Korespondenční Seminář z Programování

ZAČÁTEČNICKÁ KATEGORIE

28. ročník **KSP-Z** **Duben 2016**

Po delší přestávce opět přišel čas prozradit si řešení úloh. Doufáme, že jste nad nimi strávili příjemné chvíle a pokud se Vám nějaká nevydařila, že ji zkusíte vyřešit i po termínu. Blahopřejeme k získaným bodům!

A jako obvykle se nás nebojte zeptat, pokud vám cokoliv není úplně jasné. Obrátit se na nás můžete přes fórum na našich stránkách nebo e-mailem na *ksp@mff.cuni.cz*.

Řešení třetí série začátečnické kategorie 28. ročníku KSP

28-Z3-1 Místo oslavy

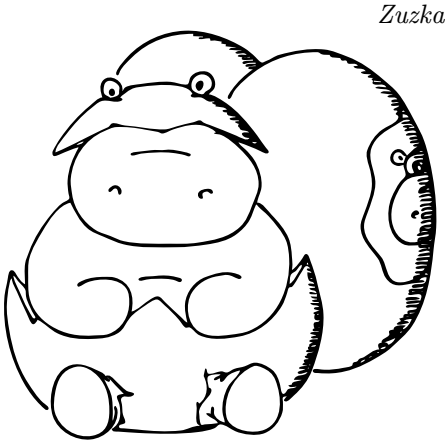
K vybrání vhodného místa pro pořádání oslavy stačí vedení školy najít první a poslední dům od začátku města, kde bydlí zájemci o oslavu. Je zřejmé, že právě obyvatelé těchto domů to budou mít nejdále. Ovšem to znamená, že stačí najít minimum a maximum v zadané posloupnosti čísel, což zvládneme během jediného průchodu vstupem.

My ale ještě chceme, aby to oba nejvzdálenější zájemci měli co nejblíže, tj. aby tato nejdelší vzdálenost byla co nejmenší. Proto budeme hledat střed mezi dvěma zmíněnými domy, vzdálenost z obou pak bude stejná.

Vzhledem k tomu, že čísla budov jsou celočíselná, chceme opět celočíselný výsledek. Pokud je vzdálenost mezi prvním a posledním domem liché číslo, jsou dvě možnosti pro konání oslavy, obě stejně výhodné. My můžeme vybrat libovolnou z nich, zvolme si vždy třeba tu s nižším číslem. Nyní bude stejný postup pro obě varianty, tedy když je ona vzdálenost liché i sudé číslo. Minimum si označíme Min, maximum jako Max, výsledná pozice sálu bude $(Max + Min)$ div 2.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/28-Z3-1.py`

Zuzka Drázdová



28-Z3-2 Zlomkovník

Z postupu, jakým byl strom tvořen, si všimneme, že pokud je čitatel menší než jmenovatel, byl poslední vybrán levý syn. Naopak, pokud je čitatel větší než jmenovatel, byl poslední pravý syn. Proč tomu tak je? Levý syn je tvořen $\frac{A}{A+B}$ a je zřejmé, že $A+B$ je větší než A (protože ve stromu jsou samá kladná čísla). Podobně u pravého syna, který je $\frac{A+B}{B}$, je $A+B$ větší než B.

Stačí nám tedy v podstatě projít zpět ke kořeni a zaznamenat, kde jsme byli. Jak taková cesta bude vypadat, zjistíme právě porovnáváním čitatele a jmenovatele. Zapamatované pořadí bude obrácené, než jaké máme vypsat v odpovědi, proto jej nesmíme zapomenout poté obrátit.

Jak přesně budeme postupovat? Nejprve porovnáme čitatele a jmenovatele, pokud je čitatel menší, tedy zlomek je ve stavu $\frac{A}{A+B}$, odečteme od jmenovatele čitatele a získáme tím otce: $\frac{A}{A+B-A} = \frac{A}{B}$. Pokud je čitatel větší, musíme k získání otce odečíst od čitatele jmenovatele: $\frac{A+B-B}{B} = \frac{A}{B}$. Podle porovnání nezapomene poznamenat, jestli jsme se jednalo o pravého nebo levého syna.

