73. ročník Matematické olympiády – 2023/2024

Úlohy krajského kola kategorie P

Krajské kolo 73. ročníku MO kategorie P se koná v úterý 23. 1. 2024 v dopoledních hodinách. Na řešení úloh máte 4 hodiny čistého času. V krajském kole MO-P se neřeší žádná praktická úloha, pro zajištění rovných podmínek řešitelů ve všech krajích je použití počítačů při soutěži zakázáno (s výjimkou nahrávání řešení do odevzdávacího systému). Zakázány jsou rovněž jakékoliv další pomůcky kromě psacích potřeb (např. knihy, výpisy programů, kalkulačky, mobilní telefony). Řešení každé úlohy vypracujte na samostatný list papíru.

Řešení každé úlohy má obsahovat:

- *Popis řešení,* to znamená slovní popis principu zvoleného algoritmu, *argumenty zdůvodňující jeho správnost* (případně důkaz správnosti algoritmu), *diskusi o efektivitě* vašeho řešení (časová a paměťová složitost; to se netýká úlohy P-II-4). Slovní popis řešení musí být jasný a srozumitelný i bez nahlédnutí do samotného zápisu algoritmu (do programu). Není možné odkazovat se na vaše řešení úloh domácího kola, opravovatelé je nemusí mít k dispozici.
- Doporučujeme uvést *zápis algoritmu* v nějakém dostatečně srozumitelném pseudokódu (případně v programovacím jazyce C/C++, Python nebo podobném). Nemusíte detailně popisovat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů, vyhledávání v poli, třídění apod. Zápis algoritmu sice není povinnou součástí řešení, ale při nejasností v popisu řešení může opravovatelům pomoci s pochopením algoritmu.

Za každou úlohu můžete získat maximálně 10 bodů. Hodnotí se nejen správnost řešení, ale také kvalita jeho popisu a efektivita zvoleného algoritmu. Algoritmy posuzujeme podle jejich časové složitosti, tzn. závislosti doby výpočtu na velikosti vstupních dat. Záleží přitom pouze na řádové rychlosti růstu této funkce. V zadání každé úlohy najdete přibližné limity na velikost vstupních dat. Efektivním vyřešením úlohy rozumíme to, že váš program spuštěný s takovými daty na současném běžném počítači dokončí výpočet během několika sekund.

Vzorová řešení úloh naleznete krátce po soutěži na webových stránkách olympiády *https://mo.mff.cuni.cz/p/*. Na stejném místě bude zveřejněn i seznam úspěšných řešitelů krajského kola a seznam řešitelů postupujících do ústředního kola. Svá opravená řešení s komentáři opravovatelů najdete v OSMO.

P-II-1 Petra lyžuje

Petra brzy vystartuje do druhého kola slalomu. Pokud by celou trať sjela čistě, momentálně vedoucí Michaelu by porazila o t setin vteřiny.

Na trati je n branek, očíslovaných shora dolů od 1 do n. O každé brance víme dvě čísla: pravděpodobnost p_i (v procentech) toho, že ji Petra projede čistě, a čas s_i (v setinách vteřiny), který oproti optimální jízdě ztratí, pokud se jí čistý průjezd této branky nepodaří.

Soutěžní úloha

Napište program, který vypočítá pravděpodobnost toho, že Petra slalom vyhraje (nebo se alespoň umístí na děleném prvním místě) – tedy toho, že dohromady chybami na trati ztratí nanejvýš t setin vteřiny. Pozorně si prohlédněte *omezení,* pro která máte tento úkol vyřešit, uvedená v sekci Bodování.

Váš program smí pracovat s reálnými čísly a při jeho psaní můžete předpokládat, že všechny operace s nimi jsou matematicky přesné. Nemusíte se tedy zabývat zaokrouhlovacími chybami, které by nastaly při běhu vašeho programu na skutečném počítači.

Formát vstupu

Na prvním řádku vstupu jsou čísla n a t ($1 \leq n, t \leq 5\,000$). Na i-tém z n následujících řádků jsou hodnoty p_i a s_i. Všechna čísla na vstupu jsou přirozená. Pravděpodobnosti p_i jsou v procentech a $1 \leq p_i \leq 99$, nemohou tedy být 0 ani 100.

