

Korespondenční Seminář z Programování

ZAČÁTEČNICKÁ KATEGORIE

38. ročník



KSP-Z

Březen 2026

Právě se díváte na leták čtvrté série 38. ročníku KSP-Z, neboli Korespondenčního Semináře z Programování, Začátečnické kategorie. Zapojit se může **každý středoškolák i základoškolák**.

V průběhu roku vydáváme několik dalších sérií úloh podobných této. Pokud budete mít jakoukoliv otázku, neváhejte se zeptat. Kontaktní adresy najdete v patičce na konci letáku. Přejeme hodně štěstí!

Termín série: neděle 19. dubna ve 32:00 (tedy další ráno v 8:00), praktické úlohy za třetinu bodů až do 26. dubna

Obsah série: 3 praktické úlohy (značené ) – K těmto úlohám je nutné napsat program (v libovolném vhodném jazyce), stáhnout si z našeho webu vstupní data a odevzdat odpovídající výstup.
1 teoretická úloha (značená ) – U této úlohy nás zajímá hlavně slovní popis řešení, ve kterém byste měli zdůvodnit jeho funkčnost a ideálně nás i přesvědčit o jeho efektivitě.

Odevzdávání: Přes web na adrese <https://ksp.mff.cuni.cz/z/odevzdavatko/>



Zadání čtvrté série začátečnické kategorie 38. ročníku KSP

38-Z4-1 Sousední čísla 8 bodů

Prázdniny se rychle blíží a Kevin se na ně tak těší, že si napsal seznam aktivit, které by chtěl během prázdnin podniknout. Protože bydlí na konci jedné velmi dlouhé vlakové trati, ke každé aktivitě napsal, jak daleko se nachází – jako počet stanic od jeho domu. Jednoho dne se Kevin podíval na svůj seznam a uvědomil si jednu zásadní věc. Nápadů měl tolik, že neví kde začít!

Dlouho nad tím přemýšlel a nakonec vymyslel, že začne návštěvou dvou sousedních stanic, kde toho chce podniknout nejvíce. Bohužel ale Kevin při vytváření seznamu nepřemýšlel nad jeho strukturou a psal svoje aktivity v nahodilém pořadí. To znamená, že různé aktivity pro každou stanici jsou rozházené po celém seznamu.

Aby Kevin zjistil, které dvě sousední stanice navštívit jako první, musí ze všech kandidátů vybrat tu dvojici sousedních stanic, pro které dohromady napsal nejvíce aktivit. Pomůžete mu ji najít?

K dispozici máte seznam, kde každý řádek reprezentuje jednu aktivitu a na řádku bude číslo reprezentující stanici, na které se aktivita bude konat. Sousední stanice poznáme tak, že to jsou dvě po sobě jdoucí čísla, tedy lišící se právě o 1 (třeba 5 a 6, 6 a 7 nebo 41 a 42).

Toto je praktická open-data úloha. V odevzdávacím systému si necháte vygenerovat vstupy a odevzdáte příslušné výstupy. Záleží jen na vás, jak výstupy vyrobíte.

Formát vstupu: Na prvním řádku dostanete číslo N – délku seznamu. Na následujících N řádcích bude následovat seznam aktivit, kde na každém řádku bude jedno (přirozené) číslo stanice, kde se aktivita koná.

Formát výstupu: Na výstupu vypište na prvním řádku dvě sousední stanice, které mají nejvíce aktivit ze všech sousedních stanic. A na druhém řádku vypište počet aktivit patřících každé stanici (ve stejném pořadí jako na prvním řádku).

Garantujeme, že vždy existuje alespoň jedna dvojice sousedních stanic. Pokud existuje víc správných dvojic, vypište libovolnou z nich.

Ukázkový vstup:

8
1
4
5
1
1
4
5
6

Ukázkový výstup:

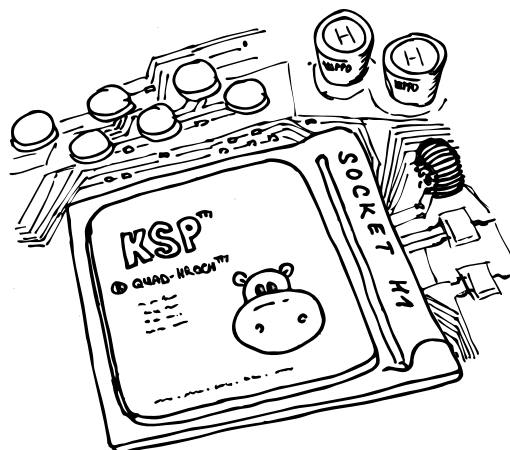
4 5
2 2

38-Z4-2 Efektivní záloha webu 10 bodů

Sára spravuje infrastrukturu KSPího webu a dnes zjistila že dochází místo na disku. Na server útočí AI scan boti a vygenerovali skoro 1 HippoByte dat ($\frac{1}{4}$ BrontoBytu) v log souborech. Sára se proto rozhodla změnit infrastrukturu. Nejprve ale potřebuje udělat zálohu, která nebude zabírat příliš místa.

