PRASK
IV. ročník, 2017/18
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Úlohy 1. kola letnej časti

Termín odoslania riešení tejto série je pondelok **25. jún 2018**.

Teoretické úlohy

V tejto časti ťa čaká niekoľko matematickejších úloh, ktoré úzko súvisia s informatikou. Ako riešenie týchto úloh treba poslať podrobne spísaný postup toho, ako si riešil danú úlohu.

A ak by ťa to zaujímalo, podobné úlohy môžeš nájsť aj v Olympiáde v informatike, kategória B (http://oi.sk/archiv/2016/sl-2016-1-zad-B.pdf). Vrelo ti ju odporúčame riešiť tiež, naučíš sa veľa nových vecí a môžeš sa dostať aj na krajské kolo Olympiády.

1. Pekelné teplo

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Andrejovi na ajo@ksp.sk

Vedúci sa na leto rozhodli podnikať. Dohodli sa, že si každý založí vlastnú farmu a na nej bude čosi pestovať. Od hrachu a fazule až po kukuricu či repu. Leto sa blíži a oni vedia, že musia všetci poriadne polievať svoju úrodu. Vedúci však majú obmedzený rozpočet a vedia postaviť iba jednu spoločnú studňu na niektorej farme. Práve teraz sa snažia vybrať z niekoľkých kandidátov vhodnú farmu na výstavbu studne. Pri porovnávaní však potrebujú často vedieť odpoveď na otázku: "Ak sa bude studňa nachádzať na farme x, ako dlho potrvá, kým z nej voda dotečie na všetky zvyšné farmy?"

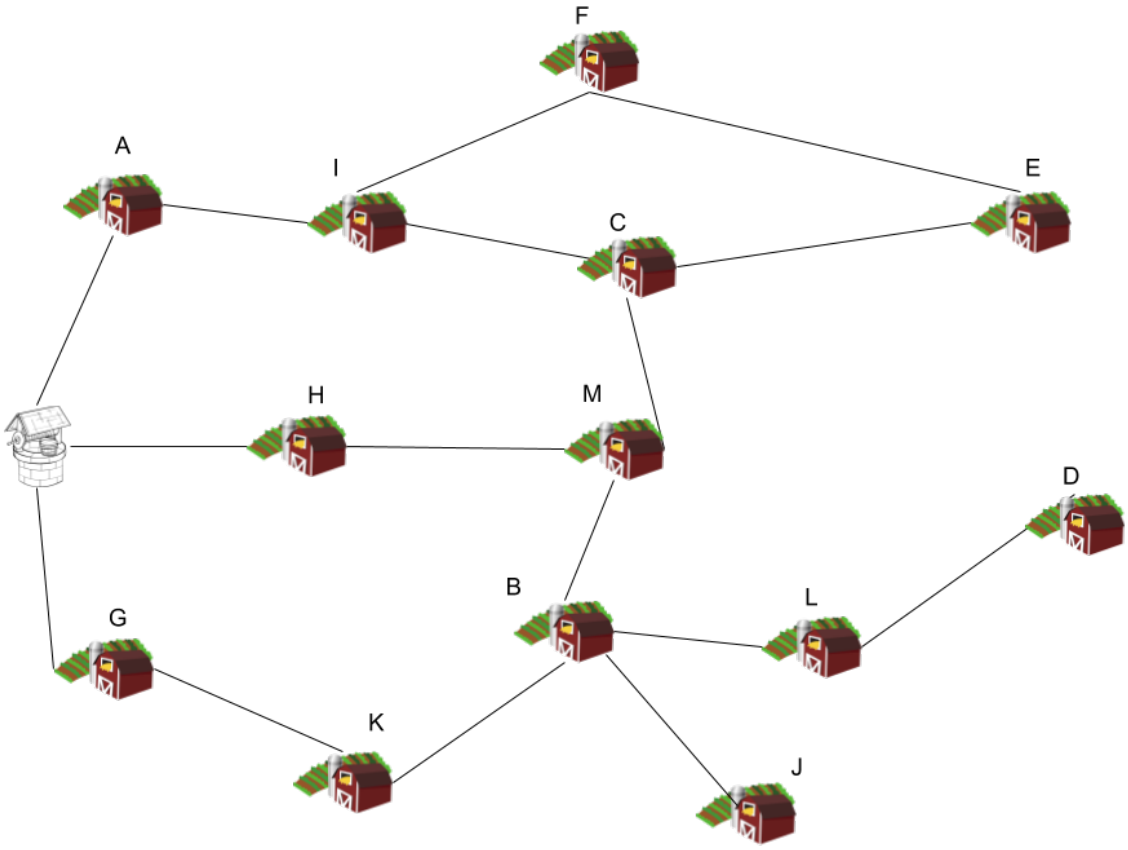
Úloha

Vašou úlohou je vyriešiť problémy súvisiace s touto otázkou. Ako zvyčajne, budete riešiť niekoľko podúloh zoradených podľa nami odhadovanej zložitosti:

a) (3 body) Na priloženej mape fariem zistite čas, ktorý bude vode trvať, pokým sa v prípade sucha dostane z vyznačeného miesta do jednotlivých miest. Pre tok vody platia nasledovné pravidlá:

   1. Ak sa voda nachádza na nejakej farme x, vie sa odtiaľ presúvať na ďalšie farmy pomocou potrubí, ktoré z farmy x vedú.
   2. Prejsť potrubím trvá vode presne jednu minútu. Pre jednoduchosť totiž predpokladáme, že potrubia sú rovnako dlhé (neriaďte sa tým ako vyzerajú na obrázku).
   3. Presun vody v rámci jednej farmy nezaberá žiaden čas. To znamená, že začať polievať alebo poslať vodu ďalším potrubím vieme v okamihu ako tam voda pritiekla.

