Korespondenční Seminář z Programování

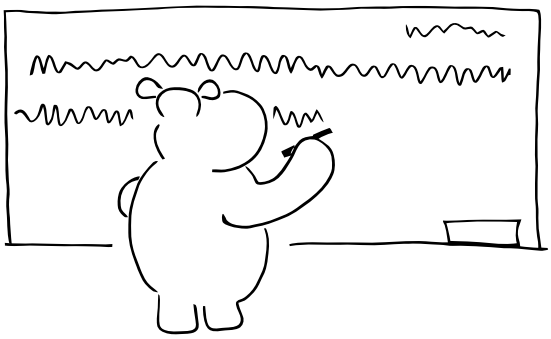
ZAČÁTEČNICKÁ KATEGORIE

35. ročník | **KSP-Z** | **Červen 2023**

Řešení páté série začátečnické kategorie 35. ročníku KSP

35-Z5-1 Jen tak mimochodem

Úlohu můžeme přirozeně rozdělit na dva podproblémy: nalezení všech míst v textu, kde začíná/končí nějaká odbočovací/vracecí fráze, a pak samotné odstranění všech (i vnořených) odboček.



Na rozehřátí vyřešíme verzi úlohy, kde se odbočky píšou do závorek, tj. odbočovací a vracecí fráze je právě jedna, a to otevírací, respektive zavírací kulatá závorka. Pak umíme lehce poznat, kde v textu začíná a končí odbočka, a můžeme se soustředit na druhý problém.

Budeme text procházet znak po znaku zleva doprava a udržovat si *úroveň zanoření,* tj. v jak hluboké odbočce se nyní nacházíme. Na začátku je úroveň nula (nejsme v žádné odbočce), a kdykoliv narazíme na otevírací závorku, zvýšíme úroveň zanoření o jedna, a kdykoliv na zavírací, naopak úroveň o jedna snížíme. Můžeme si rozmyslet, že úroveň nikdy neklesne pod nulu (to bychom museli někde mít víc zavíracích než otevíracích závorek) a že aktuální znak chceme vypsat právě tehdy, je-li aktuální zanoření na úrovni nula.

Takový algoritmus má časovou složitost $\mathcal{O}(N)$, kde N je délka vstupního textu, protože pro každý znak provádíme jen konstantně mnoho práce.

To bychom měli, jak teď ale vyřešit plnou úlohu, kde odbočovacích i vracecích frází může být více a můžou být víceznakové? To je v plné obecnosti těžké – můžou se dít všemožné divné věci, například můžeme mít dvě odbočovací fráze, kde jedna je podřetězcem druhé, nebo odbočovací a vracecí fráze, co se v textu překrývají. Naštěstí v zadání slibujeme, že žádné takové případy nenastanou.

Budeme dělat to samé, jako doposud, jen detekce odbočení nebo vrácení bude složitější. Namísto kontrolování, zda je aktuální znak `(`, musíme projít všechny odbočovací fráze a pro každou zkontrolovat, zda v textu na aktuální pozici nezačíná. V řetězcové notaci: ptáme se, zda $S[i : i+|O|] = O$, kde i je aktuální pozice, S vstupní řetězec, O aktuální fráze a $|O|$ její délka. Podobně pro kontrolu vrácení projdeme všechny vracecí fráze a zkontrolujeme, zda před aktuální pozicí *končí,* tj. zda $S[i-|V| : i] = V$. Můžete si rozmyslet, proč by byl výstup špatně, kdybychom se místo toho opět ptali, zda V na aktuální pozici začíná. Při implementaci si v obou případech si také před samotným porovnáváním chceme zkontrolovat, že dané podřetězce existují (a indexy nám z řetězce nevylézají).

Jaká je časová složitost? Testování, zda fráze délky F končí nebo začíná na aktuální pozici, provádíme v $\mathcal{O}(F)$. Pro každou pozici takto testujeme všechny odbočovací a vracecí fráze, takže označíme-li jako Z celkovou délku všech odbočovacích a vracecích frází na vstupu, strávíme na každém znaku $\mathcal{O}(Z)$ času, a celková časová složitost je tedy $\mathcal{O}(NZ)$.

