Priebeh krajského kola

Krajské kolo 41. ročníka Olympiády v informatike, kategória B, sa koná 20. 1. 2026 v dopoludňajších hodinách. Na riešenie úloh majú súťažiaci **4 hodiny čistého času**. Rôzne úlohy riešia súťažiaci na samostatné listy papiera. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Slovne popíšte algoritmus.
  Slovný popis riešenia musí byť jasný a zrozumiteľný i bez nahliadnutia do samotného algoritmu/programu.
- Zdôvodnite správnosť vášho algoritmu.
- Uveďte a zdôvodnite jeho časovú a pamäťovú zložitosť.
- Podrobne uveďte dôležité časti algoritmu, ideálne vo forme programu v nejakom bežnom programovacom jazyku (napr. C++, Python, Java, Pascal).
- V prípade, že používate vo svojom programovacom jazyku knižnice, ktoré obsahujú implementované dátové štruktúry a algoritmy (napr. STL pre C++), v popise algoritmu stručne vysvetlite, ako by ste napísali program s rovnakou časovou zložitosťou bez použitia knižnice.

Hodnotenie riešení

Za každú úlohu môžete získať od 0 do 10 bodov.

Pokiaľ nie je v zadaní povedané ináč, najdôležitejšie dve kritériá hodnotenia sú v prvom rade **správnosť** a v druhom rade **efektívnosť** navrhnutého algoritmu. Na výslednom počte bodov sa môže prejaviť aj kvalita popisu riešenia a zdôvodnenie tvrdení o jeho správnosti a efektívnosti.

Efektívnosť algoritmu posudzujeme vypočítaním jeho časovej zložitosti – funkcie, ktorá hovorí, ako dlho vykonanie algoritmu trvá v závislosti od veľkosti vstupných parametrov. Nezávisí pri tom na konštantných faktoroch, len na rádovej rýchlosti rastu tejto funkcie.

V zadaní úloh uvádzame časť „Hodnotenie“, v ktorej nájdete približné limity na veľkosť vstupných údajov. Pod pojmom „efektívne vyriešiť“ chápeme to, že váš program spustený na modernom počítači by mal dať odpoveď nanajvýš do niekoľkých sekúnd.

Údaje z tejto časti zadania by mali slúžiť hlavne na to, aby ste o riešení, ktoré vymyslíte, vedeli približne povedať, koľko bodov zaň dostanete.

B-II-1 Plátanie ciest

V Slovakistane jazdí veľa áut. Pre veľa áut treba veľa ciest. A pre veľa ciest treba veľa záplat.

Na slovakistanských cestách je n dier. Každá diera má hĺbku niekoľkých centimetrov. Koalícia Slovakistanských Plátačov (skrátane KSP) počas roka spraví niekoľko výjazdov s cieľom tieto diery zaplátať.
Počas jedného výjazdu zamestnanci KSP identifikujú jednu dieru s najväčšou hĺbkou a zaplátajú z nej **práve jeden centimeter**. Po výjazde sa teda hĺbka jednej najhlbšej diery zmenší o 1.
KSP musí každý rok vydať výročnú správu o svojej činnosti. Všetkých však aj tak zaujíma len jedno číslo – akú hĺbku mala po zaplátaní diera, ktorá bola plátaná v poslednom výjazde roka.

Súťažná úloha

Máte zadaný počet dier n a ich hĺbky na začiatku roka. Hĺbky sú nezáporné celé čísla $D[1]$ až $D[n]$.
Viete, že počas roka prebehlo k výjazdov. Každý výjazd zmenší hĺbku jednej aktuálne najhlbšej diery o 1. Našťastie sa počas roka diery ďalej neprehlbujú.
Môžete predpokladať, že k výjazdov nestačí na úplné zaplátanie všetkých dier v Slovakistane.

Zistite, aká bola hĺbka diery plátanej v poslednom výjazde po jej zaplátaní.

Formát vstupu a výstupu

Na prvom riadku vstupu sa nachádzajú dve čísla, n a k – počet dier na ceste a počet výjazdov.
Na druhom riadku vstupu sa nachádza n čísel $D[1]$ až $D[n]$ oddelených medzerami, ktoré popisujú hĺbky jednotlivých dier.

Na výstup vypíšte jedno celé číslo – hĺbku poslednej zaplátanej diery po jej zaplátaní.

