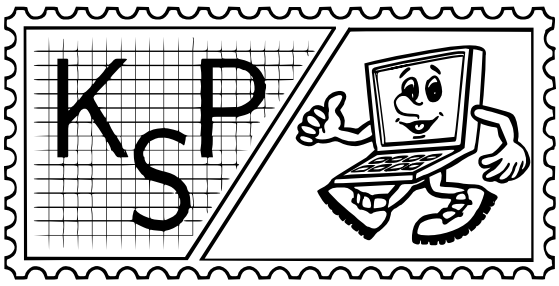

Korešpondenčný seminár z programovania

Leták zimnej časti XXX. ročníka

Korešpondenčný seminár z programovania (KSP) je súťaž programátorov – stredoškolákov a mladších – pripravovaná skupinou študentov a doktorandov FMFI UK. Jej cieľom je zdokonaliť žiakov v programovaní a v algoritmickom myslení.

Ak študuješ na strednej škole a vieš aspoň trochu programovať, neváhaj a zapoj sa do našej súťaže, má to množstvo výhod:

- Riešením súťažných úloh a štúdiom našich vzorových riešení sa môžeš naučiť mnoho nového. Získané poznatky a skúsenosti sa ti iste budú hodiť v iných súťažiach v programovaní (napríklad pri riešení Olympiády v informatike), počas vysokoškolského štúdia, či pri prijímacích pohovoroch do zamestnania. (Mnoho našich bývalých riešiteľov sa bez ťažkostí zamestnalo v špičkových IT spoločnostiach ako Google, Facebook, ESET, ...)
- Na riešenie úloh máš dosť času a môžeš ich riešiť doma bez toho, aby si niekam cestoval.
- Medzi zadaniami sa nachádzajú ľahšie aj ťažšie. Každý si môže vybrať tie, ktoré vie riešiť a ktoré považuje za zaujímavé.
- Pre najlepších riešiteľov organizujeme každoročne dve týždenné sústredenia. Sústredenie je jedinečnou príležitosťou ako spoznať nových priateľov s podobnými záujmami, naučiť sa čosi viac nielen o programovaní a zažiť kopec zábavy.

Ako KSP prebieha?

Počas školského roka prebehnú dve samostatné časti súťaže: zimná a letná. Každá časť sa skladá z dvoch kôl, každé kolo obsahuje desať súťažných úloh. Najlepších riešiteľov zimnej časti pozývame na jarné sústredenie; najlepších riešiteľov letnej časti zase na to jesenné.

Súťažiť sa dá v troch kategóriách: **Z** (pre začínajúcich riešiteľov, obsahuje úlohy 1–5), **O** (skúsenejší riešitelia, úlohy 4–8) a **T** (špeciálna kategória pre náročných, päť samostatných úloh). Každá kategória má svoju vlastnú výsledkovú listinu. Na sústredenia pozývame riešiteľov na základe výsledkov v kategóriách Z a O.

Vaše riešenia úloh môžete odovzdávať na stránke `http://www.ksp.sk/eRiesenie`, kým neuplynie termín určený v zadaniach kola. Po tom, čo riešenia opravíme, nájdete na tomto mieste aj naše komentáre k nim a počet bodov získaný za jednotlivé úlohy.

Ako má vyzerať riešenie a za čo dostanem body?

Vašou úlohou je vytvoriť program, ktorý rieši zadanú úlohu. V prvom rade sa snažte, aby bol korektný, t.j. aby dal pre každý vstup správnu odpoveď, v druhom rade aby bol čo najrýchlejší a mal čo najmenšie pamäťové nároky.

Riešenie úlohy pozostáva z programu a popisu použitého algoritmu. V zadaní vždy uvedieme, koľko bodov sa dá získať za program a koľko za popis; výsledný počet bodov za úlohu je súčtom týchto dvoch hodnotení.

Váš program hneď po odovzdaní automaticky otestujeme na viacerých vopred pripravených vstupoch. Body za neho vám pridelíme podľa toho, na koľkých vstupoch dá správnu odpoveď v časovom limite. Len čo sa program dotestuje, dozviete sa výsledok. Ak ste nezískali plný počet bodov, môžete program vylepšiť a odovzdať ho znova. Podrobnejšie informácie o odovzdávaní programov nájdete na našej webstránke.

Popis algoritmu by mal byť natoľko podrobný a zrozumiteľný, aby bolo možné podľa neho napísať program rovnako efektívny, ako ten váš. Ďalej vyžadujeme odhad časovej a pamäťovej zložitosti a zdôvodnenie (ak je to potrebné, aj dôkaz) správnosti algoritmu.

Ak vo svojom riešení používate zložitejšie dátové štruktúry (napríklad haldu, nie obyčajné pole), musíte popísať aj ich implementáciu. To platí aj v prípade, že ich váš programovací jazyk už obsahuje a vy ste ich neimplementovali. Ak si nie ste istí, či niečo môžete použiť bez popisu, radšej to popíšte, prípadne sa spýtajte vo fóre na stránke.

