Korespondenční Seminář z Programování

35. ročník | KSP | Červen 2023

Vzorová řešení páté série třicátého pátého ročníku KSP

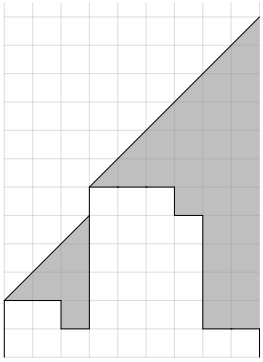
35-5-1 Písek

Začneme s jednodušší verzí úlohy, kde je na pravé straně ostrova nekonečně vysoká zeď. V takovém případě se písek může sesypat pouze z levého okraje.

Nejdříve si spočítáme, v jaké výšce (nad zemí) bude povrch hromady písku. Pro každý dům si ji můžeme reprezentovat jedním číslem – výškou na levém okraji. Od něj bude pokračovat svah s úhlem 45° do pravého okraje, kde bude sahat o patro výš. U nejlevějšího domu je výška písku rovná výšce domu – cokoliv vyššího by se sesypalo přes okraj. Nad dalším domem může svah pokračovat, takže přičteme jedna k předchozí výšce. Dům na tomto místě ale taky může být vyšší, v čemž případě bude začínat nový svah na střeše domu. Pro každý dům tedy bude výška povrchu rovna maximu z výšky domu a minulé výšky zvětšené o jedna.

Spočítat objem písku je pak už jednoduché. Pro každý sloupec odečteme od výšky povrchu výšku domu (písek začíná až nad střechou), vynásobíme čtyřmi (objem jednoho políčka) a přičteme dva za trojúhelníkové půlpolíčko se svahem.

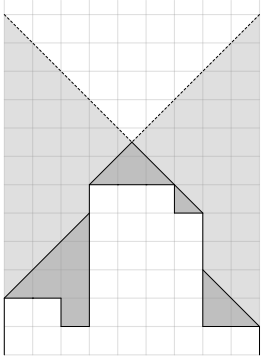
Takto by to vypadalo pro příklad ze zadání:



Objem písku je $2+6+14+2+6+10+18+38+42 = 138$.

Nyní se vrátíme k původní úloze, kde se písek sype přes oba kraje. V takovém případě se může písek udržet v dané výšce právě tehdy, když se nesesype z ani jedné strany. Můžeme tedy ostrov projít z obou směrů, spočítat maximální výšku písku, ve které se nesesype z dané strany, a vzít minimum z těchto hodnot. Pokud jsou výšky v daném sloupci ze dvou směrů různé, tak se nižší svah celý vejde pod ten vyšší a výpočet objemu bude stejný. Pokud jsou ale stejné, svahy se uprostřed zkříží a místo dvou budeme přičítat jenom jedna.

Takto by to vypadalo pro příklad ze zadání:



Objem písku je $2+6+14+2+5+2+2+6+2 = 41$.

Celý algoritmus je jen několik průchodů ostrova, kde pro každý dům provedeme výpočet v konstantním čase, takže časová složitost je $\mathcal{O}(N)$. Obdobně to platí i pro paměťovou složitost.

Program (C++):
`http://ksp.mff.cuni.cz/viz/35-5-1.cpp`

Program (Python):
`http://ksp.mff.cuni.cz/viz/35-5-1.py`

Úlohu připravili: Jan Adámek, Dan Skýpala

35-5-2 Hroší lomikámen

Co po nás úloha doopravdy chce? Máme dva řetězce A a B a máme najít u každého stejně dlouhý podřetězec, přičemž v tomto podřetězci má být rozdíl stejných a různých písmen maximální.

Nabízí se jednoduchý algoritmus, kde pro každou možnost relativního posunutí začátku podřetězce v A a v B vyzkoušíme všechny začátky a konce. Mohli bychom počítat počet rozbitých kamenů v tomto podřetězci postupným procházením, ale všimněme si, že dvojice stejných kamenů nám dá $+2$, zatímco dvojice různých -2 a my hledáme součet úseku. Na to můžeme použít prefixové součty, díky kterým spočteme součet v konstantním čase.

Co ale vlastně děláme? Hledáme úsek s největším součtem vyzkoušením všech možností. Toto můžeme ale zrychlit. Protože součet úseku od ℓ po r v prefixových součtech je $p[r] - p[\ell]$, tak dává smysl vybrat nejmenší $p[\ell]$.