Pokračovat budeme opět k novému otci aktuálního syna. Tento postup budeme opakovat, až dokud se nedostaneme do kořene stromu. Poté nezapomeneme vypsat posloupnost R a L v obráceném pořadí a máme výsledek.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/28-Z3-2.py`

Zuzka Drázdová

28-Z3-3 Posloupnost za trest

Nejdříve samozřejmě zkusíme počítat postupně členy posloupnosti $P_1, P_2, \ldots, P_N$. První člen P_1 je triviální, P_{i+1} vyrobíme z P_i tak, že budeme procházet číslice zleva doprava a počítat, jak dlouhé úseky stejných číslic potkáme. To pokaždé zvládneme v čase $\mathcal{O}(\textit{počet cifer})$.

Zpočátku to půjde dobře, jenže cifer bude rychle přibývat. (Dokonce exponenciálně: i-tý člen má přibližně 1.304^i číslic. To dokázal John Conway fascinujícím způsobem. Pokud máte chuť na trochu pokročilejší matematiky, směle se začtěte do článku ve wikipedii[1] a odkazů, které z něj vedou.)

Nás naštěstí zajímá jen prvních K číslic výsledku. Zkusíme tedy počítat jen prvních K číslic každého členu posloupnosti. Je to trochu riskantní, protože by se mohlo stát, že při vytváření dalšího členu budeme potřebovat víc číslic předchozího členu, než jsme spočítali. Pokusem si ale můžeme ověřit, že posloupnost roste dost rychle, takže se to pro $K \geq 10$ nestane.

Z toho dostaneme řešení o složitosti $\mathcal{O}(NK)$, dost rychlé na to, aby vyřešilo všechny naše vstupy během pár desítek sekund. Přesto ho zkusíme ještě trochu zrychlit.

Označme Q_i hodnotu P_i zkrácenou na prvních K číslic. Pokud je K malé, začnou se hodnoty $Q_1, Q_2, \ldots$ brzy opakovat. Náš algoritmus si proto bude ve slovníku pamatovat všechna Q_i, která už viděl, a číhat na první opakování.

[1] `https://en.wikipedia.org/wiki/Look-and-say_sequence`

Jakmile zjistí, že $Q_i = Q_j$ pro nějaké $j < i$, musí se od i-té pozice neustále opakovat úsek $Q_j, Q_{j+1}, \ldots, Q_{i-1}$. Stačí tedy zjistit, který z opakovaných členů vyjde na hledané Q_N, a vypsat ho.

Zdálo by se, že perioda bude pro velká K dostatečně daleko, takže tento trik nepomůže. Neváhejme a zkusme to. Překvapení: pro maximální povolené $K = 300\,000$ je $Q_{55} = Q_{52}$ a zjevně to platí i pro všechna menší K. Stačí tedy spočítat prvních 55 členů posloupnosti a známe odpověď pro jakékoliv N.

Program k základnímu řešení (Python 3):
`http://ksp.mff.cuni.cz/viz/28-Z3-3.py`

Program k rychlejšímu řešení (Python 3):
`http://ksp.mff.cuni.cz/viz/28-Z3-3-magic.py`

Martin „Medvěd“ Mareš

28-Z3-4 Zbývající úkoly

Nejprve vymyslíme jednoduché řešení, a pak si jej prostým trikem velmi zrychlíme. Pojďme na to.

Začneme tím, že si načteme celý vstup. Pro každý úkol si budeme chtít pamatovat jeho délku a seznam úkolů, na kterých je závislý. Toto nejde udělat jinak, než že celý vstup načteme do paměti. V poli t uschováme délky, v poli *dep* potom seznamy úkolů, které musíme provést dříve. Jak to udělat pohodlně se můžete podívat do programu v Pythonu.

Potom si napíšeme rekurzivní funkci, která spočítá, kdy nejdříve může být úkol x dokončený:

```
def finish_time(x):
  deptime = 0
  for d in dep[x]:
    deptime = max(deptime, finish_time(d))
  return deptime + t[x]
```

V této funkci vezmeme maximum z času dokončení všech úkolů, na kterých je x závislý, a přičteme dobu trvání x. Jak ji použijeme?

Abychom si zjednodušili kód, můžeme použít podlý trik. Přidáme si neexistující úkol A, který bude trvat 0 jednotek času, zato ale bude závislý na všech ostatních. Jeho čas dokončení je pak přesně to, co po nás úloha žádá.

