Korespondenční Seminář z Programování

ZAČÁTEČNICKÁ KATEGORIE

33. ročník | **KSP-Z** | **Únor 2021**

Řešení třetí série začátečnické kategorie 33. ročníku KSP

33-Z3-1 Šikmá věž z krabic

Nejprve je důležité si rozmyslet, jakým způsobem chceme krabice na sebe lepit, abychom maximalizovali celkovou šířku šikmé věže. Protože nám na výšce věže nezáleží, chceme každou krabici otočit „na šířku“ (delším rozměrem vodorovně). Zároveň musíme krabici umístit tak, aby její těžiště leželo na předchozí krabici. Těžiště se nachází právě v polovině šířky krabice, a tudíž když chceme krabici co nejvíce využít, položíme ji těžištěm na hranu předchozí krabice. Z toho plyne, že kromě první krabice každá krabice přispívá k celkové šířce věže polovinou své delší strany. Zůstává otázka, jak vybrat první krabici. Protože první krabice přispívá k věži celou svou délkou, chceme jako první umístit tu nejširší krabici. Na pořadí ostatních krabic nezáleží.

Jak tedy bude Kevin postupovat? Nejprve vybere krabici s nejdelší stranou a přilepí ji na zem. Další krabice bude brát v pořadí tak, jak mu přijdou pod ruku. Vždy je otočí na šířku a nalepí je těžištěm na pravý kraj předchozí krabice.

Protože Kevin s každou krabicí interaguje nejvýše dvakrát, časová složitost algoritmu bude lineární vzhledem k počtu krabic.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/33-Z3-1.py`

Úlohu připravili: Maruška Kalousková, Vašek Končický, Tom Sláma, Klárka Tauchmanová

33-Z3-2 Přečasování titulků

Jak zmiňuje zadání, titulky se od filmu liší rychlostí a jsou posunuté. Kdybychom obě informace znali, můžeme titulky snadno opravit – každou časovou značku nejprve vynásobíme koeficientem zrychlení a poté přičteme posun.

Jak potřebné údaje spočítat? Koeficient zrychlení nám říká, kolik vteřin titulků připadá na jednu vteřinu filmu, neboli $\frac{délka(titulky)}{délka(film)}$. Je-li > 1, pak musíme titulky zrychlit, naopak je-li mezi 0 a 1, musíme je zpomalit. Problém je, že my ale délku filmu ani titulků neznáme. K čemu nám ten vzorec tedy je? Stačí si uvědomit, že úvaha platí pro jakýkoli výsek, je-li v originále stejně dlouhý. Speciálně tedy můžeme výpočet udělat s časovým rozdílem mezi Kevinem identifikovanými titulky.

Jakmile známe koeficient, přenásobíme s ním časy titulků a zbývá určit posun. To už je ale snadné. Časy obou Kevinem nalezených titulků jsme již opravili z hlediska rychlosti, takže rozdíl mezi Kevinovým časem a časem po přenásobení koeficientem je hledaný posun.

Pozorování výše už snadno přetavíme v algoritmus. Načteme oba Kevinem identifikované titulky a pak všechny titulky. Přitom se v nich snažíme odhalit identifikované repliky (ty jsou naštěstí unikátní). S optimalizací porovnávání řetězců si těžkou hlavu dělat nemusíme – provedeme pouze konstantně (2) porovnání s každým titulkovým textem. Jakmile najdeme odpovídající si páry, provedeme již známý výpočet a znovu projdeme přes titulky a každý opravíme.

Časová složitost je lineární vzhledem k velikosti vstupu a paměťová taktéž.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/33-Z3-2.py`

Úlohu připravili: Vojta Káně, David Klement, Jirka Setnička

33-Z3-3 Chamtivý stánkař

V řešení této úlohy budeme vycházet z řešení úlohy v minulé sérii. Tam jsme řešili podobný problém s jediným rozdílem, že nyní jsme přidali druhý druh nápoje (či vlastně zákazníka) a priority ve zpracování.

