Korespondenční Seminář z Programování

ZAČÁTEČNICKÁ KATEGORIE

27. ročník | **KSP-Z** | **Březen 2015**

Už třetí série začátečnické kategorie KSP je u konce a jako obvykle vám přinášíme autorská řešení úloh. Věříme, že vám pomohou k tomu, abyste se v programování a hlavně v řešení problémů pořád zlepšovali. Zkuste se podívat, které úlohy byste řešili jinak a které zase stejně jako autoři nebo jak by se daly řešit úlohy, se kterými jste si nevěděli rady!

Gratulujeme všem, kdo získali nějaké body! A jako obvykle se nás nebojte zeptat, pokud vám cokoli není úplně jasné. Obrátit se na nás můžete přes fórum na našich stránkách nebo e-mailem *ksp@mff.cuni.cz*.

Řešení třetí série začátečnické kategorie 27. ročníku KSP

27-Z3-1 Kevin nabíječ, s.r.o.

Když jste nad úlohou chvíli přemýšleli, případně si zkoušeli různá zapojení, asi jste přišli na to, že na způsobu zapojení vůbec nezáleží, dokud budou všechny prodlužovačky připojeny k té jedné ve zdi. Každá prodlužovačka jednu zdířku použije pro své napájení a k jich poskytne pro libovolné další použití. Pokud použijeme všechny dostupné prodlužovačky, můžeme prostě spočítat součet všech $k-1$. To uděláme nejsnáze tak, že sečteme všechna k a na konci odečteme N (a přidáme jedničku za zásuvku ve zdi).

V zadání byl ale malý háček, který jsme se snažili naznačit obrázkem. Kevin si nakoupil i prodlužovačky, které měly nula zdířek, a takové zapojovat nechceme. Takže je stačí prostě z řešení vynechat. Tento chyták byl jen v posledním testu, takže i pokud jste si ho nevšimli, mohli jste dostat většinu bodů. Na takové chytáčky si ale dávejte pozor.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/27-Z3-1.py`

Ondra Hlavatý

27-Z3-2 Nedej vitagen

Maximální délka slova na vstupu byla v zadání 100. To je dostatečně malá konstanta na to, abychom se jí nemuseli zabývat. Soustředíme se hlavně na to, aby náš program byl co nejrychlejší vzhledem k počtu slov na vstupu.

Připravíme si spojový seznam všech zadaných slov a druhý spojový seznam, ve kterém budou také všechna slova, akorát napsaná pozpátku. Teď stačí jen najít slova, která jsou v obou seznamech.

To se dá dělat více způsoby. Nejsnazší je prostě každé slovo z prvního seznamu porovnat se všemi ze druhého a zapsat si ho stranou, pokud se s nějakým shoduje. To by mělo časovou složitost $\mathcal{O}(N^2)$, kde N je počet slov na vstupu.

O něco mazanější a výrazně rychlejší způsob je oba seznamy setřídit abecedně a potom je šikovně zkoumat najednou. Na třídění posloupnosti je spousta algoritmů, které skončí v čase $\mathcal{O}(N \log N)$.[1]

Jak přesně tedy budeme seznamy prohlížet? Budeme postupně procházet oba seznamy najednou a v každém kroku se podíváme na první slova v obou seznamech. Pokud jsou stejná, nalezli jsme požadovanou shodu. Jinak z nich vybereme to lexikograficky menší (neboli to, které by se ve slovníku objevilo dřív) a to smažeme. Tím se v tomto seznamu posuneme o jedno dál.

Pokaždé tak pracujeme pouze se začátkem nějakého spojového seznamu, což je velmi rychlé, protože nahlédnutí na první prvek i jeho smazání ve spojovém seznamu trvá konstantní čas (nezávisí na velikosti seznamu).

Pokud je nějaké slovo v obou seznamech, někdy během mazání se stane, že oba dva seznamy budou mít toto slovo na začátku. Proto stačí před každým mazáním zjistit, jestli se náhodou slova na začátcích seznamů neshodují. Mezi dvěma mazáními slov provedeme konstantní počet operací a mazání je $2N$, všechna dohromady budou tedy trvat $\mathcal{O}(N)$. Z toho plyne, že časová složitost celého algoritmu je $\mathcal{O}(N \log N + N) = \mathcal{O}(N \log N)$.

