	řešitel	škola	ročník	sérií	1-1	1-2	1-3	1-4	1-5	1-6	série	celkem
45.	Ondřej Bleha	GBNěmcovHK	4	3	6						9,0	9,0
46.–47.	Ondřej Daniš	GFPValMez	4	1		0	1	3			8,0	8,0
	Kristýna Prokopová	GJosBožČT	3	1			1	3			8,0	8,0
48.–54.	Robert Jaworski	GÚstavníPH	1	1	4						7,6	7,6
	Vojtěch Jedlička	GCoubTábor	2	1	4						7,6	7,6
	Petr Khartskhaev	PORGPha	2	1	4						7,6	7,6
	David Krásný	SPŠEPlzeň	2	1	4						7,6	7,6
	Petr Macháček	GTýnNVlt	3	1	4						7,6	7,6
	Jan Najman	SPSEPard	2	1	4						7,6	7,6
	Jakub Vybiral	GLovosice	2	1	4						7,6	7,6
55.	Vít Skalický	GPísnickáPH	1	5	4						6,3	6,3
56.–57.	Vít Gardoň	GPří	3	1				3			5,5	5,5
	Ondřej Chlubna	GOrlová	2	1				3			5,5	5,5
58.–60.	Matyáš Boháček	ZŠKladskáPH	1	1	2						4,7	4,7
	Tomáš Pelák	SŠkybernHK	3	1	2						4,7	4,7
	Matej Straka	SPŠEPrešov	4	1	2						4,7	4,7
61.	Ondřej Cach	SPSEPard	3	2	2						4,4	4,4
62.	Vojtěch Crha	GČeskoliPH	4	1	0	0,5	0,5	0,5			4,1	4,1
63.	Anna Hollmannová	GSRandyJN	2	4		1	1				4,0	4,0

KSP pro vás připravují studenti Matematicko-fyzikální fakulty Univerzity Karlovy.

Webové stránky: *https://ksp.mff.cuni.cz/*
E-mail: *ksp@mff.cuni.cz*
Diskusní fórum: *https://ksp.mff.cuni.cz/forum/*

Chcete-li s námi komunikovat bezpečně, můžete si ověřit náš HTTPS certifikát – jeho SHA1 fingerprint je: `E9:DB:EE:C6:62:BC:14:DE:09:E4:E8:97:DC:36:0E:87:B3:50:B0:01`.

Korespondenční Seminář z Programování

31. ročník KSP **Prosinec 2018**

Milí řešitelé, milé řešitelky!

První sníh už dávno napadl a zase roztál, Vánoce a nový rok už klepou na dveře a organizátoři si konečně našli čas na opravení vašich řešení první série. Za způsobenou prodlevu se vám omlouváme.

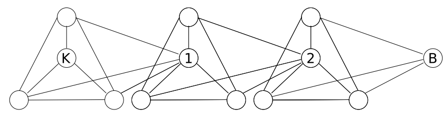
Právě si prohlížíte historicky první komentáře k úlohám, konkrétně úlohám letošní první série hlavní kategorie. Od letoška jsou totiž řešení každé série rozdělena na dvě části: na samotná autorská řešení, která vydáváme brzy po termínu série, a komentáře k došlým řešením, která vydáváme až po opravení vašich řešení.

Pokud se vám cokoliv nezdá nebo máte nějaký dotaz, neváhejte se ozvat na našem fóru nebo emailem na známou adresu.

Komentáře k první sérii třicátého prvního ročníku KSP

31-1-2 Skoro nejkratší cesta

Hlavním problémem bylo, že málokdo přečetl správně zadání úlohy, které se ptalo na počet cest, nikoliv na cesty. To je problém, protože cest může být i exponenciálně mnoho (viz graf níže).



Jinak se sešla pěkná řešení, která úlohu řešila jedním ze tří způsobů:

- Ekvivaletně ke vzorovému řešení.
- Někteří řešitelé si všimli, že lze napočítat cesty a délky od B (babiččin vrchol) a od K (Karkulčin vrchol) pomocí BFS (tak, jak je to popsáno ve vzorovém řešení). Pak můžeme projít každou hranu (z u do v), a pokud by tato hrana prodloužila cesty z K do u a z v do B na skoro nejkratší, tak vynásobením počtu nejkratších cest mezi K a u a počtu nejkratších cest mezi B a v zjistíme počet skoro nejkratších cest přes vrcholy u a v.
- Další přístup byl počítání cest délky k pro $k = 1$ až $k = \ell + 1$, kde ℓ je délka nejkratší cesty z K do B, tu si můžeme spočítat například pomocí BFS.

