Korespondenční Seminář z Programování

34. ročník | **KSP** | **Duben 2022**

Vzorová řešení čtvrté série třicátého čtvrtého ročníku KSP

34-4-1 No pun indented!

Počet validních odsazení budeme počítat postupně pro každý prefix[1] řádků a každou úroveň odsazení posledního z nich.

Pro první řádek existuje pouze jediný způsob, jak jej odsadit – nemůžeme jej odsadit vůbec, protože se zatím určitě nenacházíme za žádným řídicím příkazem.

Nyní si představme, že jsme již spočítali všechny možné způsoby, kterými lze odsadit prvních i řádků. Jak můžeme z tohohle počtu vyjít, aby se nám podařilo zjistit možnosti také pro prvních $i+1$ řádků?

Zkusíme každou z těchto možností vzít a vyzkoušet pro ni, zda může být $(i+1)$-ní řádek odsazen o $0, 1, \ldots$ úrovní tak, aby vzniklý kód byl validně odsazen. Maximální úroveň odsazení je rovna počtu řídicích příkazů, jejich počet můžeme shora odhadnout počtem řádků na vstupu, tedy N. To je sice nadsazený odhad, ovšem řádově jich tolik být může, asymptotickou složitost ani správnost řešení to tedy neovlivní.

Všimneme si, že pokud je prvních i řádků již validně odsazeno, nemusíme znovu procházet celý kód. Stačí nám pouze zkontrolovat, zda je $(i+1)$-ní řádek správně odsazen vzhledem k i-tému řádku.

Řekněme, že $(i+1)$-ní řádek má mít odsazení úrovně ℓ. Jaká všechna odsazení mu mohou předcházet?

Musíme rozlišit dvě situace:

1) *Na i-tém řádku se nachází řídicí příkaz:* Úroveň odsazení i-tého řádku může být jedině $\ell - 1$.
2) *Na i-tém řádku se nachází normální příkaz:* Odsazení i-tého řádku může být úrovně $\ell, \ldots, N$.

Abychom tedy správně spočítali všechny možnosti odsazení prvních $i+1$ řádků, musíme si také pamatovat, kolika způsoby lze odsadit prvních i řádků pro konkrétní úroveň odsazení i-tého řádku.

Zkusíme námi odvozený vztah ještě vyjádřit vzorcem. Nechť $C[i][k]$ značí počet možností odsazení prvních i řádků kódu, přičemž i-tý řádek má mít k-tou úroveň odsazení.

Potom speciálně platí:

$$C[0][0] = 1, \quad C[0][i] = 0 \text{ pro } i > 0$$

čili první řádek určitě nemůžeme odsadit jinak než tak, že nepoužijeme žádné odsazení.

Jinak obecně platí:

$$C[i][k] = \begin{cases} C[i-1][k-1] & \text{řádek } i-1 \text{ je řídicí příkaz} \\ \sum_{\ell=k}^{N} C[i-1][\ell] & \text{jinak.} \end{cases}$$

Odpověď se potom skrývá v hodnotě $C[N][0]$ – do ní totiž bude posčítaný celý poslední řádek

$$C[N-1][0],\ C[N-1][1],\ \ldots,\ C[N-1][N].$$

Časová složitost řešení záleží na tom, jakým způsobem hodláme počítat součet možností na řádku, kterému předchází normální příkaz. Pokud si součet rozepíšeme, můžeme si všimnout, že

$$C[i][k] = C[i-1][k] + C[i][k+1].$$

Když tedy jednotlivé hodnoty v C budeme v rámci řádku počítat odzadu, zvládneme každý způsob odsazení spočítat v $\mathcal{O}(1)$ operacích. Naše řešení tedy vykoná $\mathcal{O}(N^2)$ operací (konstantně mnoho operací pro každou hodnotu z tabulky $\mathcal{O}(N) \times \mathcal{O}(N)$).