   Počas riešenia úlohy nemusíte riešiť množstvo vody. Predpokladajte, že potrubia sú dostatočne veľké a keď na farmu dorazí voda, je jej dosť na to, aby ňou farmári polievali aj ju poslali do všetkých zvyšných potrubí.



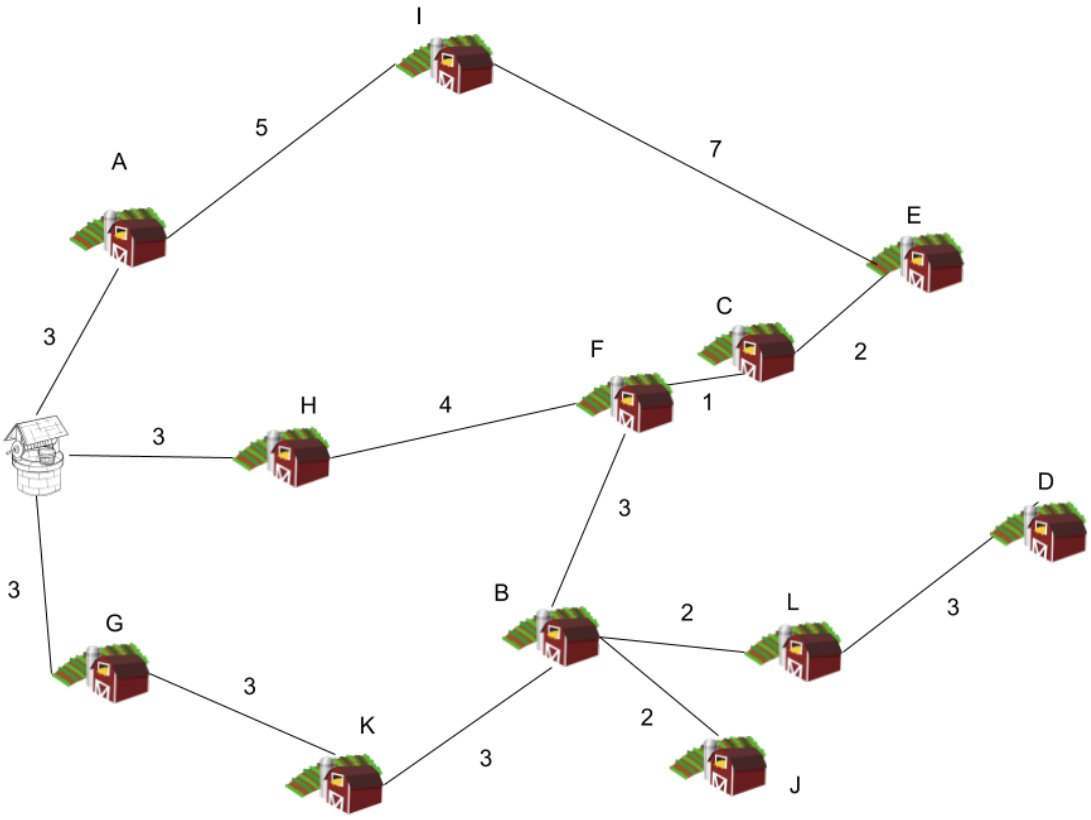
Na obrázku vidíte niekoľko fariem a jednu studňu. Čiary, spájajúce niektoré dvojice fariem, znázorňujú existenciu potrubí medzi nimi. Ak teda medzi farmami existuje potrubie, znamená to, že medzi nimi vie priamo tiecť voda. Písmená nad farmami predstavujú iba ich názvy, pomocou ktorých sa môžete na jednotlivé farmy odkazovať. Na obrázku napríklad vidíme, že voda vie za čas 3 dojsť napríklad aj na farmu F cez farmy A a I.

b) (3 body) Popíšte postup, ktorý ste použili na vyriešenie prvej podúlohy. Snažte sa, aby bol tento postup dostatočne všeobecný, nasledujúc váš postup by ste teda vedeli zistiť správnu odpoveď pre ľubovoľne vyzerajúcu mapu akéhokoľvek počtu fariem.

c) (5 bodov) Farmári si uvedomili, že im vaša predchádzajúca pomoc nestačí. Získali totiž lepší odhad o tom, ako dlho tečie voda jednotlivými potrubiami. Predsa len sú rôzne dlhé. Riešenie podúlohy b) teda zrazu nestačí.

Navrhnite postup, ktorý zistí, ako najrýchlejšie sa vie voda dostať zo studne na všetky farmy. Pritom predpokladajte, že ku každému potrubiu je priradené jedno kladné číslo – čas za ktorý voda prejde z jednej strany na druhú (viď obrázok). Tento postup by mal fungovať pre ľubovoľnú mapu fariem a dĺžky potrubí.