To sice není optimální, ale v našich vstupech jsou odbočovací i vracecí fráze krátké a je jich málo, a proto toto řešení stačilo na plný počet bodů.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/35-Z5-1.py`

Aho-Corasicková a spol.

Plný počet bodů nás ale nemůže zastavit! Pokud víte o existenci algoritmů na hledání v textu, jako například KMP, algoritmus Aho-Corasickové, nebo Rabinův-Karpův algoritmus (mimochodem, můžete se o nich dočíst víc v naší kuchařce[1]), pak si možná říkáte, že by s jejich pomocí šlo náš algoritmus zrychlit. A vskutku!

Nám se konkrétně bude hodit algoritmus Aho-Corasickové. Nebudeme zabíhat do jeho popisu, ale důležité je, co dělá: na vstupu dostane dlouhý text a několik výrazů, které v něm chceme najít, a na výstupu nám pro každý výraz dá seznam všech pozic v textu, kde se vyskytuje. Jeho časová složitost je $\mathcal{O}(N + Z + V)$, kde N je délka textu, Z součet délek hledaných výrazů a V celkový počet nalezených výskytů. V našem případě je nejvýše jeden výskyt na každé pozici, tudíž naše časová složitost je $\mathcal{O}(N + Z)$.

Pořídíme si pole stejné délky jako vstupní text, na počátku plné nul, kde si na každý index chceme poznamenat, zda tam začíná $(+1)$ nebo končí (-1) nějaká odbočka (nebo ani jedno). Za tím účelem pustíme Aho-Corasickovou na náš vstupní text a odbočovací a vracecí fráze a pak pro každý výskyt na příslušnou pozici v poli buď přičteme, nebo odečteme jedničku. (Nesmíme zapomenout, že u vracecích frází nás zajímá index, kde fráze končí, nikoliv začíná.) Zbytek algoritmu pak bude v podstatě totožný jako na začátku.

Jak už jsme slíbili, časová složitost Aho-Corasickové bude $\mathcal{O}(N+Z)$, což je zároveň i časová složitost celého algoritmu.

Poznámka na závěr: Python (a plno dalších programovacích jazyků) ve standardní knihovně nemá funkci pro Aho-Corasickovou (`str.find` je sice také lineární, ale umí hledat jen jeden výraz najednou). Pokud si ji nechceme sami implementovat a nechceme/nemůžeme použít nějakou knihovnu, můžeme si pomoct malým trikem a použít *regulární výrazy.* Ty jsou ještě mnohem silnější než Aho-Corasicková (umožňují například věci jako „najdi výraz, co začíná na `Uáá`, pak obsahuje libovolné množství `á` a končí na alespoň

[1] `http://ksp.mff.cuni.cz/viz/kucharky/hledani-v-textu`

jeden vykřičník“) a dalece přesahují rámec tohoto řešení. Důležité je, že v regulárních výrazech umíme zapsat výraz „najdi podřetězec, který se rovná řetězci R_1, nebo řetězci R_2, nebo…“, a že ačkoliv toho obecně neumíme tolik říct o jejich časové složitosti (záleží na konkrétní implementaci regexového enginu), v praxi – a speciálně v této úloze – nám můžou stačit. Pro implementační detaily se můžete podívat do vzorového řešení.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/35-Z5-1-regexp.py`

Úlohu připravili: Jan Černohorský, Říša Hladík, Vojta Káně, Michal Kodad

35-Z5-2 Zlaté stránky

Nový název budeme vymýšlet od prvního písmenka – to ovlivní abecední pořadí nejvíce. Vlastně na něm záleží „nekonečněkrát“ více než na druhém (`azzz` je abecedně před `baaa`) a to platí pro každá dvě po sobě jdoucí písmenka. Z toho vyplývá, že chceme při každém kroku sestavování názvu vybrat to aktuálně nejvhodnější písmenko, jelikož má „nekonečněkrát“ větší vliv než písmenka na všech následujících pozicích.

Takovému algoritmu se říká *hladový,* protože v každém kroku „chamtivě zhltne“ (rozuměj použije) lokálně nejlepší řešení. (Více o nich v naší kuchařce.[2]) A jak jsme se dozvěděli ve čtvrté úloze této série, tak hladové algoritmy skoro nikdy nefungují, proto musíme na závěr pečlivě ověřit, že nás zhltnutí lokálně nejlepší možnosti nepřipraví o celkově nejlepší řešení.

