Korespondenční Seminář z Programování

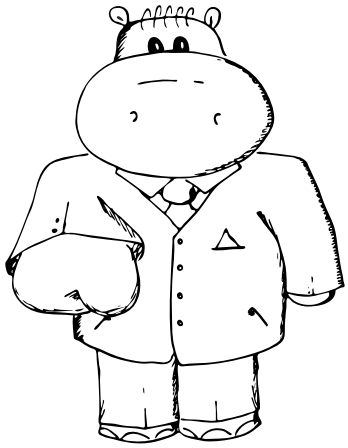
35. ročník | **KSP** | **Únor 2023**

Vzorová řešení třetí série třicátého pátého ročníku KSP

35-3-1 Čajovod

Nejprve si rozmysleme lehčí variantu, v níž neexistují manažeři. Určitě pak nemá smysl nějaké kohouty zavírat – nepomůže nám to a spotřebujeme tah navíc. Můžeme tedy ignorovat nejen prázdné místnosti, ale i programátory, k nimž aktuálně čaj proudí. Pro ty na suchu zase existuje jednoznačná množina kohoutků, které musíme otevřít (každé 2 vrcholy ve stromu spojuje právě jedna cesta).

Jak je najít? Nabízí se využít rekurzivní struktury stromu a prohledat jej do hloubky. Pro každý vrchol si zjistíme, jestli se v jeho podstromu nachází místnost s programátory. V listech to je `ANO` pro programátory a `NE` pro prázdnou místnost. Ve vnitřních vrcholech spočítáme logickou disjunkci (tzv. OR) z potomků. Jestliže je kohoutek na nějaké hraně zavřený a zároveň vrchol, do kterého hrana vede, obsahuje programátory v podstromu, otevřeme jej.



Proč nám tento postup nefunguje pro vstupy s manažery? Pořád přeci platí, že všechny kohoutky na cestách k programátorům musí být otevřené. Jenže pro manažery je situace jiná. Stačí zavřít jeden kohoutek na cestě k nim, máme tedy na výběr. Chceme zvolit takovou variantu, která povede na nejmenší počet změn.

Všechny varianty vyzkoušet nemůžeme, je jich $\mathcal{O}(2^N)$. Budou ale často sdílet společné části a i náš výpočet bude pořád dokola rozmýšlet optimální řešení téhož podstromu, akorát pro jiné způsoby řešení předků. Co takhle si mezivýsledky ukládat? Různých podstromů je přeci jen N. Tento způsob zrychlení algoritmu se obvykle nazývá dynamické programování.[1]

Stále budeme používat DFS. Náš plán bude pro každý vrchol spočítat, kolik nejméně změn v podstromu potřebujeme provést. Toto navíc spočítáme pro dva případy: pokud do vrcholu čaj přitéká a pokud ne. Pokud v daném případě nelze splnit podmínky (například by dotekl čaj k manažerům), jako počet změn použijeme nekonečno.

Jakmile tuto informaci známe pro všechny syny daného vrcholu, můžeme zpracovat vrchol samotný. Projdeme všechny hrany, které z vrcholu vedou, a pro každou se rozhodneme, jestli je výhodnější její stav změnit nebo ne. Nakonec sečteme změny na všech hranách i v jejich podstromech a součet uložíme do vrcholu, abychom tuto hodnotu mohli použít, až budeme zpracovávat rodiče.

Jakou časovou složitost máme teď? Každý vrchol zpracujeme jen jednou, poté jednou použijeme hodnoty uložené v něm. Časová složitost je proto $\mathcal{O}(N)$. Obdobně paměťová, pořídili jsme si navíc jen dvě počítadla na každý vrchol.

Poznámka k implementaci: Jako vhodné nekonečno pro tahy, které nesplňují podmínky, můžeme zvolit třeba N, neboť více tahů se ani udělat nedá.

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

Úlohu připravili: Vojta Káně, Dan Skýpala

35-3-2 Hroší hortenzie

Problém můžeme reformulovat tak, že pro každý z N intervalů koncentrací (které chtějí kamarádi pít) chceme nalézt nejlevnější (co do počtu lístků) mističku, která dokáže vytvořit koncentraci v tomto rozpětí.