Jen si musíme dát pozor, aby platilo $\ell < r$. Takže budeme pole prefixových součtů postupně procházet, udržovat si nejmenší doposud nalezené $p[\ell]$ a to vždy odečteme od aktuálního $p[i]$. Díky tomu najdeme nejlepší začátek a konec v lineárním čase.

To budeme dělat pro každou možnost posunu, což nám dá výslednou časovou složitost $\mathcal{O}((N+M)\min(N,M)) = \mathcal{O}(NM)$.

Program (Python):
`http://ksp.mff.cuni.cz/viz/35-5-2.py`

Úlohu připravili: Kristýna Petrlíková, Dan Skýpala

35-5-3 Nejkrásnější náhrdelník

Jak úlohu řešit přímočaře? Vygenerujeme si všechny možnosti, kam dát zapínání, a potom je všechny porovnáme a vybereme maximum. Např. pro náhrdelník $3, 1, 3, 2$ by možnosti byly:

$$3, 1, 3, 2$$
$$1, 3, 2, 3$$
$$3, 2, 3, 1$$
$$2, 3, 1, 3$$

Maximální je třetí z nich.

Obecně by to celé trvalo $\mathcal{O}(N^2)$. Na vygenerování spotřebujeme $\mathcal{O}(N^2)$ času a porovnávání trvá $\mathcal{O}(N)$ na sousední dvojici.

Pojďme to ale zlepšit. Pro začátek nepotřebujeme všechny možnosti generovat, ale můžeme si prostě udržovat indexy do náhrdelníku bez zapínání.

Nyní pojďme všechny možnosti porovnávat současně. Udržujme si množinu aktuálních možností, kde možnost je reprezentována aktuálním indexem a indexem začátku. (Na začátku všechny možnosti budou tvaru i, i.) Poté vždy vyškrtněme možnosti, které nebudou největší. Najděme maximální prvek na aktuálním indexu a zahoďme všechny možnosti, které ho nemají.

Pojďme se podívat, jak se tento algoritmus vyvíjí na ukázkovém vstupu:

$$\begin{array}{l} 3,\ 1,\ 2,\ 3,\ 2,\ 1,\ 3 \\ \uparrow_1 \uparrow_2 \uparrow_3 \uparrow_4 \uparrow_5 \uparrow_6 \uparrow_7 \\ 3,\ 1,\ 2,\ 3,\ 2,\ 1,\ 3 \\ \uparrow_7 \uparrow_1 \qquad \uparrow_4 \\ 3,\ 1,\ 2,\ 3,\ 2,\ 1,\ 3 \\ \quad \uparrow_7 \end{array}$$

A skončíme s tím, že největší možnost začíná na indexu 7.

Bohužel, tento algoritmus je stále pomalý. Uvažme vstup $3, 3, \ldots, 3, 1$. V první fázi projdeme n možností, v druhé $n-1$, ...

Pojďme to opravit trochu jiným pohledem na věc. V algoritmu si budeme udržujeme úseky – to jsou části délky k, které jsou největší k-tice v zadané posloupnosti. V předchozím algoritmu se nám úseky mohly překrývat, ale tentokrát jim to zakažme.

Na začátku si vytvořme úseky délky 1, které obsahují maximální prvek. (Každý maximální prvek odpovídá úseku.) Nyní je všechny zkusme prodloužit.

Buď nově přidávané číslo vždy není součástí úseku, pak přidejme ty čísla, která jsou maximální. (Ostatní úseky nám „umřou“ – nebudou se dále prodlužovat.)

Jinak nám některé úseky narazí do následujících. Protože začátek úseku je vždy maximální číslo (a opačně), tak ty, co nenarazily, nám umřou, a ty co narazily, nechť pohltí úsek přímo za nimi (a usmrtí ho v případě nutnosti). Po pohlcení nám též umřou ty úseky, které nejsou nejdelší z živých (delší úsek je z definice maximální k-tice, takže za kratším je menší číslo).

Zde se nám může stát, že všechny živé úseky (s délkou p) do sebe narazí cyklicky. Pak posloupnost je periodická s periodou p a jako odpověď stačí vydat začátek libovolného úseku.