Obmedzenia a hodnotenie

Vo všetkých vstupoch bude platiť, že $k < D[1] + D[2] + \cdots + D[n]$.

Plný počet bodov môže získať riešenie, ktoré efektívne vyrieši ľubovoľný vstup spĺňajúci nasledujúce obmedzenia: $n \leq 10^5$, $D[i] \leq 10^9$ a $k \leq 10^9$.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 1000$, $D[i] \leq 10^9$ a $k \leq 10^9$ sa dá získať 7 bodov.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 10^5$, $D[i] \leq 10^9$ a $k \leq 10^6$ sa dá získať 5 bodov.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 1000$, $D[i] \leq 1000$ a $k \leq 1000$ sa dajú získať 3 body.

Príklad

vstup

```
4 3
3 8 7 2
```

výstup

```
6
```

V prvom výjazde sa o jedna zmenší diera s hĺbkou 8. Pri druhom výjazde majú plátači na výber z dvoch dier s hlbkou 7, zaplátajú ľubovoľnú z nich. Posledný (tretí) výjazd potom zapláta ostávajúcu dieru s hĺbkou 7. Výsledkom je nová hĺbka tejto diery, čiže 6.

vstup

```
5 9
1 2 3 4 5
```

výstup

```
1
```

Tesne pred deviatym výjazdom budú existovať tri diery s hĺbkou 1 a dve diery s hĺbkou 2. Je jedno, ktorú z dier s hĺbkou 2 zaplátame, jej výsledná hĺbka bude 1.

vstup

```
5 1
1 1 1 1 1
```

výstup

```
0
```

B-II-2 Chemická továreň

V chemickej továrni prasklo kvôli vysokému tlaku niekoľko potrubí, z ktorých sa začala valiť nebezpečná kyselina. Je preto potrebné vyhlásiť poplach a evakuovať všetkých pracovníkov únikovým východom von z budovy skôr, ako sa k nim dostane kyselina.
Hlavný vedúci továrne nechce ukončiť prevádzku predčasne. Vyhlásiť poplach preto chce až v najneskoršom možnom momente takom, že sa následne ešte všetci zamestnanci stihnú zachrániť.

Továreň si môžeme predstaviť ako mriežku $r \times s$, v ktorej je každé políčko označené jedným z nasledujúcich znakov:

- `.` – prázdne políčko.
- `P` – políčko, na ktorom pracuje pracovník továrne.
- `K` – políčko, na ktorom z prasknutého potrubia vyteká kyselina.
- `U` – únikový východ. Je zaručené, že v mriežke existuje práve jedno takéto políčko.
- `#` – neprechodná stena.

Políčka, ktoré susedia stranou, nazývame susedné. Každé políčko má teda najviac štyroch susedov. Políčka mimo továrne považujeme za steny.

Okamih, v ktorom nastala nehoda, nazveme čas 0. V tomto okamihu sú kyselinou zaliate práve všetky políčka, ktoré sú označené písmenom `K`. Kyselina sa postupne šíri do všetkých smerov: ak v čase i sekúnd nejaké nezaliate políčko susedilo so zaliatym, tak v čase $(i + 1)$ už bude aj toto políčko zaliate kyselinou. Aj políčka označené písmenami `P` a `U` môžu byť zaliate a kyselina sa z nich bude normálne šíriť ďalej. Akonáhle kyselina zaleje nejaké políčko, toto políčko už navždy ostane zaliate.

Vedúci továrne si môže zvoliť ľubovoľné **celé** číslo $t \geq 0$: čas okamihu, v ktorom vyhlási poplach. Akonáhle je vyhlásený poplach, môžu sa všetci pracovníci naraz rozbehnúť k únikovému východu `U`.
Na každom políčku označenom `P` začína jeden pracovník. V okamihu t, kedy bol vyhlásený poplach, tento pracovník ešte stojí na svojom pôvodnom mieste. V okamihu $(t + 1)$ už môže byť na susednom políčku, a tak ďalej – za každú sekundu sa môže presunúť zo svojho súčasného políčka na niektoré z tých, ktoré s ním susedia. Pochopiteľne, pracovníci počas úteku nemôžu stúpiť na políčko, kde je stena, ani na políčko, ktoré je už zaliate kyselinou. Nie je dovolené stúpiť ani na políčko, kam by utekajúci pracovník prišiel v tom istom okamihu, v ktorom ho zaplaví kyselina.
Pracovníci si navzájom neprekážajú – na každom políčku ich v každom momente môže byť ľubovoľne veľa.