Formát výstupu

Na výstup vypište jedno reálné číslo, pravděpodobnost v procentech, že Petra vyhraje.

Bodování

Plný počet bodů mohou získat řešení efektivní pro $n, t \leq 5\,000$ bez dalších předpokladů.

Nejvýše 8 bodů mohou získat řešení efektivní pro $n, t \leq 5\,000$, které fungují *za dodatečného předpokladu,* že všechny ztráty s_i mají stejnou hodnotu (tedy platí $s_1 = s_2 = \cdots = s_n$).

Nejvýše 5 bodů mohou získat řešení efektivní pro $n \leq 5\,000$, které fungují *za silnějšího dodatečného předpokladu,* že $t = 25$ a $s_1 = s_2 = \cdots = s_n = 10$.

Nejvýše 3 body mohou získat řešení efektivní pro $n \leq 20$ bez dalších předpokladů.

Pokud vaše řešení využívá některý z dodatečných předpokladů, na začátku *výrazně* uveďte který.

Příklady

Vstup:	*Výstup:*
3 25 90 10 90 10 70 10	99.7

Na trati jsou tři branky. Petra má náskok $t = 25$ setin vteřiny. Chyba na každé brance ji stojí 10 setin vteřiny. O výhru tedy přijde jen tehdy, když udělá chybu na všech třech brankách. To se stane s pravděpodobností

$$10\,\% \times 10\,\% \times 30\,\% = 0.1 \times 0.1 \times 0.3 = 0.003 = 0.3\,\%.$$

S pravděpodobnosti $100\,\% - 0.3\,\% = 99.7\,\%$ *tedy vyhraje.*

Vstup:	*Výstup:*
1 10 90 10	100

I s chybou se Petra umístí na děleném prvním místě.

Vstup:	*Výstup:*
4 25 50 27 80 11 70 19 90 12	45.8

První branku musí Petra projet čistě, jinak rovnou ztratí příliš mnoho času. Jestliže pokazí třetí, všechny zbylé branky musí projet čistě. Jestliže čistě projede první a třetí branku, druhou i čtvrtou může pokazit.

P-II-2 Jedno obrácení

Máme pole $A[0 \ldots n-1]$. Toto pole obsahuje právě jednou každé z čísel od 0 do $n-1$. *Uklizenost* pole je rovna počtu indexů i, pro které platí $A[i] = i$. Brzy přijede na návštěvu Inspektor polí a my bychom mu rádi ukázali co nejvíce uklizené pole. Je zjevné, že největší možná uklizenost pole je n, čehož lze dosáhnout tím, že pole A uspořádáme vzestupně. Na to však bohužel nemáme čas – než přijde Inspektor, stíháme už jen obrátit pořadí prvků v jednom souvislém úseku pole A; tedy si vybrat indexy $i \leq j$ a prvek na pozici i prohodit s prvkem na pozici j, prvek na pozici $i+1$ s prvkem na pozici $j-1$, atd.

Soutěžní úloha

Napište program, který načte popis pole A a ze všech možných obrácení úseků najde to, kterým z pole A vyrobíme pole s největší možnou uklizeností.

Formát vstupu

Na prvním řádku vstupu je přirozené číslo n ($1 \leq n \leq 200\,000$). Na druhém řádku vstupu jsou hodnoty $A[0]$, $A[1]$, …, $A[n-1]$. Je zaručené, že v těchto hodnotách se každé celé číslo od 0 do $n-1$ nachází právě jednou.

Formát výstupu

Na výstup vypište dvě čísla $i \leq j$: index prvního a posledního prvku v úseku, který je třeba obrátit. Pokud existuje několik optimálních řešení, můžete vypsat libovolné z nich.

Bodování

Plný počet bodů mohou získat pouze řešení s lineární časovou složitostí. Jiná řešení efektivní pro $n \leq 200\,000$ mohou získat nanejvýš 9 bodů. Nejvýše 6 bodů mohou získat řešení efektivní pro $n \leq 7\,000$. Nejvýše 3 body mohou získat řešení efektivní pro $n \leq 300$.