Vojta Sejkora

31-1-3 Řazení kořenek

Většina funkčních řešení byla podobná tomu vzorovému: našla nejdelší rostoucí podposloupnost (více či méně efektivně) a pak vypsala, jak popřesouvat zbývající kořenky. Vypisování se málokdo věnoval pečlivě – konec konců, ze zadání nebylo jasné, co přesně se má vypsat. Ale jak je vidět ze vzorového řešení, vypisování je nakonec ta nejzajímavější část úlohy. Proto jsme těm, kdo ho zvládli správně a rychle, přidělili jeden bod nad maximum.

Dynamické programování

Se zajímavým řešením přišel Pepa Minařík. Nechť $a_1, \ldots, a_n$ je permutace k setřídění a a_k její nejmenší prvek. Ten musíme nějak dostat na začátek, což lze učinit dvěma způsoby:

- Buďto někdy (bez újmy na obecnosti hned na začátku) přesuneme a_k před všechny ostatní prvky. Pak zbývá dotřídit posloupnost $a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n$.
- Anebo to neuděláme, ale pak musíme někdy (bez újmy na obecnosti úplně na konci) přesunout prvky $a_1, \ldots, a_{k-1}$. Tehdy už musí být setříděné $a_{k+1}, \ldots, a_n$.

Označíme-li minimální počet přesunů $P(a_1, a_2, \ldots, a_n)$, musí tedy platit

$$P(a_1, \ldots, a_n) = \min \begin{cases} 1 + P(a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n), \\ (k-1) + P(a_{k+1}, \ldots, a_n). \end{cases}$$

Nabízí se založit na tomto vztahu dynamické programování: počítat P od nejkratších posloupností k těm nejdelším. Na první pohled to vypadá, že ho budeme muset spočítat pro exponenciálně mnoho posloupností. Zachrání nás ovšem, že všechny z nich vzniknou ze zadané permutace vynecháním nejmenších ℓ prvků a následně zahozením prvních m prvků. Jsou tedy jednoznačně určené parametry ℓ a m, takže jich je jenom $\mathcal{O}(n^2)$.

Rozmyslete si, že pokud si dvojice (ℓ, m) šikovně uspořádáme, dokážeme každé $P(\ldots)$ spočítat v konstantním čase. Vyjde z toho poměrně jednoduchý kvadratický algoritmus. Pokud si během něj budeme udržovat, která z variant byla při výpočtu $P(\ldots)$ ta menší, můžeme snadno odpovědět, jaké přesuny máme provést.

Hladové algoritmy a jiná nefunkční cháska

Nyní se podívejme na některá slibná, ale nefunkční řešení. Můžeme například zkusit použít nějaký třídicí algoritmus, který málo přesouvá. Například Insertsort – třídění vkládáním. Ten ovšem selže na permutaci 4 1 2 3. Vybere nejprve 4 a pak ostatní čísla přesouvá před ni. Spotřebuje tedy 3 přesuny. Stačil by ale jeden: přesunout 4 za ostatní prvky. Podobně se dá vyvrátit optimalita dalších třídicích algoritmů.

Někteří řešitelé také zkoušeli „hladový" přístup: spočítat pro každé číslo, jak daleko je od své správné pozice, a zahájit přesouvání od toho, které je nejdál. To nefunguje například pro vstup 3 4 1 5 2: nejdále od správné pozice je 2, takže vytvoříme 3 2 4 1 5, což už nejde dotřídit jedním přesunem. Dva přesuny přitom k setřídění vstupu stačí: například přesunout 1 na začátek a pak 2 za 1.

Klárka Tauchmanová & Martin „Medvěd" Mareš

31-1-4 Myslivci

I přes poměrně velký počet řešení se mi skoro v žádném nepovedlo najít chybu. Asi bylo ze zadání docela jasné, co se má vlastně počítat, což je super. To, že nikdo nezůstal bez bodů, ale neznamená, že by bylo samozřejmé přijít na optimální řešení. Asi všichni, komu se to povedlo, byli blízko jednomu z algoritmů popisovaných ve vzorovém řešení – buď si setřídili obě pole a pak je najednou prošli, a nebo si setřídili jedno z nich a pak v něm binárně vyhledávali odpovídající prvky. První zmíněné bude asi rychlejší v praxi, druhé má zase teoretickou výhodu se strašlivě rozdílnými velikosti vstupů, takže úplně stejná nejsou; plný počet bodů byl ale za obě.