Jak tedy bude vypadat celkový algoritmus? Dokud to je možné, bereme písmenka shodující se s oficiálním názvem Hipporty (dále jen H). Jakmile dojdou, vezmeme abecedně nejbližší dostupné písmenko. Taková můžou být jedno nebo dvě: jedno abecedně dřívější a jedno pozdější, a my si díky zadání můžeme vybrat libovolné z nich. BÚNO (bez újmy na obecnosti) řekněme, že jsme si vybrali to dřívější; kdyby náhodou nebylo k dispozici a my museli vybrat nejbližší pozdější, bude algoritmus stejný, jen bychom všude prohodili slova „menší“ a „větší“.

Když teď pokračujeme a vybíráme další písmena, chceme vždy abecedně co největší, protože ať budeme dělat, co chceme, vždy už budeme abecedně před H. Touto metodou už můžeme sestavit zbytek názvu.

Zbývá rozmyslet si správnost. Jak by musel vypadat řetězec, který se abecedně vejde mezi náš výstup a H? Určitě se s H nemůže shodovat ve více počátečních písmenkách než my (víme, že by nějakého písmenka musel použít víc než je k dispozici), což znamená, že se od našeho výstupu poprvé liší až někde v druhé části, a to použitím abecedně většího písmenka. Jenže my jsme v každém okamžiku brali největší přípustné písmenko, tudíž i to je nemožné, a takový řetězec tedy nemůže existovat, čímž je důkaz správnosti hotov.

Časová složitost je lineární s délkou názvu společnosti a velikostí abecedy, tedy $\mathcal{O}(|\Sigma| \cdot |H|) = \mathcal{O}(26 \cdot |H|) = \mathcal{O}(|H|)$. S opatrnější implementací jde dosáhnout složitosti $\mathcal{O}(|\Sigma| + |H|)$, ale v této úloze to nebylo potřeba.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/35-Z5-2.py`

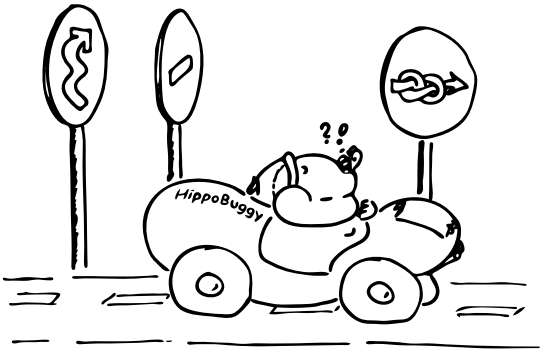
Co kdyby nám zadání nedalo na výběr mezi nejbližším předcházejícím a následujícím názvem, ale museli jsme vždy najít ten předcházející? Pak může naše řešení selhat na okrajovém případu: v okamžiku, kdy se přestaneme shodovat s H a chceme použít abecedně nejbližší dřívější písmenko, se může stát, že takové písmenko neexistuje, protože už nám všechna abecedně menší písmenka došla. Pak se budeme muset vrátit o pár písmenek zpátky (a pokaždé kontrolovat, zda k aktuálnímu znaku už abecedně nižší písmenko máme), a výsledný řetězec se pak nebude nutně shodovat s H na co nejvíce písmenkách. Jako příklad můžete uvážit H = `caaa` s dostupnými písmenky `a`, `a`, `b`, `c`, `z`, `z`, `z`. Algoritmus se zastaví s částečným řetězcem `caa`, ale výsledný řetězec bude `bzzz`. Takže jak vidno, stačí mít jen o trochu těžší úlohu a hladový algoritmus zase nefunguje. Můžete si rozmyslet, že naznačený „polohladový“ algoritmus již správný je; níže si můžete přečíst jeho implementaci.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/35-Z5-2-full.py`

Úlohu připravili: Říša Hladík, Vojta Káně, Martin Koreček, Vítek Skalický

35-Z5-3 Karneval

Uvažujme na chvíli, že je karneval již nějak roztahaný po městě, ale dál se nepohybuje. V tom případě můžeme použít prohledávání do šířky, jak lze najít např. v naší kuchařce.[3] Tím dokonce najdeme cestu nejkratší.



