The following exercises, taken from CLRS if not otherwise specified, are very good practise.

**Practice Exercises**

1. Solve Exercises 33.4-1, -3
2. Solve Exercises 15.1-1, -2, -3, -4, -5
3. Solve Exercises 15.2-1, -2, -3, -4, -5
4. Solve Exercises 15.3-1, -2, -3, -4, -5
5. Solve Exercises 15.4-1, -2, -3, -4, -5, -6
6. Solve Exercises 16.1-2, -3, -4
7. Solve Exercises 16.2-1, -2, -3, -4, -5, -6, -7
8. Solve Exercises 16.3-1, -2, -3, -4, -5, -6, -7, -8
9. Solve Exercises 22.1-1, -2, -3, -4, -5, -6, -7, -8
10. Solve Exercises 22.2-1, -2, -3, -4, -5, -6, -7, -8
11. Solve Exercises 22.3-1, -2, -3, -4, -5, -6, -7, -8, -9, -10
12. Solve Exercises 22.4-1, -2, -3, -4, -5
13. Solve Exercises 22.5-1, -2, -3, -4, -5, -6, -7
14. Solve Exercises 23.1-1, -2, -7
15. Solve Exercises 23.2-1, -4, -8
16. **The Sysadmin Problem.** We wish to store programs $P_1, P_2, \ldots, P_n$ on a disk. Program $P_i$ needs $s_i$ kB and the disk has capacity $D < \sum_{i=1}^n s_i$ kB.
   
   (i) Prove or disprove: to maximize the number of programs held on the disk, we can use a greedy algorithm that takes programs in order of nondecreasing size.
(ii) Prove or disprove: to maximize disk usage, we can use a greedy algorithm that takes programs in order of nonincreasing size.

17. **The 0/1 Knapsack Problem.** A Thief robbing a store finds \( n \) items, the \( i \)-th is worth \( v_i \) dollars and weighs \( w_i \) pounds, where \( v_i \) and \( w_i \) are integers. He wants to take as valuable a load as possible, but he can carry at most \( W \) pounds in his knapsack. Which items should he take? (0/1 means drop or take, there is no possibility of taking fractions of the items.). Design an algorithm for this problem and analyse its complexity.

18. **The Maximum Sum Contiguous Subvector.** Given an array of \( n \) integers, consider the problem of finding the maximum sum in any contiguous subvector of the input. For example, in the array

\[
(31, -41, 59, 26, -53, 58, 97, -93, -23, 84)
\]

the maximum is achieved by summing the third through seventh element: \( 59 + 26 - 53 + 58 + 97 = 187 \). (Hint: When all numbers are positive, the entire array is the answer. When all numbers are negative, the empty array is the answer.)

(i) Give a simple, clear and correct \( \Theta(n^2) \) algorithm.

(ii) Create a better, more efficient algorithm and analyze its time complexity, giving tight bounds.

19. **Game of Matches.** Suppose there are 30 matches on a table. I begin by picking up 1, 2 or 3 matches. Then my opponent must pick up 1, 2 or 3 matches. We continue in this fashion until the last match is picked up. The player who picks up the last match loses. How can I (the first player) be sure of winning the game?

### Problems to hand in

1. **The Knapsack Problem.** You plan to go hiking and you want to pack your stuff in a knapsack. Your knapsack has a total volume of \( V \) cubic inches, and you have \( n \) different items of volumes \( v_1, \ldots, v_n \) that you want to take with you. Unfortunately, you can’t take them all, since their total volume exceeds the volume of your knapsack. You need to decide which items to bring and which items to leave at home. Suppose that you want to optimize the packing so that the volume is utilized to its maximal extent. This means you want to find a subset
of the items such that the sum of their volumes is maximal with the constraint that the sum is no more than $V$. We call this the optimum packing of $v_1,\ldots,v_n$ in $V$.

(i) Fix a list of items $v_1,\ldots,v_n$. For any $i \leq n$, and any $U \subseteq V$, denote by $OPT_i(U)$ the volume of the optimum packing of $v_1,\ldots,v_i$ in $U$. Prove that for all $i$ and $U$, we have

$$OPT_i(U) = \max\{OPT_{i-1}(U), v_i + OPT_{i-1}(U - v_i)\}$$

(ii) Assume that all the volumes $v_1,\ldots,v_n$ are integral numbers of units. Design a dynamic-programming algorithm to solve the knapsack problem. Analyze your algorithm.

2. Solve problem 16-2, p 402 in CLRS.