Fakt, že koncentrace mohou být opravdu velké, nám úlohu trochu komplikuje, avšak můžeme si povšimnout, že potřebujeme jen malý zlomek těchto hodnot. V podstatě nás zajímají jen hodnoty koncentrací misek, abychom mohli určit, kterou misku použijeme. A protože našich mističek je pouze K, můžeme přečíslovat celý rozsah koncentrací na čísla 1 až K. To uděláme tak, že seřadíme mističky podle koncentrací vzestupně a přiřadíme jim čísla (indexy) 1 až K. Pak postupně přečíslujeme všech N požadovaných intervalů, aby interval popisoval rozsah indexů mističek, ve kterých může Lucka danému kamarádovi připravit čaj. Přečíslování zvládneme rychle binárním vyhledáváním.

Nyní pojďme vyřešit zbytek úlohy. Pro každý z N intervalů chceme udělat minimový dotaz, tedy chceme zjistit mističku z tohoto intervalu, na kterou potřebujeme minimum lístků. K tomu využijeme intervalový strom. Pokud intervalové stromy neznáte, podívejte se do naší kuchařky.[2]

Sestavíme si intervalový strom obsahující všechny mističky, přičemž se budeme dotazovat na minimum potřebných lístků. Tedy nám stačí udělat N dotazů, z nichž každý zabere $\mathcal{O}(\log K)$ času, celkově tedy $\mathcal{O}(N \log K)$.

Ještě však musíme zvážit původní přečíslování, které nám zabralo $\mathcal{O}(N \log K)$ (N binárních vyhledání v K mističkách) a také na začátku musíme intervalový strom sestavit, to však zabere maximálně $\mathcal{O}(K \log K)$ času.

Tedy celková složitost je $\mathcal{O}(N \log K)$.

Úlohu připravili: Kristýna Petrlíková, Lukáš Veškrna

[1] `http://ksp.mff.cuni.cz/viz/kucharky/dynamicke-programovani`
[2] `http://ksp.mff.cuni.cz/viz/kucharky/intervalove-stromy`

35-3-3 Prodlužovačky

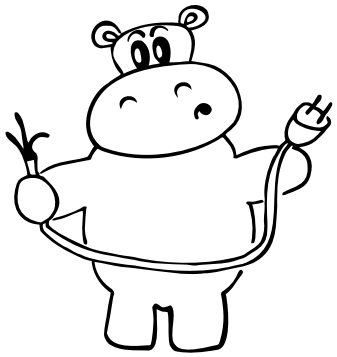
Úlohu budeme řešit po patrech odspoda, tedy začneme v patře stromu s nejvyšší hloubkou a postupně budeme spravovat vrcholy na vyšších a vyšších hladinách, až dorazíme do kořene, který je určitě v pořádku – zeď má neomezenou kapacitu.

Buďme v nějakém vrcholu v a předpokládejme, že v celém odpovídajícím podstromu jsou již zařízení popřepojovány, aby odpovídající příkon byl nižší než daná kapacita, a zbývá nám opravit v.

Pokud součet příkonu v podstromu nepřesahuje kapacitu v, není třeba nic opravovat a můžeme pokračovat do vyšších pater. V opačném případě je potřeba odpojit nějaká zařízení v odpovídajícím podstromu – odpojování mimo podstrom nijak neovlivňuje příkon do v.

Předpokládejme, že známe, která ještě neodpojená zařízení odpovídají podstromu a zajímá nás, která z nich musíme odpojit, abychom opravili v.

Nechť k je minimální číslo takové, že v lze opravit k odpojeními. Pokud však dokážeme opravit v odebráním nějakých k zařízení z daného podstromu, tak určitě dokážeme v opravit odpojením těch k předmětů s největším příkonem. To však vede na jednoduchý způsob, jak opravit vrchol v – budeme odpojovat zařízení od toho s největším příkonem, dokud v neopravíme.



Tato myšlenka vede na následující algoritmus. V obráceném BFS pořadí budeme procházet všechny vrcholy a každý spravíme odebíráním neodebraných spotřebičů s největším příkonem.