Jenže co když se průvod pohybuje? Pak máme obecně problém – v úlohách na hledání cest, kde se průběžně mění prostředí, nemusí platit předpoklady nutné pro BFS. Např. se může vyplatit chvíli počkat (třeba než projede vlak), abychom následně šli přímo, než dočasnou překážku začít dlouze obcházet. Tím přestává platit, že čím dříve na nějaké políčko dojdeme, tím určitě lépe. Je to ale případ karnevalu? Není! Průvod má naštěstí jednu důležitou vlastnost – jakmile na políčko jednou vstoupí, již nikdy neodejde. Pokud tedy odněkud vede cesta do cíle v čase t, jistě byla k dispozici v časech $< t$ (a možná byly i nějaké kratší cesty).

Po této rozvaze tedy můžeme s klidem použít téměř standardní prohledávání do šířky. Akorát ve frontě si musíme u políček pamatovat i to, jakého časového okamžiku se týkají. Až začneme zpracovávat políčko s vyšším časem, než mělo předchozí, posuneme karneval o jeden krok podle pravidel. Na políčka s karnevalem nesmíme vstoupit, tudíž je

[2] `http://ksp.mff.cuni.cz/viz/kucharky/zakladni-algoritmy`
[3] `http://ksp.mff.cuni.cz/viz/kucharky/grafy`

do fronty nevkládáme. Tuto podmínku však musíme ověřit i ve chvíli, kdy se políčko z fronty rozhodneme zpracovat, karneval na něj totiž mohl právě vstoupit. Jinak je průběh algoritmu standardní.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/35-Z5-3.py`

Úlohu připravili: Vojta Káně, Martin Koreček, Vítek Skalický, Dan Skýpala

35-Z5-4 Protipříklad

V úloze 35-Z5-2 hladový algoritmus fungoval, ale na tuto úlohu opravdu nezabere. Jak sestrojit protipříklad?

Pokusíme se nastražit na hladový algoritmus past. Najdeme dvojici řetězců, které mají velký překryv, ale jejich slepení zablokuje jiná dvě slepení. Zablokovaná slepení musí každé mít menší překryv, ale v součtu větší, než to vybrané.

Zvolíme tyto řetězce:

$A =$ `abcdef`
$B =$ `cdefgh`
$C =$ `efghbcd`
$Z =$ `bcdefz`

Velikosti překryvů vyjdou následovně:

	A	B	C	Z
A	–	4	2	5
B	0	–	4	0
C	0	2	–	3
Z	0	0	0	–

Hladový algoritmus tedy slepí $A + Z$ na $D =$ `abcdefz`. Z toho vyjde další tabulka překryvů:

	B	C	D
B	–	4	0
C	2	–	0
D	0	0	–

Teď lepíme $B + C$ na $E =$ `cdefghbcd` a dostaneme:

	D	E
D	–	0
E	0	–

Teď máme na výběr buď $D + E =$ `abcdefzcdefghbcd` nebo $E + D =$ `cdefghbcdabcdefz`, obojí o 16 znacích.

Pokud bychom ovšem neskočili na špek, mohli bychom slepit $A + B =$ `abcdefgh` a $C + Z =$ `efghbcdefz` (překryv celkem o 7). Finálním slepením získáme nadřetězec `abcdefghbcdefz`, který má jenom 14 znaků.

Hladový algoritmus jsme tedy usvědčili z neoptimálnosti.