Popis odovzdávajte vo formáte PDF alebo ako plain text. Hodnotíme hlavne korektnosť algoritmu a v druhom rade jeho efektivitu. Získaný počet bodov sa dozviete, keď vaše riešenie po termíne odovzdania opravíme.

Zopár ukážkovo vyriešených starších úloh nájdete na stránke `http://www.ksp.sk/wiki/Seminar/Riesenie`. Na tomto mieste sa tiež môžete dočítať, čo je vlastne časová a pamäťová zložitosť (ak vám tieto pojmy veľa nehovoria).

Ktoré kategórie môžem riešiť?

Kategóriu Z môžu riešiť:

- tretiaci a štvrtáci[1], ak do začiatku príslušného polroka neboli na sústredení KSP,
- druháci a mladší, ak do začiatku príslušného polroka boli najviac na jednom sústredení KSP.

Kategórií O a T sa môžu zúčastniť všetci stredoškoláci (a mladší) bez obmedzenia. Ak vám to pravidlá dovoľujú, môžete riešiť aj viacero kategórií naraz.

Čo je na kategórii T iné?

Keď sa vám zdá, že príkladov je málo, chcete sa naučiť viac, osvojiť si nové finty a vyskúšať si naprogramovať niečo, čo ste doteraz neskúšali, táto kategória je akurát pre vás.

Úlohou tejto kategórie je aj celoročná príprava riešiteľov na medzinárodné súťaže. Jej víťaza čaká večná sláva a navyše **body z kategórie T budú mierne zohľadnené pri výbere reprezentácie na Medzinárodnú olympiádu v informatike**. (Tento výber má formu týždňového sústredenia, na ktoré sú pozvaní najlepší riešitelia celoštátneho kola Olympiády v informatike, kat. A.)

Na rozdiel od kategórií Z a O, **v kategórii T nemusíte písať popis riešenia, odovzdávate len program. Zadania kategórie T už nenájdete v papierovej podobe, ale len na internetovej stránke** `http://www.ksp.sk/wiki/Zadania/Zadania`**.**

Na našej stránke sa dozviete aj viac o tejto kategórii.

Registrácia

Pred odovzdaním elektronického riešenia je potrebné zaregistrovať sa na našej webstránke a vyplniť požadované kontaktné údaje. Odporúčame sa zaregistrovať pár dní pred dňom, kedy chcete odovzdať vaše riešenie (pre prípad, že by ste mali počas registrácie nejaké problémy).

Účasťou v KSP nám dávate súhlas spracovať a archivovať údaje, ktoré nám poskytnete pri registrácii, ako aj zverejniť vaše meno, školu, ročník a získané body vo výsledkovej listine.

[1] Za štvrtákov považujeme študentov, ktorí maturujú v tomto školskom roku; tretiaci sú tí, ktorí budú maturovať budúci školský rok; ostatné ročníky analogicky.

Korešpondenčný seminár z programovania
XXX. ročník, 2012/13
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Úlohy 1. kola zimnej časti

Termín odoslania riešení tejto série je pondelok **22. októbra 2012**.

1. Za mreže s nimi!

kat. Z; 7 b za popis, 3 b za program

Náš nový minister školstva dostal opäť vynikajúci nápad! Aby nám deti nerástli pre kriminál, ukáže im, aký smutný život je v Leopoldove. A teraz sa na návšteve vo väzení strieda jedna trieda za druhou. Prehliadky ciel, besedy s vrahmi, aj nové kamarátstva nadviazané v jedálni...

Čas od času sa však stane menšia nehoda. Ono sa sem-tam, a tým sem-tam myslíme vždy, stane, že sa žiaľbohu, a niekedy je to aj chvalabohu, že žiaľbohu, práve jedno dieťa stratí. A potom je pán minister smutný. V takýchto chvíľach je potrebný zásah dozorcov.

Našťastie je vstup do väzenia dobre pokrytý kamerovým systémom. Softvér tohto systému dokonca dokáže spoľahlivo rozoznávať tváre a má úžasnú hešovaciu funkciu, ktorá každej tvári priradí unikátny číselný identifikátor.

Vedenie väzenia by teraz potrebovalo program, ktorý prezrie záznamy a zistí, ktoré že to dieťa počas návštevy záhadne zmizlo.

Úloha

Na vstupe dostanete dve postupnosti čísiel. Čísla sa môžu v postupnostiach aj opakovať. Existuje práve jedno číslo, ktoré sa nachádza v prvej postupnosti a nenachádza v druhej. Nájdite ho. Môžete predpokladať, že v druhej postupnosti nie sú pridané žiadne nové čísla.

Formát vstupu

V prvom riadku sú čísla n a m, udávajúce dĺžky oboch postupností. V druhom riadku nasledujú medzerami oddelené čísla $a_1, a_2, \ldots, a_n$ – identifikátory detí, ktoré kamera zaznamenala počas príchodu. V treťom riadku sú zase čísla $b_1, b_2, \ldots, b_m$ – identifikátory detí nahraných počas odchodu.

Pre dĺžky n a m platí $n \geq 1$, $m \geq 0$ a $n, m \leq 100\,000$. Identifikátory detí sú celé čísla z rozsahu od 1 do 10^9.

