Korespondenční Seminář z Programování

ZAČÁTEČNICKÁ KATEGORIE

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

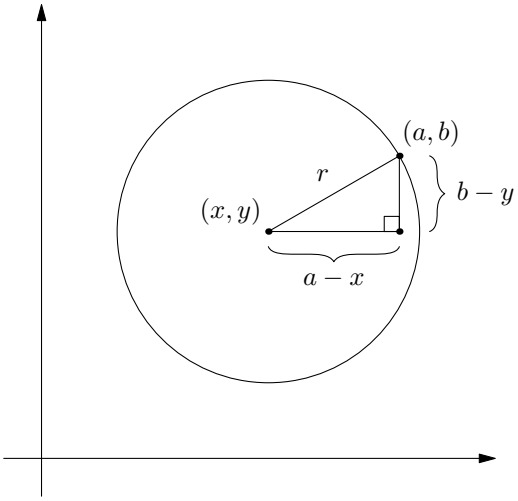
Řešení čtvrté série začátečnické kategorie 33. ročníku KSP

33-Z4-1 Střílení do terče

Naším cílem je zjistit, které kulky trefily terč. Z matematického pohledu můžeme na terč nahlížet jako na kruh se středem (x, y) a poloměrem r. Pro bod (a, b) na jeho obvodu platí následující vzoreček:

$$(a - x)^2 + (b - y)^2 = r^2.$$

To můžeme nahlédnout z obrázku a znalosti Pythagorovy věty:



Aby se kulka trefila do terče, musí být její vzdálenost od středu terče menší než poloměr terče. Proto se vlastně ptáme, zda je pro souřadnice dopadu kulky $(shot_x, shot_y)$ splněna následující nerovnice:

$$(shot_x - x)^2 + (shot_y - y)^2 < r^2.$$

Při vyhodnocování tedy budeme postupně načítat souřadnice dopadu jednotlivých kulek ze vstupu a pro každou z nich vyhodnotíme levou část nerovnice. Pokud bude výsledek menší než r^2, vypíšeme pořadové číslo kulky na výstup, jinak neděláme nic.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/33-Z4-1.py`

Úlohu připravili: Maruška Kalousková, Michal Kodad, Jirka Sejkora, Klárka Tauchmanová

33-Z4-2 Hesla nebo odpad?

Pro každé heslo potřebujeme umět rozhodnout, zda odpovídá zadanému předpisu. Předpis vůči heslu budeme muset kontrolovat po znacích, tomu se nevyhneme. Ale jak si postup co nejvíce zjednodušit?

Protože se hvězdička vyskytuje v předpisu právě jednou, můžeme předpis v místě hvězdičky rozdělit, a pak na něj nahlížet jako na předpis pro začátek (prefix) a na předpis pro konec (sufix) hesla. Heslo tedy budeme kontrolovat dvakrát – zda začátek hesla odpovídá prefixu předpisu a zda konec hesla odpovídá sufixu předpisu.

Zde si můžeme všimnout, že kontrola konce hesla vůči sufixu je ekvivalentní kontrole otočeného hesla vůči otočenému sufixu, což nám umožní zjednodušit implementaci. Dále je důležité si uvědomit, že když při kontrole dojdeme na konec předpisu, zbývající znaky hesla můžeme vesele ignorovat. Tyto znaky se totiž v předpisu schovají pod hvězdičku.

Výše popsaný postup kontroly má jeden problém – kontrola prefixu i sufixu slova může uspět, ačkoli by heslo projít nemělo, neboť je kratší než zadaný předpis (to se stane, když se prefix a sufix předpisu překrývají). Proto ještě musíme provést kontrolu délky hesla.

Časová složitost je lineární vzhledem k součtu délek hesel na vstupu, paměť nám stačí vždy jen délku jednoho hesla a prefixu.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/33-Z4-2.py`

Úlohu připravili: Maruška Kalousková, Jirka Sejkora, Jirka Setnička, Klárka Tauchmanová

33-Z4-3 Kluzké bludiště

Tuto úlohu můžeme vcelku přímočaře vyřešit prohledáváním do šířky. Budeme si značit již navštívená políčka (na začátku je to jenom start) a na každém políčku si budeme udržovat odkaz na políčko, ze kterého jsme na něj přišli.

Na začátku si vložíme startovní políčko do fronty a pak pro každé políčko z fronty opakujeme následující postup: Najdeme všechna nenavštívená políčka, na která se z aktuálního políčka umíme doklouzat (tedy nejvýše čtyři políčka ve všech čtyřech směrech), ta lehce najdeme odkrokováním v daném směru, než narazíme do zdi. Nalezená políčka označíme za navštívená, uložíme si k nim odkaz na aktuální políčko (ten se bude hodit pro vypsání cesty) a vložíme políčka do fronty.