Potřebujeme však ještě dokázat, že tento lokálně optimální algoritmus vrátí řešení, které je optimální i globálně. To však není těžké nahlédnout indukcí podle hladiny. V daném vrcholu v potřebujeme odebrat alespoň k zařízení – stejně jako pro v, tak i pro všechny jeho předchůdce však nikdy nebude na škodu vzít ty s největším příkonem. Pokud bychom jich chtěli vybrat více než k, pak jich můžeme vybrat jen k a ten zbytek později, až to bude potřeba. Tedy lokálně optimální výběr zachovává globální optimalitu.

Zbývá si tedy rozmyslet, jak algoritmus efektivně implementovat. V každém vrcholu potřebujeme hledat a odebírat minima a spojovat zařízení ze synů – tedy podporovat operaci slévání. Na to můžeme použít haldy, které podporují slévání – například binomiální (či Fibonacciho) haldu, kde slévání a odebírání minima umíme v logaritmickém čase. Algoritmus bychom pak mohli implementovat jako procházení v obráceném BFS pořadí, nebo jako DFS (můžeme si rozmyslet, že při DFS již rovněž známe všechna odebraná zařízení z podstromu).

A jaká je časová složitost? Každé zařízení odebereme nejvýše jednou a práci na slití v haldě můžeme naúčtovat synovi, celkem tak dostáváme linearitmickou složitost, tedy $\mathcal{O}(N \log N)$.

Tento algoritmus je sice po teoretické stránce uspokojivý, neboť má příjemnou asymptotickou složitost, jeho implementace by ale byla složitější, neboť vyžaduje některou z pokročilých hald.

Místo haldy bychom mohli ale použít obyčejný binární vyhledávací strom a slévání dvou stromů implementovat tak, že prvky z menšího ze dvou stromů popřesouváme do většího. Kdybychom prvky nemazali, nebylo by složité nahlédnout, že každý prvek přesuneme $\mathcal{O}(\log N)$-krát. Tedy celková časová složitost by byla $\mathcal{O}(N \log^2 N)$. Pokud prvky mažeme, každý prvek můžeme přesunout vícekrát, ale v každém kroku pracujeme jen s haldami, které nemají větší velikosti, než kdybychom prvky vůbec nemazali. To ale znamená, že provedeme nejvýše tolik práce a celková složitost tak zůstává $\mathcal{O}(N \log^2 N)$.

Úlohu připravili: Robert Jaworski, Ondra Sladký

35-3-4 Záchrana listí

Lehká varianta

Nejdříve se podívejme, jak řešit jednoduchou variantu, a potom budeme zobecňovat. Představme si, že chceme víko přivázané k i-té krabici někam položit. V takové chvíli je jedno, zdali jsou na krabicích 0 až $i-2$ položená víka, protože tam provázek nedosáhne. Stačí tedy zohlednit krabice $i-1$ až $i+1$.

To navádí na dynamické programování. Postupně budeme přidávat krabice a vždy si budeme pamatovat, kolik nejvíce listí umíme zachránit. Dle pozorování výše však kromě aktuální krabice musíme uvážit stav předešlé a následující krabice. Na každé z těchto tří krabic buď leží víko, nebo ne. Budeme si tedy pamatovat maximum pro každou z těchto kombinací. Pokud některá kombinace nemůže nastat (např. nemáme dost vík), pak si pro danou kombinaci zapamatujeme minus nekonečno.

Označme si $dp[i][c]$ hodnotu pro kombinaci c s poslední krabicí na pozici i. Např. pokud v naší kombinaci víko na předchozí krabici položené není, aktuální je, a následující také je, značme ji $dp[i][(0,1,1)]$.