Sára musí jednat rychle. Pokud se jí nepodaří uvolnit místo na disku, tak se server zhroutlí a KSP web bude nedostupný! Pomozte Sáře vytvořit efektivní zálohu, která bude zabírat méně místa na disku a půjde rychle obnovit. Sára při svém výzkumu zjistila, že optimální způsob je Hroší-LZW komprese.

Toto je praktická open-data úloha. V odevzdávacím systému si necháte vygenerovat vstupy a odevzdáte příslušné výstupy. Záleží jen na vás, jak výstupy vyrobíte.



Hroší-LZW komprese:

Místo ukládání jednotlivých znaků si udržujeme slovník frází, které se v textu objevují. Slovník na začátku obsahuje právě malá písmena a-z s kódy 0–25. Postup komprese je, že držíme aktuální posloupnost w (na začátku prázdná) a čteme vstup zleva doprava. Pro další znak c zkusíme, zda se $w + c$ ve slovníku vyskytuje. Pokud ano, rozšíříme w na $w + c$ a pokračujeme. Pokud ne, vypíšeme kód pro w , přidáme $w + c$ do slovníku (s dalším dostupným číslem) a nastavíme $w := c$. Po konci vstupu ještě vypíšeme kód pro poslední w , pokud není prázdné.

Formát vstupu: Vstupní soubor obsahuje jediný řádek skládající se z malých písmen anglické abecedy. Slovník na začátku obsahuje právě těchto 26 jednopísmenných frází s kódy 0–25. Nemusíte řešit konce řádků nebo jiné speciální znaky.

Formát výstupu: Nejdřív vypíšete celý slovník, kde každý řádek obsahuje kód, index předchozího slova a poslední znak oddělené mezerou. Pak vypíšete text, který je výstupem LZW komprese, tedy posloupnost kódů oddělených mezerou na jednom řádku.

Nevypisujete kódy 0–25, protože ty jsou implicitně známy jako jednopísmenné fráze a-z.

Ukázkový vstup:	Ukázkový výstup:
ababa	26 0 b
	27 1 a
	28 26 a
	0 1 26 0

Hledáme postupně nejdelší posloupnosti:

- načteme **a**, pak **b** – slovník neobsahuje **ab**, vypíšeme 0, přidáme kód 26 pro **ab** (slovo s indexem 0 rozšířené o **b**).
- načteme **a** – slovník neobsahuje **ba**, vypíšeme 1, přidáme kód 27 pro **ba** (slovo s indexem 1 rozšířené o **a**).
- načteme **b** – slovník obsahuje **ab**, $w := ab$.
- načteme **a** – slovník neobsahuje **aba**, vypíšeme 26, přidáme kód 28 pro **aba** (slovo s indexem 26 rozšířené o **a**).
- Na konci vypíšeme kód pro poslední znak (0).

Aktuálně se může zdát, že komprese není efektivní, ale pro delší texty se to zlepšuje, protože se budou objevovat delší fráze, které se budou opakovat.

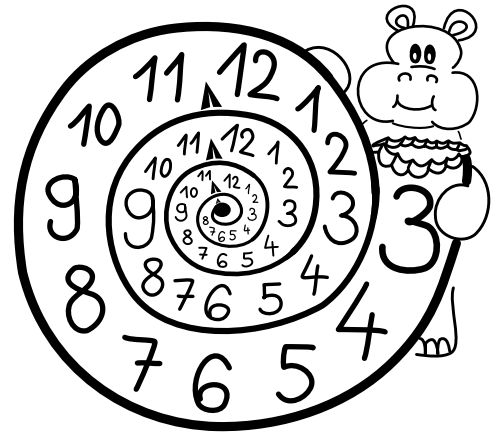
38-Z4-3 Jarní malování 12 bodů

Přišlo jaro, a s ním i radost malování. Kevin ale nemá žádné štětce ani barvy, tak se rozhodl, že bude malovat na počítači. Otevřel program na malování a dostal inspiraci, že namaluje avantgardní černobílý obraz, kde každý pixel je jen nějaký odstín šedé.

Kevin maloval a maloval, než domaloval svůj obrázek, tak si uvědomil, že na něm něco chybí. V obraze byla spousta odstínů šedé, ale žádný z nich nebyl dostatečně výrazný. Tak sáhl po nástroji „kyblík“ a začal přemýšlet nad tím, kde by ho mohl použít, aby měl co nejlepší efekt.