(Dva body za program a najviac päť za popis dostanete, ak bude fungovať pre $n, m \leq 1000$ a čísla z rozsahu od 1 po 1000. Každé riešenie, ktoré dostane tri body za program, môže dostať aj sedem bodov za popis – nie je teda nutné mať úplne optimálnu časovú zložitosť.)

Formát výstupu

Vypíšte jeden riadok a v ňom jedno celé číslo. Toto číslo sa musí nachádzať v prvej postupnosti a nesmie nachádzať v druhej.

Príklad

vstup

```
7 3
38 203 205 38 291 205 38
203 291 38
```

výstup

```
205
```

2. Zahrajme si!

kat. Z; 7 b za popis, 3 b za program

V KSPárni je mnoho vecí a medzi nimi má svoje čestné miesto aj KSPácky synťák. Má už svoje roky a naraz zvláda zahrať len jeden tón. Jano sa rozhodol, že preň spíše zbierku skladieb, a zaumienil si, že tieto skladby sa budú hrať len jednou (pravou) rukou. Jedna z otázok, ktorú Jano pri písaní skladieb rieši, je určenie ich náročnosti.

Aby Jano mohol dať svojej tvorivosti čo najväčší priestor, zišiel by sa mu na určenie náročnosti skladby nejaký program, ktorý by to robil za neho.

Úloha

Synťák má k klávesov očíslovaných 1 až k zľava doprava. Hráčova pravá ruka je široká ako ℓ klávesov. Pokiaľ má teda hráč palec na i-tom klávese, dokáže stlačiť len klávesy na pozíciách $i, i+1, \ldots, i+\ell-1$. Ak by chcel hráč zahrať iný kláves, musí pohnúť rukou. Na začiatku skladby je ruka vždy položená na klávesoch 1 až ℓ.

Počas hrania skladby je možné kedykoľvek pohnúť rukou do ľubovoľnej strany o ľubovoľný počet klávesov. Súčet vzdialeností, o ktoré sa ruka pohla počas hrania, nazvime *náročnosť zahrania*. *Náročnosť skladby* je potom najmenšia *náročnosť zahrania* pre všetky možné spôsoby, akými sa dá skladba zahrať.

Napíšte program, ktorý pre danú skladbu zistí jej náročnosť, teda najmenšiu vzdialenosť, o ktorú sa musí dokopy pohnúť ruka počas jej hrania.

V popise nezabudnite aj zdôvodniť správnosť vášho programu, napríklad prečo sa nedá skladba zahrať s menšou náročnosťou, ako vypíše váš program.

Formát vstupu

Na prvom riadku sú medzerami oddelené čísla n, k a ℓ, pričom $1 \leq n \leq 500\,000$ a $1 \leq \ell \leq k \leq 10^9$.

Nasleduje n riadkov s notami Janovej skladby v poradí, v akom ich treba zahrať. Nota je určená číslom klávesu, ktorý treba na KSPáckom synťáku stlačiť, čiže sú to celé čísla v rozsahu 1 až k.

Formát výstupu

Na výstupe bude jedno celé číslo – celková vzdialenosť, ktorú musí ruka prejsť na zahranie skladby na vstupe. Uvedomte si, že výsledok sa nemusí vždy zmestiť do 32-bitovej premennej.

Príklady

vstup

```
5 8 4
1
2
3
4
5
```

výstup

```
1
```

Celá skladba sa nedá zahrať bez pohybu. Na jeden pohyb to však už ide – napríklad po prvom tóne posunieme ruku o jeden kláves doprava.

vstup

```
3 5 1
1
5
3
```

výstup

```
6
```

Po prvom tóne posunieme ručičku o 4 klávesy doprava, po druhom tóne o 2 klávesy späť.

3. Záhadné umenie

kat. Z; 7 b za popis, 3 b za program

René si rád kreslí a jedného dňa sa rozhodol, že vyskúša kresliť v novom umeleckom smere, *trojuholníkoizme*. *Trojuholníkoizmus* je vysoko abstraktné umenie a kresby sa skladajú len z bodov, ktoré navyše musia byť umiestnené v presnej mriežke. To však nie je všetko. Cieľom tohto smeru je pomocou bodov vytvoriť pravouhlé trojuholníky a rátajú sa len tie, ktoré majú odvesny rovnobežné so súradnicovými osami.

Tieto trojuholníky tam, samozrejme, nie sú zakreslené a skúsený umelecký kritik si ich musí vedieť predstaviť. René je však ešte len začiatočník a občas sa v tých trojuholníkoch stratí. Rád by vedel, koľko trojuholníkov sa nachádza v jeho obraze, aby si vedel overiť, či našiel všetky.

Úloha

Daných je n bodov v mriežke. Napíšte program, ktorý spočíta, koľkými spôsobmi vieme vybrať tri body tak, aby tvorili pravouhlý trojuholník s odvesnami, ktoré sú rovnobežné so súradnicovými osami.