Jedním z ukazovaných řešení u minulé úlohy byla iterace podle času. Stejnou myšlenku můžeme zopakovat i u této úlohy. Akorát si musíme pořídit dvě fronty. Další postup můžeme odvozovat za pomoci představy, že běží čas a stánkař se v každý okamžik nějak rozhoduje. Události můžeme popsat například takto:

1. Noví lidé si stoupnou do správných front.
2. Dlouho čekající lidé fronty opustí.
3. Pokud je ve frontě na čaj nějaký zákazník, bude obsloužen.
4. Pokud je ve frontě na limo nějaký zákazník a zároveň nebyl obsloužen nikdo s čajem, bude podána limonáda.
5. Posuneme čas o minutu dopředu.

Můžeme si rozmyslet, že tento popis událostí zhruba odpovídá tomu, co by se ve zjednodušeném světě u stánku dělo. Implementace řešení pak může tento postup doslovně sledovat. Podobně jako v minulé úloze bude řešení dostačující, ale relativně pomalé. Jak bychom tedy mohli implementaci zjednodušit a zrychlit?

Reálně to zas tak moc nejde. Žádný z kroků simulace nemůžeme odstranit, protože pak bychom počítali něco jiného. Můžeme je akorát přesouvat jinam, či je zkusit jednotlivě zrychlit.

Jedním takovým vylepšením je pozorování, že první krok vlastně nepotřebujeme. Je to kus kódu, který přehazuje načtený vstup prakticky ve stejném formátu do dvou front. Kdybychom vstup do front načetli přímo, můžeme se ve zbytku programu tvářit, že v nich „zákazníky z budoucnosti“ nevidíme. Touto relativně jednoduchou změnou si sice zásadně nepomůžeme v rychlosti programu, ale zbavíme se zbytečné duplicity.

Druhé pozorování se týká hodin. Kdybychom přičítali pouze jedničku, tak se nám může stát, že spoustu času trávíme děláním ničeho. Kdybychom ale nějak věděli, na jaký čas v budoucnu skočit, tak bychom se prodlevám vyhnuli. Díky předchozí změně to ale přeci víme! V obou frontách jsou „zákazníci z budoucnosti“. Stačí se tedy podívat, kdy přijde

první z nich a na tento čas přímo skočit. Tím vlastně způsobíme, že vnější cyklus simulace poběží nejvýše tolikrát, kolik je zákazníků na vstupu. Předtím to bylo za každou minutu jednou.

Co jsme poslední změnou získali? Můžeme se ohlédnout zpět na druhé řešení minulé úlohy, kterému jsme říkali iterace přes zákazníky. Vlastně jsme teď vyrobili řešení, které se chová velmi podobně. Sice iteruje přes čas, ale skáče pouze do okamžiků, kdy je potřeba nějakého zákazníka zpracovat. Formálně jsme složitost převedli z lineární v čase na lineární v počtu zákazníků. To znamená, že už počet kroků algoritmu neovlivňuje čas, kdy zákazníci přicházejí.

Kdybychom chtěli podobné řešení napsat od začátku jako iteraci přes zákazníky, mohli bychom samozřejmě také. Myšlenka je to ale o něco málo komplikovanější a ničím nám proti řešení výše nepomůže. Nebudeme ji tedy zde rozebírat. Můžete si ji tak aspoň rozmyslet jako cvičení...

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/33-Z3-3.py`

Úlohu připravili: Jirka Setnička, Vašek Šraier, Eliška Vítková

33-Z3-4 Statek

Přímočaré řešení je takové, že zkusíme postavit statek postupně na každém z políček a vždy zjistíme, jak velká jsou pole kolem něj. Průběžně si pamatujeme, pro které umístění statku byla tato plocha polí největší. Jakmile vyzkoušíme všechna políčka, budeme znát optimální umístění.