Nyní bychom chtěli z kombinací a hodnot končících na pozici i určit hodnoty pro kombinace na pozici $i+1$. Nejdřív posuňme všechny kombinace. To jen z naší kombinace vypadne levá krabice a vpravo přibude nová, zatím bez víka: $(0,1,1) \to (1,1,0)$. Nový stav mohl vzniknout ze dvou různých předchozích stavů, z nich vybereme ten lepší:

$$dp[i+1][(a,b,0)] = \max(dp[i][(0,a,b)], dp[i][(1,a,b)])$$

A pokud je k současné krabici přivázané víko, zkusíme jej postupně přidat na každou volnou krabici v naší kombinaci. Pokud takto najdeme nové maximum, použijeme jej pro nově vzniklou kombinaci.

Na závěr vybereme tu kombinaci, která nám dává nejvíce suchého čaje. A na to, abychom zjistili, kam víka dávat, můžeme použít obvyklý trik – pamatujeme si, jak jsme nové hodnoty kombinací volili, a jen jdeme pozpátku.

Pro $K = 1$ bude výsledná časová složitost $\mathcal{O}(N)$.

Těžká varianta

Na těžkou variantu bychom si samozřejmě mohli udržovat všechny možné stavy, zdali příslušné krabice mají a nemají víko, těch je ale $\mathcal{O}(2^K)$. Všimněme si ale, že můžeme vždy víka dát tak, aby se provázky ke krabicím nekřížily. Je to proto, že kdybychom prohodili zkřížená víka, zmenšíme vzdálenosti vík od krabic s provázkem, čímž nic nezkazíme.

Díky tomu jsme schopni udržovat kombinace podle toho, kam jsme položili poslední víko. Posunutí krabic o jedna dozadu znamená pouze posunutí o 1 pozici zpátky.

Je-li k aktuální krabici přivázané víko, tak ho můžeme položit na kteroukoliv pozici za posledním víkem. Pro každou tuto pozici najdeme nejlepší z předešlých kombinací.

Ještě můžeme přidat jedno zrychlení, nejlepší kombinaci totiž nemusíme hledat pořád dokola. Vše zvládneme v jediném průchodu, nejlepší kombinaci stačí průběžně aktualizovat během toho, co zkoušíme pokládat víka.

Tím se dostaneme na výslednou časovou složitost $\mathcal{O}(NK)$, která stačí na plný počet bodů.

Program (Python):
`http://ksp.mff.cuni.cz/viz/35-3-4.py`

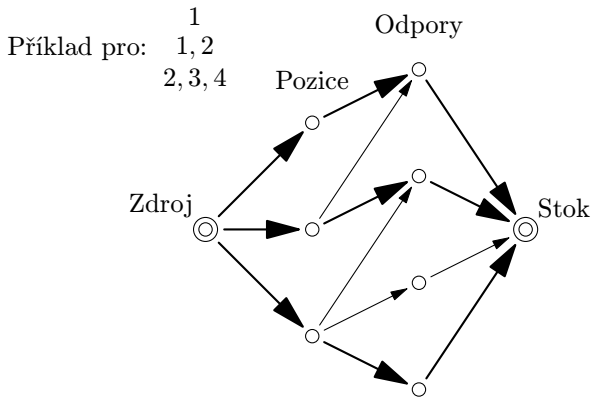
Úlohu připravili: Michal Kodad, Dan Skýpala

35-3-X1 Obvody

Naivní řešení

Nejprve se pojďme zamyslet nad tím, jak by se úloha řešila v případě, že bychom pro každou pozici měli jen seznam povolených rezistorů, nikoliv tedy intervaly. Toto je vcelku standardní úloha, kterou lze řešit pomocí toků v síti.[3] Pro každou pozici si vytvoříme vrchol napojený ze zdroje jednotkovou hranou. Každé z možných hodnot odporů vyrobíme vrchol připojený na stok taktéž jednotkovou hranou. Pak už jen přidáme hranu spojující každou pozici s na ní povoleným odporem. Na výslednou síť spustíme algoritmus, který nám najde maximální tok.

Pokud bude velikost toku stejná, jako počet pozic, tak snadno zvládneme zkonstruovat řešení. Přes každou pozici musí protékat jedna jednotka toku. Ta poteče dále do odporu. Ten použijeme na danou pozici. Jelikož má odpor jen jednu odtokovou hranu, tak jím může protékat nejvýše jedna jednotka toku, a tedy odpor může být použit v nejvýše jedné pozici. Řešení je tedy validní.