Úlohu připravil: Martin „Medvěd“ Mareš

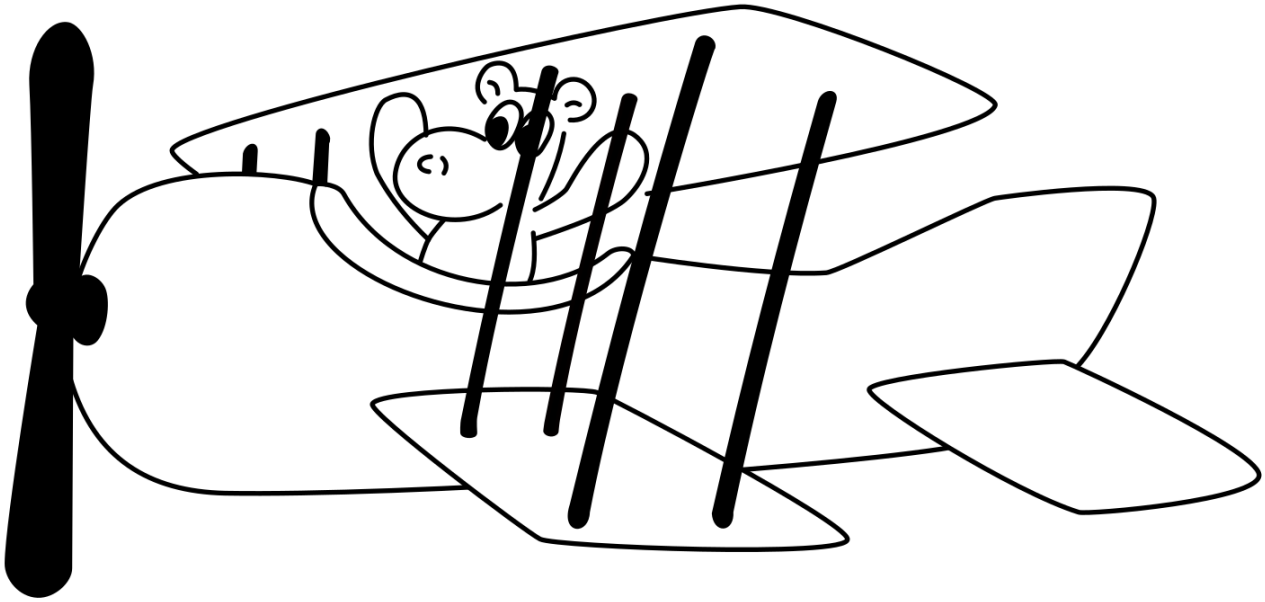
Výsledková listina páté série začátečnické kategorie 35. ročníku KSP

	řešitel	*škola*	*ročník*	*sérií*	*Z5-1*	*Z5-2*	*Z5-3*	*Z5-4*	*série*	*celkem*
0.					8	12	14	10	44,0	220,0
1.	Richard Dobíšek	MensaG	2	5	8	12	14		34,0	209,0
2.	Michael Ambros	GTomkovaOL	0	6	8	12	14	10	44,0	201,0
3.	Albert Bakoč	GZborovPH	2	8	8		14		22,0	192,5
4.	Anna-Kristina Migel	GNAlejíPH	0	10	8	12		7	27,0	191,0
5.	Martin Vagner	GVoděraPH	0	5	8	12	14	10	44,0	182,0
6.	Martin Skýpala	GJŠkodyPŘ	3	5	2		2		4,0	171,0
7.–8.	Filip Sichrovský	GČesLípa	1	5	8	12	14		34,0	152,0
	Petr Šišlák	GZborovPH	2	4					0,0	152,0
9.–10.	Jakub Binter	GČeskáČB	0	5	8	0			8,0	149,0
	Štěpán Mikéska	GJarošeBO	4	4					0,0	149,0
11.	Petr Starý	GJírovcČB	1	10	8	12	14	10	44,0	145,0
12.	Jáchym Löwenhöffer	GEvolutionJM	2	10	8	4	0	7	19,0	141,0
13.	Vít Ungermann	GZborovPH	2	4					0,0	136,0
14.	Jakub Štefan	GMělník	4	6	8	12		10	30,0	134,0
15.	Petr Karlík	GVoděraPH	0	5	8	0		10	18,0	132,0
16.	Matěj Hošek	GVolgogrOS	1	10	8	0	14		22,0	131,0
17.–18.	Erik Ježek	SPŠSmíchov	1	3					0,0	130,0
	Tomáš Kraus	KřesťGPH	1	5	8	12	14		34,0	130,0
19.	Michal Budai	G JGJ PH	−1	6					0,0	122,0
20.	Kateřina Vomelová	GÚstavníPH	3	7	8	12			20,0	117,0
21.–22.	Pavel Altmann	GMikulášPL	4	13	8	12	0		20,0	112,0
	Zuzana Aubrechtová	GHeyrovPH	4	9					0,0	112,0
23.	Petr Slonek	GJarošeBO	4	3					0,0	109,0
24.	Martin Kudrna	GKepleraPH	2	3					0,0	107,0
25.	Matúš Púll	GZborovPH	3	9					0,0	106,0