Môžete predpokladať, že pracovníci majú úplnú informáciu o stave továrne. Ak existuje aspoň jeden spôsob, ako sa vie pracovník vyhnúť šíriacej kyseline a dostať na políčko únikového východu, vyberie si zaručene jeden z týchto spôsobov.

Vašou úlohou je navrhnúť algoritmus, ktorý zistí, či sa dá zachrániť **všetkých** pracovníkov továrne. Ak áno, váš algoritmus by tiež mal určiť najväčšie možné t, teda vypočítať, ako najneskôr môže byť vyhlásený poplach, aby sa všetci pracovníci vedeli dostať k únikovému východu.

Formát vstupu a výstupu

Na vstupe dostanete najprv dve celé čísla r a s – počet riadkov a stĺpcov mriežky továrne.
Nasleduje r riadkov, z ktorých každý obsahuje s znakov zo sady `.PKU#`, ktoré opisujú interiér továrne. Políčka mimo tejto mriežky považujeme za steny.
Môžete predpokladať, že existuje **aspoň** jedno políčko `P`, **aspoň** jedno políčko `K` a **práve** jedno políčko `U`. Môžete predpokladať, že existuje cesta z aspoň jedného políčka `K` na políčko `U`.

Na výstup vypíšte jedno číslo $t \geq 0$: najneskorší čas, v ktorom musí byť vyhlásený poplach, aby všetci pracovníci bezpečne unikli z továrne. Pokiaľ nie je možné aby všetci pracovníci unikli ani v prípade, že poplach bude vyhlásený ihneď, teda v čase $t = 0$, vypíšte -1.

Obmedzenia a hodnotenie

Plný počet bodov môže získať riešenie, ktoré efektívne vyrieši ľubovoľný vstup spĺňajúci obmedzenia $r, s \leq 5000$.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $r, s \leq 1000$ sa dá získať 8 bodov.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $r, s \leq 100$ sa dá získať 6 bodov.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $r, s \leq 20$ sa dajú získať 4 body.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $r, s \leq 5$ sa dajú získať 2 body.

Za polovicu bodov môžete riešiť **jednoduchšiu verziu úlohy**, v ktorej stačí rozhodnúť, či všetci pracovníci továrne vedia uniknúť, ak bude poplach vyhlásený okamžite, teda v čase $t = 0$. Môžete teda získať najviac 1, 2, 3, 4, 5 bodov postupne pre obmedzenia $r, s \leq 5$, 20, 100, 1000, 5000.
Časti vášho riešenia, v ktorých riešite túto jednoduchšiu verziu, **viditeľne označte**.

Príklady

vstup

```
4 5
P.U..
.#.#P
.#.#.
....K
```

výstup

```
1
```

Ohrozený je najmä pracovník v najpravejšom stĺpci. V čase 0 bude kyselinou zaplavené len políčko `K`*. V čase 1 sa kyselina dostane aj na jeho dve susedné políčka. V čase 2 by sa dostala na ďalšie dve políčka (stena byť zaplavená nevie), vrátane políčka, na ktorom stojí spomínaný pracovník. Ten tam teda v čase 2 už stáť nemôže. Poplach preto musí byť vyhlásený najneskôr v čase 1. Ak vedúci vyhlási poplach v čase 1, stihnú sa zjavne obaja pracovníci zachrániť.*

vstup

```
4 5
P.U.K
.#.#P
.#.#.
....K
```

výstup

```
-1
```

Tu sú už v čase 0 zaliate obe únikové cesty pracovníka v pravom stĺpci, ten teda uniknúť nevie, ani keby bol poplach vyhlásený hneď v čase 0. Uniknúť ale nevie ani pracovník, ktorý je na začiatku v ľavom hornom rohu. Totiž aj keby bol poplach vyhlásený v čase 0, tento pracovník sa na políčko s únikovým východom vie dostať najskôr v čase 2, ale práve vtedy toto políčko už zaplaví kyselina z pravého horného rohu.