Pokud během procházení narazíme na cílové políčko, tak můžeme prohledávání ukončit a pak již jen vypíšeme cestu rekonstruovanou přes zpětné odkazy. Začneme z cíle a podíváme se na zpětný odkaz. Podle toho, kam jsme museli z minulého políčka jít, tak si do cesty přidáme `N`, `D`, `P`, nebo `L` a pokračujeme s dalším políčkem, dokud nedojdeme na start. Ale pozor na to, že teď jsme cestu zrekonstruovali pozpátku, musíme ještě cestu před vypsáním otočit. A to už je správné řešení.

Jak dlouho takový postup trvá? Pro bludiště velikosti $R \times S$ může zpracování jednoho políčka trvat až $\mathcal{O}(R + S)$ (musíme proklouzat celý řádek a sloupec). Pro každé políčko to uděláme jednou, políček je RS, takže celková časová složitost je $\mathcal{O}(RS(R + S))$.

Rychlejší řešení s grafy

Toto řešení je zbytečně pomalé, zlepšit se dá postavením vhodného grafu nad bludištěm. Ono už samotné bludiště je vlastně grafem s vrcholy odpovídajícími políčkům a s hra-

nami vedoucími vždy na sousední políčka. My si ale postavíme lepší graf, aby nám klouzání netrvalo tak dlouho.

Vrcholy našeho grafu budou také políčka bludiště, ale hrany budou pro každý vrchol vést na místa, do kterých se z něho umíme doklouzat. Navíc si zavedeme speciální případ pro to, pokud při klouzání potkáme cíl – v tom případě bude hrana směřovat do něho.

Jedná se o orientovaný graf, protože z políčka, do kterého jsme se právě doklouzali, nemusí existovat hrana nazpět (typicky protože se cestou nazpět dokloužeme dál). Počet vrcholů a hran v něm bude odpovídat počtu vrcholů a hran v původním bludišti (tedy RS vrcholů a až $4RS$ hran), ale hrany si zkonstruujeme šikovněji.

Jak si pořídíme ty správné hrany? Můžeme si vyrobit rekurzivní funkci, která dostane políčko a směr a vrátí, kam se z daného políčka v tomto směru dokloužeme. Pokud je tento směr pro dané políčko již spočítaný, tak funkce vrátí uloženou hodnotu a hotovo. Pokud směr ještě spočítaný není, tak se funkce podívá na vedlejší políčko v daném směru. Pokud je vedlejší políčko...

- zeď, tak výsledek je aktuální políčko,
- cíl, tak výsledek je cíl samotný,
- jinak funkce zavolá sama sebe na vedlejší políčko (tomuto říkáme rekurze).

Získaný výsledek si funkce uloží a vrátí.

Když máme tuto funkci, tak nám stačí projít všechna políčka a pokud ještě nemají spočítaný nějaký směr, tak na nich zavoláme tuto funkci s daným směrem. Tím si pořídíme tyto rychlé hrany ze všech políček bludiště.

Jak dlouho tento předvýpočet trvá a jak nám pomůže? Tím, že si funkce ukládá již spočítané směry u políček, tak ji pro každé políčko a směr provedeme nejvýše jednou, tedy celý předvýpočet zabere čas $\mathcal{O}(RS)$.

Díky spočítaným hranám pak již při procházení nemusíme pokaždé krokovat, ale použijeme rychlé hrany a jedno políčko tak při prohledávání zpracujeme za konstantní čas ($\mathcal{O}(1)$). Celé prohledávání pak zvládneme v čase $\mathcal{O}(RS)$, což je tedy i celková časová složitost algoritmu. Oproti prvnímu pomalejšímu řešení to může představovat rozdíl v čase běhu jedna sekunda versus velké desítky sekund.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/33-Z4-3.py`

Úlohu připravili: Kuba Pelc, Jirka Sejkora, Tom Sláma, Eliška Vítková

33-Z4-4 Opačná úloha

Zkusme nejdříve odpovědět druhou otázku: Jakou časovou složitost má program? První podmínka (pokud $S < K$), nám zvětší index j o jedna. U druhé podmínky se nám zvětší index i o jedna. Třetí podmínka není zajímavá, protože tam hned končíme. Jelikož provedení jednoho cyklu nám trvá konstantně dlouho (provádíme maximálně tři porovnání a dvě sčítací operace), tak časová složitost závisí na tom, kolik provedeme opakování cyklu.