Standa Lukeš

31-1-5 Krájení bábovky

Úloha byla podle očekávání složitá, takže body dostaly i různé částečné pokusy, pokud fungovaly alespoň v některých rozumně vymezitelných případech (pravidelný mnohoúhelník, čtyřúhelník, …); k tomu asi není co poznamenávat.

Několik řešitelů se však dopustilo předpokladu, který jsme v zadání implicitně vyvraceli obrázkem. Jeden řešitel dokonce měl tolik drzosti, že vyjádřil ještě politování nad nepřesným obrázkem v zadání, za což byl adekvátně ohodnocen sníženou bodovou sazbou. Vězte, že obrázky v zadání se snažíme kreslit tak přesné, jak to je jenom možné, a pokud se tři přímky těsně (leč viditelně) neprotínají v jednom bodě, tak je to proto, že se v tom bodě opravdu neprotínají.

Nyní tedy ukážeme, že **v žádném konvexním mnohoúhelníku s konečným počtem stran neexistuje bod, kterým by procházelo více než konečně mnoho dělících úseček**. Dělící úsečka je zde taková, jejíž vrcholy leží na obvodu mnohoúhelníka a která dělí mnohoúhelník na dvě části o stejném obsahu.

Definice: Je zadán úhel φ s vrcholem V a polopřímkami Vr a Vs. Je také zadán konstantní obsah S a bod B uvnitř úhlu φ. Přímka p prochází bodem B a protíná polopřímky Vr a Vs v bodech R_i a S_i tak, že obsah trojúhelníka VR_iS_i je roven S.

Tvrzení: Existují nejvýše dvě přímky p pro zadaný úhel φ, obsah S a bod B, které splňují definici.

Důkaz: Podle Lemmatu o ose úhlu z řešení této úlohy[1] nalezneme takové body R a S, že $|VR| = |VS|$ a $S_{\triangle VRS} = S$. Pak podle téhož lemmatu $\overrightarrow{VR_i} = q_i\overrightarrow{VR}$ a $\overrightarrow{VS_i} = \frac{1}{q_i}\overrightarrow{VS}$ pro vhodné q_i, které nalezneme postupem v následujícím odstavci.

Ve dvourozměrném vektorovém prostoru s bází tvořenou vektory $\overrightarrow{VR}$ a $\overrightarrow{VS}$ si také vyjádříme vektor $\overrightarrow{VB} = \alpha\overrightarrow{VR} + \beta\overrightarrow{VS}$, kde $\alpha \geq 0$ a $\beta \geq 0$. Bod B ale také leží na úsečce mezi body R_i a S_i, proto platí, že $\overrightarrow{VB} = w\overrightarrow{VR_i} + (1-w)\overrightarrow{VS_i}$. Vztahy pro jednotlivé bázové vektory pak můžeme zkoumat zvlášť:

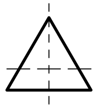
$$\alpha = w \cdot q_i, \qquad \beta = \frac{1-w}{q_i}$$

$$w = \frac{\alpha}{q_i} \quad \longrightarrow \quad \beta = \frac{1-\frac{\alpha}{q_i}}{q_i}$$

$$\beta q_i^2 = q_i - \alpha \quad \longrightarrow \quad \beta q_i^2 - q_i + \alpha = 0$$

Kvadratická rovnice pro q_i pak má nanejvýš dvě reálná řešení, takže existují nanejvýš dvě různé přímky p pro zadaný úhel, obsah S a bod B, které splňují definici.

Z tohoto lemmatu tedy vyplývá, že každým bodem uvnitř mnohoúhelníka prochází pro každou dvojici hran nanejvýš dvě dělící příčky mezi těmito hranami. A protože hran je konečný počet, tak i dělících příček procházejících libovolným zvoleným bodem je nanejvýš konečně mnoho.



Takový důkaz naštěstí nebylo třeba provádět. Stačilo si totiž uvědomit, kde leží nejkratší dělící příčka v rovnostranném trojúhelníku! Ani ta totiž neprochází těžištěm. Není to totiž těžnice/výška, jejíž délka je $\frac{a\sqrt{3}}{2}$, ale rovnoběžka se stranou, která rovnostranný trojúhelník rozdělí na lichoběžník a na menší rovnostranný trojúhelník o straně délky $\frac{a}{\sqrt{2}}$.