Dále se dá nahlédnout, že když řešení existuje, tak maximální tok musí mít velikost odpovídající počtu pozic na rezistory. Větší být evidentně nemůže, protože ze zdroje toho nemůže vytékat více. Podobně jako je popsané v předešlém odstavci, pokud bychom znali řešení úlohy, zvládli bychom zkonstruovat tok s velikostí počtu pozic. Maximální tok tedy nemůže být menší. Tedy když maximální tok není dostatečně velký, máme zaručeno, že řešení neexistuje.

Časová složitost takového řešení se odvíjí od použitého algoritmu na toky. Ten řešíme pro $\Theta(N+V)$ vrcholů a $\Theta(N+M+V)$ hran, kde N značí počet pozic na rezistory, M celkový počet intervalů (tedy součet délek seznamů povolených rezistorů v pozicích) a V je počet rezistorů, které lze použít.

Popsaný algoritmus je ve skutečnosti algoritmem na maximální párování popsaný ve výše zmíněné kuchařce.

Výše popsaný způsob můžeme využít i na řešení zadané úlohy. Prostě jen nahradíme každý interval za seznam všech celých čísel v něm. Ovšem tímto postupem může vzniknou graf s až $\Theta(NV)$ hranami, což už je příliš.

Drobná optimalizace

Dá se všimnout, že když do jedné pozice padne alespoň N rezistorů, tak ať umístíme do ostatních pozic cokoliv, nějaký z možných rezistorů nám zbude. Tedy můžeme nejprve vyřešit úlohy bez pozic s hodně možnostmi a ty začít umisťovat až poté. Do grafu nebudeme dávat vrcholy rezistorů, do kterých by nevedla žádná hrana.

Tím se nám velikost grafu omezí na $\mathcal{O}(N^2)$ vrcholů i hran. To už je algoritmus, jehož složitost závisí pouze na délce vstupu.

Laskavý čtenář nechť si rozmyslí implementační detaily stavění sítě a hledání nevyužitých odporů pro pozice s hodně možnostmi. Slibujeme, že to lze udělat v čase $\mathcal{O}(M \log M)$. Na hledání nevyužitých odporů stačí jedno setřídění a pak chytřejší průchod.

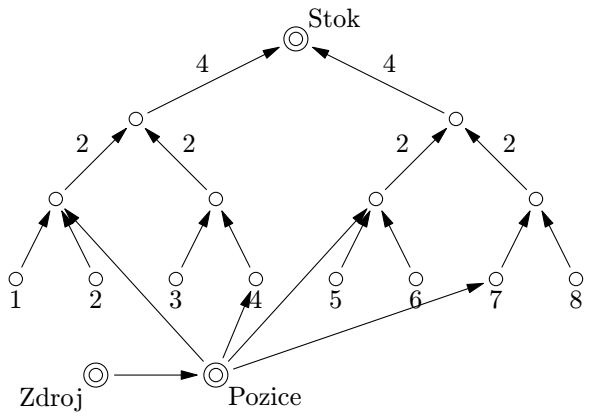
Rychlejší řešení

Vzhledem k tomu, že v úloze se pracuje s intervaly, je na čase použít intervalové stromy.[4] Tentokrát je ovšem nevyužijeme jako datovou strukturu, ale jejich princip využijeme při konstrukci sítě, na kterou pak spustíme algoritmus na toky. Postavíme si intervalový strom nad možnými odpory. Kořen bude stok a z ostatních vrcholů povede do otce hrana s hodnotou odpovídající délce intervalu, který vrchol reprezentuje.

Na tento strom dále připojíme vrcholy pozic na odpory. Připojování bude analogické operaci sečtení intervalu. Při sčítání vybereme vrcholy stromu tak, aby dohromady pokryly celý interval. My tedy připojíme jednotkové hrany tak, aby každá pozice byla pro každý interval připojená na ty vrcholy, které bychom chtěli sečíst.