Když si rozmyslíme, že indexy i a j jen zvětšujeme, tak každé číslo můžeme navštívit maximálně dvakrát. Poprvé ho přičítáme do sumy S (při zvětšení indexu j) a podruhé ho odčítáme od sumy S (při zvětšení indexu i). Z toho víme, že provedeme asymptoticky $2n$ operací. Tedy časová složitost algoritmu je $\mathcal{O}(n)$. Paměťová složitost je $\mathcal{O}(1)$ – pamatujeme si tři proměnné.

Nyní víme, že náš algoritmus je příjemně rychlý, ale je správný?

Pro celá kladná čísla nám algoritmus funguje. Zkusme přijít na to proč. U celých kladných čísel nám platí následující: Mějme podposloupnost mezi indexy i a j. Pokud do této podposloupnosti přidáme jedno číslo (posuneme j o jedna), tak se součet podposloupnosti vždy zvětší a naopak pokud podposloupnost zmenšíme o jedna (posuneme i o jedna), tak se součet vždy zmenší.

Z toho nám platí, že pokud součet podposloupnosti je menší než hodnota K a tuto podposloupnost zmenšíme, tak součet bude pořád menší než hodnota K. Toto platí samozřejmě i naopak. Z toho dostáváme, že pokud součet podposloupnosti potřebujeme zvětšit, tak musíme podposloupnost rozšířit, a toto platí i naopak. Tímto jsme ale dostali algoritmus popsaný v zadání, tudíž algoritmus je správný.

Pro čistě záporná čísla nám algoritmus nefunguje. Uvažme třeba vstup `-4 -1` a hledané $K = -5$. Sečtení hodnot -4 a -1 je očividně výsledek -5, ale náš algoritmus toto nenajde. Při první opakování není první podmínka $S < K$ pravdivá (-4 není menší jak -5) a provede se tak druhá podmínka. Na konci prvního opakování má S hodnotu 0.

Není ale těžké uvědomit si, že pro záporná čísla by šlo algoritmus lehce opravit. Buď bychom mohli celý vstup vynásobit -1 nebo bychom mohli otočit nerovnosti v algoritmu.

Pro celá čísla nám algoritmus také nefunguje. Uvažme třeba vstup `5 -1` a hledané $K = 4$. Součet čísel 5 a -1 je 4, ale algoritmus toto nenajde. Narozdíl od čistě záporných čísel nelze algoritmus ani nijak lehce opravit, v tomto případě totiž nemůžeme o posunutí jednoho ani druhého konce intervalu nic prohlásit (obě posunutí mohou součet zvětšit i zmenšit) a tím padá hlavní myšlenka algoritmu.

