Matematická olympiáda – kategorie P

Co se dozvíte?

- Základní informace o MO-P.
- Jak by mělo vypadat řešení?
- Návodné úlohy.

Co je MO-P?

- Programovací soutěž,
- zaměřená na efektivní řešení konkrétních teoretických úloh,
- vstup a výstup v jednoduchém formátu, žádná další interakce s uživatelem.

Více informací: mo.mff.cuni.cz.

Organizace soutěže (1)

Domácí kolo:

- Do 15. listopadu.
- 1 teoretická a 3 praktické úlohy.
  - Jedna v nestandardním výpočetním modelu.
- Odevzdávání viz mo.mff.cuni.cz.
- Typická postupová hranice: Cca 10 bodů (z 40).

Krajské kolo:

- 21. ledna.
- 4 hodiny, 4 teoretické úlohy řešené na papíře.

Organizace soutěže (2)

Ústřední kolo:

- 30 nejlepších účastníků.
- 2×5 hodin, 3 teoretické a 3 praktické úlohy
- 19.–21. března v Plzni

Výběrové/přípravné soustředění:

- pro cca 10 nejlepších účastníků
- duben?

Mezinárodní soutěže

Mezinárodní olympiáda v programování (IOI)

- 4 nejlepší účastníci
- Bolívie, léto(?) 2025

Středoevropská olympiáda v programování (CEOI)

- 4 nejlepší nematuranti, kteří se nekvalifikovali na IOI
- Rumunsko, léto(?) 2025

Evropská dívčí olympiáda v programování (EGOI)

- 4 nejlepší dívky
- ??? 2025

Co by mělo obsahovat řešení – praktické úlohy

- Zdrojový kód řešení:
  - C, C++, Python, C#, Java, Pascal

Vyhodnocení:

- Automatické testování na konkrétních vstupech různé velikosti.
- Musí doběhnout dostatečně rychle a vrátit správnou odpověď.

body	n
2	$n \leq 10$
2	$n \leq 1\,000$
3	$n \leq 100\,000$
3	$n \leq 1\,000\,000$

Co by mělo obsahovat řešení – teoretické úlohy

- Co nejsrozumitelnější popis řešení.
  - A případně zdrojový kód „důležitých“ částí.
- Zdůvodnění jeho správnosti.
- Určení a zdůvodnění časové a paměťové složitosti.

Vyhodnocení:

- Chybné řešení: 0 bodů.
- Jinak:
  - Cca 3 body za kvalitu popisu a zdůvodnění,
  - plus $0 - 7$ bodů dle efektivity.

Model

Poslední úloha:

- delší studijní text
- netradiční výpočetní prostředky
- v domácím kole praktická, v dalších teoretická

Příklad

Honza je s kamarády na veletrhu dortů a postupně obchází stánky, na nichž mu nabízí různě velké dorty. Honza nechce vypadat jako žrout, a proto se rozhodl, že si nikdy nedá dort na víc než dvou po sobě jdoucích stáncích. Jakou největší celkovou váhu dortů může sníst?

1 5 5 7 4 4

Příklad

Honza je s kamarády na veletrhu dortů a postupně obchází stánky, na nichž mu nabízí různě velké dorty. Honza nechce vypadat jako žrout, a proto se rozhodl, že si nikdy nedá dort na víc než dvou po sobě jdoucích stáncích. Jakou největší celkovou váhu dortů může sníst?

1 5 5 7 4 4

Příklad

Honza je s kamarády na veletrhu dortů a postupně obchází stánky, na nichž mu nabízí různě velké dorty. Honza nechce vypadat jako žrout, a proto se rozhodl, že si nikdy nedá dort na víc než dvou po sobě jdoucích stáncích. Jakou největší celkovou váhu dortů může sníst?

1 5 5 7 4 4 $\rightarrow$ 17

Příklad

Honza je s kamarády na veletrhu dortů a postupně obchází stánky, na nichž mu nabízí různě velké dorty. Honza nechce vypadat jako žrout, a proto se rozhodl, že si nikdy nedá dort na víc než dvou po sobě jdoucích stáncích. Jakou největší celkovou váhu dortů může sníst?

1 5 5 7 4 4 $\rightarrow$ 18

Soutěžní úloha

Na vstupu dostanete celé číslo n ($1 \leq n \leq 10^6$) a n kladných celých čísel, která udávají váhy dortů na stáncích v pořadí, v jakém kolem nich Honza jde. Vypište jedno celé číslo, udávající maximální celkovou váhu dortů, které Honza může sníst, tj. největší možný součet podmnožiny vstupních čísel neobsahující více než dvě po sobě jdoucí čísla.