[3] `http://ksp.mff.cuni.cz/viz/kucharky/toky-v-sitich`
[4] `http://ksp.mff.cuni.cz/viz/kucharky/intervalove-stromy`

Příklad pro: $[1,2],[4,7]$



V takovém grafu najdeme maximální tok. Jak ale přečíst výsledné řešení? Začneme od listů intervalového stromu. Když tok teče do nějakého listu, tak daný odpor přiřadíme pozici, odkud tok teče. Dále budeme vyřizovat vyšší a vyšší úrovně stromu. Pro každou jednotku toku z vrcholů pozic se pokusíme najít volný rezistor v podstromu. Celý podstrom je totiž pro danou pozici možné využít. Navíc máme zaručeno, že v podstromu bude dostatek nevyužitých odporů, protože jediná hrana vedoucí z podstromu má kapacitu stejnou jako počet odporů v podstromu.

Pro nalezení nevyužitého odporu můžeme využít intervalového stromu. V každém vrcholu si budeme pamatovat počet ještě nevyužitých odporů a vždy půjdeme do podstromu, kde ještě nějaký je.

Stejně jako v naivním řešení zvládneme konstrukci udělat i v opačném směru, takže pokud tok není dostatečně velký, řešení neexistuje.

Alternativně si můžeme všimnout, že už to, že nám tok ukázal, z kterého intervalu máme do každé pozice přiřadit odpor, je dostatečné zjednodušení úlohy. Když totiž do každé pozice patří jen jeden interval, můžeme využít hladové řešení.

Výše popsaným řešením získáme graf s $\mathcal{O}(N+V)$ vrcholy, ovšem jen s $\mathcal{O}(M \log V + V)$ hranami.

Optimalizace na závěr

Nechť $[a,b]$ je nějaký interval hodnot odporů, uvnitř něhož nezačíná ani nekončí žádný ze zadaných intervalů. Pak listy intervaláče odpovídající rezistorům $a, a+1, \ldots, b$ můžeme nahradit za jeden list odpovídající celému intervalu. Hrana z tohoto vrcholu pak bude mít kapacitu $b-a+1$ a následující hrany v intervalovém stromě také budou příslušně zvětšené.

Takto můžeme zkomprimovat všechny intervaly, kde nic nezačíná ani nekončí, a postavit intervalový strom až na nich.

Jaké všechny intervaly to jsou můžeme zjistit tak, že si setřídíme všechny začátky a konce. Tyto hodnoty pak projdeme a spojíme úseky, kde nic není. Ke každé hodnotě si rovnou můžeme psát, do kolikátého z nových listů patří.

Touto modifikací se zmenší počet listů intervaláče na $\mathcal{O}(M)$. Tedy dostaneme graf s $\mathcal{O}(M)$ vrcholy a $\mathcal{O}(M \log M)$ hranami. Až na nalezení toku mají zbylé části algoritmu časovou složitost $\mathcal{O}(M \log M)$ a paměťovou $\mathcal{O}(M)$.