Jinak opakujeme výše popsané prodlužování, dokud nám nezbude jen jeden úsek. Pak ten je jistě naší odpovědí, protože neexistuje větší k-tice se stejnou délkou, tedy ani delší.

A jakou toto bude mít složitost?

- Každé číslo bude přidáno nejvýše jednou.
- Každý úsek zemře nejvýše jednou.
- Každý úsek bude pohlcen nejvýše jednou (pak bude pohlcen i s tím úsekem, co ho pohltil).

Každá věc se při prodlužování stane nejvýše $\mathcal{O}(N)$-krát, a proto algoritmus poběží v lineárním čase.

Dva kanóny na vrabce

Jako ochutnávku na závěr zmíníme dvě silné techniky, ze kterých nám jen tak mimochodem vypadnou řešení v $\mathcal{O}(N \log N)$ čase. O obou si víc můžete přečíst v Průvodci labyrintem algoritmů.[1] Obě techniky pracují s řetězci, a proto o naší posloupnosti nyní budeme jako o řetězci přemýšlet. V obou případech se nám také hodí zbavit cykličnosti, a proto si náš řetězec zapíšeme dvakrát za sebou. Pak mezi všemi jeho podřetězci délky N hledáme ten lexikograficky největší.

První technikou jsou tzv. *suffixová pole,* která umí vzít libovolný řetězec a seřadit všechny jeho možné *suffixy* (podřetězce začínající někde v řetězci a končící na jeho konci) podle jejich lexikografického pořadí. Výstupem je pole, kde na pozici i je uložené, kde začíná i-tý lexikograficky nejmenší suffix. Lidová moudrost praví, že se suffixovými poli (a jejich příbuzným, suffixovými stromy) jde skoro libovolná řetězcová úloha vyřešit v lineárním čase.

Tak tomu (až na logaritmus navíc) je i u nás: nejkrásnější umístění zapínání totiž téměř přesně odpovídá lexikograficky největšímu suffixu naší posloupnosti. Má to jen jeden háček: zatímco my hledáme podřetězec délky N, suffixové pole nám seřadí suffixy délky 1 až $2N$. Můžeme si ale rozmyslet, že suffixy délky menší než N můžeme ignorovat, a že ty zbylé budou ve správném pořadí i vzhledem jen k jejich prvních N znakům. Stačí tedy procházet suffixové pole odzadu a zastavit se u prvního suffixu délky větší než N. To umíme v lineárním čase, a spolu s konstrukcí suffixového pole (nad řetězci s neomezeně velkou abecedou) tak dostáváme řešení v čase $\mathcal{O}(N \log N)$.

Druhá technika využívá tzv. *okénkové hešování* – techniku, pomocí které si pro náš řetězec umíme předpočítat jakési „prefixové součty“ – akorát s heši místo se součty. Umíme se pak v $\mathcal{O}(1)$ zeptat na libovolný podřetězec a obdržet zpátky jeho heš – číslo s tou vlastností, že dva stejné řetězce

[1] `http://pruvodce.ucw.cz/`

vždy dostanou ten samý heš, a různé řetězce s velmi velkou pravděpodobností (řekněme $1 - n^{-10}$) dostanou různý heš. Dovolíme-li algoritmu nějakou malou pravděpodobnost chyby, umíme pak rychle testovat, zda se dva podřetězce rovnají – stačí porovnat jejich heše a doufat, že jsme zrovna neměli smůlu a nenarazili na dva různé řetězce se stejným hešem.

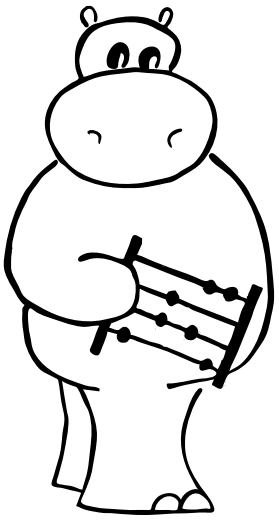
Jak s okénkovým hešováním vyřešit úlohu? Vcelku jednoduše: nyní můžeme pro dva podřetězce délky N v čase $\mathcal{O}(\log N)$ zjistit, který z nich je lexikograficky větší: stačí kombinací binárního vyhledávání a okénkového hešování najít první místo, kde se neshodují, a podívat se, který z nich má na tomto místě větší hodnotu. A tím už máme vyřešenou celou úlohu: stačí postupně brát podřetězce $[0, N), [1, N + 1), \ldots, [N, 2N)$ a každý porovnat s aktuálně lexikograficky největším. Tak N-krát provedeme binární vyhledávání a časová složitost je $\mathcal{O}(N \log N)$.