Najít mezi nimi to nejdelší už je triviální, můžeme to dělat třeba tak, že si v průběhu pamatujeme jenom dočasného adepta na vítěze a přepíšeme ho leda delším slovem.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/27-Z3-2.py`

Martin Španěl

27-Z3-3 Superstromy

Podle počtu správných řešení se zdá, že jste s úlohou neměli příliš problémů. Ono také nebylo potřeba vymýšlet nic světoborného. Stačilo si jen uvědomit, že si nikdy neuškodíme tím, že stromy rostoucí pomalu zasadíme jako první.

Proč to platí? Vždy alespoň jeden strom doroste jako poslední. Jak bychom mohli naše řešení zlepšit? Jedině tím, že tento poslední strom zasadíme v nějaký dřívější den. Pokud jsme však v dřívějších dnech sázeli pouze pomalejší stromy, prohozením si párty akorát odložíme na později.

Optimální pořadí stromů tedy dostaneme tak, že si je seřadíme sestupně dle počtu dní, které rostou. Následně jen stačí najít maximum ze součtu doby růstu a čísla dne, ve který daný strom zasadíme.

Dostali jsme tak řešení s časovou složitostí $\mathcal{O}(N \log N)$ a paměťovou $\mathcal{O}(N)$. To bohatě stačilo na plný počet bodů. Kdo si však všiml nízkého limitu na dobu růstu, mohl řešení o něco vylepšit.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/27-Z3-3.py`

Protože existuje pouze $T = 1\,000$ různých typů stromů, můžeme si v poli velikosti T uložit, kolik kterých z nich máme.

[1] `http://ksp.mff.cuni.cz/viz/kucharky/trideni`

Díky tomu zvládneme čísla třídit v čase $\mathcal{O}(N+T)$. Dokonce si i vystačíme jen s $\mathcal{O}(T)$ pamětí, protože si nemusíme pamatovat celý vstup.

Ačkoli popsaná myšlenka není vůbec složitá, vysloužila si svůj vlastní název. Takovému třídění čísel se říká *counting sort*. Vyplatí se jej použít v případech, kdy chceme seřadit velké množství stejných hodnot.

Program – counting sort (Python 3):
`http://ksp.mff.cuni.cz/viz/27-Z3-3-counting.py`

Jenda Hadrava

27-Z3-4 Robo Rally

V této úloze stačí vymyslet, jak v programu reprezentovat herní plochu a roboty a krok po kroku odsimulovat, co se děje.

Je mnoho možností, jak k problému přistoupit. Hlavní je nějakým způsobem reprezentovat robota, třeba jako objekt nebo klidně jako seznam čísel udávajících jeho pozici a orientaci. Například seznam $[7, 1, 2, \texttt{D}]$ by mohl reprezentovat robota číslo 7, který je v prvním řádku, druhém sloupci a dívá se směrem dolů.

Potom stačí vymyslet, kam roboty dát. Je rozumné mít je v poli indexovaném podle jejich identifikátorů, abychom se snadno dostali k robotovi, kterého zrovna potřebujeme.

Často navíc budeme zjišťovat, co je na nějakém políčku. Nejjednodušší způsob je prozkoumat souřadnice všech robotů a zjistit, jestli se nějaká neshoduje se zkoumaným políčkem. To by ale mohlo trvat dlouho a děláme to často. Pokud nám záleží na rychlosti, můžeme si udělat pomocné dvourozměrné pole, kde na každém políčku bude číslo robota, který na něm je (nebo rovnou odkaz na něj), nebo -1 (resp. nulový odkaz), pokud tam žádný není.

