Za co bude zápočet?

1. Úlohy ke společné diskusi
   - najít/vymyslet 2 úlohy
   - poslat cca týden před cvičením (ook@ucw.cz)
   - rozešlu před cvičením, na cvičení probereme řešení
2. Aktivní řešení zadaných soutěží/úloh.

Typický program

cca 30 minut	diskuse úloh
cca 30-60 minut	teorie
zbytek	řešení

- úloh k tématu (`https://recodex.mff.cuni.cz/`)
- minulých soutěží ICPC (`https://codeforces.com/`)

International Collegiate Programming Contest

- jedna z nejstarších programovacích soutěží ($\approx$ 1977)
- organizace:
  - předkolo CTU open (na ČVUTu, 27.-28.11.?)
    - na základě výsledků vybereme 3-4 týmy reprezentující MFF UK
  - středoevropské kolo CERC (Praha, ???)
    - 2-5 nejlepších univerzit postupuje do ...
  - celosvětové finále (???)

Kdo může soutěžit

- 3-členné týmy, z jedné univerzity
- $\leq$2 účasti na finále, $\leq$5 účastí na CERC
- zahájení studia $\geq$ 2016 nebo narozen $\geq$ 1997

Soutěž

- 3-členné týmy, jeden počítač
- 8-10 úloh, 5 hodin
- průběžné vyhodnocování
  - accepted
  - run-time error
  - time-limit exceeded
  - wrong answer; presentation-error
  - compile time error; too late; contest rule violation
- průběžné výsledky až do poslední hodiny soutěže
- možnost používat písemné materiály

Technikálie

- Soutěžní jazyky: **C++**, C, Java, Python, Kotlin?
- Vstup ze `stdin`, výstup na `stdout`.
- Nesmí se používat soubory (ani `stderr`) a thready.
- Paměťový limit $\approx 1GB$.
- Časové limity nejsou zveřejňovány, bývají v řádu sekund.

Hodnocení

primárně: počet vyřešených úloh

sekundárně: kumulativní čas; za každou vyřešenou úlohu:

- počet minut od začátku soutěže do úspěšného odevzdání
- plus 20 minut za každé neúspěšné odevzdání

Žádná penalizace za neúspěšná odevzdání úloh, které nevyřešíte.

Strategie: čas

- Vyřešíme $k \approx 8$ úloh.
- i-tou odevzdanou úlohu řešíme t_i minut

Kumulativní čas bez penalizace je

$$k \cdot t_1 + (k-1) \cdot t_2 + \cdots + t_k$$

- Nejprve řešit nejsnažší (**nejrychleji naprogramovatelné**) úlohy!
- Rychle ji najdětě:
  - paralelizujte čtení zadání
  - sledujte průběžné výsledky
- Na začátku jsou 2-3 minuty ladění/testování navíc horší než 20 minut penalizace.
- Typicky $t_1 \approx 10 - 20$ minut.

Strategie: využití počítače

- V max. míře využívat k programování, ne k ladění.
- Programovat bez chyb:
  - Celkovou strukturu a klíčová místa si předem rozmyslet na papíře.
  - Rovnováha mezi přehledností a rychlostí.
- Hledání chyb/ladění na výtisku.
- Minimalizovat přerušení?

Strategie: tým

Mimo počítač lze

- ladit/hledat chyby
  - občas pomůže vysvětlovat kód někomu jinému
- připravovat si detaily řešení

Alternativa: 2 programátoři +

- “teoretik”
  - čte zadání a stručně je vysvětluji, vybírá jednoduché úlohy
  - vymýšlí řešení a vysvětluje je programátorům
  - pomáhá s laděním
- “specialista”
  - řeší matematické a geometrické úlohy

Písemné materiály

- CTU Open, CERC: Libovolné.
- Finále: 25 stran. Příklad:
  `iuuk.mff.cuni.cz/~rakdver/acm/reference.pdf`

Například:

- matematické vzorce (geometrie, kombinatorika, . . .)
- datové struktury a algoritmy
  - intervalový strom, union-find, vyvážený vyhledávací strom, lokalizace bodu v rovině
  - Gaussova eliminace, Euklidův algoritmus vč. hledání lineární kombinace, čínská věta o zbytcích, 2-SAT
  - Dijkstra, Bellman–Ford, komponenty 2-souvislosti
  - KMP, Aho-Corasick, suffixové pole/automat
  - nejdelší společná/rostoucí podposloupnost
  - nejbližší společný předchůdce, heavy-light dekompozice
  - největší tok a nejmenší řez, párování, vertex-cover a nezávislá množina v bipartitních grafech
  - Voroného diagramy a Delaunayova triangulace
  - lineární programování, párování v obecných grafech