Úlohu připravili: Ríša Hladík, Dan Skýpala

35-5-4 Kalkulačka

Stavy kalkulačky si můžeme představit jako nekonečný orientovaný graf. Pokud se dá nějaké číslo a převést operací na b, existuje v tomto grafu mezi vrcholy odpovídajícími a a b hrana. Nás zajímá minimální počet operací, kterými z nějakého z čísel $0-9$ dostaneme K. Tedy vlastně nejkratší cesta mezi libovolným z vrcholů $0-9$ k vrcholu odpovídajícímu K. Tu můžeme najít s pomocí BFS, přičemž za výsledek prohlásíme první cestu, kterou najdeme. Ta bude jistě nejkratší, protože BFS se chová tak, že vždy prozkoumává vrcholy v bližších vzdálenostech a pak až v těch vzdálenějších.

Ale jak uložit nekonečný graf? Jednoduše: nebudeme ho ukládat. Hrany a vrcholy budeme vytvářet až za běhu, když do nich dojdeme. Tomu se říká prohledávání implicitního grafu.



Nyní pojďme rozebrat složitost. Nejprve najdeme horní odhad počtu potřebných operací, k čemuž se nám bude hodit reprezentovat čísla v devítkové soustavě. Pak můžeme čísla zapsat do kalkulačky po číslicích, např. číslo 1234 (již zapsané v devítkové soustavě) bychom zapsali jako $((1 * 9 + 2) * 9 + 3) * 9 + 4$. Na každou cifru tedy potřebujeme dvě operace plus jednu na poslední přičtení. Jakékoli číslo tedy můžeme na kalkulačce získat tak, že ho převedeme do devítkové soustavy a pak dané číslo vložíme výše uvedeným způsobem do kalkulačky. Např. 1234 je v devítkové soustavě 1621 a $((1 * 9 + 6) * 9 + 2) * 9 + 1 = 1234$. Číslo K tedy určitě lze vyrobit na $2\lceil \log_9 K \rceil + 1$ operací.

Prohledávání se tedy může dostat jenom do vrcholů, které jsou nejvýše takto daleko od počátku. To nám dá i horní odhad časové složitosti: z každého vrcholu vede konstantní počet hran, takže jedním vrcholem strávíme konstantní čas.

Nepotkáme tedy čísla větší než $9^{2\lceil \log_9 K \rceil + 1}$ (i kdybychom opakovaně násobili 9). Navíc nemusíme prozkoumávat čísla větší než $9^{\frac{3}{2} \log_9 K + 2}$, protože by nám pak trvalo ještě přibližně $\frac{1}{2} \log_9 K$ kroků, abychom ho znovu snížili (abychom se dostali ke K).

Tedy finální složitost vychází jako $\mathcal{O}(9^{\frac{3}{2} \log_9 K}) = \mathcal{O}(K^{\frac{3}{2}})$. Pečlivějším rozborem by odhad nejspíše ještě šel zlepšit.

Úlohu připravili: Ján Plachý, Lukáš Veškrna

35-5-S Determinant

Úkol 1 – Obsah mnohoúhelníku [6b]:

Pro začátek si uvědomíme, jak spočítat obsah trojúhelníku mezi body $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$. Vyrobíme vektory $\mathbf{x} = \mathbf{p}_2 - \mathbf{p}_1$ a $\mathbf{y} = \mathbf{p}_3 - \mathbf{p}_1$. Následně pomocí determinantu spočítáme obsah rovnoběžníku určeného vektory $\mathbf{x}$ a $\mathbf{y}$. Obsah trojúhelníku bude polovina obsahu rovnoběžníku.

Jakmile toto umíme, spočítat obsah konvexního mnohoúhelníku již zvládneme hravě. Rozdělíme mnohoúhelník na trojúhelníky a jejich obsah sečteme.

Rozdělení lze elegantně realizovat takto: Vezmeme tři po sobě jdoucí vrcholy, spočítáme obsah trojúhelníku mezi nimi a tento trojúhelník z mnohoúhelníku odštípneme vynecháním prostředního vrcholu. Tento postup opakujeme, dokud jsou v mnohoúhelníku alespoň tři vrcholy.