Teď už nebude problém načíst pozice a orientace robotů a umístit je do naší datové struktury. Otáčení robotů je taky triviální. Pokud směry reprezentujeme písmeny, stačí k tomu několik podmínek. Pokud reprezentujeme směry čísly $0, 1, 2, 3$, můžeme si pomoci operací modulo, která nám umožní zjistit zbytek po dělení čtyřmi. Tedy když se chceme otočit o tři doprava, tak k současnému směru přičteme trojku a výsledek vymodulíme čtyřmi.

Zajímavější je pohyb. Musíme si dát pozor, že je potřeba simulovat krok po kroku. Tedy posun o pět políček budeme muset rozložit na pět kroků a každý vyhodnotit zvlášť. Pokud nám v nějakém kroku nestojí nic v cestě, tak je to snadné. Stačí přepsat posunutému robotovi souřadnice a případně upravit pomocnou tabulku.

Pokud nám stojí něco v cestě, řekneme tomu, ať udělá krok stejným směrem a podle stejných pravidel jako my. Tedy pokud mu bude stát něco v cestě, tak tomu taky řekne, ať se to posune. Takto se o tom dozví celá řada. Je důležité ošetřit případ, kdy se řada už nemá kam posunout (je před ní zeď). Kvůli tomu je potřeba ještě před samotným posunem nejdříve ověřit, zda se vůbec můžeme pohybovat, a teprve potom posun provést.

Po provedení všech kroků stačí vypsat postupně souřadnice a orientace robotů.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/27-Z3-4.py`

Martin Španěl

27-Z3-5 Dřevěná slacklajna

Jak už všichni víme z hodin matematiky, trojúhelník, jehož nějaká strana je delší než součet všech ostatních, nejde nakreslit. A stejně tak to platí i pro mnohoúhelník. Kdyby Kevin se Sárou položili všechna prkna kromě nejdelšího za sebe do jedné linie a tato linie by byla kratší než ono nejdelší prkno, nemohou si okruh vůbec vyrobit.

Základem řešení bude tedy tato myšlenka. Délky prken si nejprve setřídíme, ideálně od největšího po nejmenší. Součet všech délek označíme S, délku nejdelšího prkna D.

Pokud je:

- $S - D > D$, okruh lze postavit ze všech prken, jejichž délka je D nebo menší.
- $S - D \leq D$, tak nejdelší prkno nelze použít. Od S odečteme D a celé porovnání provedeme znova, akorát D již bude délka dalšího nejdelšího prkna v pořadí.

Na konci stačí zkontrolovat, že nám zbyla alespoň tři prkna, ze kterých může Kevin se Sárou okruh postavit.

Časová složitost bude $\mathcal{O}(N \log N)$, kde N je počet prken. Projití setřízeného seznamu délek prken bude lineární, jelikož pro každé prkno provedeme pouze konstantně mnoho operací. Ovšem setřízení prken bude trvat $\mathcal{O}(N \log N)$.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/27-Z3-5.py`

Katka Zákravská

27-Z3-6 Red Bull dává křídla

Nedostali jsme příliš mnoho řešení, což nás mrzí. Pojďme si ukázat, že tato úloha nebyla tak těžká, jak se vám možná zdála.

Jelikož procházíme stavovým prostorem (což neznamená nic jiného, než že máme hrací plochu a na každém políčku můžeme být buď jako král, nebo jako kůň), tak si musíme rozmyslet, jak jej budeme reprezentovat.

Pro tuto konkrétní úlohu se nám zdá vhodné mít dvě dvourozměrná pole, která budou odpovídat velikosti hrací plochy, tedy velikosti $M \times N$.

Jedno pole bude pro místa, kam jsme došli jako král, a druhé bude pro místa, kam jsme došli jako kůň.

Vždy z každého dosaženého políčka zkusíme jít do všech políček ve druhém poli, na která se daná figurka může dostat a ještě jsme tam nebyli. (Ve druhém poli, protože se tahy koněm a králem střídají.)

To, jestli jsme v některém stavu už byli, si budeme značit číslem udávajícím, v kolikátém kroku jsme tam došli. Což nám pomůže při debugování, neboť víme, kam nám to kdy skočilo. Ale taky podle toho dokážeme zrekonstruovat cestu.

