Informácie a pravidlá

Pre koho je súťaž určená?

Do **kategórie B** sa smú zapojiť len tí žiaci základných a stredných škôl, ktorí ešte ani v tomto, ani v nasledujúcom školskom roku nebudú končiť strednú školu.
Do **kategórie A** sa môžu zapojiť všetci žiaci (základných aj) stredných škôl.

Odovzdávanie riešení domáceho kola

Riešitelia domáceho kola odovzdávajú riešenia sami, v elektronickej podobe, a to priamo na stránke olympiády: http://oi.sk/. Odovzdávanie riešení bude spustené niekedy v septembri.

Riešenia kategórie A je potrebné odovzdať najneskôr **15. 11. 2022**.
Riešenia kategórie B je potrebné odovzdať najneskôr **30. 11. 2022**.

Priebeh súťaže

Za každú úlohu domáceho kola sa dá získať od 0 do 10 bodov. Na základe bodov domáceho kola stanoví Slovenská komisia OI (SK OI) pre každú kategóriu bodovú hranicu potrebnú na postup do **krajského kola**. Očakávame, že táto hranica bude približne rovná **tretine maximálneho počtu bodov**.

V krajskom kole riešitelia riešia štyri teoretické úlohy, ktoré môžu tematicky nadväzovať na úlohy domáceho kola. V kategórii B súťaž týmto kolom končí.

V kategórii A je približne najlepších 30 riešiteľov krajského kola (podľa počtu bodov, bez ohľadu na kraj, v ktorom súťažili) pozvaných do **celoštátneho kola**. V celoštátnom kole účastníci prvý deň riešia teoretické a druhý deň praktické úlohy. Najlepší riešitelia sú vyhlásení za víťazov. Približne desať najlepších riešiteľov následne SK OI pozve na týždňové výberové sústredenie. Podľa jeho výsledkov SK OI vyberie družstvá pre Medzinárodnú olympiádu v informatike (IOI) a Stredoeurópsku olympiádu v informatike (CEOI).

Ako majú vyzerať riešenia úloh?

V praktických úlohách je vašou úlohou vytvoriť program, ktorý bude riešiť zadanú úlohu. Program musí byť v prvom rade korektný a funkčný, v druhom rade sa snažte aby bol čo najefektívnejší.

V kategórii B môžete použiť ľubovoľný programovací jazyk.

V kategórii A musíte riešenia praktických úloh písať v jednom z podporovaných jazykov (napr. C++, Pascal alebo Java). Odovzdaný program bude automaticky otestovaný na viacerých vopred pripravených testovacích vstupoch. Podľa toho, na koľko z nich dá správnu odpoveď, vám budú pridelené body. Výsledok testovania sa dozviete krátko po odovzdaní. Ak váš program nezíska plný počet bodov, budete ho môcť vylepšiť a odovzdať znova, až do uplynutia termínu na odovzdávanie.

Presný popis, ako majú vyzerať riešenia praktických úloh (napr. realizáciu vstupu a výstupu), nájdete na webstránke, kde ich budete odovzdávať.

Ak nie je v zadaní povedané ináč, riešenia teoretických úloh musia v prvom rade obsahovať **podrobný slovný popis použitého algoritmu, zdôvodnenie jeho správnosti** a diskusiu o efektivite zvoleného riešenia (t. j. posúdenie časových a pamäťových nárokov programu). Na záver riešenia uveďte program. Ak používate v programe netriviálne algoritmy alebo dátové štruktúry (napr. rôzne súčasti STL v C++), súčasťou popisu algoritmu musí byť dostatočný popis ich implementácie.

Usporiadateľ súťaže

Olympiádu v informatike (OI) vyhlasuje *Ministerstvo školstva SR* v spolupráci so *Slovenskou informatickou spoločnosťou* (odborným garantom súťaže) a *Slovenskou komisiou Olympiády v informatike*. Súťaž organizuje *Slovenská komisia OI* a v jednotlivých krajoch ju riadia *krajské komisie OI*. Na jednotlivých školách ju zaisťujú učitelia informatiky. Celoštátne kolo OI, tlač materiálov a ich distribúciu po organizačnej stránke zabezpečuje IUVENTA v tesnej súčinnosti so Slovenskou komisiou OI.