Maria Matějka

31-1-6 Hroznýšovo okno

To tak přijde QA inženýr do hospody, objedná si jedno pivo, dvě piva, nula piv, 9 999 999 piv, −1 pivo, ještěrku, xyzzy … První normální zákazník vejde do hospody, poprosí o jídelní lístek, hospoda vzplane jasným plamenem a všichni uhoří. (Zdroj: Twitter)[2]

Takový tester navštívil váš kód a zjistil, že zhusta funguje. Sice občas spadne, když zmáčknete nějaké tlačítko, u kterého jste to na začátku nečekali, to by GUI dělat nemělo, ale jinak v zásadě dělá, co má. To je moc fajn!

První úkol se ukázal jako velmi jednoduchý, neboť jej splnil prakticky každý. Někdo využil pojmenovanou funkci, jiný lambda funkci.

Druhý úkol byl přidat tlačítko Stop, které časovač zastaví. To bylo taky celkem bezproblémové. Nikde nebylo specifikováno, jestli se po zastavení má ještě aktualizovat čas, proto to také řada z vás neudělala, což je v pořádku. Mělo by ale aspoň nespadnout, když jej zmáčknete a časovač zatím ještě neběží; za takové opomenutí jste přišli o bod. Jedno z rozumných řešení bylo inicializovat časovač (`self.timer = QTimer()`) už v konstruktoru.

Třetí úkol se nakonec ukázal, že nebude činit větší problémy. Stejně jako ve druhém úkolu se vyskytovaly potíže s neexistencí časovače, pokud jsem si chtěla nastavit nejdřív délku intervalu a pak teprve časovač spustit (za to byl mínus bod). Někteří z vás interval násobili a dělili 2, jiní přičítali a odčítali zvolenou konstantu (třeba 100 milisekund), což bylo obojí zcela v pořádku.

Tiše jsem pro tento úkol ignorovala, když program při změně intervalu zahazoval délku předchozího ještě nedoběhlého intervalu, neboť to přisuzuji snaze vecpat do jednoho programu řešení všech tří úkolů. Stejně tak jsem neřešila případ, kdy jste nevypisovali čas, ale počet tiků časovače od začátku programu.

Ve **čtvrtém úkolu** ale program zhusta nefungoval dle zadání. Dá se to otestovat třeba tak, že spustíte časovač, vyskáčete s limitem na nějakou dlouhou hodnotu (třeba 32 sekund), počkáte těsně před konec lhůty a rychle oklikáte mínus, takže se dostanete třeba na jednu sekundu, než skončí ten původní limit 32 sekund.

Co se vlastně má stát? Toť zajímavá otázka. Stále té věci chceme říkat stopky, takže by bylo fajn, kdyby po nějaké dlouhé době, ať už budeme mezitím klikat jakkoliv zběsile na aktualizační tlačítka, zobrazoval reálný čas od spuštění

[1] `http://ksp.mff.cuni.cz/viz/31-1-5/reseni`
[2] `https://twitter.com/brenankeller/status/1068615953989087232`

do poslední aktualizace. Takže za nějakou dobu by měl čas aktualizovat a nenechat se zmást změnou intervalu.

Rozhodně se ale nemá stát tohle:

1. po doběhnutí 32 sekund se zobrazený čas zvedne jen o 1 s (protože tahle hodnota je nastavená v době, kdy časovač doběhl);
2. po kliknutí na snižovací/zvyšovací tlačítko se restartuje časovač bez toho, aby program vzal v úvahu již uplynulý čas od posledního tiku.

Takové chování programu jsem za správné řešení úkolu 4 nepovažovala. Někteří spoléhali na to, že metoda `setInterval` objektu `QTimer` to celé vyřeší, to tak ovšem není. V dokumentaci asi nikde není explicitně napsané, co se vlastně stane, když změníte interval za běhu časovače; zdá se však, že se tím nastaví čas od teď do vypršení časovače, což ale nechceme – zahodili bychom si tím předchozí uběhlý čas. To ostatně potvrzuje i testovací program, přinejmenším na mém stroji.