Jelikož nám ke spočtení kombinací pro každý následující řádek stačí vycházet pouze z hodnot v předchozím řádku, můžeme si udržovat pouze dvě pole o velikosti $\mathcal{O}(N)$, jejichž obsahy budeme po zpracování každého řádku prohazovat.

Pokud navíc budeme dostatečně opatrní v přepisování počítaných hodnot, vystačíme si dokonce s jediným polem. Celková paměťová složitost našeho řešení v obou případech bude $\mathcal{O}(N)$.

Program (C):
`http://ksp.mff.cuni.cz/viz/34-4-1.c`

Úlohu připravili: Jirka Kalvoda, Kristýna Petrlíková, Ondra Sladký

34-4-2 Písemka z analýzy

Budeme si udržovat množinu M spolužáků, kteří ještě neudělali chybu. Každý den se jich zeptáme a vybereme si tu odpověď, kterou řekla většina. Bude-li to půl na půl, zvolíme libovolně.

Pokaždé, když se naše předpověď nevyplní, odstraníme z M všechny spolužáky, kteří předpovídali špatně. Jelikož jsme se řídili většinou, množina M se zmenší alespoň dvakrát.

Ukážeme, že tato strategie udělá nejvýš $\log_2 n$ chyb, kde n je počet spolužáků. Na počátku jsou v M všichni spolužáci, po k chybách klesne velikost M na nejvýše $n/2^k$. Kdyby nastalo $k > \log_2 n$, musela by M už být prázdná. To ale není možné, protože vždy v ní zůstane aspoň ten jeden spolužák, který se nikdy nemýlí.

Náš příběh o písemce z analýzy je zjednodušenou verzí úlohy známé pod názvem The Experts Problem. V té místo spolužáka, který se nikdy nemýlí, existuje takový, jenž chybuje maximálně c-krát. Pojďme vyřešit i tuto verzi.

Už nefunguje přestat se ptát spolužáků poté, co se zmýlili. Místo toho si u každého spolužáka budeme udržovat nějakou váhu mezi 0 a 1. Ta vyjadřuje, jak moc mu věříme. Vždy se zeptáme všech a zvolíme tu odpověď, kterou nám dali spolužáci s vyšší celkovou vahou. Pokud tato předpověď nevyšla, pak těm spolužákům, kteří se nestrefili, vydělíme váhu dvěma.

Označme W součet všech vah a počítejme, jak se v průběhu semestru mění. Na počátku má každý spolužák váhu 1, takže $W = n$. Pokud písemku předpovíme správně, W se

[1] Prefix je posloupnost prvních několika řádků.

nezmění. Pokud špatně, spolužákům o celkové váze aspoň $W/2$ vydělíme váhu dvěma, takže celkovou váhu snížíme o aspoň $W/4$. Nová celková váha tedy bude nejvýš $3/4 \cdot W$. Obecně po k chybách máme $W \leq n \cdot (3/4)^k$.

Na druhou stranu víme, že existuje spolužák, který udělá nejvýš c chyb. Jeho váha proto nikdy neklesne pod 2^{-c}, takže bude vždy platit $W \geq 2^{-c}$.

Teď už stačí obě nerovnosti pro W spojit. Dostaneme:

$$n \cdot (3/4)^k \geq 2^{-c}$$

Obě strany nerovnosti zlogaritmujeme:

$$\log_2 n + k \cdot \log_2(3/4) \geq -c$$

Přehodíme $\log_2 n$ na druhou stranu:

$$k \cdot \log_2(3/4) \geq -c - \log_2 n$$

Nakonec si uvědomíme, že $\log_2(3/4) \doteq -0.415 \doteq -1/2.409$. Když tímto číslem vydělíme obě strany, otočí se nerovnost a dostaneme:

$$k \leq 2.41 \cdot (c + \log n).$$

Uděláme tedy pouze $\mathcal{O}(c + \log n)$ chyb.