vstup

```
3 10
U......P..
#####.####
.........K
```

výstup

```
3
```

B-II-3 Zber jahôd

Hanka si povedala, že ďalšie prázdniny sa už nechce nudiť doma, zohnala si preto aj s kamarátom Jurom brigádu na španielskej farme. Prvý deň na farme ich privítal starý farmár, ktorý im nakázal zbierať jahody vo veľkom jahodovom záhone. S namosúreným výrazom ich varoval, že ak chcú večer jesť, majú ich nazbierať čo najviac. Deti z mesta sa celý deň bezradne motali po jahodovisku a večer išli spať hladné. Našťastie sa nad nimi zľutoval farmárov pomocník a vysvetlil im, že zbieranie jahôd má svoje presné pravidlá, ktoré musia dodržiavať. A hoci im poradil ako sa po jahodovisku správne pohybovať, nevedel im už pomôcť s tým, ako zber jahôd maximalizovať. Hankina unavená informatická duša sa však pustila do rozmýšľania. Taký problém sa totiž určite dá riešiť algoritmicky.

Súťažná úloha

Jahodovisko na farme si vieme predstaviť ako veľkú mriežku rozmerov $r \times s$. Na každom políčku tejto mriežky rastie nejaký kladný počet jahôd. Na začiatku dňa odvezie farmár Hanku a Jura k ľavému hornému políčku mriežky (pozícia $(1, 1)$) a tam ich vysadí. Počas dňa sa musia postupne presunúť do pravého dolného rohu (pozícia (r, s)), kde sa nachádza ubytovňa.
Aby pole zbytočne nepošliapali, hýbať sa môžu vždy len **dodola alebo doprava**. To znamená, že ak sa niektorý z nich nachádza na políčku (x, y), môže sa presunúť buď na políčko $(x + 1, y)$ alebo $(x, y + 1)$. Pri prechode mriežkou vyzbierajú všetky jahody na políčkach, ktoré pri prechode navštívili.

a) (3 body) Keďže sú Hanka s Jurom ešte neskúsení, pri prechode poľom musia ísť celý čas spolu a pri zbere si pomáhať. Navrhnite algoritmus, ktorý pre zadanú mriežku nájde tú cestu z políčka $(1, 1)$ do políčka (r, s), pre ktorú je súčet hodnôt na navštívených políčkach maximálny možný.

Po pár dňoch sú z Hanky a Jura ostrieľaní zberači a po poli už nemusia chodiť spoločne. Každý z nich si môže zvoliť svoju vlastnú cestu. Vždy keď jeden z nich stúpi na nejaké políčko mriežky, vyzbiera **všetky** jahody, ktoré na tomto políčku rastú. Ak ale cez niektoré políčko predchádzajú počas dňa obaja, jahody tam vyzbiera *len jeden z nich*. Na konci farmár spočíta, koľko jahôd vyzbierali obaja *dohromady*.

b) (1 bod) Hanke napadlo, že hľadať dve cesty s najväčším možným súčtom by mohla nasledovne: Ako v podúlohe a) nájde v mriežke cestu s najväčším súčtom. Všetky políčka tejto cesty nahradí hodnotami 0. V upravenej mriežke potom opäť nájde cestu s najväčším súčtom. Tieto dve nájdené cesty budú tvoriť výsledok.

   Takéto riešenie ale **nefunguje**. Nájdite konkrétny príklad mriežky, v ktorej sa Hankino riešenie pomýli a nenájde dve cesty s najväčším možným celkovým súčtom.

c) (6 bodov) Navrhnite algoritus, ktorý vyrieši zadanú úlohu a zistí, koľko najviac jahôd vedia spoločne vyzbierať, dodržiac všetky vyššie uvedené podmienky.

Formát vstupu a výstupu

Na prvom riadku vstupu sú dve čísla r a s – počet riadkov a stĺpcov jahodoviska.
Nasleduje r riadkov. Na každom z nich je s celých nezáporných čísel oddelených medzerami, ktoré popisujú koľko jahôd rastie na jednotlivých políčkach mriežky.
Všimnite si, že jahody možno rastú aj na začiatočnom a cieľovom políčku. Ak tam nejaké sú, určite budú vyzbierané.

Na výstup vypíšte jedno celé číslo – maximálny počet jahôd, ktoré vedia spoločne vyzbierať pri prechode mriežkou z ľavého horného rohu do pravého dolného rohu.