Úlohu připravili: Jirka Kalvoda, Dan Skýpala

Výsledková listina třetí série třicátého pátého ročníku KSP

	řešitel	*škola*	*ročník*	*sérií*	*3-1*	*3-2*	*3-3*	*3-4*	*3-S*	*3-X1*	*série*	*KSP-X*	*celkem*
0.					10	10	12	13	15	10	60,0	30,0	180,0
1.	Benjamin Swart	MensaG	4	8	10	10	12	13	15	8	60,0	26,0	179,5
2.	Štěpán Mikéska	GJarošeBO	4	3	10	10	12	13	14		59,0	9,0	176,5
3.	Kateřina Doubková	GNAlejíPH	4	4	10	10	12	13	14,5		59,5	0,0	167,5
4.–5.	Erik Ježek	SPŠSmíchov	1	3	10	10	12	13	13		58,0	5,0	165,0
	Filip Prášil	GJiříPoděb	4	4	10	10	12	13	14		59,0	1,0	165,0
6.	David Kolář	GJírovcČB	4	8	10	10	12	13			45,0	0,0	157,5
7.	Ondřej Pupík	GRožnovPR	3	5	10	10	9	13	6		48,0	0,0	149,5
8.	Viktor Helmich	GTMannaPH	4	3	10	4	6	5	13,5		38,5	0,0	147,5
9.	Jáchym Kouba	GJŠkodyPŘ	3	8	10	10	8	3	9,5		40,5	0,0	146,0
10.	Patrik Číhal	SŠKKamPard	3	7	10	10	12	13		4	45,0	21,0	144,0
11.	Marek Raška	GTři	4	3	10	10	12	0	9		41,0	0,0	140,0
12.	Vojtěch Lančarič	SPŠG Třebešín	4	8	10	1	6	3			20,0	0,0	137,5
13.	Adam Červenka	GJarošeBO	4	3	2	10	8	13	9,5		42,5	0,0	126,0
14.	Kryštof Tahal	GUBalvanJN	4	3	10	4	6	11	3,5		34,5	0,0	125,0
15.	Matúš Púll	GZborovPH	3	6	10	5	4	3	12		34,0	0,0	122,5
16.	Anna-Kristina Migel	GNAlejíPH	0	3	10	10	11	3			34,0	6,0	122,0
17.	Zuzana Aubrechtová	GHeyrovPH	4	7	10	10	6	1			27,0	0,0	120,0
18.	Daniel Culliver	GZborovPH	3	5	10	10	5	3	13		41,0	0,0	116,5
19.	Honza Kocourek	ParkLane	3	4	10	4	6		7,5		27,5	0,0	113,0
20.	Adam Kolník	SSŠVTPraha	4	12	10	10	12	13			45,0	0,0	110,0
21.	Adam Jahoda	GKepleraPH	4	5	10	4	6	0	10		30,0	0,0	109,5
22.–23.	Albert Bakoč	GZborovPH	2	7	10		2		6		18,0	0,0	107,0
	Patrik Prítrský	GGrössBA	2	3	10			13			23,0	1,0	107,0
24.	Vít Kaděra	G Wicht	1	3	10		7	3	6,5		26,5	0,0	99,5
25.	Petr Slonek	GJarošeBO	4	3	10	10	7				27,0	0,0	86,5
26.	Martin Šindelář	GGrössBA	3	3	10	8	10				28,0	0,0	84,5
27.	Jakub Ondroušek	GTomkovaOL	3	13	10			9			19,0	0,0	81,5
28.	Michael Jarvis	GŠpitálsPH	1	3	10						10,0	0,0	78,5
29.	Jakub Konc	GPároNitra	4	2	10			13			23,0	0,0	71,0
30.	Kryštof Marek	SGPCE	3	4	10	4	6	3			23,0	0,0	65,0
31.	Robert Klimt	GDobříš	3	3	1						1,0	0,0	63,0
32.	Kateřina Vomelová	GÚstavníPH	3	5	10		6	1			17,0	0,0	55,0
33.	Jakub Hampl	GMělník	3	4	10			1			11,0	0,0	52,5
34.	Jan Ševeček	G UherBrod	1	3	10			1			11,0	0,0	51,0
35.	Petr Němec	G Wicht	1	2							0,0	0,0	46,0
36.	Jan Slíva	MensaG	2	6							0,0	0,0	45,0
37.	Viktor Číhal	SPŠSmíchov	3	6							0,0	0,0	42,0
38.	Ondřej Sedláček	GOPavla PH	2	1	10	10	8	3	10		41,0	0,0	41,0
39.	Erik Sabol	GČeskoliPH	3	3							0,0	2,0	36,5
40.	Adam Houdek	SOŠ Březová	−2	2	10		7				17,0	0,0	36,0
41.	Oto Skýpala	GJŠkodyPŘ	−1	3	7						7,0	0,0	35,0
42.	Finley Stuart	GPísnickáPH	2	3	10			0			10,0	0,0	34,0
43.	Richard Dobíšek	MensaG	2	2	10						10,0	0,0	32,0
44.	Julie Krejčí	PraKonz	3	4	5						5,0	0,0	31,0
45.	Jakub Binter	GČeskáČB	0	1							0,0	0,0	30,0
46.	Lucian Poljak	GJŠkodyPŘ	1	3	7			0			7,0	0,0	27,0
47.	Vladimír Sklenár	GTerVans	3	8	10		4	0			14,0	0,0	24,5
48.	Ondřej Novák	G Brandýs	1	3	2				5		7,0	0,0	23,0
49.	Martin Skýpala	GJŠkodyPŘ	3	1							0,0	0,0	21,0
50.	Ivan Žemlička	GÚstavníPH	2	1	10	10					20,0	0,0	20,0
51.	Janek Hlavatý	GJirsíkaČB	4	10							0,0	0,0	19,0
52.–54.	David Bojko	GMělník	1	3	5			1			6,0	0,0	16,0
	Petr Dymanus	GŠpitálsPH	3	1							0,0	0,0	16,0
	Richard Tichý	SPŠSmíchov	1	4	7	1	8				16,0	0,0	16,0
55.–56.	Michal Martínek	GÚstavníPH	2	3							0,0	0,0	15,0
	Vojtěch Procházka	MensaG	2	2	10						10,0	0,0	15,0