Jak to bude rychlé? Představte si takový vstup:

$$1 \leftarrow 2 \leftarrow 3 \leftarrow 4 \leftarrow 5$$

V takovém případě se funkce zavolá postupně s těmito parametry:

$$1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1$$

Zde je nutno si všimnout, že se funkce volá mnohokrát třeba pro jedničku, přestože její návratová hodnota bude vždy stejná. Vždyť si výsledky můžeme uložit:

```
def finish_time(x):
  if x not in fintime:
    deptime = 0
    for d in dep[x]:
      deptime = max(deptime, finish_time(d))
    fintime[x] = deptime + t[x]
  return fintime[x]
```

Jak bude vypadat seznam volání teď?

$$1, 2, 1, 3, 2, 4, 3, 5, 4$$

Nemůžeme říct, že by se nyní volala pro každé x pouze jednou. Čeho jsme se ale zbavili, je neustálé opakování „ocásků“, pro které už známe odpověď, a můžeme ji vrátit daleko dříve.

Pokud bychom chtěli časovou složitost vyčíslit, všimneme si že funkci zavoláme jednou jako závislost na virtuálním úkolu A, a jednou pro každou závislost jiného úkolu y na tomto úkolu x – to právě tehdy, když poprvé počítáme y. Dohromady víme, že závislostí je K, takže celkový počet volání bude $N + K$. N (počet úkolů) musíme započítat, protože může být klidně větší než K.

Úplně stejně se na to podíváme, pokud budeme počítat čas pro funkci samotnou. Všechna volání dohromady ve vnitřním for cyklu stráví $\mathcal{O}(K)$ času, všechno ostatní už je jen konstantní zpomalení. Proto bude mít výpočet časovou složitost $\mathcal{O}(N + K)$. Do tohoto času se vejdeme i s načtením vstupu a vypsáním výstupu, a stejnou funkcí odhadneme i prostorovou složitost.

Doteď se úloha zdála růžová. Bohužel jsme si při testování úlohy nevšimli výrazného omezení jazyka, který pro řešení úloh propagujeme – Pythonu. Ten totiž ve výchozím nastavení neumožňuje zanořit volání funkcí více než tisíckrát.

Tentokrát tedy přidáváme tři zdrojové kódy. První problém obchází tím, že nastavení změní. Bohužel, bude fungovat jen na Linuxu. O důvodech se můžeme pobavit na fóru.

Program (Python 3, Linux):
`http://ksp.mff.cuni.cz/viz/28-Z3-4-linux.py`

Druhý jej řeší tím, že nepoužívá volání funkcí, ale simuluje jej pomocí seznamu. Technika je to obecná a dá se použít vždy, jen zdrojový kód není úplně dobře čitelný.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/28-Z3-4-explicit.py`

Program (C++):
`http://ksp.mff.cuni.cz/viz/28-Z3-4.cpp`

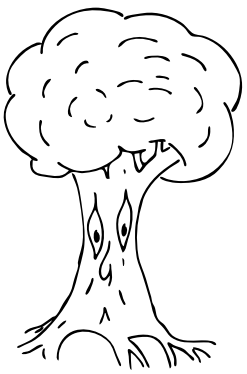
Ondra Hlavatý

28-Z3-5 Ukotvení stromu

Pro začátek předpokládejme, že žádné dva body nemají stejnou x-ovou ani y-ovou souřadnici.

Nejprve body seřadíme podle x-ové souřadnice.

Pokud jsou všechny body středově souměrné, musí být bod nejvíce vlevo souměrný s tím nejvíce vpravo. Tím pádem podle těchto dvou bodů jednoznačně určíme střed S – bude přesně v polovině mezi nimi.



Zbývá ověřit, že všechny ostatní body jsou podle S souměrné. Druhý musí být souměrný s předposledním, třetí se třetím od konce a tak dál pro všechny body. Pokud má být souměrný bod A s bodem B podle středu S, musí platit $S_x - A_x = B_x - S_x$ a zároveň $S_y - A_y = B_y - S_y$.

Pokud se jediná podmínka poruší, odpovíme, že nejsou souměrné. Když žádný test neselže, odevzdáme střed S.