Za nájdenie správnej odpovede na nižšie uvedenom obrázku môžete získať 1 bod. Čísla pri potrubiach symbolizujú koľko času vode trvá, pokým prejde od jednej farmy k druhej.



Na obrázku znovu vidíme studňu a niekoľko fariem, pospájaných potrubiami. Čísla pri jednotlivých potrubiach určujú ich dĺžku. Teda voda sa vie dostať zo studne na farmy K za 3+3 minút cez farmu G.

d) (4 body) Studňa je aj vďaka vašej pomoci postavená. Na farme y však zrazu zistili, že potrebujú trochu viac vody. Starší farmári vám poradili, že to viete docieliť tak, že v potrubiach, ktorými voda preteká nastriedačku znížite a zvýšite tlak. To krátkodobo vytvorí podtlak a do cieľovej farmy sa dostane o niečo viac vody.

Vidíme však, že voda zrazu nemôže ísť potrubiami úplne bez obmedzení, aby tento postup fungoval, musí prejsť cez **párne** veľa potrubí. Vašou úlohou je nájsť takú postupnosť potrubí, že voda sa na farmu y dostane čo najrýchlejšie a pritom prejde cez párny počet potrubí.

Navrhnite postup, ktorý pre zadanú mapu fariem, dĺžky potrubí a farmu x kde je studňa a farmu y kde potrebujú viac vody nájde túto najrýchlejšiu postupnosť párneho počtu potrubí.

2. Prechádzka dobre uzátvorkovanými jaskyňami

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Žabovi na zaba@ksp.sk

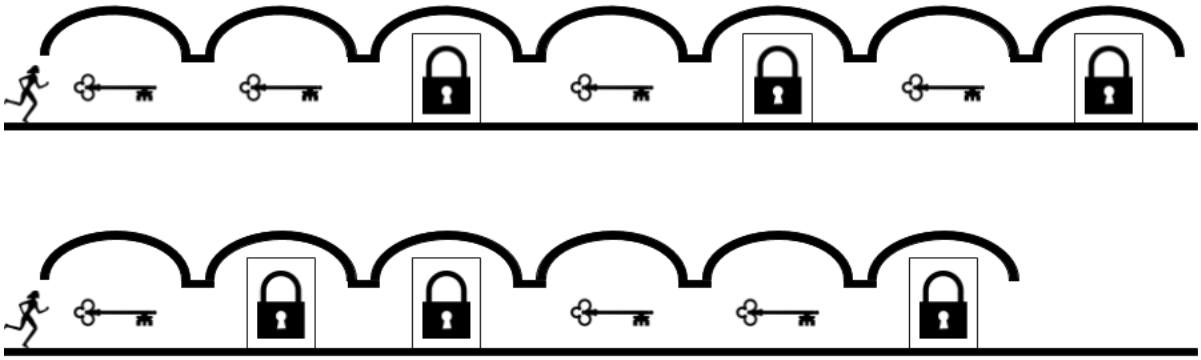
Úloha

Timka sa nedávno hrala svoju obľúbenú hru: Legend of Zelda. So svojou postavičkou sa pohybovala v podzemných jaskyniach (dungeonoch), zabíjala príšerky, zbierala mince a kľúče a pomocou kľúčov otvárala dvere do ďalších častí jaskyne.

V jednej chvíli sa ocitla na začiatku dlhočizného radu jaskyní. V každej z nich bol buď kľúč, ktorý mohla zobrať, alebo zamknuté dvere, ktoré bolo treba otvoriť na to, aby mohla pokračovať do nasledujúcej jaskyne. Kľúče, ktoré sa v tejto jaskyni nachádzali boli univerzálne, ale iba na jedno použitie. Každý kľúč teda vedel otvoriť ľubovoľné z dverí, mohol sa však použiť iba raz, po otvorení dverí zmizol. Aby to ale nebolo ťažké, Timka mohla niesť naraz ľubovoľne veľa kľúčov.

a) (3 body) Timka si na internete našla obrázok toho, ako vyzeral tento rad jaskýň – kde sa nachádzali kľúče a kde dvere. Zaujímalo by ju, či vie prejsť cez všetky tieto jaskyne, alebo je v hre naopak nejaká chyba a na koniec sa dostať nedá.

Popíšte postup fungujúci pre ľubovoľne vyzerajúci rad jaskýň, ktorý Timke pomôže zistiť, či vie prejsť cez všetky z nich. Dajte si pozor, aby tento postup skutočne fungoval pri ľubovoľnom počte jaskýň a ľubovoľnom rozmiestnení kľúčov a dverí. Takisto napíšte, prečo je váš postup správny.

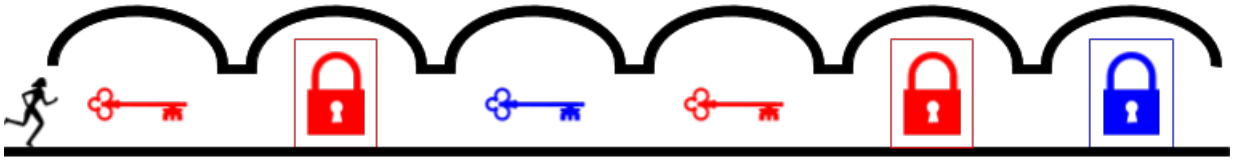


Na prvom obrázku sú jaskyne, cez ktoré Timka vie prejsť. Napríklad tak, že prvý kľúč nechá tak, zoberie až ten druhý, ten použije na otvorenie dverí v tretej jaskyni a následne už len bude zbierať kľúče a otvárať dvere. Mohla dokonca na začiatku zobrať aj ten prvý kľúč, aj keď by jej na nič nebol.