Obmedzenia a hodnotenie

Vo všetkých podúlohách predpokladajte, že počet jahôd na jednom políčku mriežky nepresiahne 10^9 a celkový výsledok sa pohodlne zmestí do 32-bitovej premennej.

V podúlohe a) viete získať plný počet bodov za riešenie efektívne pre $r, s \leq 5000$. Ľubovoľné funkčné riešenie vie získať 2 body.

V podúlohe c) viete získať plný počet bodov za riešenie efektívne pre $r, s \leq 500$. Najviac 4 body viete získať za riešenie efektívne pre $r, s \leq 100$. Ľubovoľné funkčné riešenie vie získať 2 body.

Príklad

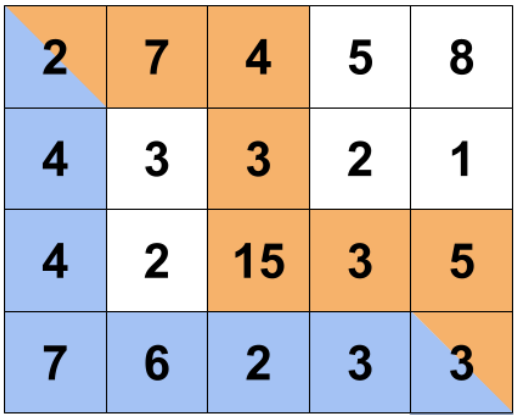
vstup

```
4 5
2 7 4 5 8
4 3 3 2 1
4 2 15 3 5
7 6 2 3 3
```

výstup

```
68
```

Na obrázku sú znázornené dve cesty, ktorými mohli ísť Hanka a Juraj, aby spoločne vyzbierali čo najviac jahôd. Všimnite si, že hodnoty 2 a 3 v rohoch sú do výsledku započítané len raz, aj keď tieto políčka navštívia obaja. Vyznačená cesta vedúca cez políčko s 15 jahodami je zároveň maximálnou cestou cez túto mriežku. Jej súčet 42 by bol preto výsledkom pre podúlohu a).
Všimnite si, že v tomto konkrétnom prípade by Hankin postup z podúlohy b) fungoval, lebo keď zmeníme hodnoty na políčkach tejto cesty na 0, druhá vyznačená cesta bude mať v takto zmenenej mriežke najväčší súčet.



B-II-4 Pomalý krtko

K tejto úlohe patrí študijný text uvedený na nasledujúcich stranách. Tento študijný text je totožný so študijným textom domáceho kola.

Vo *všetkých podúlohách* krajského kola sa budeme zaoberať krtkami, ktorí sú pomalí.
Krtko je pomalý, ak každý deň vie vyrobiť nanajvýš jednu dokončenú miestnosť. Povolené sú teda len tie staviteľské pravidlá, na ktorých pravej strane je *nanajvýš jedna dokončená miestnosť*. Napr. pravidlo $\texttt{X} \to \texttt{cddcX}$ je zakázané, ale pravidlá $\texttt{X} \to \texttt{cX}$, $\texttt{X} \to \texttt{Y}$, $\texttt{X} \to \texttt{c}$ či $\texttt{X} \to \varepsilon$ sú povolené.

Okrem toho si pripomeňme, že krtko označuje nedokončené miestnosti veľkými písmenami anglickej abecedy. K dispozícii má teda **práve 26 rôznych typov nedokončených miestností**. Viac typov nedokončených miestností nevie použiť.

Podúloha A (4 body)

Nájdite sadu pravidiel pre pomalého krtka, ktorej portfólio $\mathcal{A}$ je tvorené práve všetkými norami, ktorých zápis reprezentuje číslo zapísané v desiatkovej sústave deliteľné 35. Teda napríklad $0, 35, 70, 105, \ldots$ Zápis nesmie začínať prebytočnými nulami, teda 0035 v portfóliu nesmie byť.

Dva body za podúlohu A viete získať, ak namiesto čísel deliteľných 35 vygenerujete portfólio čísel deliteľných 3.

Podúloha B (3 body)

Nájdite sadu pravidiel pre pomalého krtka, ktorej portfólio $\mathcal{B}$ je tvorené všetkými norami skladajúcimi sa z malých písmen anglickej abecedy, ktoré *neobsahujú* súvislý podúsek `ananas`. Teda napr. nory `xyzaxsz` aj `bananakukuricastol` v portfóliu majú byť, zatiaľ čo `dalbysomsitakytoananasik` ani `annananastrojhrala` v portfóliu nesmú byť.