	řešitel	*škola*	*ročník*	*sérií*	*Z5-1*	*Z5-2*	*Z5-3*	*Z5-4*	*série*	*celkem*
26.	Daniel Culliver	GZborovPH	3	3					0,0	104,5
27.	Jakub Dobiáš	GZborovPH	2	4					0,0	104,0
28.	Lukáš Franta	GZborovPH	1	3					0,0	103,0
29.	Vojtěch Lančarič	SPŠG Třebešín	4	8					0,0	100,0
30.	Marian Šámal	GMělník	3	4					0,0	96,0
31.	Alexandr Bihun	GJírovcČB	3	10	2	0			2,0	94,0
32.	Lucian Poljak	GJŠkodyPŘ	1	6	0				0,0	92,0
33.	Petr Kroča	G UherBrod	2	12					0,0	84,0
34.	Ondřej Pupík	GRožnovPR	3	6					0,0	82,0
35.	Erik Sabol	GČeskoliPH	3	13					0,0	73,0
36.	Jakub Hampl	GMělník	3	9					0,0	68,0
37.–38.	Lubomir Habarta	G UherBrod	2	3					0,0	65,0
	Filip Kolomazník	GMnichHrad	4	2					0,0	65,0
39.	Petr Laškevič	GNAlejíPH	2	3	8	12			20,0	63,0
40.–41.	Luka Králík	GArc	2	7	8	0		1	9,0	62,0
	Kryštof Maxera	GJírovcČB	2	14					0,0	62,0
42.	Kateřina Doubková	GNAlejíPH	4	5					0,0	60,0
43.	Jan Theodor Hrdý	G UherBrod	2	3					0,0	59,7
44.	Daniel Babický	GHořovice	4	2					0,0	58,3
45.–47.	Jan Bradáč	GBoskovice	−1	2					0,0	57,0
	Michal Hrbek	GZborovPH	1	3					0,0	57,0
	Matyáš Vrňák	SŠPUHodonín	2	4					0,0	57,0
48.	Miroslav Kolouch	GJírovcČB	3	5					0,0	55,0
49.–50.	Richard Tichý	SPŠSmíchov	1	6					0,0	54,0
	Ondřej Tulach	GZborovPH	2	2					0,0	54,0
51.–52.	Michal Martínek	GÚstavníPH	2	3					0,0	53,0
	Matěj Smetana	AkademGPH	2	2					0,0	53,0
53.	Lída Kačenková	GBudějovPH	4	3					0,0	52,0
54.	Kryštof Nondek	GZborovPH	2	2					0,0	51,0
55.	Karel Novák	GNVPlániPH	2	4	1,7				1,7	49,7
56.	Tadeáš Těhan	GVolgogrOS	1	2					0,0	49,0
57.	Adam Červenka	GJarošeBO	4	2					0,0	48,0
58.	Ondřej Novák	G Brandýs	1	5	2				2,0	47,0
59.	Oliver Petrovič	GPároNitra	3	2					0,0	46,0
60.–61.	Matyáš Sirotek	GJirsíkaČB	4	3					0,0	45,0
	Václav Tichý	GKepleraPH	3	2					0,0	45,0
62.–65.	Šimon Hanák	CMG Brno	0	7	2				2,0	44,0
	Filip Jarolím	G Wicht	1	4	2				2,0	44,0
	Vít Kaděra	G Wicht	1	2					0,0	44,0
	Beata Šišláková	GZborovPH	4	1					0,0	44,0
66.	Adam Jahoda	GKepleraPH	4	9					0,0	42,0
67.–68.	Dominik Dembinný	ZŠMR Kladno	−1	4					0,0	41,0
	Honza Kocourek	ParkLane	3	4					0,0	41,0
69.–70.	Matěj Bittner	GArabskáPH	3	1					0,0	40,0
	Antonín Zlevor	GVídeňskBO	1	2					0,0	40,0
71.	Jan Tichon	GZborovPH	1	2					0,0	39,0
72.	Adam Peer	SPŠSmíchov	1	2	8	0			8,0	38,0
73.–75.	Marek Plachý	GJateční ÚL	4	5					0,0	36,0
	Sylvie Troubilová	BiGy Žďár	2	3					0,0	36,0
	Jakub Vlček	GPříbor	4	1					0,0	36,0
76.	Martin Müller	GZborovPH	2	5					0,0	35,7
77.–78.	Thuy Linh Bui	GHořovice	4	1					0,0	35,0
	Adam Houdek	SOŠ Březová	−2	1					0,0	35,0
79.	Michal Glasnák	???	0	1	8	12	14		34,0	34,0
80.–81.	Štěpán Dvořák	???	2	1	8	0	14	10	32,0	32,0
	Jan Prosecký	GNoMěsNMor	4	5					0,0	32,0
82.–83.	Jakub Kodym	GZborovPH	4	1					0,0	31,0
	Jakub Kornel	GRožnovPR	4	2					0,0	31,0
84.–87.	Adam Kolník	SSŠVTPraha	4	9					0,0	30,0
	Petr Němec	G Wicht	1	1					0,0	30,0
	Patrik Prítrský	GGrössBA	2	1					0,0	30,0
	Ondřej Sedláček	GOPavla PH	2	2					0,0	30,0