Konstanta 2.41 by se jestě dala vylepšovat, ale tento příběh si už budeme vyprávět někdy jindy.

Úlohu připravili: Říša Hladík, Martin „Medvěd“ Mareš

34-4-3 Korelace nejsou tranzitivní

Úlohu řešme pomocí teorie grafů – veličiny budou vrcholy a studie hrany mezi nimi. Abychom však měli situaci jednodušší, pro každou veličinu si pořídíme vrcholy dva – jeden pro původní veličinu a jeden pro její negaci (například pro veličinu „být vysoký“ bychom měli vrcholy „být vysoký“ a „být nízký“). Hrana povede mezi vrcholy, pokud dané veličiny jsou podle nějaké studie kladně korelované. Tedy například, pokud existuje studie „čím více mléka lidé pijí, tím jsou vyšší“, pak povedeme hranu mezi vrcholy „být nízký“ a „nepít mléko“ a mezi „být vysoký“ a „pít mléko“.

Všimněme si, že v takovémto grafu odpovídá protipříklad pro Ondru cestě mezi veličinou a její negací. To proto, že hrany odpovídají jednotlivým studiím v kýžené posloupnosti studií. My potřebujeme takovou cestu nalézt.

Nejprve potřebujeme nalézt nějakou veličinu, ze které taková cesta vede. To můžeme udělat tak, že si pomocí prohledávání do hloubky (nebo do šířky či čehokoliv jiného) najdeme komponenty souvislosti grafu. Nyní nás zajímá, zda existuje nějaká veličina, že její negace je ve stejné komponentě jako ona. To můžeme udělat například tak, že si komponenty předtím očíslujeme, a porovnáváme čísla komponent, do kterých bod patří.

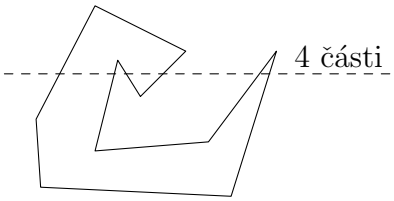
Jakmile takovou veličinu máme, musíme ještě zkonstruovat cestu mezi veličinou a její negací, což opět můžeme udělat například prohledáváním do hloubky.

Rozklad na komponenty souvislosti zvládneme v lineárním čase, kontrola pro každou veličinu je konstantní, takže celkově také lineární. I druhé prohledávání je lineární, celková časová složitost je tak lineární vzhledem k velikosti nového grafu. Zároveň je lineární i vzhledem k velikosti grafu na vstupu, protože jsme počet hran a vrcholů pouze zdvojnásobili. Paměťová složitost je rovněž lineární.

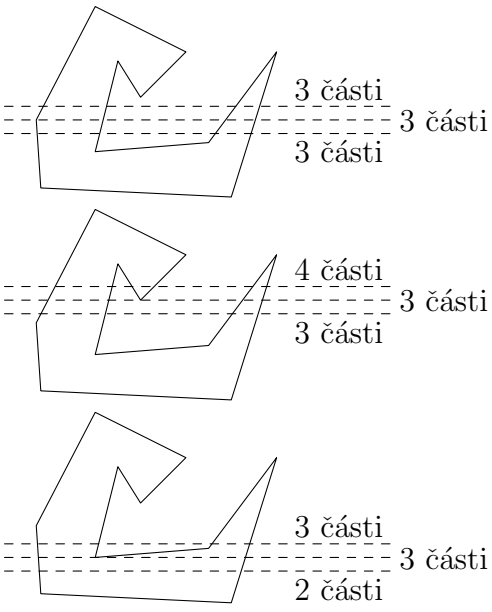
Úlohu připravili: Jirka Kalvoda, Ondra Sladký

34-4-4 Geometrie

Zamysleme se nad tím, co se stane, když dort někde rozřízneme přímkou. Nejprve chvíli předpokládejme, že přímka zatím neprochází žádným z vrcholů N-úhelníku. Tedy přímka prostupně prochází hranami, kterými se dostává dovnitř a ven z dortu. Nahlédneme, že za každou takovou oblast, kterou řez navštíví, se počet částí dortu o jedna zvýší. Jelikož na začátku se dort skládal z jedné části, po řezu, který navštívil dort K-krát, bude $K + 1$ částí dortu.