Algoritmus funguje i pokud mají nějaké body stejnou x-ovou souřadnici. V takovém případě je potřeba při řazení porovnávat primárně podle x-ové a sekundárně podle y-ové souřadnice.

Proč to funguje? Protože přesně takové pořadí bychom dostali, kdybychom celou rovinu nepatrně pootočili, aby se žádné x-ové souřadnice neshodovaly.

Zbývá dořešit časovou složitost. Body, který je N, umíme seřadit v čase $\mathcal{O}(N \log N)$ třeba MergeSortem. Pokud je uchováváme v poli, bude každý z N testů trvat konstantní čas. Celková časová složitost je tedy $\mathcal{O}(N \log N)$.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/28-Z3-5.py`

Martin Španěl

28-Z3-6 Šíření drbů

Dobrým prvním krokem může být zkusit správný počet potřebných přestávek odhadnout. Proto se pokusme spočítat správné výsledky pro malý počet kamarádek.

Máme-li jen jednu kamarádku, ta zná svůj drb hned (tedy nepotřebuje ani jednu přestávku). Dvě kamarádky potřebují jednu přestávku na to, aby si vyměnily své drby. Čtyři kamarádky (jak máme v zadání) potřebují dvě přestávky. S trochou trpělivosti můžeme zkoušením zjistit, že pro osm kamarádek jsou potřeba tři přestávky.

Z toho (a s nápovědou v zadání, že máme uvažovat pouze N, která jsou mocniny dvojky) můžeme odhadnout, že potřebujeme $\log_2 N$ přestávek. Jinak řečeno, pokud je 2^k kamarádek, klepy si vymění během k přestávek.

Pojďme se tedy podívat, jestli se nám podaří najít takové přiřazení kamarádek, aby se klepy šířili tak rychle. Začněme u malého počtu kamarádek. Když máme pouze dvě kamarádky, tak stačí když se potkají během jediné přestávky. Řešení pro čtyři kamarádky máme rovnou v zadání (nejprve se potkají A–B a C–D, poté A–C a B–D).

Jak můžeme přistupovat k tomu, když je ve třídě kamarádek osm? Rozdělme si je na dvě skupiny po čtyřech a nechme nejprve rozšířit drby v rámci jednotlivých skupinek. Jak jsme ukázaly výše, aby každá kamarádka znala všechny skupinové drby, stačí nám dvě přestávky.

Všimněte si, že na rozšíření drbů mezi skupinkami nám pak stačí už jen jedna přestávka – první kamarádka z první skupinky se podělí o drby s první kamarádkou z druhé skupiny, druhá s druhou a tak dále. Každá kamarádka pak bude vědět všechny drby ze své skupiny (ty znala i před touto poslední přestávkou) a tím, že se potkala s někým z druhé skupiny, tak se dozvěděla i všechny drby z druhé skupiny.

Stejný trik můžeme použít i pro šestnáct kamarádek – stačí je rozdělit na dvě skupiny. Každá skupinka během tří přestávek rozšíří drby v rámci své skupiny (tak jak bylo ukázáno v předchozím odstavci). Během následující přestávky si pak každá kamarádka popovídá s příslušnou kamarádkou z druhé skupiny, čímž se dozví všechny drby.

Ve skutečnosti takto můžeme postupovat pro libovolný počet kamarádek. Kamarádky rozdělíme na dvě skupiny a zamyslíme se nad tím, jak rozšířit drby v rámci skupin. Potom během jedné přestávky už rozšíříme drby napříč skupinami.

Tímto způsobem spotřebujeme za každé zdvojnásobení počtu kamarádek jen o jednu přestávku více. Dostali jsme se tedy přesně na náš vytyčený cíl v počtu přestávek.