Jaskyňami na druhom obrázku Timka prejsť nevie. Nech spraví čokoľvek, ostane stáť pred druhými dverami bez kľúča.

b) (2 body) V ďalšej úrovni hry čakal na Timku ďalší rad jaskýň, situácia však bola o niečo komplikovanejšia. Dvere aj kľúče mohli mať dve farby – modrú a červenú, pričom modrý kľúč otváral iba modré dvere a červený kľúč iba čerevené. Timka teraz stojí pred tou istou otázkou. Dokáže prejsť cez všetky jaskyne?

Popíšte postup, ktorý Timke povie, či cez zadanú postupnosť jaskýň vie alebo nevie prejsť. V tomto prípade však už musíte počítať s dvoma typmi kľúčov a dverí. Odôvodnite správnosť vášho postupu.



Obrázok vyššie ukazuje, ako by mohla vyzerať takáto sústava jaskýň. Je jasné, že v tomto prípade by sa Timke podarilo úspešne prísť až do konca.

c) (3 body) Na ďalší deň v škole riešili problém dobrého uzátvorkovania. Úloha znie tak, že máme zadaný výraz, napríklad $((3+1)\cdot(7-3))$ a máme rozhodnúť, či je dobre uzátvorkovaný, teda či sa nám nejaká zátvorka nestratila, alebo nie je navyše. Napríklad vo výraze $(7+(3\cdot4)$ nám chýba jedna `)`, aj keď nevieme na ktorom miesta[1].

Aby sme sa zbytočne netrápili s číslami, uzátvorkovaný výraz si môžeme predstavovať iba ako postupnosť zátvoriek, teda prvý výraz by vyzeral `(()())` a druhý `(()`. Pri zostavovaní dobre uzátvorkovaných výrazov sa riadime troma pravidlami:

- () je dobre uzátvorkovaný výraz
- ak je x dobre uzátvorkovaný výraz, tak aj (x) je dobre uzátvorkovaný – toto je úplne klasické uzátvorkovanie, napríklad $4\cdot(3+2)$ vieme uzátvorkovať na $(4\cdot(3+2))$, ignorujúc čísla a znamienka, z `()` sme dostali `(())`.
- ak sú x a y dobre uzátvorkované výrazy, tak aj xy je dobre uzátvorkovaný výraz – dva dobre uzátvorkovné výrazy vieme priložiť k sebe, napríklad ak máme $(4+2)$ a $(3\cdot(2-1))$, tak vieme spraviť $(4+2)-(3\cdot(2-1))$. Z `()` a `(())` sme dostali `()(())`.

Keď sa Timka pozerala na tieto výrazy zátvoriek, veľmi jej to pripomínalo jej včerajšiu hru. Stačilo, keď si znak `(` predstavila ako kľúč a znak `)` ako dvere. A skutočne, tieto dva problémy boli rovnaké. Ak chcela zistiť, či je výraz dobre uzátvorkovaný, stačilo jej zistiť, či by prešla danou sústavou jaskýň.

[1] Aj výraz $(7)+(3\cdot4)$, aj $(7+(3\cdot4))$ sú totižto správne.

Teda skoro... Riešenie podúlohy a) malo predsa len jeden skutočne drobný nedostatok. Aký?

Pomôžte Timke nájsť taký rad jaskýň obsahujúcich kľúče a dvere jednej (čiernej) farby, že nimi všetkými dokáže prejsť, ale ak zmeníme každý kľúč na `(` a každé dvere na `)`, **nedostaneme** korektne uzátvorkovaný výraz. Upravte riešenie podúlohy a) tak, aby skutočne rozpoznávalo iba dobre uzátvorkované výrazy.

d) (3 body) Timka začala rozmýšľať nad tým, čo by sa stalo, keby jej výrazy mohli obsahovať dva typy zátvoriek – `(` ku ktorej patrí `)` a `[` ku ktorej patrí `]`. V jej hre s jaskyňami by zátvorky `(` a `)` boli modré kľúče a dvere a zátvorky `[` a `]` červené kľúče a dvere. Veľmi rýchlo však zistila, že jej riešenie podúlohy b) sa na tento problém použiť nedá.

Nájdite takú postupnosť jaskýň obsahujúcu kľúče a dvere dvoch farieb, že keď zmeníme všetky červené kľuče na `(`, modré kľúče na `[`, červené dvere na `)` a modré dvere na `]`, **nedostaneme** dobre uzátvorkovaný výraz.

