## Reading material

We will introduce the paradigm *dynamic programming*. You should read KT Section 6.1 and 6.2. $[w]$ on an exercise means it is a warmup exercise.

## Exercises

**1** $[w]$ **Weighted interval scheduling** Solve the following weighted interval scheduling problem using both the recursive method with memoization and the iterative method. The intervals are given as triples $(s_i, f_i, v_i)$: $S = \{(1,7,4),(10,12,2),(2,5,3),(8,11,4),(12,13,3),(3,9,5),(3,4,3),(4,6,3),(5,8,2),(4,13,6)\}$.

**2 Grid Paths** Consider an $n \times n$ grid whose squares may have traps. It is not allowed to move to a square with a trap. Your task is to calculate the number of paths from the upper-left square to the lower-right square. You can only move right or down.

**2.1** Give an algorithm that return the number of paths and analyse its running time.

**2.2** Implement your algorithm on CSES: https://cses.fi/problemset/task/1638

**3 Job planning** Solve KT 6.2.

**4 Balloon Swap Party (from the Exam 2025)** You are playing a new puzzle game called The Balloon Swap Party. There are $n$ party guests, numbered from 1 to $n$. Each guest is holding a balloon of one of three colors: red, blue, or green. You also have a balloon in your hand; at the start of the game, your balloon is red. To earn points, you visit each guest in order from 1 to $n$. For each guest, you can either keep your balloon or swap it with the balloon the guest is holding.

- If you swap your balloon for another balloon of the same color, you earn 1 point.
- If you swap your balloon for a balloon of a different color, you lose 1 point. (Your score can be negative.)
- If you decide not to swap, your score does not change.

You must visit the guests in order, and once you visit a guest, you cannot return to them.

You want an efficient algorithm to compute your maximum possible score. Your input is a list $C[1 \ldots n]$, where $C[i]$ is the color of the balloon that the $i$th guest is holding.

**Example** Let $n = 6$. The colors of the balloons of guest 1 to 6 are (in this order): blue, red, green, blue, green, green. If you swap balloons with all the guests, you get $-4$ points. If you swap balloons with guest 2, 3, 5, and 6 you get 2 points.

Let $\mathsf{OPT}(c,i)$ be the maximum score you can obtain if you have a balloon of color $c$ after visiting the $i$th guest. If it is not possible to have a balloon of color $c$ after visiting the $i$th guest then $\mathsf{OPT}(c,i) = -\infty$. We define $\mathsf{OPT}(c,0) = 0$ if $c = \text{red}$ and $-\infty$ otherwise.

**4.1** Fill out the table below for $\mathsf{OPT}(c,i)$ when $C = [\text{red}, \text{green}, \text{green}, \text{red}, \text{blue}, \text{blue}, \text{green}]$.

| c \ i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| red | | | | | | | | |
| blue | | | | | | | | |
| green | | | | | | | | |

**4.2** We want to define a recurrence for $\mathsf{OPT}(c,i)$. The base case ($i = 0$) is already defined. For $i > 0$, write the recurrence for the two cases $C[i] = c$ and $C[i] \neq c$.

**4.3** Describe an efficient algorithm to compute your maximum possible score. Your input is the list $C[1 \dots n]$, where $C[i]$ is the color of the balloon that the $i$-th guest is holding. Your algorithm should be based on the recurrence for $\mathsf{OPT}(c,i)$. (You do not have to write pseudo code for the algorithm.)

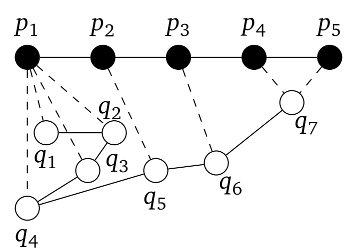
**4.4** Analyze the time and space complexity of your algorithm.

**5 Discrete Fréchet distance** Consider Professor Bille going for a walk with his dog. The professor follows a path of points $p_1, \dots, p_n$ and the dog follows a path of points $q_1, \dots, q_m$. We assume that the walk is partitioned into a number of small steps, where the professor and the dog in each step either both move from $p_i$ tp $p_{i+1}$ and from $q_j$ to $q_{j+1}$, respectively, or only one of them moves and the other one stays.

The goal is to find the smallest possible length $L$ of the leash, such that the professor and the dog can move from $p_1$ and $q_1$, resp., to $p_n$ and $q_n$. They cannot move backwards, and we only consider the distance between points. The distance $L$ is also known as the discrete Fréchet distance.

We let $L(i,j)$ denote the smallest possible length of the leash, such that the professor and the dog can move from $p_1$ and $q_1$ to $p_i$ and $q_j$, resp. For two points $p$ and $q$, let $d(p,q)$ denote the distance between the them.

In the example below the dotted lines denote where Professor Bille (black nodes) and the dog (white nodes) are at time 1 to 8. The minimum leash length is $L = d(p_1, q_4)$.




**5.1** Give a recursive formula for $L(i,j)$.

**5.2** Give pseudo code for an algorithm that computes the length of the shortest possible leash. Analyze space and time usage of your solution.

**5.3** Extend your algorithm to print out paths for the professor and the dog. The algorithm must return where the professor and the dog is at each time step. Analyze the time and space usage of your solution.

**Puzzle of the week: 101 ants** [1]There are 101 ants on a rod of length 1 metre. 100 of them are black and positioned randomly along the 1 metre rod. The 101st ant is red and is positioned in the middle of the rod. All ants are travelling at 1 metre/minute either right or left at the start. The ants are also perfectly elastic, so that if two ants collide they simply turn round and carry on at 1 metre/minute in the opposite direction. The rod has been capped at both ends so that an ant that reaches the end of the rod simply turns round and carries on travelling back towards the middle of the rod (still at 1 metre/minute). Each ant starts off heading in a random direction. What is the probability that after exactly 1 hour the red ant is back exactly in the middle of the rod?

The ants are arbitrarily positioned on the rod and are travelling at 1 metre/minute either right or left at the start.

[1]I got this puzzle from Raphäel Clifford