Příklady

Vstup:	*Výstup:*
7 5 4 3 2 1 0 6	0 5

Když obrátíme pořadí prvních šesti prvků tohoto pole, dostaneme uspořádané pole. Toto je zjevně optimální.

Vstup:	*Výstup:*
7 6 5 2 4 3 1 0	0 6

Obrácením celého pole dostaneme pole $(0, 1, 3, 4, 2, 5, 6)$*. Jeho uklizenost je 4: hodnoty 0, 1, 5 a 6 jsou na správných místech. Všimněte si, že hodnotu 2 toto obrácení „pokazilo“ – přesunulo ji pryč ze správného místa.*

Vstup:	*Výstup:*
7 5 6 0 1 2 3 4	2 6

Pro tento vstup žádné obrácení nevytvoří pole s uklizeností větší než 1, proto je správnou odpovědí libovolné z obrácení, které vyrobí pole s uklizeností 1. V našem příkladu jsme si vybrali obrácení úseku na indexech 2 až 6. To vyrobí pole $(5, 6, 4, 3, 2, 1, 0)$*. Uklizenost tohoto pole je 1, neboť hodnota 3 se nachází na indexu 3.*

Vstup:	*Výstup:*
7 0 1 2 3 4 5 6	4 4

Toto pole je již optimální, nejlepším řešením je nic nepokazit, a tedy obrátit nějaký úsek délky 1.

P-II-3 Těžkotonážní skokobot

Bludiště pro robota je tvořeno $r \times s$ čtvercovými políčky. Na plánu bludiště jsou řádky očíslovány od 0 do $r-1$ shora dolů a sloupce od 0 do $s-1$ zleva doprava. Některá políčka v bludišti jsou překážky (značené `X`), ostatní jsou volná (značená tečkou).

Robot začíná v levém horním rohu plánku bludiště. Vaším úkolem je na co nejmenší počet akcí robota dostat do pravého dolního rohu. Existují dva typy akcí: kroky a skoky.

- Při kroku si robot vybere jeden ze čtyř základních směrů (doleva, doprava, nahoru či dolů) a pohne se jím na sousední políčko. Krok samozřejmě může udělat jen tehdy, vede-li na volné políčko v bludišti.
- Při skoku si robot také vybere jeden ze čtyř základních směrů, ale tentokrát si navíc zvolí vzdálenost $d > 1$. Následně vyskočí zvoleným směrem ze svého současného políčka, přeskočí přes $d-1$ políček (ta mohou být libovolného typu) a dopadne na d-té (to musí být prázdné).

Skoky však mají i jednu nevýhodu: robot je těžký a má velkou hybnost, a proto neumí po skoku hned zastavit. Po *každém* skoku proto musí následující pohyb (krok nebo skok) vést *tím samým směrem;* speciálně skok nesmí provést, pokud by se dalším krokem ve stejném směru dostal mimo bludiště. Do cíle cesty robot také nesmí dorazit skokem, protože potřebujeme, aby tam zůstal stát.

Soutěžní úloha

Napište program, který načte rozměry a mapu bludiště a zjistí, jestli lze robota dostat do cíle, a pokud ano, na jaký nejmenší počet akcí to lze udělat.

Formát vstupu

Na prvním řádku vstupu jsou rozměry bludiště, čísla r a s ($1 \leq r, s \leq 10\,000$). Zbytek vstupu tvoří mapa bludiště: r řádků, na každém z nich s znaků '`X`' a '`.`'. Je zaručeno, že první a poslední z těchto znaků (start a cíl robota) jsou '`.`'.

Formát výstupu

Na výstup vypište minimální počet akcí potřebných k přesunu robota z políčka $(0, 0)$ na políčko $(r-1, s-1)$, nebo -1, jestliže řešení neexistuje.

Bodování

Plný počet bodů mohou získat řešení efektivní pro $r, s \leq 10\,000$ bez dalších omezení.

Nejvýše 8 bodů mohou získat řešení efektivní pro $r, s \leq 1000$.