	řešitel	*škola*	*ročník*	*sérií*	*Z5-1*	*Z5-2*	*Z5-3*	*Z5-4*	*série*	*celkem*
88.	Ondřej Nevěřil	GZábřeh	1	2	5	12		10	27,0	29,0
89.	Samuel Bloomfield	GNAlejíPH	0	2					0,0	28,3
90.–92.	Daniel Kubík	BiGy Žďár	2	1					0,0	26,0
	Jonáš Menšík	GJŠkodyPŘ	1	1					0,0	26,0
	Jan Roubal	GPacov	3	1					0,0	26,0
93.–94.	Olga Cinková	ArcibisGPH	3	12					0,0	25,0
	Michal Mentzl	G UherBrod	3	1					0,0	25,0
95.	Samuel Luis Štúber	GJirsíkaČB	4	2					0,0	24,0
96.–97.	Jozef Remiš	G Bilíkova	4	3					0,0	23,0
	Jakub Salinger	GZborovPH	2	4					0,0	23,0
98.	Jaromír Trnka	VOŠVarnsdorf	2	1					0,0	22,0
99.	Vít Olšovec	GPřípotoPH	1	7					0,0	21,7
100.–101.	Jan Straka	VOSPŠŽďár	3	5					0,0	21,0
	Kryštof Tahal	GUBalvanJN	4	1					0,0	21,0
102.	Daniel Kratochvíl	GTel	2	1					0,0	20,0
103.–105.	Michal Mík	SSŠVTPraha	2	4					0,0	18,0
	Radim Novák	GZborovPH	3	3					0,0	18,0
	Matej Stepan	SPSŠ Kladno	2	1					0,0	18,0
106.–111.	Adam Jirásek	G Brandýs	3	3					0,0	17,0
	Kryštof Kadlčák	1.ITGPH	2	1					0,0	17,0
	Andrea Mikulová	BGOstrava	4	3					0,0	17,0
	David Pacák	G Brandýs	2	4					0,0	17,0
	Lukáš Stanovský	PORGPha	−2	2					0,0	17,0
	Jáchym Tuma	G FrýdlNOs	2	7					0,0	17,0
112.–114.	Kryštof Marek	SGPCE	3	6					0,0	16,0
	Radek Zach	SPŠSmíchov	1	3					0,0	16,0
	Martin Zeman	GZborovPH	1	1					0,0	16,0
115.	Jaromír Obitko	GArabskáPH	1	3					0,0	15,3
116.–117.	Adam Bureš	SPŠ Přerov	3	4					0,0	14,0
	Filip Kopecký	SPŠDMasPH	3	2	2	0			2,0	14,0
118.–120.	Vladimír Jančár	GRaymanaPV	4	1					0,0	13,0
	Jakub Kopčil	GMikulášPL	4	10					0,0	13,0
	Matyáš Martinek	GLesníZlín	3	2	5				5,0	13,0
121.–122.	Radek Bláha	GČeskáČB	1	7		12			12,0	12,0
	Matěj Zábranský	GZborovPH	3	1					0,0	12,0
123.–129.	Jan Holý	GZborovPH	1	1					0,0	9,0
	Tomáš Kazimír	GNPr	3	1					0,0	9,0
	Emil Pashayev	GZborovPH	2	1					0,0	9,0
	Josef Randacek	ZŠHusovaLI	0	1					0,0	9,0
	Richard Smutny	GZborovPH	2	1					0,0	9,0
	Honza Stanovsky	GZborovPH	−2	1					0,0	9,0
	Norbert Suchý	GZborovPH	2	2					0,0	9,0
130.–143.	Ondrej Bondor	VOŠVarnsdorf	1	1					0,0	8,0
	František Borisjuk	SPŠEMasLI	3	1					0,0	8,0
	Filip Cába	GEBenešeKL	2	2					0,0	8,0
	Šimon Durda	PORG Ostrava	2	4					0,0	8,0
	Michal Forgó	SŠKlatovskáPL	1	1	8	0			8,0	8,0
	Radim Guichen	GJírovcČB	1	2					0,0	8,0
	Lukáš Hlaváč	EupSchoolLux	2	1	8				8,0	8,0
	Michaela Kontrišová	GJHroncaBA	2	1					0,0	8,0
	Jan Kotovský	GPísnickáPH	4	16					0,0	8,0
	Julie Krejčí	PraKonz	3	2					0,0	8,0
	Filip Neubauer	AkademGPH	3	3					0,0	8,0
	Thomas Riedle	BRG APP	4	13					0,0	8,0
	Matěj Rýdl	SPŠSmíchov	1	1					0,0	8,0
	Petr Zaoral	GTep	1	2					0,0	8,0
144.–147.	Vojtěch Janáček	GFXŠaldyLI	1	1					0,0	7,0
	Ondřej Jurčík	GZborovPH	2	1					0,0	7,0
	Daniel Kašpárek	GNadŠtolPH	1	1					0,0	7,0
	Kateřina Přibylová	GZborovPH	2	1					0,0	7,0
148.–149.	Jan Koška	GJírovcČB	3	9					0,0	6,0
	Tomáš Wróbel	G Wicht	0	1					0,0	6,0