Úlohu připravili: Jirka Kalvoda, Michal Kodad, Jirka Setnička

Výsledková listina čtvrté série začátečnické kategorie 33. ročníku KSP

	řešitel	*škola*	*ročník*	*sérií*	*Z4-1*	*Z4-2*	*Z4-3*	*Z4-4*	*série*	*celkem*
0.					8	10	12	14	44,0	176,0
1.	Robert Jaworski	GÚstavníPH	3	21	8	10	12	14	44,0	176,0
2.	Dominik Farhan	GMikulášPL	4	7	8	10	12	14	44,0	170,5
3.	Jakub Ondroušek	GTomkovaOL	1	12	8	10	12	13	43,0	169,5
4.	Kryštof Maxera	GJírovcČB	0	6	8	10	12	13	43,0	156,0
5.	Jonáš Dej	G Wicht	2	4	8	10	12	14	44,0	151,0
6.	Jan Hlavsa	GMělník	3	4	8	10	12	13	43,0	150,0
7.–8.	Robert Gemrot	GKomHavíř	4	13	8	10	12		30,0	134,0
	Tomáš Janovec	GMnichHrad	3	5	8	10	12		30,0	134,0
9.	Thomas Riedle	BRG APP	2	8	8	10	12		30,0	128,0
10.	Vojtěch Venzara	GMělník	3	4	8	10	12		30,0	124,0
11.	Erik Sabol	GČeskoliPH	1	7	8	10	12	13	43,0	116,0
12.–13.	Klára Grinerová	GZborovPH	4	4	8	10	1	13	32,0	110,0
	Pavel Šrytr	GMělník	4	4	8	10			18,0	110,0
14.	Veronika Jůzková	MensaG	3	8	8	10			18,0	107,0
15.	Adam Kolník	SSŠVTPraha	2	3	8	10	12		30,0	104,0
16.	Lukáš Létal	GJŠkodyPŘ	2	4	8	10	8		26,0	101,0
17.	Martin Fof	MendelGOP	3	4	8	10			18,0	92,0
18.	Yahor Herashchanka	ZŠ Turnov	0	4	8	7	0	6,9	21,9	90,6
19.–20.	David Holas	SPŠEMasLI	1	2	8	10	12	14	44,0	88,0
	Michal Mlynár	GJSeiferPH	0	2	8	10	12	14	44,0	88,0
21.	Jakub Smolik	GEBenešeKL	3	3	8	10			18,0	83,0
22.	Olga Cinková	ArcibisGPH	1	4	8	10	12		30,0	81,5
23.	Kryštof Latka	PORG Krč	3	2	8	10	12	14	44,0	81,0
24.	Oto Skýpala	GJŠkodyPŘ	−3	2	8	10	12	8	38,0	80,0
25.	Jakub Mikeš	GJŠkodyPŘ	3	2	8	10	12	13,5	43,5	76,5
26.	Ondřej Piroutek	GČeskoliPH	3	4	8	10			18,0	70,0
27.	Pavel Altmann	GMikulášPL	2	6	8	10			18,0	68,0
28.	Michal Bravanský	GBílovec	3	12	8	10	12		30,0	67,0
29.	Vít Novotný	GMělník	3	2					0,0	57,0
30.	Daniel Mencl	GMělník	3	3					0,0	53,0
31.	Vojtěch Skyba	G UherBrod	3	2	8	10	1		19,0	49,0
32.	Jan Soukup	GJiříPoděb	3	3	8				8,0	46,0
33.	Kryštof Marek	SGPCE	1	2	8	10			18,0	45,0
34.	Štěpán Filipčík	GTomkovaOL	3	2					0,0	43,0
35.	Jonáš Koziorek	GSOŠ FrMís	4	2	8	10			18,0	42,0
36.	Jakub Nevařil	G UherBrod	3	12					0,0	40,0
37.–39.	Matouš Bohoněk	GMělník	3	2					0,0	35,0
	Adam Húšťava	EupSchoolLux	3	13	8	0			8,0	35,0
	Michal Žáček	MensaG	4	2	8				8,0	35,0
40.	Zdeněk Pezlar	GJarošeBO	3	2					0,0	34,0
41.	Jonáš Bína	ZŠŠtáflovaHB	−4	4	2,7				2,7	32,7
42.–43.	Vojtěch Gaďurek	PORGPha	4	1					0,0	32,0
	Matěj Strnad	ZŠRiegraSM	0	5					0,0	32,0
44.–48.	Viktor Čubík	G UherBrod	3	1					0,0	30,0
	Martina Daňková	GBystrc	4	9	8	10	12		30,0	30,0
	Jáchym Hájek	ZŠTřebechovice	−2	2	8	7			15,0	30,0
	Tomáš Kašpárek	G FrýdlNOs	3	1					0,0	30,0
	Antonin Spaniel	GÚstavníPH	2	1					0,0	30,0
49.	Jan Kotyk	G Kolín	4	1					0,0	20,0
50.	Vojtěch Čermák	SPŠEMasLI	2	4	8	10			18,0	19,0
51.–54.	Viktor Číhal	SPŠSmíchov	1	1	8	10			18,0	18,0
	Petr Hladík	GMikulášPL	3	5					0,0	18,0
	Tomáš Chabada	SPŠEMasLI	4	5					0,0	18,0
	Tomáš Pražák	GJSeiferPH	0	1	8	10			18,0	18,0
55.	Radek Bláha	GČeskáČB	−1	4		4			4,0	15,0

	řešitel	*škola*	*ročník*	*sérií*	*Z4-1*	*Z4-2*	*Z4-3*	*Z4-4*	*série*	*celkem*
56.	Honza Kocourek	ParkLane	1	1					0,0	14,0
57.	Jan Najman	SPSEPard	4	8					0,0	10,0
58.–61.	Jakub Drobný	ŠPMNDaGB	3	1					0,0	8,0
	Adam Jahoda	GKepleraPH	2	1	8				8,0	8,0
	Jan Machourek	GBBr	0	3					0,0	8,0
	Matyas Oliva	G UherBrod	3	1					0,0	8,0
62.	Matěj Hošek	GVolgogrOS	−1	1					0,0	7,0
63.	Martin Bulíř	SPŠEMasLI	4	1					0,0	6,0
64.	Marek Maškarinec	GFXŠaldyLI	0	2					0,0	3,0
65.	Antonín Musil	PORGPha	4	3					0,0	2,0