Nejvýše 5 bodů mohou získat řešení efektivní pro $r, s \leq 50$.

Libovolně pomalé správné řešení může získat až 3 body.

Příklady

Vstup:	*Výstup:*
4 7 ..XXXXX X..XXXX XX..XXX XXX....	8

Provedeme kroky vpravo, dolů, vpravo, dolů, vpravo, dolů, potom skok o 2 doprava a nakonec krok doprava.

Vstup:	*Výstup:*
1 5 .XXX.	-1

Skočit o čtyři doprava nesmíme, neboť bychom neuměli udělat následně další pohyb stejným směrem.

Vstup:	*Výstup:*
4 7 ..XXX.. X..XXXX XX..X.X XXX....	6

Začneme skokem doprava o pět a následným krokem doprava, čímž se dostaneme do pravého horního rohu. Dále uděláme krok zpět doleva, poté skok o dva dolů a po něm zabrzdíme dalším krokem dolů. Pak ještě krok doprava a jsme v cíli.

P-II-4 O Vekslákbotovi a Pokladničce

K této úloze se vztahuje studijní text uvedený na následujících stranách, který je stejný jako v domácím kole.

Jednotlivé podúlohy jsou hodnoceny nezávisle, můžete je řešit v libovolném pořadí. Při hodnocení úkolů krajského kola budeme přihlížet jen ke správnosti vašich programů – na jejich efektivitě (časové složitosti) nám nebude záležet. Nezapomeňte kromě samotného programu uvést i jeho slovní popis, včetně zdůvodnění správnosti.

Podúloha A (2 body): Přebarvení na maximum

V Pokladničce je na začátku $c \geq 0$ červených, $m \geq 0$ modrých a $z \geq 0$ zelených žetonů (a nic jiného), přičemž je zaručeno, že tyto počty *jsou navzájem různé.* Napište program, po jehož skončení bude v Pokladničce přesně $c + m + z$ žetonů, přičemž všechny musí mít tu barvu, které bylo na začátku nejvíc.

Podúloha B (3 body): Přebarvení na minimum

Počáteční stav Pokladničky je stejný jako v podúloze A. Napište program, po jehož skončení bude v Pokladničce přesně $c + m + z$ žetonů, přičemž všechny musí mít tu barvu, které bylo na začátku *nejméně.*

Podúloha C (5 bodů): Mocnina tří

Na začátku je v Pokladničce $c \geq 0$ červených žetonů (a nic jiného). Napište program, po jehož skončení bude červených žetonů v Pokladničce přesně 3^c. Navíc v ní může být i libovolně mnoho žetonů jiných barev.

Upozorňujeme, že váš program musí skončit – pro každé c se po konečném počtu kroků musí dostat do situace, ve které již nelze provést žádnou z jeho instrukcí.

Studijní text

Za devatero horami a devatero řekami byla jednou jedna prazvláštní země. V této zemi žili roboti a místo peněz používali žetony všech možných barev. V domečku na úpatí desáté hory spolu žili dva hrdinové našeho příběhu: roboti Vekslákbot a Pokladnička.

Jak asi tušíte z jejího jména, Pokladnička v sobě ráda ukládá všechny možné žetony. Jak možná z jejího jména netušíte, Pokladnička také velmi ráda vykonává různé programy. A jak jste si po přečtení předchozí věty naprosto jistí (ostatně, toto je studijní text Matematické Olympiády kat. P a ne jen tak nějaká pohádka), právě tyto programy pro ni budete psát vy jako řešení soutěžních úloh.

Vekslákbot je mistr vyměňování žetonů. Ať už chce vyměnit dva červené za tři modré, anebo žlutý a černý za $10^9 + 7$ šedých, Vekslákbot určitě zná robota, který zná robota, jenž s ním právě takovou směnu rád provede. Vekslákbot má moc rád Pokladničku, takže pokud ho ona o nějakou výměnu poprosí, okamžitě (nebo, jak říkáme my, v konstantním čase) ji pro ni provede.

Pojďme se nyní podívat na to, jak budou vypadat programy pro Pokladničku.