Jak ale spočítáme velikost polí kolem statku? Z políčka se statkem spustíme prohledávání do šířky.[1] Najdeme všechna políčka polí, se kterými statek přímo sousedí. Poté najdeme jejich sousední políčka, a tak dále, dokud nenajdeme všechna políčka, na která lze ze statku kombajnem dojet. Políčka průběžně počítáme, musíme si však dát pozor, abychom některé nezapočítali vícekrát, ta již započítaná si proto budeme označovat.

Bude-li počet políček N, z každého políčka můžeme prohledat až celou mapu. Časová složitost proto bude $\mathcal{O}(N^2)$.

To samé prohledávání děláme několikrát, pojďme to vylepšit. Jednotlivé plochy polí si od sebe odlišíme a pro každé políčko si budeme pamatovat, jakého pole je součástí a jak je toto pole velké. Uděláme to tak, že postupně projdeme všechna políčka polí, a pokud u daného políčka ještě nevíme, do jakého pole patří, spustíme z něj nové prohledávání a tím si všechna políčka v tomto novém poli označíme.

Dále můžeme postupovat stejně jako v přímočarém řešení, ovšem nyní dokážeme velikost polí kolem statku spočítat okamžitě. Podíváme se na čtyři sousední políčka a zjistíme, do jakých polí patří. Velikosti těchto polí sečteme, ale nesmíme totéž pole započítat vícekrát. Proto jsme si u každého políčka kromě velikosti pamatovali i to, do jakého pole patří.

Toto řešení již běží v čase $\mathcal{O}(N)$. Musíme ukládat informace pro daná políčka, paměťová složitost tedy bude také lineární.