	řešitel	*škola*	*ročník*	*sérií*	*3-1*	*3-2*	*3-3*	*3-4*	*3-S*	*3-X1*	*série*	*KSP-X*	*celkem*
57.	Radomír Budínek	GŘíč	4	1							0,0	0,0	12,0
58.–59.	Tomáš Kazimír	GNPr	3	1							0,0	0,0	11,0
	Michal Mentzl	G UherBrod	3	1	10			1			11,0	0,0	11,0
60.–62.	Jakub Kopčil	GMikulášPL	4	3							0,0	0,0	10,0
	Jan Kotovský	GPísnickáPH	4	10							0,0	0,0	10,0
	Filip Maňas	GJarošeBO	2	1	10						10,0	0,0	10,0
63.	Dominik Malý	GBSučany	4	1							0,0	0,0	7,0
64.–68.	Vojtěch Bízek	GPísnickáPH	3	1							0,0	0,0	6,0
	Adam Bureš	SPŠ Přerov	3	1							0,0	0,0	6,0
	Lída Kačenková	GBudějovPH	4	2		3					3,0	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
69.–70.	Matěj Smetana	AkademGPH	2	1							0,0	0,0	5,0
	Jan James Soukup	GKlatovy	4	1							0,0	0,0	5,0
71.	Jáchym Löwenhöffer	GEvolutionJM	2	1							0,0	0,0	4,0
72.–76.	Martin Dobruský	SŠKKamPard	4	1							0,0	0,0	3,0
	Miroslav Kolouch	GJírovcČB	3	1							0,0	0,0	3,0
	Andrea Mikulová	BGOstrava	4	3							0,0	0,0	3,0
	Vít Mitáš	GPolička	1	2							0,0	0,0	3,0
	Petr Šišlák	GZborovPH	2	1							0,0	0,0	3,0
77.–79.	Alexandr Bihun	GJírovcČB	3	1							0,0	0,0	2,0
	Vojtěch Kosina	G Chrudim	2	1	2	0					2,0	0,0	2,0
	Jakub Vlček	GPříbor	4	1							0,0	0,0	2,0
80.	Michael Ambros	GTomkovaOL	0	1			1				1,0	0,0	1,0

Výsledková listina KSP-X po třetí sérii třicátého pátého ročníku

	řešitel	*škola*	*ročník*	*sérií*	*1-X1*	*2-X1*	*3-X1*	*celkem*
0.					10	10	10	30,0
1.	Benjamin Swart	MensaG	4	8	8	10	8	26,0
2.	Patrik Číhal	SŠKKamPard	3	7	8	9	4	21,0
3.	Štěpán Mikéska	GJarošeBO	4	3	1	8		9,0
4.	Anna-Kristina Migel	GNAlejíPH	0	3	6			6,0
5.	Erik Ježek	SPŠSmíchov	1	3	5			5,0
6.	Erik Sabol	GČeskoliPH	3	3	2			2,0
7.–8.	Filip Prášil	GJiříPoděb	4	4	1			1,0
	Patrik Prítrský	GGrössBA	2	3	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.

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/*