1st tutorial

Introduction

Tutorial principles.

- **You** want to learn something.
- **I** am trying to create space for it.
- That's why the highest priority is what **you** need:
  - ... understand something from the lecture
  - ... understand a task from the previous tutorial
  - ... understand some homework assignment, etc.
- **Don't be afraid** to ask questions
- A few general **tips**:
  - When you feel lost, try to play with examples. (E.g., if you can't solve a task for general n, k, try what it says for small values. Don't know what an algorithm does? Try it on small input.)
  - Try to explain where you got stuck to your classmates. It will help you understand what's really going on, get your thinking clearer.
  - If you still don't know, reach out, I will try to help.

Logistical details.

- Email me at `koutecky+ads2@iuuk.mff.cuni.cz`.
- The tutorial website is at `http://research.koutecky.name/db/teaching:ads2223_tutorial`
- You need to get 100 points to gain credit. Points are awarded for:
  - Homework. There will be a batch of tasks every other week. The total amount of points obtainable will be at least 150. You need to solve at least one task from each batch, not necessarily within the deadline.
  - Solutions submitted after the deadline earn two thirds of possible points (i.e., a perfect solution of a 9 point task gets you only 6 points; a 50% solution gets you 3 and not 4.5 points, etc.)
  - I will not accept solutions more than 2 months after the deadline.
  - You can ask me for exceptions and I'm likely to grant them when justified, but please ask first.
  - The use of AI: ideally don't; if you really have to, I treat it as discussing with a friend. You **must** write your own solution. I will report all cheating incidents to PlagUE, even if I only have a (strong) suspicion.
  - Grading / giving feedback on each other's homework. If you are confident about your solution, you can email me and say you want to be a student grader; if your solution is indeed correct, I will let you correct the other solutions and gain extra points for it. **Do this!**
  - Quizzes - maybe? **Discuss**
  - A small project. *(But this is hard.)*
  - *I don't require attendance. If you feel that you can benefit more from self-study or anything else, go for it.*
- Tasks are handed in the OWL system: `https://kam.mff.cuni.cz/owl/`
- We can discuss anything over Discord.
- If you need further help, try getting it from your classmates. If there's more of you who need help with the same thing, email me and we'll set up a meeting :)

---

1. Asymptotic complexity: Sort the following functions into groups of those that grow asymptotically at the same rate (for each f and g it holds that $f = \Theta(g)$), and subsequently sort the groups themselves from slowest to fastest growing:
n, $42n + 7$, n^2, $\log n$, $\log(n^2)$, $(\log n)^2$, $\sqrt{n}$, 2^n, 2^{2n}, 4^n, $2^{n \log n}$, $2^{2 \log n}$, $2^{(\log n)^2}$, n^n, $n!$, $(n + 1)!$.

2. Recurrence: Solve the following recurrence (always assume $T(1) = 1$):

- $T(n) = T(n/2) + \Theta(1)$
- $T(n) = T(n/2) + \Theta(n)$
- $T(n) = 2T(n/2) + \Theta(n)$
- $T(n) = 8T(n/2) + \Theta(n^2)$
- $T(n) = 7T(n/2) + \Theta(n^2)$

3. Naive search: The text search problem is as follows: we have a string ι (a "needle", which is tiny, like the smallest Greek letter) and a (presumably longer) string σ (for "hayσtack"), and we want to decide whether ι is a contiguous substring of σ, that is, there is an index i s.t. $\sigma[i : |\iota|] = \iota$.

A naive algorithm works like this: try all start indices i, and test whether $H[i : |\iota|] = N$. *(Think about how you would implement this!)*
Prove that the naive algorithm can take as much as $\Omega(|\iota||\sigma|)$ steps, even in the case when it doesn't find anything.

4. Substring: Decide whether a string H contains N as a substring (not necessarily contiguous). (Design a fast algorithm.) What if we wanted to count the number of such occurences?

5. Guard. We have a maze on an $N \times N$ grid with walls, a hero, and a treasure. We want to find a way for the hero to reach the treasure as quickly as possible, but not run into a guard (that is, appear in the same cell as the guard at the same time). The guard has a fixed route of length L as a sequence of adjacent cells and it goes back and forth on this route. How to find such a shortest route as quickly as possible? What is the complexity of your algorithm in terms of N and L?

6. Lost Robots. There is a maze and 2 robots at different places in the maze. However, these robots are controlled by the same remote control, which has four buttons – going north, south, east, and west. When a robot receives a command which cannot be executed, it ignores it. How to find a sequence of commands which takes both robots out of the maze? (Once a robot is out of the maze, it stops listening to commands.) How to find the shortest such sequence?