MOP

Matěj Konečný

Abstrakt. I když MO-P znamená Matematická olympiáda, kategorie programování, k jejímu (úspěšnému) řešení není třeba umět programovat. Každé kolo je alespoň z části teoretické, přičemž teoretické úlohy jsou rozhodně blíže úlohám z matematické olympiády než nějaké práci s počítačem. Na přednášce si ukážeme, jak se dostat do celostátka MO-P jen s minimem práce.

Úvod

MO-P se v posledních letech potýká s nedostatkem řešitelů, což také znamená, že je velmi jednoduché dostat se z domácího kola do krajského a z krajského do celostátního. Například ve školním roce 2013/14 stačilo pro postup z domácího kola 5 bodů ze 40 a pro postup z krajského kola 15 bodů ze 40. Nenaučíme se, jak MO-P vyhrát ani jak vyřešit úlohy optimálně. Naučíme se pouze, jak vyřešit úlohy nějak, což pro postup do celostátka stačí.

Úloha 1. Dostaneme číslo a několik operací s ním (např. $+2$, $\cdot 6$, -100, $!$, ^{4}, ...) a chceme najít takovou posloupnost těchto operací (operace se vyhodnocují postupně zleva doprava), aby byl výsledek co největší.

Úloha 2. Máme posloupnost čísel a chceme najít takovou souvislou podposloupnost, aby měla co největší součet.

Úloha 3. Chceme vykrást zlatnictví. Každý předmět ve zlatnictví má cenu a váhu. My uneseme jenom m a chceme si vzít takové předměty, abychom si odnesli co největší bohatství.

Úloha 4. (Bonus) Máme posloupnost čísel a chceme je seřadit od nejmenšího po největší.

Co to vlastně je programování? O čem úlohy jsou?

> **Citát.** Napsal jsem tam, že vyzkoušíme všechny možnosti ve faktoriálové složitosti, a dostal jsem se na celostátko.
>
> (Totožnost autora výroku je redakci známa.)

Definice (vágní). *Program* je posloupnost kroků, má vstup a výstup. Vstup je třeba číslo, několik čísel, nějaký text, graf atd. V programu můžete používat proměnné, konstanty, základní aritmetické operace a přiřazování. Navíc můžete používat podmínky (pokud je tohle pravda, udělej a, jinak b) a cykly (dokud je tohle pravda, dělej a).

Příklad 5.

1. Načti čísla a, b.
2. Nastav $c := 0$.
3. Dokud $b > 0$:
4. Nastav $c := c + a$.
5. Sniž b o 1.
6. Vrať c.

Příklad 6.

1. Načti posloupnost čísel $a[i]$ dlouhou n.
2. Nastav $b := -\infty$.
3. Pro každé i od 1 do n:
4. Pokud $a[i] > b$:
5. Nastav $b := a[i]$.
6. Vrať b.

Příklad 7.

1. Načti posloupnost čísel $a[i]$ dlouhou n.
2. Pro každé i od 1 do n:
3. Označ j pozici minima z prvků $a[i], \ldots, a[n]$.
4. Prohoď $a[i]$ s $a[j]$.
5. Vrať posloupnost $a[i]$.

Složitost

U úloh je třeba určit časovou a paměťovou složitost. Časová složitost znamená, kolik řádově operací program provede v závislosti na velikosti vstupu. Paměťová složitost znamená, kolik řádově zabere paměti v závislosti na velikosti vstupu. Velikost vstupu je typicky třeba počet čísel nějaké posloupnosti či počet vrcholů a hran nějakého grafu.

Zajímá nás pouze, „jak rychle ta časová složitost roste“, tzn. provede-li program například $3n^2 - n + 6$ kroků, chceme jen, že to roste řádově jako n^2. Potom říkáme, že časová složitost programu je v $\mathcal{O}(n^2)$.

Odkazy

[1] KSP Kuchařka:
http://ksp.mff.cuni.cz/encyklopedie/kucharky/programatorske-kucharky.html

[2] Stránky MO-P: *http://mo.mff.cuni.cz/p/*