Pokud budeme brát vrcholy vždy v pořadí po obvodu, znaménko determinantu bude pro každý trojúhelník stejné. Pokud nám vyjde záporný součet, stačí jako výsledný obsah vzít jeho absolutní hodnotu.

Co když mnohoúhelník není konvexní? Tentýž postup bude stále fungovat. Kdykoli totiž vytyčíme trojúhelník vně mnohoúhelníku, bude mít opačné znaménko. Vynecháním prostředního bodu jím mnohoúhelník doplníme.

Spočítání obsahu jednoho trojúhelníku trvá konstantní čas. Kroků provedeme méně, než kolik je vrcholů, takže dostáváme celkovou časovou složitost $\mathcal{O}(N)$.

Ještě ve zkratce uvedeme alternativní řešení. Zvolíme si libovolný bod $\mathbf{z}$, pro jednoduchost $\mathbf{z} = (0, 0)$. Poté sečteme obsahy trojúhelníků $\mathbf{z}\mathbf{p}_1\mathbf{p}_2, \ldots, \mathbf{z}\mathbf{p}_{N-1}\mathbf{p}_N, \mathbf{z}\mathbf{p}_N\mathbf{p}_1$ a vezmeme absolutní hodnotu tohoto součtu. Podobně jako dříve platí, že se části uvnitř mnohoúhelníku započítají kladně a části vně mnohoúhelníku záporně.

Úkol 2 – Polynom [6b]:

Nechť má polynom předpis $p(x) = a_2x^2 + a_1x + a_0$, kde a_2, a_1, a_0 jsou neznámé koeficienty. Aby polynom procházel zadanými body, musí platit následující rovnice:

$$p(x_0) = a_2x_0^2 + a_1x_0 + a_0 = y_0$$
$$p(x_1) = a_2x_1^2 + a_1x_1 + a_0 = y_1$$
$$p(x_2) = a_2x_2^2 + a_1x_2 + a_0 = y_2$$

Zapíšeme-li tuto soustavu maticově jako $\mathbf{A}\mathbf{x} = \mathbf{y}$, matice $\mathbf{A}$ bude vypadat následovně:

$$\mathbf{A} = \begin{pmatrix} x_0^2 & x_0 & 1 \\ x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \end{pmatrix}$$

Aby byl polynom jednoznačně určený, determinant matice $\mathbf{A}$ musí být nenulový.

$$\begin{aligned} \det \mathbf{A} &= x_0^2 x_1 + x_1^2 x_2 + x_2^2 x_0 - x_0^2 x_2 - x_1^2 x_0 - x_2^2 x_1 \\ &= -(x_0 - x_1)(x_1 - x_2)(x_2 - x_0) \end{aligned}$$

Každá závorka je nenulová, proto je nenulový i celý determinant.

Úkol 3 – **Součin** [3b]:

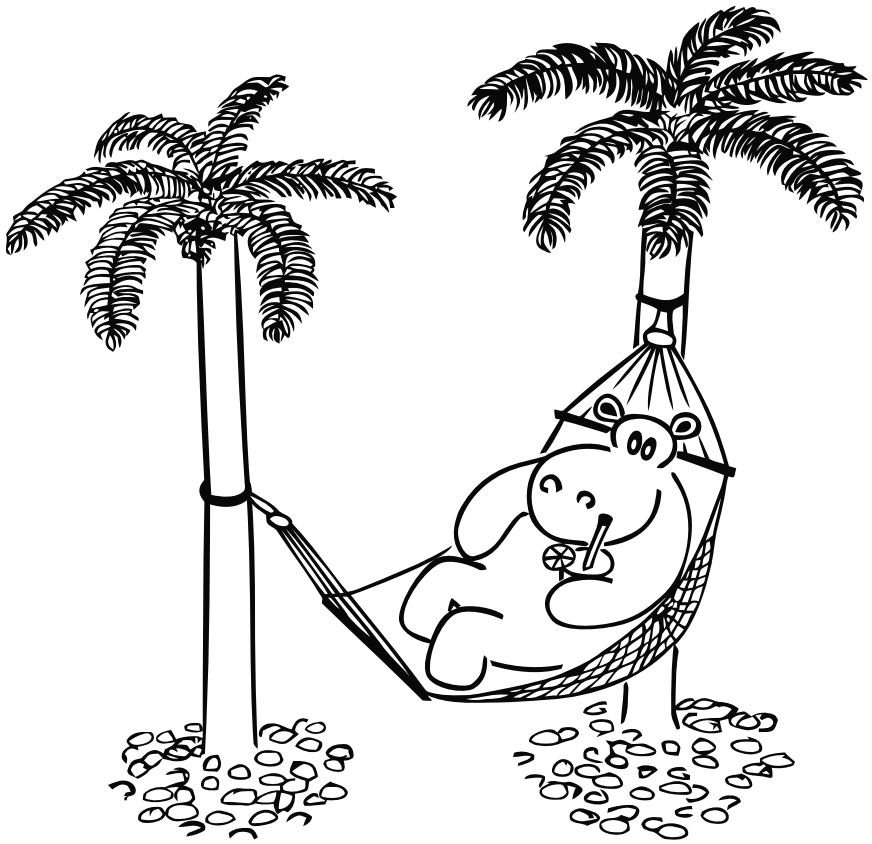
Nahlédneme vztah pomocí zobrazení. Víme, že determinant matice zobrazení popisuje, kolikrát se tímto zobrazením zvětší obsah.