A-I-1 Najmenejkrát rozsvieť

*Toto je **praktická úloha**. Pomocou webového rozhrania odovzdajte **funkčný, odladený program**.*

Nočný matfyz je zväčša temný. Budovu si zjednodušene vieme predstaviť ako mriežku rozmerov $n \times n$, rozdelenú na štvorcové políčka. Každé políčko má buď rozsvietené, alebo je tam tma.

Matúš stojí v severozápadnom rohu budovy a chce sa dostať do protiľahlého, juhovýchodného rohu. Tmavé políčka sú nebezpečné – človek ľahko o niečo zakopne, do niečoho narazí, alebo si nejak ináč ublíži. Matúš preto zásadne odmieta na takéto políčka vstúpiť. Matúšov začiatok aj cieľ cesty majú rozsvietené.

Riadky aj stĺpce mriežky si očíslujeme od 1 po n začínajúc v rohu, kde Matúš začína. Políčko v riadku r a stĺpci s budeme označovať (r, s). Matúš sa vie pohybovať štyrmi základnými smermi – teda z políčka (r, s) sa vie pohnúť na ľubovoľné z políčok $(r-1, s)$, $(r, s-1)$, $(r, s+1)$ a $(r+1, s)$. Samozrejme, Matúš sa smie pohnúť len na políčko, ktoré existuje a je tam rozsvietené.

Pri poslednej rekonštrukcii budovy pribudli na všetkých políčkach tlačidlá, ktorými sa dá rozsvietiť. Z nejakého záhadného dôvodu každé tlačidlo funguje tak, že rozsvieti svetlá nielen na políčku, na ktorom je, ale dokonca v celom jeho riadku mriežky. Stlačenie tlačidla v riadku r a stĺpci s teda rozsvieti všetky nasledovné miestnosti: $(r, 1), (r, 2), \ldots, (r, n)$.

V dnešnej dobe treba šetriť elektrinou. Matúš by preto chcel svoj cieľ dosiahnuť tak, aby cestou čo najmenejkrát musel tlačidlom rozsvietiť.

Formát vstupu a výstupu

V prvom riadku vstupu sú dve celé čísla n a m: rozmer mriežky a počet miestností, v ktorých sa už teraz svieti. V každom z nasledujúcich m riadkov sú súradnice jednej z týchto miestností vo formáte „riadok stĺpec“. Môžete predpokladať, že všetky súradnice sú korektné, že všetky miestnosti na vstupe sú navzájom rôzne a že medzi nimi sú miestnosti $(1, 1)$ a (n, n).

Na výstup vypíšte jeden riadok a v ňom jedno celé číslo. Ak sa Matúš do cieľa svojej cesty vôbec nevie dostať, vypíšte číslo -1. V opačnom prípade vypíšte najmenší počet riadkov, ktoré potrebuje cestou rozsvietiť.

Obmedzenia a hodnotenie

Vstupy sú rozdelené do štyroch sád. Za každú sadu, ktorú vaše riešenie celú správne vyrieši, dostanete príslušný počet bodov.

Vo všetkých vstupoch platí $2 \leq m \leq n^2$. Ďalšie obmedzenia pre jednotlivé sady vstupov sú v tabuľke.

číslo sady	1	2	3	4
body	3	2	2	3
maximálne n	10	100	2 000	2 000
maximálne m	—	—	6 000	—

Príklady

vstup
```
2 3
1 1
1 2
2 2
```
výstup
```
0
```

vstup
```
3 3
3 3
1 1
2 2
```
výstup
```
2
```

vstup
```
3 3
3 3
1 1
2 1
```
výstup
```
1
```

vstup
```
3 3
3 3
1 1
1 2
```
výstup
```
-1
```

V prvom príklade už existuje rozsvietená cesta do protiľahlého rohu. V druhom príklade Matúš môže stlačiť tlačidlo na štarte, potom prejsť na $(1, 2)$, odtiaľ na $(2, 2)$, tam znova stlačiť tlačidlo, následne prejsť na $(2, 3)$ a odtiaľ do cieľa cesty. V treťom príklade môže Matúš prejsť na políčko $(2, 1)$, tam rozsvietiť druhý riadok, ním prejsť na $(2, 3)$ a odtiaľ už rovno do cieľa. Vo štvrtom príklade sa Matúš nemá ako dostať do druhého riadku.