Formát vstupu

Prvý riadok vstupu obsahuje celé číslo n – počet bodov ($3 \leq n \leq 100\,000$). Nasledujúcich n riadkov obsahuje celé čísla $x_i,\ y_i$ ($1 \leq x_i,\ y_i \leq 100\,000$) udávajúce súradnice i-teho bodu. Všetky body sú navzájom rôzne.

Formát výstupu

Vypíšte jedno číslo – počet trojuholníkov.

Príklad

vstup

```
6
10 10
20 10
10 20
20 20
30 20
30 30
```

výstup

```
8
```

4. Závody korytnačiek

kat. Z a O; 10 b za popis, 5 b za program

Na poslednom sústredení KSP sa počas Náboja hrala zaujímavá hra. Táto hra sa v češtine volá Želví závody;[2] ak by ste mali záujem dozvedieť sa o nej viac, nájdete ju pod týmto názvom alebo pod názvom Schildkrötenrennen.

Pre úžasnosť tejto hry by ju chceli vedúci niekedy zopakovať v novej monumentálnej verzii, ktorá sa od pôvodnej trocha líši, napríklad aj tým, že je oveľa väčšia a je v nej oveľa viac korytnačiek. V tejto hre má každý hráč svoju figúrku korytnačky a snaží sa ju dostať ako prvý k šalátu.

Hracia plocha pozostáva z niekoľkých políčok, v každom momente hry je na každom políčku niekoľko (nula až všetky) korytnačiek. Keď je na políčku viac korytnačiek ako jedna, tak sú korytnačky poukladané na sebe. Na začiatku hry sú všetky korytnačky na prvom políčku, poukladané v nejakom poradí.

Originalita hry spočíva v spôsobe, akým sa korytnačky pohybujú.

Hráči môžu používaním kariet hýbať korytnačkami. Presnejšie, hráč za jedno kolo pohne jednou korytnačkou o niekoľko políčok dopredu alebo dozadu. A samozrejme, keď sa korytnačka pohne, zoberie so sebou aj všetky korytnačky, ktoré na nej sedia. Keď sa má korytnačka pohnúť na políčko, kde už sú nejaké korytnačky, sadne si tej najvyššej na chrbát.

Príklad: Predstavme si, že máme na prvom políčku 1. korytnačku, na nej 2. a na nej položenú 5. korytnačku. Na treťom políčku je korytnačka č. 4. Pohneme 2. korytnačkou o dve políčka dopredu. Potom na prvom políčku ostane iba 1. korytnačka a na treťom políčku bude 4., na nej 2. a na nej 5. korytnačka (ktorú 2. korytnačka odniesla so sebou). Nie je možné napríklad presunúť 2. korytnačku bez toho, aby sa presunula 5. korytnačka.

Voľakedy sa všetky tieto presuny robili ručne, ale pri novej monumentálnej verzii čosi také nie je možné. Teraz sa jednoducho nahádžu potrebné údaje do počítača, ktorý situáciu vyhodnotí. A samozrejme, program, ktorý to bude vyhodnocovať, napíšete vy.

Úloha

Na vstupe dostanete zadané poradie korytnačiek na začiatku hry a postupnosť ťahov. Ťahy budú tvaru „zober korytnačku s číslom k_i, ktorá je na políčku p_i, a pohni ju o t_i políčok dopredu“. Pohyb dopredu o $-x$ je pohyb dozadu o x. Vypíšte, ako bude vyzerať hracia plocha po odohraní týchto ťahov.

Korytnačiek je K a sú očíslované prirodzenými číslami od 1 po K. Políčok je P a sú očíslované číslami od 1 po P. Počet ťahov je T.

Formát vstupu

Na prvom riadku vstupu sú tri celé čísla K, P a T ($1 \leq K, P, T \leq 200\,000$).

Na druhom riadku je K čísel od 1 po K, každé práve raz. i-te z týchto čísel hovorí, ktorá korytnačka je i-ta zospodu.

Nasleduje T riadkov popisujúcich jednotlivé ťahy. Na každom sú tri celé čísla k_i, p_i a t_i ($1 \leq k_i \leq K; 1 \leq p_i \leq P; 0 \leq |t_i| < P$), čo znamená, že korytnačka s číslom k_i na políčku p_i sa pohne o t_i políčok dopredu (a samozrejme so sebou zoberie aj všetky korytnačky na jej chrbte). Môžete predpokladať, že po každom ťahu budú všetky korytnačky na políčkach od 1 po P, a že p_i je naozaj políčko, na ktorom sa korytnačka k_i nachádza.

Formát výstupu

Na výstup vypíšte P riadkov, každý pre jedno políčko.

Na začiatku i-teho riadku bude počet korytnačiek n_i na i-tom políčku. Za ním bude nasledovať n_i čísel oddelených medzerami. j-te z nich musí byť číslo j-tej korytnačky zospodu.

[2] a nie železničná závora, ako si niektorí ľudia vysvetľovali skratku žel. záv.

Príklad

vstup

```
5 5 6
1 2 5 4 3
5 1 2
1 1 3
4 3 -1
5 3 1
3 2 1
2 4 -2
```

výstup

```
0
3 4 2 5
1 3
1 1
0
```

5. Oko kukne, oko vidí (časť prvá)

kat. Z a O; 0 b za popis, 16 b za program

„Oko kukne, oko vidí! Oko nie je buldozér!“ hovorí známa ľudová múdrosť.