Mějme obrazec s obsahem S. Zobrazme jej nejprve maticí $\mathbf{B}$, tím se jeho obsah změní na $S \cdot \det \mathbf{B}$. Poté jej zobrazme maticí $\mathbf{A}$, tím se jeho obsah změní na $S \cdot \det \mathbf{A} \cdot \det \mathbf{B}$.

Totéž jsme však mohli popsat jedním složeným zobrazením s maticí $\mathbf{AB}$. Toto zobrazení musí zvětšit obsah stejným způsobem, proto $\det \mathbf{AB} = \det \mathbf{A} \cdot \det \mathbf{B}$.

Stejný argument můžeme použít pro matice rozměrů $n \times n$, jen použijeme zobrazení mezi prostory dimenze n. Determinant matice zobrazení popisuje, jak se změní n-rozměrná analogie objemu, což bychom mohli nahlédnout stejně jako v případě obsahu.

Úlohu připravili: David Klement, Martin „Medvěd“ Mareš



Výsledková listina páté série třicátého pátého ročníku KSP

	řešitel	*škola*	*ročník*	*sérií*	*5-1*	*5-2*	*5-3*	*5-4*	*5-S*	*série*	*KSP-X*	*celkem*
0.					9	12	12	12	15	60,0	40,0	300,0
1.	Benjamin Swart	MensaG	4	10	9	12	10	11	15	57,0	39,0	298,2
2.	Štěpán Mikéska	GJarošeBO	4	5	9	12	10	12	14,5	57,5	9,0	292,5
3.	David Kolář	GJírovcČB	4	10	9	12	11,5	12	12	56,5	0,0	262,8
4.	Jáchym Kouba	GJŠkodyPŘ	3	10	9	12	12	8	15	56,0	0,0	262,6
5.	Patrik Číhal	SŠKKamPard	3	9	9	12	12	9	14,5	56,5	21,0	243,3
6.	Vojtěch Lančarič	SPŠG Třebešín	4	10	9	12	6	5	15	47,0	3,3	235,3
7.	Viktor Helmich	GTMannaPH	4	5	9	10	6	5	10	40,0	0,0	233,2
8.	Ondřej Pupík	GRožnovPR	3	7	9	12	10			31,0	0,0	203,8
9.	Albert Bakoč	GZborovPH	2	9	9	12			15	36,0	0,0	199,8
10.	Adam Červenka	GJarošeBO	4	5	9	12	6		15	42,0	0,0	199,0
11.	Erik Ježek	SPŠSmíchov	1	5	9					9,0	5,0	185,0
12.	Filip Prášil	GJiříPoděb	4	5						0,0	1,0	180,2
13.	Anna-Kristina Migel	GNAlejíPH	0	5	9	2			4	15,0	6,0	178,8
14.	Matúš Púll	GZborovPH	3	8	9	10		9		28,0	0,0	177,0
15.	Kryštof Tahal	GUBalvanJN	4	4						0,0	0,0	175,9
16.	Daniel Culliver	GZborovPH	3	7	9	12				21,0	0,0	173,9
17.	Zuzana Aubrechtová	GHeyrovPH	4	9	9	12	11,5	3	4	39,5	0,0	170,4
18.	Kateřina Doubková	GNAlejíPH	4	4						0,0	0,0	167,5