Keďže riešenie musí počítať so všetkými malými písmenami, dovolíme si nasledovný skrátený zápis pravidiel. Zápis $\texttt{X} \to [\texttt{d} - \texttt{h}]\texttt{X}$ reprezentuje 5 pravidiel $\texttt{X} \to \texttt{dX} \mid \texttt{eX} \mid \texttt{fX} \mid \texttt{gX} \mid \texttt{hX}$. Do hranatých zátvoriek teda môžeme vložiť interval písmen a reprezentovať ním sadu viacerých podobných pravidiel, ktoré sa líšia iba vo vytvorenej dokončenej miestnosti.

Podúloha C (3 body)

Dokážte, že neexistuje sada pravidiel pre pomalého krtka, ktorej portfólio $\mathcal{C}$ by obsahovalo práve nory pozostávajúce z malých písmen anglickej abecedy, ktoré končia *presne* 47 a-čkami. (Teda napríklad nory $\texttt{a}^{47}$, $\texttt{ba}^{47}$ a $\texttt{parostroja}^{47}$ do portfólia majú patriť, naproti tomu nory $\texttt{ba}^{46}$, $\texttt{a}^{47}\texttt{b}$, $\texttt{a}^{512}$, $\texttt{a}^{42}$ ani ε do neho patriť nemajú.)

Hodnotenie

V podúlohách A a B je potrebné jednak nájsť konkrétnu sadu pravidiel, ale aj zdôvodniť jej správnosť Za uvedenie správnej sady pravidiel bez zdôvodnenia, môžete získať za každú z týchto podúloh nanajvýš 2 body. V zdôvodnení správnosti musíte ukázať, že sada pravidiel vie vytvoriť ľubovoľnú zo zadaných nôr, a že žiadnu inú noru postaviť nemôže.

Študijný text: Krtko staviteľ

Krtko rád kope nory. Krtia nora sa skladá z niekoľkých za sebou idúcich *dokončených* miestností a najviac jednej *nedokončenej* miestnosti na konci. Krtko potom každý deň príde do ešte nedokončenej miestnosti a prerobí ju na niekoľko (občas aj nula) dokončených miestností a najviac jednu nedokončenú miestnosť (ktorá, ak existuje, je opäť na konci). Keď sa krtia nora skladá už iba z dokončených miestností, považujeme ju za *dostavanú* a krtko na nej už ďalej nič nemení.
Existuje viacero typov dokončených aj nedokončených miestností. Dve nory rovnakej dĺžky sú rôzne, ak na niektorej pozícii majú noru iného typu.

Formálnejšia definícia

Krtia nora je postupnosť niekoľkých (možno žiadnych) miestností, pričom najviac jedna posledná miestnosť môže byť nedokončená. Dokončené miestnosti budeme označovať malými písmenami anglickej abecedy a číslicami; nedokončené miestnosti budeme označovať veľkými písmenami anglickej abecedy. Krtko teda pozná presne 36 typov dokončených miestností a 26 typov nedokončených miestností.
Prázdnu postupnosť písmen (teda 0 miestností v rade) budeme označovať ε (aby sme ju v texte videli). Ak máme v zápise za sebou k miestností `a`, môžeme ich skrátene zapísať ako $\mathtt{a}^k$. Špeciálne $\mathtt{a}^0 = \varepsilon$.
Teda napríklad $\mathtt{a}^{13}\mathtt{ba}^{15}\mathtt{X}$ je ešte nedostavaná krtia nora, ktorá má 29 dokončených a 1 nedokončenú miestnosť. $\mathtt{aXb}^3\mathtt{Y}b$ nie je korektná nora, lebo obsahuje dve nedokončené miestnosti (`X` a `Y`). Nora $\mathtt{aaXb}^2$ nie je korektná, lebo jediná nedokončená miestnosť nie je na konci postupnosti.
Na poradí a type miestností záleží, `abc` a `cba` sú dve rôzne dostavané nory.