Příklad vstupu:

6	$n = 6$ stánků
1 5 5 7 4 4	váhy dortů

Odpovídající výstup:

18

Jednoduché řešení

- Vyzkoušíme všechny podmnožiny.
- Například rekurzivně; `generuj(začátek, p)` vygeneruje všechny podmnožiny, které se před stánkem p shodují ze začátkem.
  - Jestliže $p =$ konec, `začátek` je jedna z generovaných podmnožin.
  - Jinak:
    - `generuj` (`začátek` a prvek na pozici p, $p + 1$)
    - `generuj` (`začátek`, $p + 1$)

Zdrojový kód

```
int vysledek = 0;

void generuj(vector<bool> &zacatek, int p)
{
  if (p == n)
    {
      if (neobsahuje_tri_po_sobe (zacatek))
        vysledek = max (vysledek,
                        soucet_vah (zacatek));
      return;
    }

  zacatek[p] = true; generuj (zacatek, p + 1);
  zacatek[p] = false; generuj (zacatek, p + 1);
}

vector<bool>(n) mnozina; generuj (mnozina, 0);
```

Časová složitost – O-notace

- Určit počet provedených operací přesně je složité.
- Operace trvají různě dlouho, čas jejich provedení závisí na stavu paměti, ...

Místo toho: „Časová složitost je $O(f(n))$.“

- Existuje nějaká konstanta c taková, že pro každé n algoritmus provede nejvýše $c \cdot f(n)$ operací.
- Více viz ksp.mff.cuni.cz/kucharky/slozitost/

Určení časové složitosti

- 2^n podmnožin
- Pro každou ověříme, zda neobsahuje 3 po sobě jdoucí prvky, sečteme váhy.
  - Cykly přes n prvků.

$$O(2^n \cdot n)$$

Odhad:

- Cca 10^9 operací za sekundu.
- Cca 10 – 100 operací ve vnitřním cyklu.
- Efektivní pro $n \approx 25$.

Běžné složitosti

Současné počítače provádí řádově 10^9 operací za sekundu:

- $O(n)$ algoritmy bývají efektivní pro cca $n \leq 10^7$
- $O(n^2)$ algoritmus pro cca $n \leq 10\,000$
- $O(2^n)$ pro $n \leq 30$

Lepší řešení – dynamické programování

Jsem-li na nějakém stánku, zajímá mě pouze

- zda jsem něco snědl na dvou předchozích stáncích a
- kolik jsem zatím snědl dortu.

Budeme počítat `nejv`[p][u]:

- největší váha dortů, které jsem mohl sníst do st. p, jestliže
- jsem si dort dal na $u \in \{0, 1, 2\}$ stáncích na konci (vč. p).

```
nejv[0][0] = nejv[0][2] = 0; nejv[0][1] = dort[0];
for (p = 1; p < n; p++) {
    nejv[p][0] = max (nejv[p-1][0], nejv[p-1][1],
                      nejv[p-1][2]);
    nejv[p][1] = nejv[p-1][0] + dort[p];
    nejv[p][2] = nejv[p-1][1] + dort[p];
}
vysledek = max (nejv[n-1][0], nejv[n-1][1],
                nejv[n-1][2]);
```

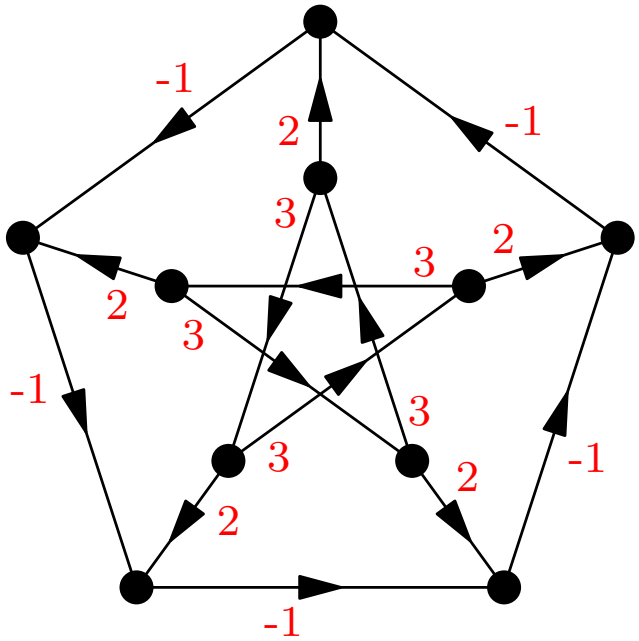
Složitost: $O(n)$

Co u teoretických úloh nedělat

- Odevzdat pouze zdrojový kód (i komentovaný).
- Přepis programu do češtiny:
  - „Ve for cyklu procházím p od 1 do $n - 1$. V každé iteraci `nejv[p][0]` nastavím na . . . “
- Nevhodně pojmenované proměnné:
  - „Zavedeme si pole `pole`. Dále si zavedeme pole `pole2`.“
  - „K právě spočítanému číslu přičteme druhý z počtů určených v předminulém odstavci.“
- Pouze popis běhu algoritmu na jednom konkrétním příkladě.
- Rozebírání nedůležitých technických detailů.
- Vynechání důležitých technických detailů.