	řešitel	*škola*	*ročník*	*sérií*	*Z5-1*	*Z5-2*	*Z5-3*	*Z5-4*	*série*	*celkem*
150.–151.	Magdaléna Juříková	GVídeňskBO	1	2					0,0	5,0
	Jakub Kaman	ZŠŠtúraBA	−1	1					0,0	5,0
152.–154.	Filip Bujdák	GPároNitra	2	1					0,0	4,0
	Tomáš Pražák	GJSeiferPH	2	7					0,0	4,0
	Jozef Smolár	GNámestovo	2	1					0,0	4,0
155.–161.	Lukáš Gramskopf	GMělník	2	2					0,0	3,0
	Alexandr Matveev	GZborovPH	2	1					0,0	3,0
	Jaroslav Guryča	G UherBrod	4	1					0,0	3,0
	Jan Hradil	G UherBrod	4	1					0,0	3,0
	Adam Kovařík	G UherBrod	4	2					0,0	3,0
	Monika Rozínková	GMělník	2	1					0,0	3,0
	Martin Starý	G UherBrod	4	2					0,0	3,0
162.	Jolana Štraitová	GBudějovPH	4	3					0,0	2,7
163.	Artur Holindak	GMercuryBA	4	1					0,0	2,3
164.–170.	Michal Hruboš	G UherBrod	4	1					0,0	2,0
	Jáchym Němeček	G Brandýs	1	1	2				2,0	2,0
	Jan Palma	GSOŠRok	1	1					0,0	2,0
	Michal Svoboda	G UherBrod	4	1					0,0	2,0
	Filip Šimek	GTurnov	4	4					0,0	2,0
	Jiří Špelina	G Brandýs	1	1	2				2,0	2,0
	Tomáš Zerzánek	GOpenGaBab	0	1					0,0	2,0
171.	Lukáš Podávka	G Brandýs	1	1					0,0	1,0



KSP pro vás připravují studenti Matematicko-fyzikální fakulty Univerzity Karlovy.
Realizace projektu byla podpořena Ministerstvem školství, mládeže a tělovýchovy.

Webové stránky: *https://ksp.mff.cuni.cz/*
E-mail: *ksp@mff.cuni.cz*
Organizátoři a kontakty: *https://ksp.mff.cuni.cz/kontakty/*