Tvorba dobre uzátvorkovaného výrazu pre dva typy zátvoriek sa riadi nasledujúcimi pravidlami:

- () a [] sú dobre uzátvorkované výrazy
- ak je x dobre uzátvorkovaný výraz, tak `(`x`)` a `[`x`]` sú dobre uzátvorkované výrazy
- ak sú x a y dobre uzátvorkované výrazy, tak aj xy je dobre uzátvorkovaný výraz

e) (4 body) Popíšte postup, ktorý Timke povie, či je výraz pozostávajúci z dvoch typov zátvoriek (guľatých a hranatých) dobre uzátvorkovaný. Odôvodnite, prečo je váš postup správny.

Dobre uzátvorkované výrazy sú napríklad `([])([[]]())` alebo `[([()])]`, zle uzátvorkované sú `]([])(())` alebo `([()(][])`.

Praktická úloha

Pri práci s počítačom je potrebné vedieť pracovať aj s rôznymi nástrojmi, ktoré slúžia na úpravu obrázkov, prácu so zvukom či vyhľadávaním na internete. V tejto časti ťa preto zakaždým čaká nejaká netradičná úloha.

3. Prudký poryv prievanu

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Adamovi na kral@ksp.sk

5.5.2018, 14:33 - "Ale Samko, veď ja nie som tvoj strýko, ja som tvoja teta."
7.5.2018, 18:57 - Ozýva sa zlovestné zavŕzganie a prudký poryv prievanu zmietol Samuelovu prácu zo stola.
25.6.2018, 19:53 - "Samo, zajtra pôjdeme na návštevu"

Samuel nemá rád návštevy svojej rodiny. Nie preto, že by nemal členov svojej rodiny rád. Skutočný dôvod jeho nenávisti plynie z neschopnosti zapamätať si všetkých svojich príbuzných a spôsoby ako ich má oslovovať. Len nedávno prišiel na rozdiel medzi strýkom a tetou (a to za veľmi trápnych okolností). 5.5.2018, 14:34 preto spravil jedno veľmi dôležité rozhodnutie. Rozhodol sa nájsť spôsob, ktorý mu umožní predchádzať podobným omylom a tak si bude môcť naplno užiť rodinné oslavy.

Pustil sa preto hneď do práce. Jeho pamäť uchováva veľké množstvo nanajvýš dôležitých informácií (mená hrdinov z doty, texty popových pesničiek, ktorý deň ide po strede, 21 desatinných miest čísla π) a preto si je schopný pamätať iba to, kto v jeho rodine je koho rodič a aké má pohlavie. Tieto informácie si teda spísal na kartičky.

7.5.2018, 18:56 mal pred sebou v stromovej štruktúre krásne zoradené 4 generácie svojej rodiny. Na druhej strane domu jeho mama však práve otvorila balkónové dvere a celá jeho práca skončila na zemi. Keď upratal všetok ten bordel, rýchlo prepísal všetky záznamy do počítača. Tam mu ich totiž prievan určite nerozfúka.

Okoloidúca náhodná anomália však popridávala k Samovmu rodokmeňu ešte jeden rodokmeň, jeho dáta teda tvoria popisy dvoch takýchto stromov. Samuel si však z toho ťažkú hlavu nerobí, aspoň budú jeho výsledky zaujímavejšie a dozvie sa niečo aj o cudzích ľuďoch.

Úloha

Samo má zoznam rodinných vzťahy v podobe `rodič(Rodic,Dieta)`. Každý takýto záznam znamená, že človek `Rodic` je rodičom človeka `Dieta`. Okrem toho o každom človeku vie aj to, či je muž (zapísané `muž(Meno)`) alebo žena (`žena(Meno)`). Všetky tieto záznamy tvoria dva nezávislé rodokmene.

Oba rodokmene sú navyše korektné, teda môžete predpokladať, že každý človek má najviac dvoch rodičov a rodokmene neobsahujú žiadne cykly. Šťastnou zhodou okolností majú všetci ľudia v Samových záznamoch iné krstné mená. Nemôže sa teda stať, že by v rodokmeňoch boli dvaja rôzni Petrovia.

Pred blížiacou sa návštevou potrebuje Samo zo záznamov zistiť nasledovné informácie. Pomôžte mu.

a) (1 bod) Je Ladislav Evin syn?

b) (2 bod) Kto je rodič Samuela?

c) (2 body) Koľko detí má Jozef?

d) (3 body) Koľko vnukov má Marta?

e) (3 body) Kto je manžel Lucie?

f) (4 body) Koľko sesterníc má Ema?

Ako riešiť túto úlohu

Samko všetky záznamy o svojej rodine vložil do interaktívneho prostredia `SWISH`, kde ich môžete nájsť vy. Toto prostredie je dostupné online na tejto adrese: https://swish.swi-prolog.org/p/prudkyporyvprievanu.pl

Okrem toho, že v tomto prostredí sú uložené Samuelove záznamy, v tomto prostredí viete písať a spúšťať programy v jazyku `Prolog`. Je to jednoduchý jazyk na spracovávanie logických výrokov, pomocou ktorého viete zistiť odpovede na uvedené otázky. Krátky tutoriál k tomuto jazyku obsahujúci všetko čo potrebujete vedieť k riešeniu úlohy nájdete tu: https://král.com/prolog

Odovzdávanie

K tejto úlohe odovzdávate popis vašeho riešenia. Okrem samotnej odpovede na otázku uveďte aj Prolog programy (takzvané dotazy), ktoré ste použili pri riešení, a popis toho, prečo ste tieto programy použili a odôvodnite správnosť vašeho postupu.