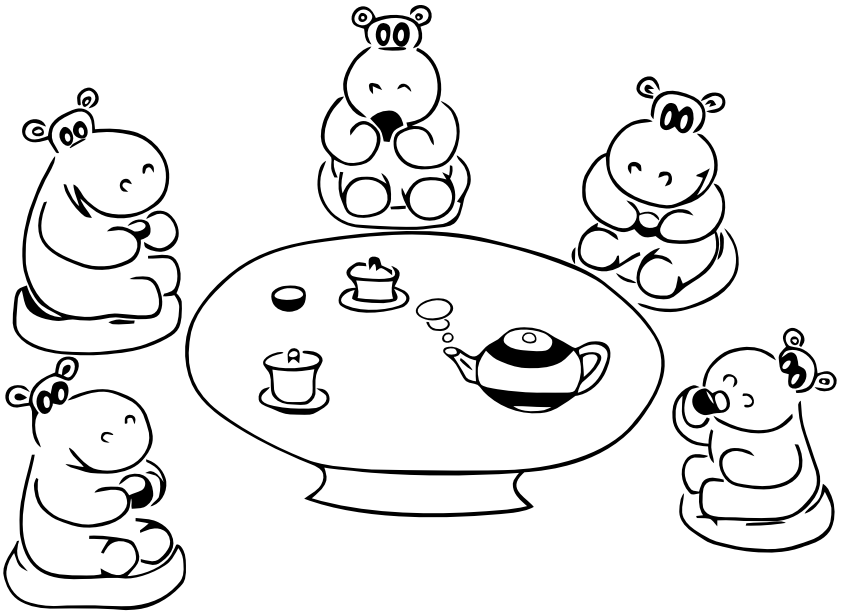
Tato technika rozdělování problémů na několik menších, které se řeší vlastně stejnou technikou, je v informatice poměrně populární a nazývá se *Rozděl a panuj*. Pokud tě zaujala a chceš se o ní dozvědět více, tak další informace najdeš v naší kuchařce.[2]

Na závěr si pojďme ještě ukázat, že rychleji to skutečně nejde. Představme si na chvíli, že už proběhlo několik přestávek. Označme si d počet drbů, které zná nejpopulárnější kamarádka (tedy ta, která zná ze všech kamarádek nejvíce klepů). Druhá nejpopulárnější tudíž zná nejvýše také d drbů. Když se tyto dvě kamarádky během další přestávky potkají, obě budou znát nejvýše $2d$ drbů.

Všimněme si, že (jelikož se jednalo o nejpopulárnější kamarádky) nikdo nebude po další přestávce znát více jak $2d$ drbů. Také si uvědomme, že toto platí pro každou přestávku. Takže, bude-li p značit počet drbů, které zná nejpopulárnější kamarádka, p se nám každou přestávku nejvýše zdvojnásobí.

Toto už nám (s tím, že na začátku je p jedna), dává právě naší formulku, že po k přestávkách bude p nejvýše 2^k. Tedy při 2^k kamarádkách potřebuje i nejpopulárnější kamarádka alespoň k přestávek.