19.	Patrik Prítrský	GGrössBA	2	5	9	12	11			32,0	1,0	167,0
20.	Honza Kocourek	ParkLane	3	6	6	6		9	13	34,0	0,0	162,5
21.	Marek Raška	GTři	4	5		7				7,0	0,0	159,9
22.	Vít Kaděra	G Wicht	1	5	9	12		3	4	28,0	0,0	155,5
23.	Adam Kolník	SSŠVTPraha	4	13						0,0	0,0	151,7
24.	Ondřej Sedláček	GOPavla PH	2	5	9	12	9	5	14,5	49,5	0,0	146,1
25.	Adam Jahoda	GKepleraPH	4	6						0,0	0,0	137,9
26.	Antonín Maloň	GJarošeBO	3	4	9	12	9,5	11	12	53,5	0,0	117,7
27.	Martin Šindelář	GGrössBA	3	4						0,0	0,0	112,5
28.	Jakub Ondroušek	GTomkovaOL	3	14						0,0	0,0	112,4
29.	Erik Sabol	GČeskoliPH	3	6	9	6	1	5	14,5	35,5	2,0	110,4
30.	Robert Klimt	GDobříš	3	5		12				12,0	0,0	107,2
31.	Petr Slonek	GJarošeBO	4	4						0,0	0,0	107,0
32.	Michael Jarvis	GŠpitálsPH	1	4						0,0	0,0	104,2
33.	Kryštof Marek	SGPCE	3	5						0,0	0,0	100,2
34.	Kateřina Vomelová	GÚstavníPH	3	7		1	4	5		10,0	0,0	86,8
35.	Adam Houdek	SOŠ Březová	−2	4	9	12				21,0	0,0	84,0
36.	Petr Starý	GJírovcČB	1	5	9	12	6	10	15	52,0	0,0	79,6
37.	Jakub Konc	GPároNitra	4	3						0,0	0,0	77,0
38.	Ivan Žemlička	GÚstavníPH	2	3	9	12				21,0	0,0	71,5
39.	Jakub Hampl	GMělník	3	5						0,0	0,0	71,1
40.	Jan Ševeček	G UherBrod	1	4						0,0	0,0	52,2
41.	Vladimír Sklenár	GTerVans	3	9						0,0	0,0	50,1
42.	Petr Němec	G Wicht	1	2						0,0	0,0	46,0
43.	Jan Slíva	MensaG	2	6						0,0	0,0	45,0
44.	Ondřej Novák	G Brandýs	1	5				0	13,5	13,5	0,0	44,5
45.	Lucian Poljak	GJŠkodyPŘ	1	5	3	1	6			10,0	0,0	42,6
46.	Viktor Číhal	SPŠSmíchov	3	6						0,0	0,0	42,0
47.	Finley Stuart	GPísnickáPH	2	5	0					0,0	0,0	40,7
48.	Julie Krejčí	PraKonz	3	5						0,0	0,0	39,0
49.	Richard Dobíšek	MensaG	2	3						0,0	0,0	37,8
50.	Oto Skýpala	GJŠkodyPŘ	−1	3						0,0	0,0	35,0
51.	Andrea Mikulová	BGOstrava	4	5					13,5	13,5	0,0	34,8
52.	Jakub Binter	GČeskáČB	0	1						0,0	0,0	30,0
53.	Filip Maňas	GJarošeBO	2	3		12				12,0	0,0	27,8
54.–55.	Daniel Čech	GBezručeFM	2	1	9	12				21,0	0,0	21,0
	Martin Skýpala	GJŠkodyPŘ	3	1						0,0	0,0	21,0
56.	Janek Hlavatý	GJirsíkaČB	4	10						0,0	0,0	19,0
57.	Vít Mitáš	GPolička	1	3		0	1		13	14,0	0,0	17,0