Za riešenie obsahujúce iba správne odpovede získate najviac polovicu bodov.

Programátorské úlohy

Tieto úlohy sú zamerané na praktickú tvorbu programov v niektorom vyššom programovacom jazyku ako je napríklad Python, C++ alebo Pascal. Na stránke odovdzávaš **iba zdrojový kód** svojho programu riešiaceho zadanú úlohu, ktorý bude okamžite automaticky otestovaný a do pár sekúnd sa dozvieš, koľko bodov tvoj program získal. Tieto body ti už nikto nemôže zobrať, ale ak si nezískal plný počet bodov, môžeš opakovane odovzdávať opravený program, až kým nebudeš spokojný s výsledkom.

Ak už vieš programovať, ale ešte si nepracoval s naším testovacím systémom, odporúčam ti zájsť na Programátorskú Liaheň (http://liahen.ksp.sk), kde si o tom môžeš prečítať úvodný text a vyriešiť si niekoľko jednoduchých úloh.

Ak však **nevieš programovať, tak nezúfaj!** Pripravili sme pre teba **Programátorskú Liaheň**, ktorá ťa **naučí základy programovania** v jazyku `C++`. Navyše, za riešenie týchto tutoriálových úloh na Liahni môžeš získať body priamo do PRASKu a tým si vynahradiť neriešenie niektorej z programátorských úloh.

Presnejšie to funguje takto. Na Liahni sa nachádzajú dve sady úloh, prvá zameraná na premenné a druhá na podmienky v jazyku `C++`. V týchto sadách sa nachádzajú bodované aj nebodované úlohy, ktoré môžeš postupne riešiť a ktoré ti postupne vysvetlia danú problematiku. Dokopy sa v jednej sade dá získať až 15 bodov.

Týmito bodmi si potom môžeš nahradiť úlohy 4 a 5. Samozrejme, toto môžeš urobiť **s každou sadou najviac raz**.

No a v budúcej sérii budeš môcť za body riešiť ďalšie dve sady z Liahne.

Samozrejme, nič ti nebráni riešiť aj úlohy z Liahne aj klasické programátorské úlohy v PRASKu.

Programátorskú Liaheň nájdeš na tejto stránke: http://liahen.ksp.sk

4. Pesničky

15 bodov za riešenie

Ak nevieš programovať, nezúfaj! Môžeš sa to naučiť a ešte za to získať body, ktoré sa ti budú počítať namiesto tejto úlohy.

Stačí, že pôjdeš na stránku Programátorskej Liahni (liahen.ksp.sk). Keď budeš riešiť sadu **variables_cpp**, bodmi, ktoré získaš si môžeš nahradiť riešenie tejto úlohy. Stačí ak na spodku tejto stránky odovzdáš pdf-ko s prezývkou, ktorú používaš na Liahni.

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Dávidovi na davidb@ksp.sk

Roman je hipster. Nosí hipsterské oblečenie, je hipsterské jedlo, pije hipsterskú kávu a počúva hipsterskú hudbu. Hudba sa však veľmi rýchlo vyvíja. Jeden deň je pesnička hipsterská, a na druhý deň ju začne hrať Rádio a stane sa mainstreamovou. Inú pesničku zase Rádio hrať prestane, vypadne z mainstreamu a stane sa hipsterskou.

Samozrejme Roman si nemôže dovoliť počúvať Rádio, je predsa hipster. Potrebuje preto vašu pomoc pri rozdhodovaní, ktoré pesničky môže alebo nemôže počúvať.

Úloha

Rádio sa rozhoduje o tom, ktoré pesničky bude hrať, výlučne podľa názvu. Konkrétne, podľa toho, aké písmená sa v jej názve nachádzajú. Pesničku zahrá Rádio iba vtedy, keď sa celý názov skladá len z mainstreamových písmen. O tom, ktoré písmená sú mainstreamové rozhoduje Rádio a verejnosti to oznamuje počas vysielania. Občas tiež Rádio vyhlási, že niektoré písmeno sa stalo hipsterským a písmená, ku ktorým sa vôbec nevyjadruje sú hipsterské už len z princípu. Na začiatku sú všetky písmená hipsterské.

Na vstupe dostanete zoznam vyhlásení Rádia o mainstreamovosti/hipsterskosti písmen a pomedzi to Romanove otázky na nejaké pesničky, všetko v chronologickom poradí.

Na každú Romanovu otázku zistite odpoveď podľa toho, ktoré písmená sú v danom momente mainstreamové.

Formát vstupu

V prvom riadku je číslo n ($1 \leq n \leq 10^5$) – počet udalostí. Nasleduje n riadkov. Každý obsahuje jednu z nasledujúcich troch správ:

`mainstream x` – písmeno `x` je od teraz mainstreamové

`hipster x` – písmeno `x` je od teraz hipsterské (nie je mainstreamové)

`aka NazovPesnicky` – Roman sa pýta či je pesnička `NazovPesnicky` mainstreamová alebo hipsterská.

Písmeno `x` je vždy buď malé alebo veľká písmeno abecedy (a-z, A-Z) a `NazovPesnicky` je tvorený jedným slovom, ktoré sa skladá z najviac 20 takýchto písmen. Názov piesne teda neobsahuje žiadne špeciálne znaky ani medzery.