Pro nalezení cesty pak stačí jít z cíle do startu po stavech, která mají právě o jedna menší číslo, než ve kterém jsme. Před výpisem tuto posloupnost musíme ještě otočit. Rozmyslete si, že v tomto případě nelze udělat trik, že bychom hledali cestu z cíle do startu, a tím se vyhnuli otáčení posloupnosti.

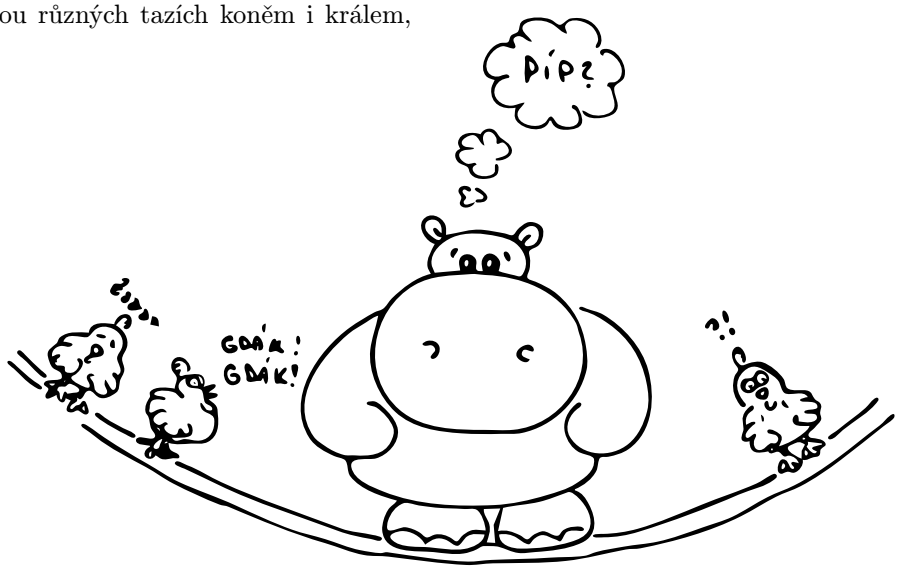
Posledním krokem je si rozmyslet, jak rychle zjistit, kam jsme již skočili. Nejlepší bude si to pamatovat v nějaké struktuře, do které budeme umět v konstantním čase vkládat prvky na konec a vyjmout první prvek. Takové struktuře se říká fronta.

Nyní již jen zbývá zanalyzovat, jak dobrý algoritmus jsme vymysleli. Projití všech stavů nám bude trvat $\mathcal{O}(N \cdot M)$, protože právě tolik je stavů a v každém stavu děláme konstantní množství operací, a stejně tolik paměti budeme potřebovat.

Závěrem ještě dodáme, že použít jen jedno společné pole pro kroky koněm i králem nestačí, protože na tom, jakou figurkou jsme se do které pozice dostali, závisí i následné možné pohyby z tohoto políčka dál. A pokud políčka zvládneme dosáhnout ve dvou různých tazích koněm i králem, tak je nutné zkoumat obě varianty. Zkuste si to třeba na příkladu Kevina na pozici 5×5 a cíle na pozici 2×2. Na takovéto políčko se dá dostat za 3 tahy, ale pokud bychom neuvažovali obě možnosti (pokud bychom použili jen jedno společné pole dosažitelnosti), tak nám algoritmus bude říkat, že se tam dostaneme za 4 tahy.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/27-Z3-6.py`