Co se bude dít o okolí vrcholů? V případě, že z vrcholu odchází hrany dortu na opačné strany řezu, tak je jedno, jakou z těchto stran dortu řez prochází. Složitější situace ovšem nastává v momentě, když obě hrany dortu směřuji na jednu stranu od řezu. Například nahoru. Kdybychom dort rozřízli pod vrcholem, tak počet částí bude o jedna menší než rozříznutí nad ním, protože řez potká o dvě hrany méně. Když povedeme řez přesně daným vrcholem, tak bude počet částí stejný jako v jednom z případů řezu posunutého o kousek na nějakou stranu. Záleží na tom, jestli se jedná o konvexní nebo nekonvexní úhel dortu. To naštěstí ale nebudeme muset řešit.



Speciální případ nastane, když jedna z odchozích hran je rovnoběžná s řezem. V ten moment můžeme dvojici vrcholů propojených touto hranou považovat za jeden vrchol a použít předchozí myšlenky.

Jednodušší varianta

Nyní se podíváme na jednodušší variantu. Tedy budeme hledat jen ty řezy dortu, kde nůž je rovnoběžný s osou x. Na tento případ se nám bude hodit technika zvaná zametání roviny přímkou. To si můžeme představit tak, že nad dortem budeme pohybovat nožem z jedné strany na druhou a budeme se dívat, na kolik částí by se aktuálně rozpadl. Nakonec odpovíme maximem z tohoto počtu.

To zní poměrně jednoduše a kdyby člověk řešil úlohu se skutečným dortem bez pomoci počítače, tak by to nejspíše

dělal přesně takto. Jenže jak v počítači simulovat spojitý pohyb nože?

Všimneme si, že zajímavé věci se dějí jen v momentě, když nožem přecházíme pres nějaký vrchol N-úhelníku. Jindy se totiž počet částí nemůže změnit. Tedy části mezi vrcholy může náš program prostě přeskočit a stačí vždy spočítat počet částí dortu po řezu, který vedeme v blízkosti vrcholu.

Ovšem každý výpočet počtu částí nemusíme začínat od znovu. Můžeme postupovat odshora dolů a vždy využít předcházejícího výsledku. Stačí udělat pouze malou úpravu pro každý vrchol na dané souřadnici. Když vrchol má hrany na různé strany řezu, tak už jsme nahlédli, že se počet nijak nezmění. Když hrany z vrcholu vychází směrem, odkud s řezem přicházíme, tak se počet částí při průchodu bodem o jedna sníží. V opačném případě se o jedna zvýší. Zadání slibuje, že žádné dva body nemají stejnou y-ovou souřadnici, takže se nemusíme bát, že by maximum nastávalo jen v nějaké y-ové souřadnici a už ne v okolí alespoň na jednu stranu.

Celý algoritmus tedy bude probíhat následovně. Projdeme seznam bodů N-úhelníku a pro každý z nich zjistíme, který z případů změny počtů částí u něj nastane (stačí se podívat na y-ové souřadnice okolních bodů. Následně vrcholy seřadíme podle y-ové souřadnice. Posloupnost pak jen projdeme a budeme si počítat aktuální počet částí dortu. Odpovíme maximem z nich. To lze chápat také jako maximum z prefixových součtů změn hodnot.

Lehčí variantu tedy umíme řešit v čase $O(N \log N)$ s paměťovou složitostí $O(N)$.