Formát výstupu

Pre každý riadok na vstupe, ktorý začína `aka` vypíšte na jeden riadok buď `hipster` alebo `mainstream` podľa toho, či je daná pesnička aktuálne hipsterská alebo mainstreamová.

Hodnotenie

Vaše riešenie bude otestované na 3 sadách vstupov. Za vyriešenie každej sady získate 5 bodov. Tieto sady sa líšia veľkosťou vstupných údajov. Okrem toho v prvej sade názvy pesničiek, na ktoré sa Roman pýta, obsahujú iba písmená `a` až `d`.

Vo vstupných sadách platí:

Číslo sady	1	2	3
maximálne n	1 000	10 000	100 000

Načítanie vstupu

Ak ste ešte nepracovali s textovými reťazcami, nezúfajte, je to vskutku jednoduché.

V **C++** chcete použiť premenné typu `string`, ktorú načítate pomocou `cin`. Načítavanie v tomto prípade zoberie všetky znaky po najbližšiu medzeru alebo koniec riadka. Načítať dva reťazce v jednom riadku oddelené medzerou (ako potrebujeme v úlohe) vieme jednoducho:

```
string a,b;
cin >> a >> b;
```

Na premennú typu `string` sa potom vieme pozerať ako na pole znakov, teda ak chceme zistiť znak reťazca `a` na pozícii i, stačí napísať `a[i]`. Tento znak potom môžeme napríklad porovnať:

```
if(a[0] == 'b') cout << "Slovo zacina na pismeno b.";
```

V jazyku **Python** môžeme načítavať vstup pomocou príkazu `input()`. Tento príkaz načíta celý nasledovný riadok, vrátane medzier. Ak teda chceme načítať dva reťazce na jednom riadku, musíme to po načítaní rozdeliť príkazom `.split()`.

```
a, b = input().split()
```

Tento príkaz načíta riadok vstupu, rozdelí ho podľa medzery a výsledné dve časti vloží do premenných `a` a `b`. Tento príkaz by nefungoval ak by bola na riadku viac ako jedna medzera.

Na premennú, v ktorej je reťazec, sa môžeme pozerať tiež ako na list znakov, akurát tento list nevieme meniť. Takisto sa však vieme pozrieť na i-te písmeno:

```
if a[0] == 'b': print('Slovo zacina na pismeno b.')
```

V **oboch jazykoch** si na internete môžete nájsť, ako sa dá zmeniť znak na číslo podľa ASCII tabuľky.

Príklady

vstup

```
6
aka caba
mainstream a
mainstream b
aka caba
mainstream c
aka baca
```

výstup

```
hipster
hipster
mainstream
```

Na začiatku sú všetky písmená hipsterské, pri druhej otázke pesnička `caba` *obsahuje hipsterské písmeno* `c`*, kvôli ktorému je hipsterská. Pesnička* `baca` *je však v momente otázky už mainstreamová.*

vstup

```
11
mainstream a
mainstream A
aka Aaaaa
hipster a
hipster b
mainstream D
aka Da
aka DA
mainstream S
aka SAD
aka LoL
```

výstup

```
mainstream
hipster
mainstream
mainstream
hipster
```

Rádio môže označiť za hipsterské/mainstreamové jedno písmeno aj viac krát za sebou.
Takýto vstup sa v prvej sade nevyskytne, pretože obsahuje skladby z iných písmen ako `a` *až* `d`*.*

5. Prehádzané magnetky

15 bodov za riešenie

Ak nevieš programovať, nezúfaj! Môžeš sa to naučiť a ešte za to získať body, ktoré sa ti budú počítať namiesto tejto úlohy.

Stačí, že pôjdeš na stránku Programátorskej Liahni (liahen.ksp.sk). Keď budeš riešiť sadu **conditions_cpp**, bodmi, ktoré získaš si môžeš nahradiť riešenie tejto úlohy. Stačí ak na spodku tejto stránky odovzdáš pdf-ko s prezývkou, ktorú používaš na Liahni.

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Romanovi na roman.sobkuliak@trojsten.sk

Jozef sa pri raňajkách zahľadel na chladničku, na ktorej bolo množstvo magnetiek. Sestra Vendula je totiž ich nadšená zberateľka. Svoj prvý kúsok dostala od tety z Gruzínska, ktorá sa vracala z brigády na arašidových plantážach. Vendule sa vtedy magnetka zapáčila natoľko, že od rodiny a kamarátov už nechcela iné darčeky ako magnetky. Tento mesiac nimi konečne zaplnila celú chladničku. Teraz ale nevie, ako magnetky usporiadať.

Jozefa, dojedajúc svoje raňajky, napadlo, že by bolo zaujímavé vytvoriť z magnetiek na chladničke čo najviac rovnakých riadkov. Svoj nápad hneď povedal Vendule a tá bola nadšená. Teraz spolu rozmýšlajú koľko takýchto riadkov vedia vytvoriť. Ponáhľajú sa však do školy, a tak potrebujú pomoc!