No keď je toho tak veľa, že oko nevidí, treba použiť buldozér. Toto je obzvlášť pravda, ak vyhľadávame niečo v počítači. A tým buldozérom, ktorý pri vyhľadávaní pomôže, sú *regulárne výrazy*.[3] V tomto zadaní si popíšeme základy postačujúce na riešenie zadaných úloh. Pokojne si toho ale o regulárnych výrazoch prečítajte aj viac – možno nájdete niečo, čo vám riešenie úloh uľahčí, a takmer určite sa vám získané vedomosti ešte budú hodiť v praxi.

Objekty, medzi ktorými budeme vyhľadávať, budú reťazce znakov. Vyhľadávať budeme tak, že napíšeme regulárny výraz – teda nejakú *vzorku* (pattern). Vzorka popisuje, ako vyzerajú reťazce, ktoré nás zaujímajú. Presnejšie, každej vzorke zodpovedá (matches the pattern) nejaká množina reťazcov.

Základné použitie v praxi vyzerá nasledovne: Máme nejaký veľký textový súbor. Z neho nás zaujímajú len niektoré riadky. Napríklad si pamätáme, že nám Fero niečo písal o sýkorkách, ale stratilo sa to niekde medzi megabajtmi iných logov z jabberu. Chceli by sme teda v danom súbore nájsť tie riadky, ktoré majú správny tvar – sú od Fera a píše sa v nich o sýkorkách. Umenie je potom ukryté v nasledujúcom kroku: na základe tejto predstavy napísať takú vzorku, ktorej budú zodpovedať práve tie riadky, ktoré nás zaujímajú, a žiadne iné. V našom príklade by sme si napr. povedali, že chceme riadky, ktoré začínajú „`Fero:`“ a obsahujú niekde podreťazec „`sykork`“. Tomuto zodpovedá vzorka „`^Fero:.*sykork`“.

Teraz si popíšeme základy tvorby vzoriek – teda povieme si, z čoho sa taká vzorka môže skladať a ktoré reťazce jej potom zodpovedajú.

- Základným stavebným kameňom vzoriek sú samotné znaky. Vzorke tvorenej jediným znakom zodpovedá reťazec tvorený dotyčným znakom a nič iné.
- Základná operácia so vzorkami je zreťazenie. Keď napíšeme dve vzorky za seba, dostaneme novú vzorku. Tej zodpovedajú reťazce zložené z dvoch častí, pričom prvá zodpovedá prvej vzorke a druhá druhej. Zreťaziť samozrejme môžeme ľubovoľne veľa vzoriek. Napr. vzorke „`jablko`“ zodpovedá len reťazec „`jablko`“ a nič iné.
- Vzorky môžeme uzatvárať do obyčajných zátvoriek. Napr. vzorke „`(jablko)`“ zodpovedá len reťazec „`jablko`“ a nič iné.
- Logický or je označovaný symbolom „`|`“. Ak ním spojíme dve vzorky, dostaneme novú, ktorej zodpovedajú aj reťazce zodpovedajúce prvej, aj reťazce zodpovedajúce druhej vzorke. Napr. vzorke „`jabl(k|ck)o`“ zodpovedajú reťazce „`jablko`“ a „`jablcko`“.
- Inú možnosť, ako dať vo vzorke na výber, predstavuje množina znakov, ktoré sa na danom mieste môžu vyskytnúť. Tú zapisujeme „`[znaky]`“. Napríklad vzorke „`jablk[oa]`“ zodpovedajú reťazce „`jablko`“ a „`jablka`“.

  Najvšeobecnejšia množina znakov sa skrátene zapisuje „`.`“. Teda vzorke „`j....a`“ zodpovedá každý 6-znakový reťazec začínajúci na j a končiaci na a.

[3] Presnejšie pôjde o tzv. *rozšírené regulárne výrazy podľa štandardu POSIX*. Viac o nich si môžete začať čítať napr. na `http://en.wikipedia.org/wiki/Regular_expression#POSIX_Extended_Regular_Expressions`. Regulárnych výrazov existuje viacero odrôd. Všetky sú si veľmi podobné, líšia sa ale napríklad drobnými detailmi v syntaxi, preto sme si v zadaní úlohy radšej presne zvolili jednu z existujúcich odrôd.

Zložitejšie množiny znakov môžeme zapísať pomocou rozsahov, napr. „`[a-zA-Z0-9]`“ je ľubovoľné jedno písmeno alebo číslica. Ak popis množiny znakov začína znakom „`^`“, znamená to negáciu – teda danej množine zodpovedajú všetky znaky okrem vymenovaných. Napríklad vzorke „`[a-z][^a-zA-Z]`“ zodpovedajú všetky dvojznakové reťazce, ktorých prvý znak je malé písmeno a druhý znak nie je ani malé, ani veľké písmeno.