Zobecnění na kompletní úlohu

V obecné verzi potřebujeme uvážiti řezy, které nejsou rovnoběžné s osou x. Zase můžeme začít se spojitou simulací. Budeme dort otáčet a pro každé otočení si pomocí předchozího algoritmu spočítáme maximální počet částí, na které se rozpadne, když řez povedeme vodorovně s aktuálním natočením dortu.

Nyní se zbavíme spojité simulace. Všimneme si, že výsledek předešlého algoritmu se změní jen, když se prohodí pozice vrcholů v setříděném seznamu nebo se změní, jakým směrem vůči řezu směřují hrany kolem vrcholu. Obojí ovšem může nastat jen v momentě, když simulace projde stavem, kdy dva body mají stejnou y-ovou souřadnici. Můžeme tedy uvážit všechny spojnice dvojic vrcholů a simulaci zastavit jen v momentě, kdy nějaká z nich bude rovnoběžná s osou x. Pro každou takovou provedeme výpočet v natočení, kdy je spojnice rovnoběžná s řezem, a těsně poté.

Nakonec ještě poznamenejme, že díky tomu, že žádné tři body neleží na stejné přímce, nemusíme uvažovat natočení, kde nějaká spojnice je vodorovná. Můžete si rozmyslet, že mírným pootočením na jednu nebo na druhou stranu se situace nezhorší. Díky tomu zejména odpadne složitý případ, kdy dva body mají stejnou y-ovou souřadnici.

Když pro každou z $\mathcal{O}(N^2)$ spojnic bodů budeme volat algoritmus řešící předchozí lehčí variantu, získáme řešení se složitostí $\mathcal{O}(N^3 \log N)$.

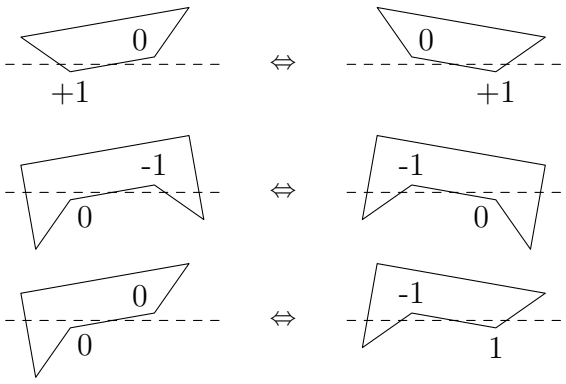
Zrychlujeme

I pro obecný případ můžeme využít toho, že předchozí řešení jen nepatrně poupravíme. Spojnice bodů budeme procházet v pořadí dle jejich úhlu od vodorovného směru (tedy jako kdybychom dortem skutečně spojitě otáčeli). Všimneme si, že u každé spojnice se v setříděném seznamu vrcholů jen prohodí daná dvojice. Když mezi dvojicí vede hrana, změní se to, na jakou stranu vedou sousední hrany od vrcholu. Místo toho, abychom vrcholy znovu třídili, můžeme jen danou dvojici pouze prohodit a přepočítat u ní změny hodnot. Pokud nemáme trojici bodů na stejné přímce, máme zaručeno, že prohazujeme sousední body.

Bohužel i když jsme se zbavili opakovaného třídění, moc jsme si nepomohli. Algoritmus má složitost $\mathcal{O}(N^3)$. Pokaždé musíme procházet celou setříděnou posloupnost a hledat, kde je nejlepší udělat řez. Nebo nemusíme?

Kromě setříděné posloupnosti se můžeme pokusit si udržovat i prefixové součty změn hodnot (tedy počet částí za každým rozříznutím). Jako celková odpověď pak bude největší číslo, které jsme do prefixových součtů za celý běh algoritmu zapsali. Dále si ukážeme, že prefixové součty se mění jen lokálně. Pro každou dvojici prohozených vrcholů bude stačit jen změnit součet prefixu po prvním z vrcholů.