A-I-2 Vizualizácia firmy

*Toto je **praktická úloha**. Pomocou webového rozhrania odovzdajte **funkčný, odladený program**.*

Firma, v ktorej pracuje Michal, má stromovú hierarchiu. Vo firme je $n > 1$ zamestnancov. Tí majú navzájom rôzne čísla od 1 po n. Číslo 1 má Marta – riaditeľka firmy. Každý iný zamestnanec má práve jedného priameho nadriadeného, a to tak, že v hierarchii nie sú žiadne cykly.

Riaditeľka nedávno dala Michalovi pravidelný n-uholník a povedala mu, že doň má spraviť vizualizáciu hierarchie firmy: každého zamestnanca umiestniť do iného vrcholu a následne nakresliť modrú úsečku medzi každým zamestnancom a jeho priamym nadriadeným. Dobrá vizualizácia je taká, pri ktorej sa žiadne dve z týchto $n - 1$ modrých úsečiek nekrižujú – nanajvýš môžu mať niektoré dvojice úsečiek spoločný koncový bod.

Zistite, koľko rôznych dobrých vizualizácií existuje. Keďže toto číslo môže byť veľké, stačí, keď vypočítate jeho zvyšok po delení $10^9 + 7$. (Vizualizácie, ktoré sa líšia otočením a/lebo preklopením, považujeme za rôzne.)

Formát vstupu a výstupu

V prvom riadku vstupu bude číslo n. V druhom riadku bude $n - 1$ prirodzených čísel $p_2, \ldots, p_n$, pričom p_i je číslo priameho nadriadeného zamestnanca i. Pre každé i platí $p_i < i$, vďaka čomu je zaručené, že v hierarchii zamestnancov nie sú žiadne cykly.

Na výstup vypíšte jeden riadok a v ňom jedno číslo: počet dobrých vizualizácií, modulo $10^9 + 7$.

Obmedzenia a hodnotenie

Je päť sád vstupov. Za každú, ktorú váš program celú správne vyrieši, dostanete príslušné body.

- V prvej sade (3 body) platí $n \leq 10$.
- V druhej sade (1 bod) platí $n \leq 100$ a pre všetky i platí $p_i = 1$ (všetci sú priami podriadení riaditeľky).
- V tretej sade (1 bod) platí $n \leq 100$ a pre všetky i platí $p_i = i - 1$ (celá firma tvorí jednu „reťaz“).
- Vo štvrtej sade (2 body) platí $n \leq 100$.
- V piatej sade (3 body) platí $n \leq 10^6$.

Príklady

vstup

```
3
1 1
```

výstup

```
6
```

Firma má troch zamestnancov: riaditeľku, Michala a vrátnika. Pri vizualizácii môžeme riaditeľku umiestniť do ľubovoľného vrcholu trojuholníka (3 možnosti), následne Michala do ľubovoľného ešte voľného vrcholu (2 možnosti) a vrátnik skončí v poslednom voľnom vrchole. Všetkých 6 takto získaných vizualizácií je dobrých.

vstup

```
4
1 2 3
```

výstup

```
16
```

Štyri z vyhovujúcich možností vyzerajú tak, že na obvode štvorca máme postupne v smere ručičiek zamestnancov 1, 2, 3, 4. Naopak, nevyhovujú možnosti, pri ktorých máme na obvode postupne zamestnancov 1, 3, 2, 4. Pri každej takejto vizualizácii sa križujú úsečky spájajúce 1-2 a 3-4.

vstup

```
11
1 1 1 2 2 2 3 4 4 5
```

výstup

```
38016
```

vstup

```
15
1 1 1 2 2 2 3 4 4 5 8 8 8 10
```

výstup

```
2488320
```

A-I-3 Nevhodný darček

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách.

Zuzka zbožňuje binárne postupnosti. Jej priateľ Peter nemá talent na dávanie darčekov, a tak jej namiesto toho dal k narodeninám prirodzené číslo c.

