 3 & 10 \\ 2 & 4 & \infty & \infty \\ 5 & 8 & \infty & \infty \\ 10 & 11 & \infty & \infty \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & \infty \\ 5 & \infty & \infty \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 3 & 9 \\ 2 & 4 & \infty & \infty \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 5 & 10 \\ 70 & 90 \\ 81 & 91 \end{bmatrix}$$

Figure 1: Some valid Young tableau states.

In this homework, you will use a Young tableau to sort an array of integers and return the number of swaps required in the Young tableau to observe the complexity of the designed algorithm.

```
long sort_with_young_tableau(int* arr,
                           const size_t size,
                           const std::pair<size_t, size_t> table_size)
```

This function should sort the array arr in ascending order (That means, at the end the array must be in sorted order.) using a Young tableau of given dimensions and return the total number of swaps performed inside the Young tableau.

Parameters:

- $\text{int}^* \text{arr}$: pointer to the array that will be sorted in place.
- $\text{size_t} \text{size}$: number of elements in the array.
- $\text{std::pair<} \text{size_t}, \text{size_t}> \text{table_size}$: dimensions of the tableau, where
 - $\text{table_size.first} = \text{number of rows (n)}$
 - $\text{table_size.second} = \text{number of columns (m)}$

2 Example I/O

Example 1

```
Size: 9
Table size: (3, 3)
Array elements: {624, 15098, 4179, 8812, 4045, 3630, 8361, 11182, 9400}
Swap: 41
Sorted array elements: {624, 3630, 4045, 4179, 8361, 8812, 9400, 11182, 15098}
```

Example 2

```
Size: 2
Table size: (2, 1)
Array elements: {11594, 7357}
Swap: 3
Sorted array elements: {7357, 11594}
```

Example 3

```
Size: 4
Table size: (1, 4)
Array elements: {3564, 2148, 11535, 5388}
Swap: 14
Sorted array elements: {2148, 3564, 5388, 11535}
```

3 Specifications and Hints

- To find the correct swap number, follow the following steps:
 - Generate an empty Young Tableau with given dimensions.
 - Insert each element into Young Tableau in a given order. (First, insert a new element into the bottom-right position, and swap it with its neighbours until the matrix is a valid Young Tableau.)
 - Extract the minimum element repeatedly until your Young Tableau is empty. Change the minimum element with ∞ and swap the element until the tableau is valid.
- After each insert or extract operation, the matrix must be a valid Young tableau.
- If both the upper and left neighbours have the same value in the insertion phase and a swap is required, you must perform the swap with the upper neighbour (or lower neighbour in extraction) (specifically the one with the smaller row index).
- You can safely use `std::numeric_limits<int>::max()` to indicate an empty position in the tableau.
- You will implement your solutions in the `the1.cpp` file.
- You are free to add other functions to `the1.cpp`.
- Do not change the first line of `the1.cpp`, which is `#include "the1.h"`
- Do not change the arguments and the return value of the function `sort_with_young_tableau()` in the file `the1.cpp`.

- Do not include any other library or write include anywhere in your `the1.cpp` file (not even in comments).
- You are given a `test.cpp` file to test your work on ODTUCLASS or your locale. You can and you are encouraged to modify this file to add different test cases.
- You can test your `the1.cpp` on the virtual lab environment. If you click run, your function will be compiled and executed with `test.cpp` with random I/O. If you click evaluate, you will get feedback for your current work and your work will be temporarily graded with a limited number of inputs.
- The grade you see in VPL is not your final grade, your code will be reevaluated with more inputs after the exam.
- If you want to test your work and see your outputs on your locale you can use the following commands:

```
> g++ test.cpp the1.cpp -Wall -std=c++11 -o test
> ./test
```

4 Constraints and Limits

- The maximum array size is 25000.
- Maximum element is $2^{15} - 1$.
- The system has the following limits to test the complexity of your solutions:
 - a maximum execution time of 1 minute
 - a 256 MB maximum memory limit
 - a stack size of 64 MB for function calls (ie. recursive solutions)
- Solutions with longer running times will not be graded.
- If you are sure that your solution works in the expected complexity constraints but your evaluation fails due to limits in the lab environment, the constant factors may be the problem.
- If your solution is correct, the time and memory limits may be adjusted to accept your solution after the lab. Please send an email if that is the case for you.

5 Regulations

- **Implementation and Submission:** The template files are available in the Virtual Programming Lab (VPL) activity called “THE1” on ODTUCLASS. At this point, you have two options:
 - You can download the template files, complete the implementation, and test it with the given sample I/O on your local machine. Then submit the same file through this activity.
 - You can directly use the editor of the VPL environment by using the auto-evaluation feature of this activity interactively. Saving the code is equivalent to submitting a file.

Please make sure that your code runs on ODTUCLASS. There is no limitation in running your code online. The last save/submission will determine your final grade.

- **Programming Language:** You must code your program in C++11. Your submission will be tested on the VPL environment in ODTUCLASS, hence you are expected to make sure your code runs successfully there.
- **Cheating:** This assignment is designed to be worked on individually. Additionally, the use of any LLMs (chatgpt, copilot, the other one that you are thinking about...) and copying code directly from the internet for implementations is strictly forbidden. Your work will be evaluated for cheating, and disciplinary action may be taken if necessary.
- **Evaluation:** Your program will be evaluated automatically using “black-box” testing, so make sure to obey the specifications. No erroneous input will be used. Therefore, you don’t have to worry about invalid cases.

Important Note: The given sample I/O’s are only to ease your debugging process and NOT official. Furthermore, it is not guaranteed that they cover all the cases of required functions. As a programmer, it is your responsibility to consider such extreme cases for the functions. Your implementations will be evaluated by the official test cases to determine your final grade after the deadline.