Staviteľské pravidlo popisuje ako môže krtko svoju noru budovať. Každý krtko má vlastnú konečnú sadu pravidiel, ktorou sa riadi. Pravidlo popisuje, ako sa môže zmeniť *jedna nedokončená* miestnosť na nejakú postupnosť dokončených a najviac jednej nedokončenej miestnosti.
Napríklad $\mathtt{X} \to \mathtt{abc}$ ukazuje, že nedokončená miestnosť `X` sa môže zmeniť na uvedenú postupnosť troch dokončených miestností. Pravidlo $\mathtt{X} \to \mathtt{Y}$ dovoľuje zmeniť nedokončenú miestnosť typu `X` na nedokončenú miestnosť typu `Y`. A pravidlo $\mathtt{X} \to \mathtt{cddcX}$ môže zmeniť `X` na postupnosť `cddc` dokončených miestností, na konci ktorých bude opäť nedokončená miestnosť `X`.
Povolené je aj použitie pravidla $\mathtt{X} \to \varepsilon$, ktoré nahradí nedokončenú miestnosť prázdnou postupnosťou, táto miestnosť teda v podstate zmizne.
Ak máme pre jednu nedokončenú miestnosť viacero pravidiel, môžeme ich navyše zjednodušene zapísať tak, že za šípkou uvedieme postupne všetky možnosti oddelené znakom |. Pre pravidlá z vyššie uvedeného príkladu by to vyzeralo nasledovne: $\mathtt{X} \to \mathtt{abc} \mid \mathtt{Y} \mid \mathtt{cddcX} \mid \varepsilon$.
Vždy, keď má krtko zmeniť miestnosť `X` a existuje viacero pravidiel, ako zmeniť túto miestnosť, môže si vybrať a použiť ľubovoľné z nich. Ak má zmeniť miestnosť `X` a neexistuje žiadne pravidlo, ako zmeniť `X`, stavba nory sa zasekne – takáto nora teda nikdy nebude dostavaná.

Každá nora sa na začiatku skladá z jedinej nedokončenej miestnosti `Z`. Pre konkrétnu sadu pravidiel budeme slovom **portfólio** označovať množinu všetkých *dostavaných nôr*, ktoré dokáže krtko zo začiatku `Z` vybudovať postupnou aplikáciou zadaných pravidiel. (Upozorňujeme, že nory, v ktorých ešte máme nedokončenú miestnosť, do portfólia sady pravidiel nikdy nepatria.)

Majme sadu pravidiel $\mathtt{Z} \to \mathtt{hlina}$. Portfóliom tejto sady pravidiel je množina obsahujúca jedinú noru, a to noru tvaru `hlina`. Nič iné totiž krtko pomocou týchto pravidiel dostavať nevie.

Ak by sme mali dve pravidlá $\mathtt{Z} \to \varepsilon \mid \mathtt{aZ}$, portfóliom tejto sady pravidiel sú všetky nory tvorené iba miestnosťami `a`. Zapísané formálne, sú to všetky nory tvaru $\mathtt{a}^n$ pre $n \geq 0$. Patria sem teda nory $\varepsilon, \mathtt{a}^1, \mathtt{a}^2, \mathtt{a}^3 \ldots$ Všetky tieto nory sa naozaj postaviť dajú, pre zvolené n nám stačí n-krát použiť pravidlo $\mathtt{Z} \to \mathtt{aZ}$ a následne pravidlo $\mathtt{Z} \to \varepsilon$.

Nakoniec, pre sadu pravidiel $\mathtt{Z} \to \mathtt{bcY} \mid \mathtt{aY}$ a $\mathtt{Y} \to \mathtt{bZ} \mid \mathtt{Z}$ je portfólio prázdne. Takýmito pravidlami totiž nikdy žiadnu noru nedostiavame, keďže po každej aplikácii niektorého pravidla nám na konci ostane nedokončená miesnosť.

Príklad 1

Zaujíma nás, či existuje taká sada pravidiel, ktorej portfólio je tvorené všetkými norami, ktoré sa skladajú iba z ľubovoľného počtu miestností `a` alebo `b` v ľubovoľnom poradí.
Takýmito norami sú napríklad nory $\mathtt{a}, \mathtt{b}, \mathtt{a}^{600}, \mathtt{abbaa}$ a nekonečne veľa ďalších kombinácií týchto dvoch znakov. (Vrátane nory ε.)

Riešenie

Hľadané pravidlá môžu vyzerať nasledovne: $\mathtt{Z} \to \varepsilon \mid \mathtt{aZ} \mid \mathtt{bZ}$.