Co naopak dělat

- Popis srozumitelný i bez nahlédnutí do programu.
  - Kód důležitých částí nám ale může pomoci objasnit detaily.
- Soustředit se na vysvětlení významu.
  - „Hodnota `nejv`[p][u] bude největší váha dortů, které můžeme sníst do pozice p tak, abychom si něco dali na $u \in \{0, 1, 2\}$ posledních stáncích (včetně stánku p).“
- Inspirovat se autorskými řešeními.


-1
2
-1
3
2
3
2
3
-1
3
3
2
-1
2
-1

Příklady

Dopravní síť ve městě:

- vrcholy: křižovatky
- hrany: jednosměrné ulice s délkami

Vztahy (graf bez vah):

- vrcholy: lidé
- hrany: má ho rád?

Molekuly (graf bez vah, orientací):

- vrcholy: atomy
- hrany: vazby mezi nimi

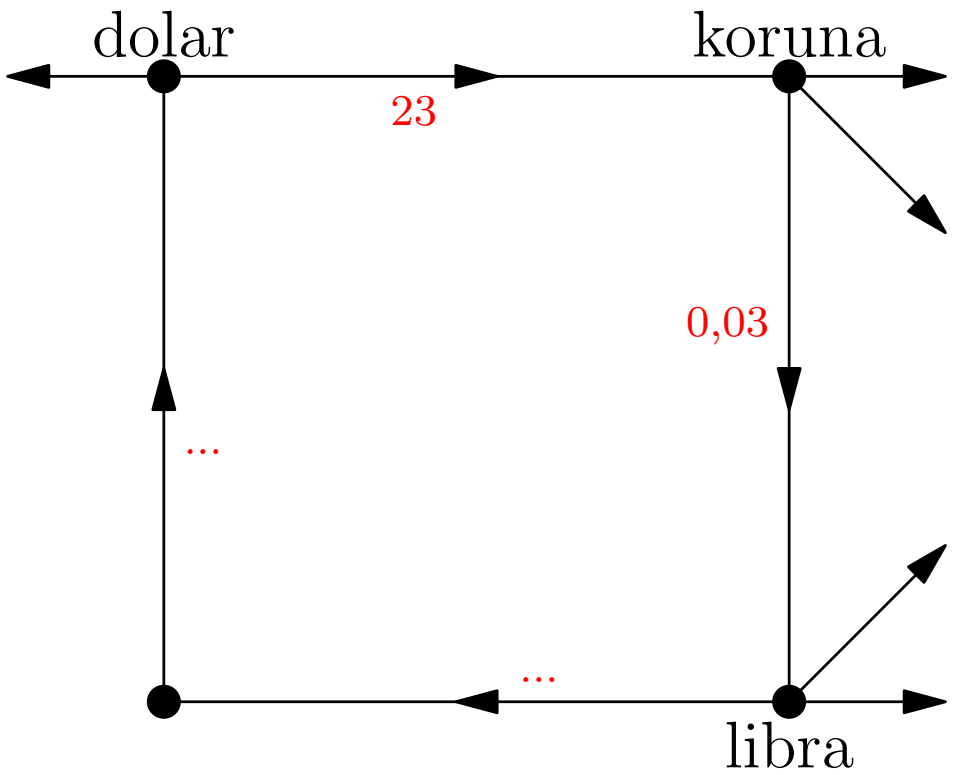
Grafy – více informací

ksp.mff.cuni.cz/kucharky/grafy/

Příklad

Máme zadán seznam kurzů měn (např. za 1 dolar dostanete 23 korun, za 1 korunu dostanete $0,03$ liber, ...). Zjistěte, zda můžete na směnách vydělat; tj. zda můžete začít např. s 1 dolarem, ten směnit na koruny, ty pak na libry, ..., a ty pak zpět na dolary, a takto skončit s více než 1 dolarem.

Převod na grafovou úlohu



Existuje cyklus se součinem vah > 1?

- Zlogaritmujeme váhy: existuje cyklus se součtem vah > 0?
- Znegujeme váhy: existuje cyklus se záporným součtem vah?

Jak rozhodnout, zda graf obsahuje záporný cyklus?

Bellman-Fordův algoritmus:

- brilliant.org/wiki/bellman-ford-algorithm/
- V knize Průvodce labyrintem algoritmů, volně ke stažení pruvodce.ucw.cz

Další zdroje

- Kuchařky KSP: ksp.mff.cuni.cz/kucharky
- Programátorská liaheň: liahen.ksp.sk
- Průvodce labyrintem algoritmů: pruvodce.ucw.cz
- MO-P FAQ: mo.mff.cuni.cz/p/otazky_a_odpovedi.html