Demonstrace fungování metody `setInterval` (Python 3): `http://ksp.mff.cuni.cz/viz/31-1-6-setinterval.py`

Maria Matějka

Výsledková listina první série třicátého prvního ročníku KSP

	řešitel	*škola*	*ročník*	*sérií*	*1-1*	*1-2*	*1-3*	*1-4*	*1-5*	*1-6*	*série*	*celkem*
0.					12	10	10	8	13	15	60,0	60,0
1.	Daniel Oravec	GVaršŽilina	4	1	12	10	8	8	12	10	57,7	57,7
2.	Jiří Kalvoda	GJarošeBO	2	1	12	10		8	13	13	57,5	57,5
3.	Josef Minařík	GJarošeBO	4	4	12	9	7	8	7	13	54,3	54,3
4.	Kristýna Petrlíková	VOŠJičín	1	1	12	9	9	4		14	52,8	52,8
5.	Lucia Krajčoviechová	GJHroncaBA	3	2	12	8	10		4	10	52,1	52,1
6.	Petr Budai	G JGJ PH	2	1	12	9,5	3	8	1	13	50,4	50,4
7.	Ondřej Jamelský	G Cheb	1	1	6	3		8	7	15	49,2	49,2
8.	Jan Provazník	GVoděraPH	3	1	6	5	9	4		15	48,6	48,6
9.	Jiří Šáda	GVoděraPH	3	1	6	5	8,5	4		15	48,4	48,4
10.	Dalibor Kramář	G BO-Řeč	4	2	12		11	6		12	44,5	44,5
11.	Vladimír Chudý	G Chrudim	2	6	12	2	1,5	4	7	14	44,1	44,1
12.	Lucie Vomelová	GŠpitálsPH	3	2	2	2	9	3	4	15	41,9	41,9
13.	Jiří Kvapil	GTomkovaOL	1	6	12	1	8,5	8	0	8	41,3	41,3
14.	David Klement	GNAlejíPH	3	4	12	2		8		15	38,7	38,7
15.	Jindřich Dítě	VOSPŠŽďár	3	4	12	2		6		15	37,8	37,8
16.	Michal Kodad	SPŠSmíchov	3	14	12		9	8		10	37,3	37,3
17.	Daniel Kurek	GTomkovaOL	3	1	9	1,5	1	4		10	37,1	37,1
18.	Vojtěch Žák	GŠpitálsPH	3	2	2		8,5	8		15	37,0	37,0
19.	Petr Zahradník	GaSOŠ ÚL	4	4	4		7	6		12	36,2	36,2
20.	Jakub Pánek	SPŠERožnov	4	1	6	1,5	1	4		11	35,9	35,9
21.	Daniel Skýpala	GTomkovaOL	1	9	4	2	7,5	8	0	10	34,2	34,2
22.	František Kmječ	StOlavVGS	3	9	12			8		11	31,8	31,8
23.	Petr Kolář	GMilevsko	3	1		1,5	5	7		8	31,6	31,6
24.	Marek Černoch	GFPValMez	3	1	4	1		8		10	31,5	31,5
25.	Daniil Barabashev	GNadKavaPH	3	1	0	1	7,5	3		10	30,7	30,7
26.	Jakub Profota	GŘíč	4	1	2	1,5	1,5	3		9	30,3	30,3
27.	Tomáš Černý	GArabskáPH	3	3	2	1	1	7		12	30,2	30,2
28.	Jan Piroutek	GŠpitálsPH	3	2	0		5,5	6		13	29,9	29,9
29.	Jakub Komárek	GUHradiště	4	6	0	7		8		12	29,7	29,7
30.	Janek Hlavatý	GJirsíkaČB	0	1		2	1,5	3		13	28,1	28,1
31.	Martin Zimen	GJMasarJI	4	2		2		6		14	26,3	26,3
32.	Matěj Kripner	GEBenešeKL	4	6	12					12	25,4	25,4
33.	Václav Pavlíček	SPSEPard	3	13	4	1	1,5	8	3	11	25,1	25,1
34.	Jáchym Mierva	BiGy Žďár	2	4	6					15	23,7	23,7
35.	Jakub Šťastný	G BO-Řeč	4	1				8		14	22,8	22,8
36.	Martin Hubata	GMikulášPL	3	1				6		14	22,2	22,2
37.	Ondřej Gonzor	G Brandýs	2	10	4	2	1,5	3		10	22,1	22,1
38.	Linda Kimrová	GEvolutionJM	3	1	6	2,5		4			21,2	21,2
39.	Matěj Volf	GCoubTábor	1	1	2				0	15	19,7	19,7
40.	Vojtěch Březina	GCoubTábor	2	2	2					12	18,5	18,5
41.	Martin Miller	GVoděraPH	4	2	4			7			15,0	15,0
42.–43.	Filip Hejsek	GPísnickáPH	2	2	12						12,0	12,0
	Jan Kaifer	GKepleraPH	3	11	12						12,0	12,0
44.	Patrik Vácal	SPŠEPlzeň	2	1	6						9,5	9,5