Vojta Sejkora



Výsledková listina třetí série začátečnické kategorie 27. ročníku KSP

	řešitel	*škola*	*ročník*	*sérií*	*Z3-1*	*Z3-2*	*Z3-3*	*Z3-4*	*Z3-5*	*Z3-6*	*série*	*celkem*
0.					8	10	10	12	12	14	66,0	198,0
1.	Jakub Pelc	G UherBrod	1	7	8	10	10	12	12	10	62,0	187,8
2.	Jan Kaifer	GČesBrod	−1	3	8	10	10	12	11	12	63,0	184,0
3.	Martin Scheubrein	G MNám Třb	3	3	8	10	10	12	12	5	57,0	183,0
4.	Jakub Matěna	GČeskoliPH	3	3	8	10	10	12	12	4,5	56,5	169,0
5.	Jiří Štěpanovský	G MNám Třb	3	3	8	10	10	12	9	8	57,0	166,0
6.	Lukáš Červený	G_Trutnov	1	3	8	10	10	12	5	8	53,0	162,5
7.	Vojtěch Lukeš	GPikaPL	3	3	8	10	10	12	9		49,0	151,0
8.	Lukáš Mičan	GČeskáČB	1	3	6	10	10	12	12		51,3	148,3
9.	Lukáš Vlček	GMikulášPL	1	3	8	10	10	12	10	1	51,0	145,0
10.	Jakub Jirkal	GJungmanLT	0	3	8	10	10	12	9,5	4	53,5	140,5
11.	Miroslav Hrabal	GTomkovaOL	1	3	8	10	10	0			28,0	140,0
12.	Michal Töpfer	G DrJPekMB	2	6	8	10	10		12		40,0	129,0
13.–14.	Karolína Kuchyňová	GMLerchaBO	4	3	8	10	10	12			40,0	120,0
	Jiří Moravčík	GUHradiště	1	3	8	10	10	12			40,0	120,0
15.	Roman Beňo	GJHroncaBA	2	2	8	10	10	12	12	6	58,0	113,0
16.	Tereza Kotěšovcová	GKlatovy	4	3	8	10	10	12			40,0	110,8
17.	Lukáš Fruněk	GLesníZlín	2	3	8	10	10	12	11		51,0	100,0
18.–19.	Tomáš Terem	GTajBanBys	3	2	8	10	10		11		39,0	99,0
	Tomáš Trojan	G Cheb	−1	3	8	0	10	12			30,0	99,0
20.–21.	Zdeněk Pavlátka	GMikulášPL	3	5	8	10		12			30,0	98,0
	Pavel Souček	G_Nymburk	3	3	8	10	10	12			40,0	98,0
22.	Matěj Fencl	GOA Chodov	1	2	8	4	10	10	9	1,5	42,5	93,5
23.	Jakub Rozlivek	GPikaPL	3	2	8	10	10	1			29,0	84,0
24.	David Žáček	GZborovPH	2	3	8						8,0	82,0
25.	Michal Převrátil	GKlatovy	2	4	8	10	10	12			40,0	80,0
26.	Tomáš Chvosta	GPří	4	3	6						7,3	76,3
27.	Matúš Maďar	GHorMichal	3	3	6		10		11,5		28,8	75,8
28.	Daniel Nigrin	GÚstavníPH	2	3	8		6				14,0	74,0
29.–30.	Nhat Minh Dinh Huy	GKadaň	2	3	6	10	10				26,0	72,0
	Jan Mráz	G_Holice	1	2	8	10	10	12	9	12	61,0	72,0