V případě, že se nezmění změny hodnot u vrcholů, tak je situace jednoduchá. Jelikož prohazujeme sousední vrcholy, všechny až na jeden prefixový součet obsahují buď oba, anebo žádný z vrcholů, a tedy se nezmění. Složitější situace nastane v momentě, kdy se změní změna hodnoty. To nastane pouze v případě, že prohazovaná dvojice je spojena hranou dortu. Rozborem případů, kam dort zatáčí v okolí vrcholů na následujícím obrázku, však zjistíme, že součet zůstane zachovaný, a tedy i tentokrát se prefixové součty obsahující oba dva vrcholy nezmění.



Tím jsme získali algoritmus, který spotřebuje $\mathcal{O}(N^2 \log N)$ času a $\mathcal{O}(N^2)$ paměti. Nejnáročnější je setřídit spojnice bodů dle úhlu.

Trošku méně paměti

Na závěr si ještě ukážeme, jak zmenšit paměťové nároky programu.

Víme, že vždy prohazujeme sousední vrcholy, ovšem v setříděných posloupnostech hran máme naplánovaná všechna prohození již od začátku. Tedy i pro dvojice vrcholů, které jsou momentálně v posloupnosti daleko od sebe. Budeme si plánovat jen omezené množství budoucích událostí. Tedy jen prohození sousední dvojice vrcholů v posloupnosti, protože o ostatních víme, že zatím nemohou nastat. Místo setříděné posloupnosti budeme využívat haldu, abychom mohli za běhu události přidávat a případě odebírat. Když budeme prohazovat nějakou dvojici, smažeme prohození vrcholů dvojice se sousedem mimo dvojici a naplánujeme prohození dle aktuálního pořadí vrcholů.

Časová složitost zůstane stále $\mathcal{O}(N^2 \log N)$, ovšem paměťová se sníží na $\mathcal{O}(N)$.