	řešitel	*škola*	*ročník*	*sérií*	*5-1*	*5-2*	*5-3*	*5-4*	*5-S*	*série*	*KSP-X*	*celkem*
58.–60.	David Bojko	GMělník	1	4						0,0	0,0	16,0
	Petr Dymanus	GŠpitálsPH	3	1						0,0	0,0	16,0
	Richard Tichý	SPŠSmíchov	1	4						0,0	0,0	16,0
61.–62.	Michal Martínek	GÚstavníPH	2	3						0,0	0,0	15,0
	Vojtěch Procházka	MensaG	2	2						0,0	0,0	15,0
63.	Radomír Budínek	GŘíč	4	1						0,0	0,0	12,0
64.	Lída Kačenková	GBudějovPH	4	4			5			5,0	0,0	11,6
65.–67.	Pavel Altmann	GMikulášPL	4	3						0,0	0,0	11,0
	Tomáš Kazimír	GNPr	3	1						0,0	0,0	11,0
	Michal Mentzl	G UherBrod	3	1						0,0	0,0	11,0
68.	Michael Ambros	GTomkovaOL	0	2						0,0	0,0	10,3
69.–71.	Jakub Kopčil	GMikulášPL	4	3						0,0	0,0	10,0
	Jan Kotovský	GPísnickáPH	4	10						0,0	0,0	10,0
	Matěj Rýdl	SPŠSmíchov	1	1	9	1				10,0	0,0	10,0
72.–73.	Kryštof Maxera	GJírovcČB	2	7						0,0	0,0	9,0
	Václav Tichý	GKepleraPH	3	1	9					9,0	0,0	9,0
74.	Dominik Malý	GBSučany	4	1						0,0	0,0	7,0
75.–78.	Vojtěch Bízek	GPísnickáPH	3	1						0,0	0,0	6,0
	Adam Bureš	SPŠ Přerov	3	1						0,0	0,0	6,0
	Samuel Lipovský	GBSučany	4	1						0,0	0,0	6,0
	Tomáš Macholda	GCoubTábor	4	1						0,0	0,0	6,0
79.	Jakub Štefan	GMělník	4	1						0,0	0,0	5,2
80.	Michal Hrbek	GZborovPH	1	1						0,0	0,0	5,1
81.–82.	Matěj Smetana	AkademGPH	2	1						0,0	0,0	5,0
	Jan James Soukup	GKlatovy	4	1						0,0	0,0	5,0
83.	David Pacák	G Brandýs	2	2						0,0	0,0	4,6
84.–85.	Jáchym Löwenhöffer	GEvolutionJM	2	1						0,0	0,0	4,0
	Matyáš Sirotek	GJirsíkaČB	4	1						0,0	0,0	4,0
86.–89.	Martin Dobruský	SŠKKamPard	4	1						0,0	0,0	3,0
	Patrik Havlík	GNAlejíPH	0	1	3					3,0	0,0	3,0
	Miroslav Kolouch	GJírovcČB	3	1						0,0	0,0	3,0
	Petr Šišlák	GZborovPH	2	1						0,0	0,0	3,0
90.	Olga Cinková	ArcibisGPH	3	1						0,0	0,0	2,6
91.–93.	Alexandr Bihun	GJírovcČB	3	1						0,0	0,0	2,0
	Vojtěch Kosina	G Chrudim	2	1						0,0	0,0	2,0
	Jakub Vlček	GPříbor	4	1						0,0	0,0	2,0
94.	Vít Faltus	GZborovPH	3	1						0,0	0,0	1,0

Výsledková listina KSP-X po páté sérii třicátého pátého ročníku

	řešitel	*škola*	*ročník*	*sérií*	*1-X1*	*2-X1*	*3-X1*	*4-X1*	*celkem*
0.					10	10	10	10	40,0
1.	Benjamin Swart	MensaG	4	9	8	10	8	13	39,0
2.	Patrik Číhal	SŠKKamPard	3	8	8	9	4		21,0
3.	Štěpán Mikéska	GJarošeBO	4	4	1	8		0	9,0
4.	Anna-Kristina Migel	GNAlejíPH	0	4	6				6,0
5.	Erik Ježek	SPŠSmíchov	1	4	5				5,0
6.	Vojtěch Lančarič	SPŠG Třebešín	4	9				3,3	3,3
7.	Erik Sabol	GČeskoliPH	3	4	2				2,0
8.–9.	Filip Prášil	GJiříPoděb	4	5	1				1,0
	Patrik Prítrský	GGrössBA	2	4	1				1,0

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.

matfyz

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/*