- Ak chceme, aby sa mohla časť vzorky v reťazci zopakovať, použijeme *kvantifikátor*. Základné kvantifikátory sú „`?`“ (nulakrát alebo jedenkrát) a „`*`“ (ľubovoľne veľakrát, vrátane nulakrát).

  Napríklad vzorka „`jablc?ko`“ je ekvivalentná so vzorkou „`jabl(k|ck)o`“. Vzorke „`pe*s`“ zodpovedajú okrem iného reťazce „`ps`“, „`pes`“ a „`peeeees`“. Vzorke „`(ab)*c`“ *nezodpovedá* reťazec „`aabbc`“, len reťazce „`c`“, „`abc`“, „`ababc`“, atď.

  Pri opakovaní vzorky jej nemusí zakaždým zodpovedať ten istý reťazec. Napríklad vzorke „`j.*a`“ zodpovedá nielen reťazec „`jxxxa`“, ale aj reťazec „`jahoda`“.

  Vzorke „`<[^>]*>`“ zodpovedá každý XML tag – reťazec začína znakom „`<`“, za ním nasleduje ľubovoľne veľa znakov iných od „`>`“ a za nimi nasleduje znak „`>`“.

- Regulárne výrazy budeme v prvom rade používať na vyhľadávanie v texte. Pri tom niekedy môžeme chcieť povedať, že sa daná vzorka musí vyskytnúť na špecifickom mieste. Na to slúžia napríklad symboly „`^`“ (začiatok reťazca), „`$`“ (koniec reťazca) a reťazec „`\b`“ (hranica slova, pričom slovom sa rozumie najdlhší podreťazec zodpovedajúci vzorke „`[a-zA-Z0-9_]*`“).

  Teda napr. vzorku „`\bje\b`“ by sme v reťazci „`dnes je teplo`“ našli, ale v reťazci „`chutne jedlo`“ nie.

- Prvých 9 zátvoriek použitých vo vzorke dostane svoje poradové číslo.[4] Vo zvyšku vzorky sa potom pomocou reťazca „`\n`“ (kde `n` je od `1` po `9`) môžeme odvolávať na presný reťazec, ktorý zodpovedal vzorke v zátvorke. (Takéto odkazovanie sa zvykne nazývať *spätná referencia*.)

  Teda napríklad vzorke „`(.)\1`“ zodpovedajú práve reťazce tvorené dvoma rovnakými znakmi.

Vyhľadávanie pomocou regulárnych výrazov podporuje veľa programov, napríklad aj OpenOffice Writer (v rozšírenom dialógu Find&Replace) alebo editor `vim` (normálne pri hľadaní cez príkaz „`/`“). Na jednoduché vyhľadávanie v textových súboroch slúži program `grep`. Ten[5] budeme používať aj na testovanie vašich riešení tejto úlohy. Najjednoduchší spôsob použitia programu `grep` je dať mu ako parametre vzorku a zoznam súborov. Program `grep` z každého súboru vypíše tie riadky, v ktorých sa niekde vyskytol podreťazec zodpovedajúci vzorke. Tu je jednoduchý príklad použitia:

```
$ egrep 'dynamick' *
prikl10-2.tex:pre celú rovinu dynamickým programovaním, ktoré políčko je ako zasiahnuté. Ale
prikl8.tex:Riešenie založíme na dynamickom programovaní. Študentov a~jedlá si označíme
prikl8.tex:Opäť existuje jednoduchý algoritmus riešiaci túto úlohu založený na dynamickom
prikl9-1.tex:spôsob, ako vygenerovať hľadaný zápis, obrátime sa na dynamické programovanie.

$ egrep '^\\komentar.*opravovala' *
prikl1.tex:\komentar{Zablatená cesta}{opravovala Dominika}
prikl4.tex:\komentar{Zapeklitý Top Ológ}{opravovala Halucinka}
```

Úloha

Vašou úlohou bude napísať niekoľko regulárnych výrazov, ktoré v danom texte vyhľadajú všetky požadované riadky. Odovzdajte obyčajný textový súbor obsahujúci presne 8 riadkov, pričom v i-tom riadku bude riešenie i-tej z nasledujúcich podúloh.

Zostrojte regulárny výraz, pre ktorý program `grep` vypíše všetky riadky, v ktorých...

1. ... je druhé písmeno „`a`“.
2. ... sa písmeno „`a`“ nevyskytuje vôbec.
3. ... sa písmeno „`a`“ vyskytuje aspoň 3×.
4. ... sa písmeno „`a`“ vyskytuje presne 3×.
5. ... niektoré slovo začína písmenom „`a`“ alebo „`A`“.
6. ... sú aspoň dva znaky a prvý znak je rovnaký ako posledný.
7. ... sa niektoré slovo zopakuje.
8. ... sa vyskytuje (možno aj 1-znakové) slovo začínajúce aj končiace tým istým znakom.

[4] Syntax popisovaná v tomto bode zadania nie je súčasťou štandardu POSIX, ide ale o rozšírenie podporované v skoro všetkých moderných odrodách regulárnych výrazov.

[5] Presnejšie, `grep` s prepínačom `-E`, resp. `egrep`, ktoré používajú našu odrodu regulárnych výrazov.