Po dlouhém přemýšlení vybral několik pozic, na které by mohl kliknout kyblíkem, ale nevěděl, jaký bude výsledek. Pomůžete mu s tím?

Toto je praktická open-data úloha. V odevzdávacím systému si necháte vygenerovat vstupy a odevzdáte příslušné výstupy. Záleží jen na vás, jak výstupy vyrobíte.



Formát vstupu: Začátek vstupu je validní obrázek ve formátu PGM¹ (můžete zkusit celý vstup otevřít v aplikaci, která tento formát podporuje, například v GIMPu). Použitý formát vypadá následovně: Na prvním řádku je P2, což značí, že se jedná textový PGM obrázek. Na druhém řádku jsou dvě čísla W a H značící šířku a výšku obrázku. Na třetím řádku je číslo udávající rozsah hodnot, které mohou pixely v obrázku nabývat. Ve všech vstupech je toto číslo 255, takže ho můžete ignorovat. Na dalších H řádcích je vždy W čísel reprezentujících jas pixelů (od 0 do 255).

Tím končí část vstupu, která reprezentuje obrázek, a začíná část, která reprezentuje pozice pro použití nástroje „kyblík“. Na dalším řádku je číslo K , které značí počet pozic na použití nástroje „kyblík“. Následuje K řádků, kde každý řádek je obsahuje tři čísla x , y a t . (x, y) jsou souřadnice pixelu – x je sloupec a y je řádek – na který se klikne kyblíkem, a t je absolutní tolerance barvy při použití kyblíku. To znamená, že pokud se použije kyblík na pixel s jasnem j a tolerancí t , tak se změny všechny pixely, které jsou *dosažitelné* z tohoto pixelu. Pixel je dosažitelný, pokud existuje cesta přes sousední pixely (sousedící hranou), kde každý pixel na cestě má jas v rozmezí $\langle j - t, j + t \rangle$ kde j je vždy původní jas pixelu, na který bylo kliknuto.

Formát výstupu: Vypíšte K řádků, kde každý řádek bude obsahovat jedno číslo reprezentující počet pixelů, které by se změnilo po použití kyblíku na danou pozici, ve stejném pořadí jako na vstupu. Je důležité poznamenat, že žádná změna pixelů doopravdy nenastane (ostatně na vstupu ani není informace, na jakou barvu kyblík přebarvuje), Kevin pouze chce vědět, kolik pixelů by se změnilo, kdyby použil kyblík jedině na danou pozici.

Ukázkový vstup:	Ukázkový výstup:
P2	17
5 4	7
255	5
107 107 106 104 104	
135 135 195 196 169	
134 136 135 196 168	
135 135 135 165 166	
3	
0 1 50	
3 1 32	
4 0 20	

¹ https://en.wikipedia.org/wiki/Netpbm_format

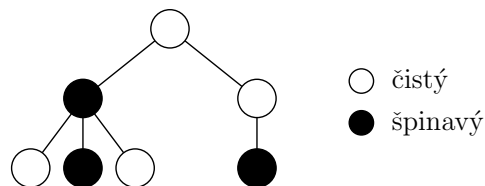
Kevin se rozhodl, že využije svůj zimní kurz vaření a půjde vařit na letní tábor. Na rozdíl od kurzu, kde se jen naučil vařit, tu musí dodržovat hygienická opatření, aby ho nezavřela hygiena. Opatření jsou různá, ale pro Kevina je z nich nejotravnější to, že si musí umývat ruce po zpracování některých přísad.

Přísady jsou dvou typů: buď „čisté“ nebo „špinavé“. Když Kevin bude chtít zpracovat čistou přísadu, ale předtím zpracovával špinavou, tak si musí umýt ruce. Kevin si chce ruce umýt co nejméněkrát, aby nespotřeboval víc mýdla než je potřeba, kvůli omezení v rozpočtu. Pomůžete mu?

Uvaření jídla se dá popsat stromem závislostí, kde každý vrchol představuje mezivýsledek přípravy. Vrchol může být čistý nebo špinavý a na jeho zpracování musíme mít do-

končeny všechny jeho děti. Tedy při uvaření nějakého jídla postupujeme od listů (základních přísad) přes vnitřní vrcholy (mezivýsledky) až ke kořeni (hotové jídlo).

Vášim úkolem je navrhnout algoritmus, který určí, v jakém pořadí by měl Kevin zpracovávat jednotlivé vrcholy stromu, aby si musel umýt ruce co nejméněkrát a tím pádem spotřeboval co nejméně mýdla.



Toto je teoretická úloha. Není nutné ji programovat, odevzdává se pouze slovní popis algoritmu. Více informací zde: <http://ksp.mff.cuni.cz/viz/tinfo>