Úlohu připravili: Maruška Kalousková, Vojta Káně, David Klement

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

Výsledková listina třetí série začátečnické kategorie 33. ročníku KSP

	řešitel	*škola*	*ročník*	*sérií*	*Z3-1*	*Z3-2*	*Z3-3*	*Z3-4*	*série*	*celkem*
0.					8	10	12	14	44,0	132,0
1.	Robert Jaworski	GÚstavníPH	3	20	8	10	12	14	44,0	132,0
2.–3.	Dominik Farhan	GMikulášPL	4	6	8	10	12	14	44,0	126,5
	Jakub Ondroušek	GTomkovaOL	1	11	3	10	12	13,5	38,5	126,5
4.	Kryštof Maxera	GJírovcČB	0	5	8	10	12	14	44,0	113,0
5.–6.	Jonáš Dej	G Wicht	2	3	8	10	12	7	37,0	107,0
	Jan Hlavsa	GMělník	3	3	8	10	12	8	38,0	107,0
7.–8.	Robert Gemrot	GKomHavíř	4	12	8	10	12		30,0	104,0
	Tomáš Janovec	GMnichHrad	3	4	8	10	12	13	43,0	104,0
9.	Thomas Riedle	BRG APP	2	7	8	10	12	7	37,0	98,0
10.	Vojtěch Venzara	GMělník	3	3	8	10	12		30,0	94,0
11.	Pavel Šrytr	GMělník	4	3	8	10	12		30,0	92,0
12.	Veronika Jůzková	MensaG	3	7	8	10	12		30,0	89,0
13.	Klára Grinerová	GZborovPH	4	3	8	10	12		30,0	78,0
14.	Lukáš Létal	GJŠkodyPŘ	2	3	8	10	12		30,0	75,0
15.–16.	Martin Fof	MendelGOP	3	3	8	10	12		30,0	74,0
	Adam Kolník	SSŠVTPraha	2	2	8	10	12	14	44,0	74,0
17.	Erik Sabol	GČeskoliPH	1	6	8	0	12		20,0	73,0
18.	Yahor Herashchanka	ZŠ Turnov	0	3	8	10		8	26,0	68,7
19.	Jakub Smolik	GEBenešeKL	3	2	8	10	12		30,0	65,0
20.	Vít Novotný	GMělník	3	2	8	10	9		27,0	57,0
21.	Daniel Mencl	GMělník	3	3	8				8,0	53,0
22.	Ondřej Piroutek	GČeskoliPH	3	3	8	10		5	23,0	52,0
23.	Olga Cinková	ArcibisGPH	1	3	5	10	12	13,5	40,5	51,5
24.	Pavel Altmann	GMikulášPL	2	5	8	10	12		30,0	50,0
25.–26.	David Holas	SPŠEMasLI	1	1	8	10	12	14	44,0	44,0
	Michal Mlynár	GJSeiferPH	0	1	8	10	12	14	44,0	44,0
27.	Štěpán Filipčík	GTomkovaOL	3	2	8				8,0	43,0
28.	Oto Skýpala	GJŠkodyPŘ	−3	1	8	10	12	12	42,0	42,0
29.	Jakub Nevařil	G UherBrod	3	12	8	10	12		30,0	40,0
30.	Jan Soukup	GJiříPoděb	3	2	8	10	12		30,0	38,0
31.–32.	Michal Bravanský	GBílovec	3	11	8	10	12	7	37,0	37,0
	Kryštof Latka	PORG Krč	3	1	8	10	12	7	37,0	37,0
33.	Matouš Bohoněk	GMělník	3	2	8				8,0	35,0
34.	Zdeněk Pezlar	GJarošeBO	3	2					0,0	34,0
35.	Jakub Mikeš	GJŠkodyPŘ	3	1	8	10	9	6	33,0	33,0
36.–37.	Vojtěch Gaďurek	PORGPha	4	1					0,0	32,0
	Matěj Strnad	ZŠRiegraSM	0	5	8		0		8,0	32,0
38.–42.	Jonáš Bína	ZŠŠtáflovaHB	−4	3					0,0	30,0
	Viktor Čubík	G UherBrod	3	1	8	10	12		30,0	30,0
	Tomáš Kašpárek	G FrýdlNOs	3	1					0,0	30,0
	Vojtěch Skyba	G UherBrod	3	1	8	10	12		30,0	30,0
	Antonin Spaniel	GÚstavníPH	2	1	8	10	12		30,0	30,0
43.–45.	Adam Húšťava	EupSchoolLux	3	12					0,0	27,0
	Kryštof Marek	SGPCE	1	1	8	10	9		27,0	27,0
	Michal Žáček	MensaG	4	1					0,0	27,0
46.	Jonáš Koziorek	GSOŠ FrMís	4	1	8	10	6		24,0	24,0
47.	Jan Kotyk	G Kolín	4	1					0,0	20,0
48.–49.	Petr Hladík	GMikulášPL	3	5	8	10			18,0	18,0
	Tomáš Chabada	SPŠEMasLI	4	5					0,0	18,0
50.	Jáchym Hájek	ZŠTřebechovice	−2	1	5	10			15,0	15,0
51.	Honza Kocourek	ParkLane	1	1					0,0	14,0
52.	Radek Bláha	GČeskáČB	−1	3	1				1,0	11,0
53.	Jan Najman	SPSEPard	4	8					0,0	10,0
54.–56.	Jakub Drobný	ŠPMNDaGB	3	1	8				8,0	8,0
	Jan Machourek	GBBr	0	3					0,0	8,0
	Matyas Oliva	G UherBrod	3	1	8				8,0	8,0

	řešitel	*škola*	*ročník*	*sérií*	*Z3-1*	*Z3-2*	*Z3-3*	*Z3-4*	*série*	*celkem*
57.	Matěj Hošek	GVolgogrOS	−1	1					0,0	7,0
58.	Martin Bulíř	SPŠEMasLI	4	1					0,0	6,0
59.	Marek Maškarinec	GFXŠaldyLI	0	2					0,0	3,0
60.	Antonín Musil	PORGPha	4	3					0,0	2,0
61.	Vojtěch Čermák	SPŠEMasLI	2	3					0,0	1,0