Vo všetkých podúlohách si opäť pod slovom predstavujeme najdlhší podreťazec tvorený veľkými a malými písmenami, číslicami a podčiarkami (_).

Vaše riešenia budú testované na textovom súbore obsahujúcom slovenský text bez diakritiky (teda všetky znaky sú zo 7-bitového ASCII). Tu je ukážka z neho:

```
ako novu hodnotu  vk  otca a odpocitat ju od vysiek synov. Zaroven aktualizujeme
 cas nasledovnym sposobom. Ak ma niektory zo synov hodnotu   vk ge1 , tak vdaka
prvej podmienke maju vsetky jeho policka vysku aspon 1, a teda do hodnoty
 cas  otca prispieva celkovym poctom policok, ktore obsahuje. Ak ma niektory syn
hodnotu   vk 0 , nema jeho  vk  vplyv na vysku policok v jeho podstrome. Do
hodnoty  as  otca teda prispieva svojou hodnotou  as.
```

Príklad

Keby zadanie obsahovalo jedinú podúlohu „všetky riadky, ktoré začínajú veľkým písmenom a končia bodkou“, jedným zo správnych riešení by bolo odovzdať súbor obsahujúci jeden riadok s textom „`^[A-Z].*[.]$`“.

Iným, tiež správnym riešením, by bolo „`^[A-Z].*\.$`“. Všimnite si spätné lomítko pred poslednou bodkou. Spätné lomítko slúži ako tzv. únikový znak (escape character) – keď potrebujeme napísať znak, ktorý by mal nejaký špeciálny význam, ako napríklad bodku alebo zátvorku, vieme to spraviť tak, že pred ním použijeme spätné lomítko.

6. Odmena za lenivosť

kat. O; 13 b za popis, 7 b za program

Na poslednom stretnutí vedúcich KSP sa rozhodlo, že sa do T2-ky donesie veľká biela tabuľa, na ktorú sa bude značiť, kto bol ako produktívny. Každý KSP-ák má teraz vlastný riadok, kde si každý deň pripíše jedno celé číslo vyjadrujúce mieru toho, čo v ten deň vykonal. KSP-áci sú veľmi zodpovední a objektívni ľudia a preto vedia presne odhadnúť svoj výkon, aj keby mal byť záporný (to v prípade, že celý deň len spali a chrápaním vyrušovali zvyšných vedúcich). Ale aj záporná produktivita sa u nás cení. Veď keď je niekto lenivý, znamená to, že ostatní museli byť produktívni, no nie?

Hlavný vedúci Jano sa jedného dňa rozhodol, že ide zistiť, kto bol najužitočnejší, aby ho mohol náležite odmeniť. Každý KSP-ák musí Janovi nahlásiť, za ktorý súvislý úsek ℓ dní chce byť hodnotený. Jano potom sčíta produktivitu v nahlásenom úseku a absolútnu hodnotu súčtu prehlási za užitočnosť tohoto KSP-áka.

Všimnite si, že aj veľmi lenivý vedúci má veľmi vysokú užitočnosť, lebo svojou lenivosťou pobáda iných k lepším výkonom; týmto by som sa chcel za celé KSP poďakovať Luxuskovi za dobre odvedenú prácu.

Prichádza deň veľkého zúčtovania a Hermi sa rozhodol, že bude trochu švindľovať, aby vyhral hlavnú cenu: sladkú limonádu.[6] Chce sfalšovať niektoré hodnoty produktivity vo svojom riadku. Aby to však nebolo príliš podozrivé, bude meniť len znamienka a aj to spraví najviac k-krát. Keďže však okrem toho musí ešte behať za dievčatami, poprosil vás, aby ste mu pomohli. A to čo najrýchlejšie.

Úloha

Na vstupe dostanete postupnosť celých čísiel a hodnoty ℓ a k. Nájdite takú súvislú podpostupnosť dĺžky ℓ, že po prenásobení **najviac** k jej prvkov číslom -1 dostaneme v absolútnej hodnote najväčší možný súčet. Tento súčet vypíšte.

Formát vstupu

Na prvom riadku sú čísla n, ℓ ($1 \leq \ell \leq n \leq 10^5$) udávajúce dĺžku celej postupnosti a dĺžku hľadanej podpostupnosti. Na druhom riadku je n celých čísiel, každé v rozsahu od -10^9 po 10^9. Posledný riadok obsahuje číslo k ($0 \leq k \leq 10^9$) – maximálny počet prvkov, ktoré môžeme prenásobiť číslom -1.

Formát výstupu

Vypíšte jedno číslo – maximálnu užitočnosť, ktorú vie Hermi takto dosiahnuť.

Príklad

vstup

```
5 3
0 -2 3 -5 1
2
```

výstup

```
10
```

Môžeme zmeniť znamienko číslam -2 a -5, alebo číslu 3. Tak či tak získame úsek s užitočnosťou 10.

[6] Ó, sladkú limonádu.