Úlohu připravil: Jirka Kalvoda

Výsledková listina čtvrté série třicátého čtvrtého ročníku KSP

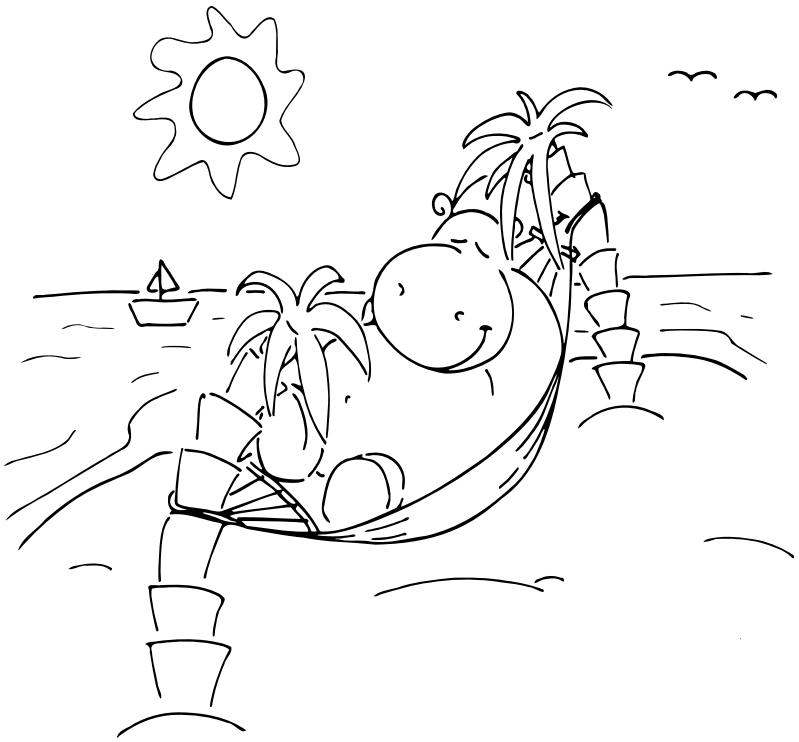
	řešitel	*škola*	*ročník*	*sérií*	*4-1*	*4-2*	*4-3*	*4-4*	*4-S*	*série*	*KSP-X*	*celkem*
0.					9	10	11	15	15	60,0	36,0	240,0
1.	Benjamin Swart	MensaG	3	4	9	10	11	15	17,5	62,5	34,5	237,0
2.	Daniel Skýpala	GTomkovaOL	4	24	9	10	11	15	17	62,0	12,0	234,5
3.	Robert Jaworski	GÚstavníPH	4	11	9	10	11	15	17	62,0	16,0	219,5
4.	Jakub Ondroušek	GTomkovaOL	2	9	9	10	11	8	15,5	53,5	0,0	179,5
5.	Jáchym Kouba	GJŠkodyPŘ	2	4	9	10	11	7	14,5	51,5	4,0	176,0
6.	David Kolář	GJírovcČB	3	4	9	10	11	15	14,5	59,5	0,0	159,5
7.	Lukáš Tomoszek	GTři	4	4						0,0	4,0	136,4
8.	Jan Slíva	MensaG	1	4	9	10	11	15		45,0	0,5	130,5
9.	Adam Kolník	SSŠVTPraha	3	9					7	7,0	8,0	122,5
10.	Jan Kotovský	GPísnickáPH	3	9						0,0	0,0	120,5
11.	Lukáš Veškrna	GKepleraPH	4	9	9	10				19,0	0,0	115,5
12.	Viktor Číhal	SPŠSmíchov	2	4	9	10	11	8		38,0	0,0	111,5
13.	Patrik Číhal	SŠKKamPard	2	3	9	10	11	12	13,5	55,5	0,0	110,0
14.	Alex Olivier Michaud	GJírovcČB	3	4	9	10	11	9		39,0	0,5	109,0
15.	Vladimír Sklenár	GTerVans	2	4	1	7	2		5,5	15,5	4,0	89,5
16.	Vojtěch Lančarič	SPŠG Třebešín	3	3	3	10	11	14		38,0	0,0	88,0
17.	Prokop Randáček	GFXŠaldyLI	3	9						0,0	0,0	80,0
18.	Ján Plachý	G VBN Prie	4	4	9					9,0	4,0	77,5
19.	Vít Skalický	GPísnickáPH	4	22						0,0	0,0	77,0
20.	Richard Tichý	SG Kladno	0	3						0,0	0,5	69,0
21.	Kryštof Maxera	GJírovcČB	1	5						0,0	0,0	60,0
22.	Jan Černohorský	G Brandýs	4	4						0,0	0,0	59,5
23.	Petr Šícho	GKepleraPH	4	4						0,0	0,0	55,0
24.	Jakub Mikeš	GJŠkodyPŘ	4	3						0,0	0,0	54,5
25.–26.	Patrik Herman	GTomkovaOL	3	8						0,0	0,0	44,5
	Jiří Kvapil	GTomkovaOL	4	20						0,0	0,0	44,5
27.	Eliška Macáková	CENADA BA	2	4						0,0	0,0	43,5
28.	Adam Kuča	PORG Krč	4	3						0,0	1,0	42,0
29.	Matúš Duchyňa	GGrössBA	3	3						0,0	0,0	36,0
30.	Ivan Trenčanský	GLSáru	3	3	9					9,0	0,0	33,0
31.	Daniel Šoltýs	GTřeKošice	4	6						0,0	0,0	32,0
32.	Jonáš Dej	G Wicht	3	2			11			11,0	0,0	29,0
33.	Nikolay Fomichev	SSŠVTPraha	3	2						0,0	0,0	26,0
34.	Veronika Jůzková	MensaG	4	6						0,0	0,0	23,0
35.–36.	Martin Belluš	GGrössBA	3	1						0,0	0,0	22,0
	Filip Siviček	GTimLučen	3	1						0,0	0,0	22,0
37.–38.	Petr Hladík	GMikulášPL	4	4						0,0	0,0	21,0
	Bobur Toshtemirov	GMikulášPL	3	2						0,0	0,0	21,0
39.	Zuzana Aubrechtová	GHeyrovPH	3	2	9					9,0	0,0	19,0
40.–42.	Honza Kocourek	ParkLane	2	1						0,0	0,0	18,0
	Karel Procházka	GPBystrica	4	1	7		11			18,0	0,0	18,0
	Matúš Púll	GZborovPH	2	2						0,0	0,0	18,0
43.	Jakub Švojgr	GČeskáČB	3	1						0,0	0,0	14,0
44.–45.	Michal Pavlíček	MendelGOP	4	2						0,0	0,0	12,0
	Ondřej Skácel	GTomkovaOL	3	6						0,0	0,0	12,0
46.–50.	Michal Bernat	GZborovPH	2	1						0,0	0,0	11,0
	Daniel Culliver	GZborovPH	2	1			11			11,0	0,0	11,0
	Vojtěch Franc	GArabskáPH	2	1						0,0	0,0	11,0
	Matouš Mišta	GTomkovaOL	1	1						0,0	0,0	11,0
	Filip Neubauer	AkademGPH	2	1						0,0	0,0	11,0
51.	Pavel Jordán	GPOA Znojmo	3	3						0,0	0,0	10,0
52.	Ondřej Pupík	GRožnovPR	2	1	3	6				9,0	0,0	9,0
53.–54.	Martin Fof	MendelGOP	4	1						0,0	0,0	8,0
	Jakub Kopčil	GMikulášPL	3	1						0,0	0,0	8,0
55.	Michal Martínek	GÚstavníPH	1	1						0,0	0,0	7,0