	řešitel	*škola*	*ročník*	*sérií*	*Z2-1*	*Z2-2*	*Z2-3*	*Z2-4*	*Z2-5*	*Z2-6*	*série*	*celkem*
31.–32.	Jan Burda	G_Holice	0	6	8	10	3	12			33,0	69,5
	David Ucháč	eduSOŠ PA	2	2	8	10	10	12	10,5	1	51,5	69,5
33.	Jakub Neruda	GTNovákBO	4	3							0,0	69,0
34.	Pavel Turinský	G Brandýs	2	2	8	10	10				28,0	63,8
35.–36.	David Bělíček	GSOŠ Podb	3	2	8	10					18,0	57,0
	Zuzana Šimečková	GČeskáČB	4	3	8		10		11		29,0	57,0
37.	Daniel Pluskal	G BO-Řeč	1	2							0,0	56,0
38.	Zuzana Drázdová	GČeskáČB	4	2	6	1	2		11		20,0	52,0
39.	Ondřej Švanda	G BO-Řeč	4	2							0,0	50,0
40.	Václav Fabík	ZŠKřídloBO	0	6	8						8,0	48,0
41.	Josef Vávra	SJec	4	2	6	10	10				27,4	47,4
42.–43.	Petr Klanica	GJarošeBO	2	1							0,0	45,0
	Zoltán Onódy	SPŠE NZám	4	1							0,0	45,0
44.	Ondřej Měkota	SPŠMasarLI	3	2	6	1					7,0	42,8
45.–50.	Patrik Bak	G Sobrance	4	1							0,0	40,0
	Tat Dat Duong	G Wicht	2	1							0,0	40,0
	Jakub Lukeš	GNAlejíPH	2	5							0,0	40,0
	Jiří Sejkora	GVoděraPH	3	1							0,0	40,0
	Jan Soukup	GKlatovy	4	1							0,0	40,0
	Jan Václavek	GUnOrl	3	2							0,0	40,0
51.–52.	Janek Hlavatý	ZŠ_DukelČB	−4	6	6						6,0	39,0
	Michaela Štolová	G_Sokolov	3	2	8	10	10				28,0	39,0
53.	David Tvrdý	GHeyrovPH	2	1							0,0	38,0
54.	Jan Vozár	G UherBrod	1	6	8	0	10				18,0	37,5
55.	Jonáš Přibyl	ZŠ ČBrod	0	1	4	10	10	12			36,0	36,0
56.	Michaela Svatošová	GKepleraPH	1	3	8		5				13,0	35,0
57.–59.	Lukáš Holeczy	GTep	3	2							0,0	28,0
	Ivana Krumlová	GJarošeBO	2	3							0,0	28,0
	Benedikt Žour	G UherBrod	0	5	0						0,0	28,0
60.–61.	Michael Kozel	GZborovPH	1	3	6						6,0	24,0
	Roman Ondráček	GBoskovice	1	3	6	0					6,0	24,0
62.	Dominika Tanglová	G_Nymburk	2	1							0,0	22,0
63.–64.	Jan Sliacky	G_Benesov	4	1							0,0	21,0
	Andrej Čermák	G JF Šaľa	1	2	6		6				12,0	21,0
65.	Victoria María Nájares Romero	GZborovPH	1	1							0,0	20,0
66.–70.	Mirolsav Březík	GLesníZlín	0	1							0,0	18,0
	Michal Nekvinda	BiGyBBHK	4	1							0,0	18,0
	Alexej Popovič	SlovanGOL	3	1							0,0	18,0
	Vojta Staněk	PORGPha	1	1							0,0	18,0
	Petr Šíma	GKlatovy	1	5							0,0	18,0
71.	Martin Zima	G_Holice	1	1	8				8		16,0	16,0
72.	Vojtěch Pejša	G Kolín	4	1							0,0	15,0
73.	Milan Kubala	GTajBanBys	3	1							0,0	14,0
74.	Filip Priečinský	SG Žilina	2	3	2	2	0		3		7,0	13,5
75.–81.	Michaela Bačová	G UherBrod	4	3							0,0	10,0
	Antonín Bruštík	G UherBrod	4	5							0,0	10,0
	David Dvořáček	G UherBrod	4	3							0,0	10,0
	Ľuboš Kolumber	SpojŠ Popr	3	3	4	3	0				7,0	10,0
	Viktor Kovařík	G UherBrod	4	4							0,0	10,0
	Markéta Machalová	G Wicht	2	3	2						2,0	10,0
	Jakub Šmahovský	G Pezinok	3	1							0,0	10,0
82.	Matěj Hudec	CírkG Plzeň	3	1	6		2				9,4	9,4
83.–86.	Matej Hockicko	TAPoprad	1	1							0,0	8,0
	Ján Pavlus	GTNovákBO	2	1							0,0	8,0
	Martin Sklenár	GTajBanBys	3	1							0,0	8,0
	Petr Zelina	GJarošeBO	2	1							0,0	8,0
87.	Vít Gadurek	Neuvedená	0	2	4						4,0	5,0