7. Obodované preteky mravcov

kat. O; 12 b za popis, 8 b za program

KSPáci postavili pre mravce prekážkovú dráhu. Tá vyzerá ako mriežka z $r \times c$ políčok, kde niektoré políčka sú voľné a na niektorých sú prekážky. Mravce lezú z ľavého horného rohu mriežky do pravého dolného, pričom sa vždy pohnú buď o jedno políčko nadol alebo doprava (a nikdy nevstúpia na políčko s prekážkou).

Cesta mravca je obodovaná nasledovne: Na začiatku má mravec 0 bodov. Keď sa pohne, počet jeho bodov sa zdvojnásobí. Navyše ak sa pohol doprava, tak sa k jeho počtu bodov pripočíta ešte jeden. Takže napríklad mravec, ktorý pôjde doprava, dole a doprava, získa: $((0 \cdot 2 + 1) \cdot 2) \cdot 2 + 1 = 5$ bodov.

Úloha

Na vstupe bude zadaná mriežka a vašou úlohou bude odpovedať na niekoľko otázok tvaru: Pre zadané číslo k vypíšte cestu mravca, za ktorú získa k-ty najmenší počet bodov.

Formát vstupu

V prvom riadku vstupu sú čísla r, c, q ($2 \leq r, c, q \leq 1\,000$). V ďalších r riadkoch nasleduje popis mriežky. Každý riadok obsahuje c čísel, kde 0 znamená voľné políčko a 1 znamená políčko s prekážkou. Môžete predpokladať, že ľavý horný a pravý dolný roh budú vždy voľné. V ďalších q riadkoch sú otázky. Každá otázka pozostáva z jedného kladného čísla menšieho ako 10^{17}.

Formát výstupu

Pre každú otázku vypíšte jeden riadok. Tento riadok má obsahovať reťazec „`neexistuje`“, ak daná cesta neexistuje, alebo reťazec zo znakov `P`, `D`, ktorý popisuje cestu mravca (`P` znamená vPravo, `D` znamená Dole).

Príklad

vstup

```
3 3 3
0 0 1
0 0 0
1 0 0
1
3
20
```

výstup

```
DPDP
PDDP
neexistuje
```

Možné cesty sú (v poradí podľa bodov): `DPDP`, `DPPD`, `PDDP`, `PDPD`.

8. Obdĺžnikové pečiatky

kat. O; 10 b za popis, 15 b za program

Jörg Gutenberg, mladší brat slávneho Johannesa, sa tiež pokúšal preraziť na európskom trhu so svojím vynálezom. Vymyslel jednoduchý prenosný variant kníhtlače určený do domácností, ktorý mal urýchliť zapisovanie pranostík, receptov, klebiet a iných foriem ľudovej slovesnosti.

Vynález pozostáva zo sady n pečiatok. Na každej pečiatke je vyrytý nejaký reťazec písmen, ktorý sa po namočení do farby dá odtlačiť na papier. Jednotlivé odtlačky sa môžu robiť aj jeden cez druhý – avšak len v prípade, že sa písmená v prekrytých častiach presne zhodujú. Napríklad pomocou pečiatok „`bankov`“ a „`kovy`“ sa dá vytvoriť nielen slovo „`bankovkovy`“, ale aj „`bankovy`“ (prekrytým, teda dvakrát odtlačeným, úsekom textu je „`kov`“).

Jörgov vynález trpí zjavným nedostatkom: S nevhodnou sadou pečiatok sa niektoré texty nedajú kompletne odtlačiť – zopár písmen musíme dopísať perom. Vždy sa však snažíme odtlačiť čo najviac písmen textu.

Úloha

Na vstupe je zadaný text a sada pečiatok. Každú pečiatku môžeme použiť ľubovoľne veľakrát. Pečiatky však nemožno otáčať ani rozpíliť.

Vypočítajte, najmenej koľko písmen textu sa nedá odtlačiť, a teda ich bude nutné dopísať perom.

Formát vstupu

Prvý riadok vstupu obsahuje neprázdny reťazec malých znakov anglickej abecedy – zadaný text. Jeho dĺžka nepresiahne 300 000. V druhom riadku je číslo n – počet pečiatok ($1 \leq n \leq 5\,000$). Nasledujúcich n riadkov obsahuje reťazce písmen na jednotlivých pečiatkach. Každý z reťazcov je neprázdny, pozostáva len z malých písmen anglickej abecedy a jeho dĺžka nepresiahne 5 000.

Formát výstupu

Na výstup vypíšte jeden riadok obsahujúci počet písmen textu, ktoré treba dopísať perom.

Príklady

vstup

```
bankovy
2
bankov
kovy
```

výstup

```
0
```

Vieme vyrobiť celý text, nič netreba dopísať.

vstup

```
rokokovy
2
oko
vyr
```

výstup

```
3
```

Druhú pečiatku nevieme použiť, lebo by nám vyrobila za koncom aj písmeno, ktoré už nechceme. Zato prvú môžeme použiť hneď dvakrát.

Zadania kategórie T

Nájdete ich na našej stránke `http://www.ksp.sk/wiki/Zadania/Zadania`. Nezabudnite sa na ne pozrieť, čaká na vás ďalších **päť** zaujímavých úloh rôznych obtiažností.

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09