Janka Bátoryová & Dominik Smrž



[2] `http://ksp.mff.cuni.cz/viz/kucharky/rozdel-a-panuj`

Výsledková listina třetí série začátečnické kategorie 28. ročníku KSP

	řešitel	*škola*	*ročník*	*sérií*	*Z3-1*	*Z3-2*	*Z3-3*	*Z3-4*	*Z3-5*	*Z3-6*	*série*	*celkem*
0.					8	10	10	12	12	14	66,0	198,0
1.	Václav Brož	GZborovPH	1	3	8	10	10	10	10	14	62,0	173,0
2.	Roman Bujdák	G JM Galanta	2	3	8	10	10	12	8,5	8	56,5	171,5
3.	Pavel Koch	GTěš	4	3	8	10	10	6,7	9	14	57,7	165,7
4.	Jakub Šťastný	G BO-Řeč	1	3	8	10	10	12	12	14	66,0	165,5
5.	Daniel Skýpala	GTomkovaOL	−2	3	8	10	10	12	8,5	0	48,5	161,5
6.	Michael Bausano	GTěš	4	3	8	10	10	4	3		35,0	133,5
7.	Vendula Kuchyňová	GMLerchaBO	2	3	8	10	10	12			40,0	129,0
8.	Jiří Moravčík	GUHradiště	2	7	8	10	10	12	4	5	49,0	127,0
9.	Václav Pavlíček	ZŠ Ždíred nD	0	3	8	10	10	10			38,0	111,0
10.	Dennis Pražák	GJirsíkaČB	1	3	8	10	10	4			32,0	107,0
11.	Tomáš Terem	GTajBanBys	4	6	8	10	10				28,0	106,0
12.–13.	Vojtěch Březina	GCoubTábor	−1	3	8	10		6			24,0	104,0
	Lukáš Riedel	G Bílina	4	2	8	10	10	0	7	14	49,0	104,0
14.	Martin Bencko	GOhradníPH	−1	3	8	5	10	12	1,5	4	40,5	102,5
15.	Jiří Löffelmann	GLitoměřPH	2	3	8	10	10	12			40,0	99,5
16.	Vojtěch Hudec	G_ČTřebová	2	2	8	10	10	0	4	10	42,0	96,0
17.	Michal Rickwood	G_ČTřebová	2	2	8	10	10	0	5	4	37,0	93,5
18.	Jakub Jirkal	GJungmanLT	1	7	8	10	10	10			38,0	87,0
19.	Anna Hollmannová	GSRandyJN	−1	3	8	2	10		1,5	3	24,5	82,5
20.	Martin Picek	GJirsíkaČB	1	4	8	10	10		8,5		36,5	81,5
21.	Jáchym Solecký	PORGPha	3	2	8	10	10				28,0	80,0
22.	Samuel Schneider	GTajBanBys	4	4	8	10	10	10			38,0	78,5
23.	Jindřich Dítě	ZŠKom2ŽS	0	3	8				6	14	28,0	67,5
24.–25.	Petr Dedek	MensaG	0	2							0,0	66,0
	Josef Pospíšil	GÚstavníPH	2	3	8			6			14,0	66,0
26.	Ondřej Gonzor	G Brandýs	−1	2	8	5		0		5	18,0	63,5
27.	Antonín Prantl	G Strakon	3	2	8	10	10	2	3,5		33,5	61,0
28.	Roman Solař	GJarošeBO	4	2							0,0	60,0
29.	Jan Kaifer	GČesBrod	0	6							0,0	59,5
30.	Jonáš Havelka	GJírovcČB	0	1	8	10	10	12	6	13	59,0	59,0
31.	Tomáš Troján	G Cheb	0	7	8	10	2				20,0	58,5
32.–33.	Adam Husník	GArabskáPH	2	1	8	10	10	10	6	14	58,0	58,0
	Pavel Souček	G_Nymburk	4	5							0,0	58,0
34.	Ľuboš Kolumber	SpojŠ Popr	4	6							0,0	56,5
35.–36.	Michael Olšavský	GNadKavaPH	1	1							0,0	56,0
	Petra Štefaníková	GOlgHavl	4	3	8	5					13,0	56,0
37.	Ondřej Krsička	Integra BO	0	3	8	10					18,0	55,0
38.	David Nápravník	GLitoměřPH	3	3	8						8,0	54,0
39.	Vít Gaďurek	Neuvedená	1	5							0,0	53,0
40.	David Blažek	SPŠÚžlabPH	3	1							0,0	52,0
41.	Vojtěch Kuchař	ZŠ Sobotka	−1	3	8	2	2				12,0	50,9
42.	Tomáš Domes	MendelG_OP	3	1	8	10	10		8,5	14	50,5	50,5
43.	Vojtěch Káně	G Brandýs	0	3	8	10		6			24,0	50,0
44.	Michaela Štolová	G Sokolov	4	5	8						8,0	49,5
45.–46.	Dominik Krasula	GKrnov	3	4	4				8	14	26,0	48,0
	Václav Šraier	GČeskoliPH	3	2	8	10					18,0	48,0
47.–48.	Radoslav Hašek	G_Čáslav	2	3	8						8,0	47,0
	Tomáš Novotný	G BO-Řeč	2	1	8	10	10	2	3	14	47,0	47,0
49.	Andrej Čermák	G JF Šaľa	2	4							0,0	46,0
50.	Janek Hlavatý	GJirsíkaČB	−3	9	8						8,0	44,3
51.	Michal Szymik	G Wicht	2	2	8	10	10				28,0	44,0
52.	Radek Jančík	GJarošeBO	3	3	4						4,0	43,0
53.	Ondřej Cach	ZŠPolab	0	3	8		8				16,0	40,5
54.–59.	Ondřej Baumgartner	GMost	4	1							0,0	40,0
	Ondřej Borýsek	GJarošeBO	3	1							0,0	40,0
	Hana Hladíková	GNadKavaPH	2	1							0,0	40,0
	Vojtěch Lanz	GZborovPH	2	1							0,0	40,0
	Jakub Matěna	GČeskoliPH	4	5							0,0	40,0
	Lucien Šíma	PORGPha	4	1							0,0	40,0