Chladničku si pre jednoduchosť predstavme ako mriežku veľkosti $n \times n$ a magnetky ako znaky `+`, `@`, atď. Potom mriežka môže vyzerať napríklad takto:

+	@	+	@
@	%	@	+
&	&	@	&
#	&	@	#

Všimnite si, že znaky v mriežke sa môžu opakovať – inak by sme nemohli vytvoriť rovnaké riadky. Pre mriežku vyššie vieme vytvoriť 3 rovnaké riadky, takže jedno z možných pekných usporiadaní je nasledovné:

@	@	+	&
@	@	+	&
@	@	+	&
#	%	&	#

Úloha

Každá mriežka veľkosti $n \times n$ obsahuje n^2 magnetiek, z ktorých sa ale niektoré môžu opakovať. Počet rôznych magnetiek si označíme r. Na mriežke sa teda vyskytuje r typov magnetiek: $t_1, t_2, ..., t_r$. Každý typ t_i sa môže na mriežke vyskytovať viackrát, takže počet výskytov typu t_i si označíme p_i. Platí, že:

$$p_1 + p_2 + ... + p_r = n^2$$

Napríklad mriežka vyššie obsahuje 5 rôznych symbolov, takže $r = 5$. Takýto vstup by sa preto dal zapísať pomocou nasledovnej tabuľky:

i	t_i	p_i
1	+	3
2	@	6
3	%	1
4	&	4
5	#	2

Variant A

Pre zadanú mriežku a jednotlivé počty magnetiek zistite, či sa dá vytvoriť aspoň 47 rovnakých riadkov.

Variant B

Na získanie plného počtu bodov stačí vyriešiť Variant B

Pre zadanú mriežku a jednotlivé počty magnetiek zistite, koľko **najviac** rovnakých riadkov je možné vytvoriť.

Formát vstupu

Na prvom riadku vstupu dostaneme dve celé čísla n a r – veľkosť mriežky $n \times n$ a počet rôznych magnetiek, ktoré sa na mriežke nachádzajú.

Nasleduje riadok s r medzerou oddelenými číslami, pričom i-te číslo reprezentuje počet magnetiek p_i typu t_i na mriežke. Samotné typy magnetiek nás nezaujímajú, vieme že všetkých r je navzájom rôznych.

Pri riešení v jazyku `C++`, `Pascal`, `Java` a podobných, si dávajte pozor na maximálne číslo, ktoré sú schopné celočíselné premenné uložiť. Počet magnetiek jedného typu môže v posledných dvoch sadách tieto limity prekročiť. Odporúčame preto použiť 64-bitové premenné, v jazyku `C++` preto namiesto typu `int` použite typ `long long int`.

Formát výstupu

Variant A

Na jediný riadok výstupu vypíšte `ano`, ak sa v mriežke dá zostrojiť aspoň 47 rovnakých riadkov. V opačnom prípade vypíšte `nie`.

Variant B

Na jediný riadok výstupu vypíšte koľko najviac rovnakých riadkov sa dá zostrojiť zo zadaných magnetiek.

Hodnotenie

Úloha síce obsahuje 2 varianty, ale odovzdávanie bude fungovať štandardne. K úlohe sme pripravili 5 sád vstupov. Za vyriešenie každej sady získate 3 body. Tieto sady sa líšia veľkosťou vstupných údajov.

Ak sa rozhodnete riešiť *Variant A*, vaše riešenie bude otestované na prvých 2 sadách a môžete tak získať 6 bodov. Pri riešení *Variantu B* bude odovzdaný program otestovaný na všetkých 5 sadách, takže môžete získať 15 bodov. Interne testovanie vašich riešení funguje jednoduchým pravidlom. Ak je výstupom vášho programu číslo, testovač sa k nemu správa ako k riešeniu *Variantu B*. V opačnom prípade ho označí za riešenie *Variantu A*.

V nasledujúcej tabuľke označuje n veľkosť mriežky a r počet rôznych magnetiek. Vo vstupných sadách platí:

Číslo sady	1	2	3	4	5
maximálne n	100	100	2 000	50 000	50 000
maximálne r	300	500	2 000	60 000	60 000

Príklady

Variant B

vstup

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

výstup

```
3
```

Ide o príklad z úvodu pre mriežku 4×4. Najviac vieme vytvoriť 3 rovnaké riadky, napríklad tak, ako je ukázané na druhom obrázku.

vstup

```
6 6
12 8 4 7 3 2
```

výstup

```
4
```

Vieme vytvoriť najviac 4 rovnaké riadky. Môžeme napríklad použiť 8 magnetiek prvého typu, 8 druhého, 4 tretieho a 4 štvrtého.

Variant A

Uvádzame ešte pár príkladov pre *Variant A*. Prvé dva príklady sú rovnaké ako pri *Variante B*.

vstup

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

výstup

```
nie
```

Dajú sa vytvoriť najviac 3 rovnaké riadky, čo je menej ako 47, takže odpoveď je `nie`.

vstup

```
6 6
12 8 4 7 3 2
```

výstup

```
nie
```

Opäť vieme vytvoriť iba 4 rovnaké riadky, čo je menej ako 47 a odpovedáme **`nie`**.

vstup

```
60 6
537 690 108 395 193 1677
```

výstup

```
ano
```

V tomto príklade už vieme vytvoriť až 57 *rovnakých riadkov, takže odpoveď je* **`ano`**.