Základy: vstup, výstup, program

Vstupem pro Pokladničku budou žetony, jež má na začátku uložené v sobě. Neexistuje žádný výstup. V zadání jednotlivých úloh budeme různě definovat cíle, jichž mají vaše programy dosáhnout.

Program pro Pokladničku se skládá ze dvou částí: *omezení*, která musí dodržet, a *pokynů*, které má vykonávat.

Omezení

První částí programu pro Pokladničku je **konečná množina** omezení (může být i prázdná). Omezení pro Pokladničku určují, kolik nejvýše žetonů konkrétní barvy, případě kombinací žetonů různých barev, může mít najednou v sobě. Formálně, omezení jsou lineární nerovnosti následujícího tvaru:

$$k_1 \cdot \text{barva}_1 + k_2 \cdot \text{barva}_2 + \cdots + k_n \cdot \text{barva}_n \leq \text{limit},$$

kde všechny koeficienty k_i i limit jsou konkrétní kladná celá čísla a barva_i jsou proměnné označující počet žetonů příslušné barvy v Pokladničce. Tečky $(\cdot)$ můžeme vynechat, pokud je zřejmé, kde končí koeficient a kde začíná název barvy. Příklady omezení jsou například nerovnosti modrá ≤ 7 nebo modrá $+ 2$červená ≤ 3.

Pokud se nějaké barvy žádné omezení netýká, může být v Pokladničce libovolné množství žetonů této barvy.

Instrukce

Každá instrukce pro Pokladničku má následující tvar:

$$k_1 \cdot \text{barva}_1, \ldots, k_n \cdot \text{barva}_n \to \ell_1 \cdot \text{barva}_{n+1}, \ldots, \ell_m \cdot \text{barva}_{n+m}$$

Všechno nalevo od $\to$ budeme nazývat levou stranou instrukce, všechno napravo zase pravou stranou. Tečky $(\cdot)$ můžeme vynechat, pokud je zřejmé, kde končí koeficient a kde začíná název barvy.

Všechna k_i i ℓ_j musí být kladná celá čísla. V rámci pravé i levé strany instrukce musí být všechny použité barvy různé. Je povolené použít stejnou barvu na obou stranách instrukce. Je povolené mít $n = 0$ či $m = 0$, tedy instrukci, které má některou stranu prázdnou. (Je povolené mít prázdné i obě strany, ale jak se brzy dozvíme, program, v němž bychom takovou instrukci použili, by nikdy neskončil.)

Příklady instrukcí:

- 2červená $\rightarrow$ 3modrá
- 1žlutá, 1černá $\rightarrow$ $(10^9 + 7)$šedá
- 3červená $\rightarrow$ 333červená, 334cyklámenová, 335purpurová
- 2zelená $\rightarrow \emptyset$

Poslední instrukce z příkladu má prázdnou pravou stranu. Pro zápis prázdné strany instrukce budeme používat symbol prázdné množiny, aby bylo jasné, že je prázdná úmyslně.

V našich příkladech budeme používat skutečné názvy barev. Ve svých programech můžete jako názvy barev používat i libovolné jiné alfanumerické řetězce. Je také povoleno v zápisech vynechávat koeficient 1. Druhou z výše uvedených instrukcí lze tedy zapsat také jako žlutá, černá $\rightarrow$ $(10^9 + 7)$šedá.

Pokladnička vykoná instrukci tak, že ze sebe vyndá sadu žetonů na levé straně instrukce, dá ji Vekslákbotovi a poprosí jej, aby jí za ně přinesl sadu žetonů z pravé strany. Vekslákbot to samozřejmě okamžitě zajistí a Pokladnička do sebe vloží žetony, které jí přinesl.

Pokud v sobě Pokladnička nemá všechny žetony, které vyžaduje levá strana instrukce, nelze danou instrukci v tu chvíli provést. Pokud by například měla pouze dva červené žetony, nešlo by provést instrukci 3červená $\rightarrow$ 333červená.

Pokladnička také nemůže provést instrukci, po jejímž provedení by bylo porušené některé omezení.