Je zjavné, že každá dostavaná nora bude pozostávať iba z miestností `a` a `b`. Potrebujeme však ešte ukázať, že každú takú noru dokáže krtko naozaj postaviť.
Kým krtkova nora nie je dostavaná, nutne bude končiť nedokončenou miestnosťou `Z` (keďže iné nedokončené miestnosti sa v pravidlách nenachádzajú). Zoberme si teraz nejakú noru tvorenú len z miestností `a` a `b`, napríklad noru `abbaab`. Krtko ju vie postaviť nasledovne: na stavbu použije druhé alebo tretie pravidlo podľa toho, ktorá miestnosť má byť v tejto nore na ďalšej pozícii. V našom prípade by teda použil druhé pravidlo, potom dvakrát tretie pravidlo, dvakrát opäť druhé pravidlo, a nakoniec tretie pravidlo. A keď už postaví všetky zadané miestnosti, aplikuje prvé pravidlo, čím noru dostavia, lebo odstráni nedokončenú miestnosť. Každá nora správneho tvaru teda patrí do portfólia týchto pravidiel.

Príklad 2

Existuje sada pravidiel, ktorej portfólio je tvorené dvoma norami – `hlina` a $\texttt{a}^{55}$?

Riešenie

Áno, napríklad sada pravidiel $\texttt{Z} \to \texttt{Y}$ a $\texttt{Y} \to \texttt{hlina} \mid \texttt{a}^{55}$. Krtko pri stavaní nemá veľmi na výber, nič okrem zadaných dvoch nôr nepostaví.
Určite ste si všimli, že v tejto sade je prvé pravidlo v podstate zbytočné a postačovalo by mať len jeden typ nedokončenej miestnosti a dve pravidlá: $\texttt{Z} \to \texttt{hlina} \mid \texttt{a}^{55}$. Samozrejme, máte pravdu. Rôzne sady pravidiel môžu mať rovnaké portfólio. Obe sady sú teda korektnými riešeniami nášho problému.

Príklad 3

Označme si portfólio z príkladu 1 ako $\mathcal{A}$ (teda nory, ktoré obsahujú ľubovoľnú kombináciu písmen `a` a `b`).
Označme si portfólio z príkladu 2 ako $\mathcal{B}$ (teda dvojica nôr `hlina` a $\texttt{a}^{55}$).
Existuje sada pravidiel, do ktorej portfólia $\mathcal{C}$ patria aj nory z $\mathcal{A}$ a zároveň aj nory z $\mathcal{B}$ (a žiadne iné)? Formálne: Chceme, aby platilo $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$.

Riešenie

Pravidlá môžu byť nasledovné:

$$\texttt{Z} \to \texttt{A} \mid \texttt{B}$$

$$\texttt{A} \to \varepsilon \mid \texttt{aA} \mid \texttt{bA}$$

$$\texttt{B} \to \texttt{hlina} \mid \texttt{a}^{55}$$

Túto sadu pravidiel sme vyrobili nasledovne:

- V sade pravidiel z príkladu 1 sme zmenili všetky výskyty nedokončenej miestnosti `Z` na `A`.
- V sade pravidiel z príkladu 2 sme zmenili všetky výskyty nedokončenej miestnosti `Z` na `B`.
- Zobrali sme obe tieto sady dokopy a navyše sme pridali pravidlá $\texttt{Z} \to \texttt{A} \mid \texttt{B}$.

Pri stavaní nory sa teda krtko v prvý deň rozhodne, či prerobí `Z` na `A` alebo na `B`. Ak vyrobí `A`, ďalej môže používať len pravidlá analogické príkladu 1, a ak vyrobí `B`, bude potom musieť postupovať rovnako ako v príklade 2.
Použitím prvého pravidla sa krtko rozhodne, či bude stavať noru z $\mathcal{A}$ alebo $\mathcal{B}$. Následne vie používať iba pravidlá z danej podúlohy. A aby sa tieto pravidlá navzájom neovplyvňovali, zmenili sme nedokončené miestnosti zo `Z` na dve unikátne nedokončené miestnosti `A` a `B`.
(Všimnite si ešte, že noru a^{55} vie vyrobiť aj prvým, aj druhým spôsobom.)

ŠTYRIDSIATY PRVÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Truc Lam Bui, Ján Hozza, Jakub Šimo
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: NIVAM – Národný inštitút vzdelávania a mládeže, Bratislava 2026