	řešitel	*škola*	*ročník*	*sérií*	*4-1*	*4-2*	*4-3*	*4-4*	*4-S*	*série*	*KSP-X*	*celkem*
56.	Jonáš Menšík	GJŠkodyPŘ	0	1						0,0	0,0	6,0
57.	Marek Maškarinec	SPŠEMasLI	1	1						0,0	0,0	4,0
58.	Jan Šuráň	GZborovPH	4	1						0,0	0,0	3,0
59.	Albert Bakoč	GZborovPH	1	1						0,0	0,0	2,0
60.	Jakub Surga	ParkLane	4	6						0,0	0,0	1,0

Výsledková listina KSP-X po čtvrté sérii třicátého čtvrtého ročníku

	řešitel	*škola*	*ročník*	*sérií*	*1-X1*	*1-X2*	*2-X1*	*3-X1*	*celkem*
0.					8	8	10	10	36,0
1.	Benjamin Swart	MensaG	3	3	8,5	8	8	10	34,5
2.	Robert Jaworski	GÚstavníPH	4	10		4	2	10	16,0
3.	Daniel Skýpala	GTomkovaOL	4	23		8		4	12,0
4.	Adam Kolník	SSŠVTPraha	3	8		8			8,0
5.–8.	Jáchym Kouba	GJŠkodyPŘ	2	3		4			4,0
	Ján Plachý	G VBN Prie	4	3		4			4,0
	Vladimír Sklenár	GTerVans	2	3		4			4,0
	Lukáš Tomoszek	GTři	4	4		4			4,0
9.	Adam Kuča	PORG Krč	4	3			1		1,0
10.–12.	Alex Olivier Michaud	GJírovcČB	3	3	0,5				0,5
	Jan Slíva	MensaG	1	3	0,5	0			0,5
	Richard Tichý	SG Kladno	0	3	0,5				0,5

Bonusové úlohy z jednotlivých sérií se nepočítají do bodování ročníku. Mají svou vlastní výsledkovou listinu a za jejich úspěšné vyřešení (alespoň polovina bodů za úlohu) udělujeme speciální odměny.



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/*