	řešitel	*škola*	*ročník*	*sérií*	*Z3-1*	*Z3-2*	*Z3-3*	*Z3-4*	*Z3-5*	*Z3-6*	*série*	*celkem*
60.–62.	Jan Mráz	G Holice	2	5	8	10	10	0			28,0	36,0
	Jiří Sejkora	GVoděraPH	4	3	8						8,0	36,0
	David Žáček	GZborovPH	3	4	8	10		12	6		36,0	36,0
63.	Pavel Svoboda	ZŠJílovsPH	0	3	8						8,0	32,0
64.	David Pavlík	GJarošeBO	3	1							0,0	30,0
65.–66.	Jakub Suchánek	GOpatovPHA	2	1	8	10	10				28,0	28,0
	Marek Černoch	GFPValMez	0	1							0,0	28,0
67.	Alexej Popovič	SlovanGOL	4	2							0,0	27,5
68.–70.	Ludmila Bujnovská	MendelG_OP	2	1	1	5			6	14	26,0	26,0
	Robert Jaworski	GÚstavníPH	−2	2							0,0	26,0
	Matěj Šmíd	SPŠÚžlabPH	2	3	8						8,0	26,0
71.	Jiří Kvapil	GTomkovaOL	−2	1	8	10			6		24,0	24,0
72.	Ondrej Potúček	G_GolNitra	2	1	8	10	5				23,0	23,0
73.	Filip Čermák	MendelG_OP	2	1	1				4,5	13	18,5	18,5
74.–76.	Radim Buráň	G UherBrod	1	1	8	10					18,0	18,0
	Jakub Dostál	SlovanGOL	2	3	8						8,0	18,0
	Pavel Turinský	G Brandýs	3	3	8	10					18,0	18,0
77.–80.	Jan Burda	G Holice	2	8	8	0					8,0	16,0
	Ladislav Töpfer	G DrJPekMB	1	1							0,0	16,0
	Jan Vaník	GRNK	1	1							0,0	16,0
	Adam Šánta	GJHroncaBA	1	1							0,0	16,0
81.	Milan Kubala	GTajBanBys	4	3							0,0	15,0
82.	Jan Vozár	G UherBrod	2	9	8						8,0	13,0
83.–85.	Vanda Hendrychová	GHeyrovPH	4	1							0,0	12,0
	Jakub Růžička	G_Nymburk	1	1						12	12,0	12,0
	Lenka Vincenová	GTomkovaOL	2	1							0,0	12,0
86.	Michal Mlčoch	G UherBrod	1	1	0	10					10,0	10,0
87.–92.	Martin Hofbauer	G BO-Řeč	1	1	8						8,0	8,0
	Pavel Martinec	GLesníZlín	2	1	8						8,0	8,0
	Filip Matějka	GZborovPH	2	1							0,0	8,0
	Václav Plavec	GTep	3	1							0,0	8,0
	Daniel Pluskal	G BO-Řeč	2	3	8						8,0	8,0
	Martin Zima	G Holice	2	3	8						8,0	8,0
93.	Dominik Karas	GTěš	4	2							0,0	7,0
94.	Matěj Hudec	CírkG Plzeň	4	4	4						4,0	6,5
95.–96.	Josef Martínek	GPelhřimov	2	1							0,0	5,0
	Robert Wiesner	GJosefskPH	4	1							0,0	5,0
97.–98.	Martin Hubata	GMikulášPL	0	1							0,0	2,0
	Daniel Perout	GJarošeBO	−1	1		2					2,0	2,0
99.–101.	Tomáš Baják	ZŠ VBílovice	−1	2							0,0	1,0
	Rajmund Hruška	GPošKošice	3	2	1						1,0	1,0
	Martin Kolář	GÚstavníPH	3	1	1						1,0	1,0
102.–106.	Lenka Duongová	SvobChebŠ	−2	1							0,0	0,5
	Martin Mikšík	SPŠ Bo	1	1							0,0	0,5
	Filip Novotný	GJMasar_JI	0	1							0,0	0,5
	Petr Zahradník	GaSOŠ ÚL	1	1							0,0	0,5
	Petr Šícho	GKepleraPH	−2	1							0,0	0,5