Program

Program pro Pokladničku tvoří **konečná posloupnost** instrukcí výše uvedeného typu. Zdůrazňujeme, že jde o posloupnost, tedy **záleží** na pořadí instrukcí.

Vykonávání programu

Program se vykonává v krocích. V každém kroku Pokladnička začne číst program od začátku a čte jej, dokud nenajde první instrukci, kterou momentálně může vykonat, a tu vykoná. (Na zbytek programu se v tomto kroku už ani nepodívá a v dalším kroku začne znovu číst program od začátku.)

Vykonávání programu skončí, když už žádnou instrukci v programu nelze vykonat.

Příklad #1: Více červených

Úloha: Na začátku jsou v Pokladničce nějaké červené a nějaké modré žetony. Napište program, po jehož skončení bude v Pokladničce právě jeden zlatý žeton a nic jiného, pokud bylo červených žetonů více než modrých. Ve všech ostatních případech musí Pokladnička skončit úplně prázdná.

Řešení 1 (bez omezení)

Nebudeme mít žádná omezení. Pokyny budou vypadat následovně:

1. červená, modrá $\rightarrow \emptyset$
2. červená, zlatá $\rightarrow$ zlatá
3. červená $\rightarrow$ zlatá
4. modrá $\rightarrow \emptyset$

Při vykonávání tohoto programu bude nejprve Pokladnička používat instrukci 1, dokud to lze. Až to přestane být možné, má v sobě už buď jen červené, anebo jen modré žetony (anebo žádné žetony a program skončí). Pokud jsou modré, jediná vykonatelná instrukce je instrukce 4, jejímž opakovaným používáním se Pokladnička vyprázdní a program skončí.

Pokud po skončení instrukce 1 zůstanou v Pokladničce červené žetony, bude to o trochu složitější: Jediná instrukce, která se v danou chvíli dá použít, je instrukce 3, která vymění jeden červený žeton za jeden zlatý. Od této chvíle se však instrukce 3 již nepoužije, a to proto, že lze vykonávat instrukci 2, která je v posloupnosti dříve. Pomocí té se Pokladnička postupně zbaví zbývajících červených žetonů. Jakmile v ní zůstane jen samotný zlatý žeton, nelze vykonat žádnou instrukci, a program tedy skončí.

Řešení 1 (s omezením)

Stejnou úlohu můžeme vyřešit s použitím omezení. Vystačíme si s jediným:

- zlatá ≤ 1

Program vypadá následovně:

1. červená, modrá $\rightarrow \emptyset$
2. červená $\rightarrow$ zlatá
3. červená $\rightarrow \emptyset$
4. modrá $\rightarrow \emptyset$

Tentokrát se v situaci, kdy jsou v Pokladničce jen samé červené žetony, nejprve jednou vykoná instrukce 2 (jeden červený žeton vyměníme za jeden zlatý) a potom se už bude používat instrukce 3, dokud červené žetony nedojdou. Omezení, které jsme si zvolili, totiž Pokladničce brání znovu využít instrukci 2.

Příklad #2: Součet

Úloha: Na začátku je v Pokladničce $c > 0$ červených žetonů, $m > 0$ modrých žetonů a jeden zelený. Napište program, po jehož skončení bude v Pokladničce přesně c červených, m modrých a $c + m$ fialových žetonů.

Řešení: Kdybychom pouze chtěli dostat $c + m$ fialových žetonů, stačilo by vyměnit všechny červené i modré žetony za fialové „s kurzem jedna ku jedné“. Jak ale nepřijít o původní žetony?

Naše řešení nebude mít žádná omezení a program bude vypadat následovně:

1. červená, zelená → tmavočervená, zelená
2. modrá, zelená → tmavomodrá, zelená
3. zelená → žlutá
4. tmavočervená, žlutá → červená, fialová, žlutá
5. tmavomodrá, žlutá → modrá, fialová, žlutá
6. žlutá → $\emptyset$

Všimněte si, jak náš program využívá přítomnost zeleného a žlutého žetonu v Pokladničce: Pomocí nich umíme rozlišit, zda ještě měníme původní červené a modré žetony, anebo zda již zpátky vznikají nové.