Našťastie v miestnom matematickom servise ponúkajú službu, ktorá Zuzke pomôže. Táto služba funguje nasledovne: Ak im prinesiete postupnosť zloženú len z núl a jedničiek, vrátia vám ju nezmenenú. Ak ale prinesiete postupnosť, kde sú aj väčšie čísla, jedno z nich ($x > 1$) si vyberú, zoberú kladivo a pobúchajú po ňom tak, že sa rozpadne na tri menšie čísla x_0, x_1, x_2, pre ktoré platí $x_0 = x_2 = \lfloor x/2 \rfloor$ a $x_1 = x \bmod 2$. (Slovne: x vydelíme dvoma. Prvé aj tretie nové číslo sú rovné celej časti podielu, druhé nové číslo je zvyšok po delení.) Zvyšok postupnosti zostane nezmenený.

Zuzka do servisu prvýkrát priniesla jednoprvkovú postupnosť tvorenú jej číslom c. Následne servis navštevovala znova a znova až dovtedy, kým nedostala binárnu postupnosť čísel. Rozmyslite si, že výsledná Zuzkina postupnosť nezávisí od toho, ako si v servise vyberajú, po ktorom prvku kedy pobúchať kladivom.

Príklad: Zuzka začala s postupnosťou (6). Po prvom pobúchaní kladivom po šestke sa táto postupnosť zmenila na $(3, 0, 3)$. Potom napríklad mohli pobúchať kladivom po prvej trojke a vyrobiť postupnosť $(1, 1, 1, 0, 3)$. No a následne by rozbili aj druhú trojku a tak vznikla výsledná postupnosť $(1, 1, 1, 0, 1, 1, 1)$. Aj keby v servise rozbili najskôr druhú trojku a až potom prvú, skončila by Zuzka s touto postupnosťou.

Súťažná úloha

Na vstupe dostanete číslo c. Tým je jednoznačne určená Zuzkina výsledná postupnosť $z_1, \ldots, z_n$.

Ďalej na vstupe dostanete dva indexy ℓ a r.

Vašou úlohou je zistiť, koľko jedničiek sa nachádza v podpostupnosti $z_\ell, z_{\ell+1}, \ldots, z_r$.

Formát vstupu a výstupu

V jedinom riadku vstupu sú čísla c, ℓ a r. (Hodnota n, teda dĺžka výslednej Zuzkinej postupnosti, nie je súčasťou vstupu. Je ale zaručené, že ℓ a r spĺňajú nerovnosti $1 \leq \ell \leq r \leq n$.)

Na výstup vypíšte hľadaný počet jedničiek.

Obmedzenia a hodnotenie

Pri písaní svojich riešení môžete predpokladať, že váš programovací jazyk vie efektívne robiť aritmetické operácie s ľubovoľne veľkými celými číslami. (Tieto teda netreba implementovať, stačí napr. použiť normálny operátor `+` pre sčítanie a podobne.)

Plný počet bodov dostanete za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $c \leq 10^{1000}$.

Nanajvýš 6 bodov môžete dostať za riešenie, ktoré je efektívne za dodatočného predpokladu $r - \ell \leq 100$.

Nanajvýš 3 body môžete dostať za riešenie, ktoré je efektívne pre $c \leq 10^5$.

Nanajvýš 2 body môžete dostať za riešenie, ktoré funguje aspoň keď $\ell = 1$ a $r = n$, čiže za riešenie, ktoré vie vypočítať, koľko jednotiek je v celej výslednej postupnosti.

Príklady

vstup	výstup
`5 2 4`	`2`

Z čísla 5 vznikne postupnosť $(1, 0, 1, 1, 1, 0, 1)$. Z tej chceme úsek $(z_2, z_3, z_4) = (0, 1, 1)$.

vstup	výstup
`6 1 7`	`6`

Postupnosť pre číslo 6 sme videli vyššie. V celej postupnosti je šesť jedničiek.

A-I-4 Zoznámte sa s Hviezdnym impériom

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách. K tejto úlohe patrí študijný text uvedený na nasledujúcich stranách. Odporúčame najskôr prečítať ten a až potom sa vrátiť k samotným súťažným úlohám.

Podúloha A (2 body).
Navrhnite algoritmus, ktorý zistí, či Hviezdne impérium tvorí kružnicu: teda či vieme všetky systémy zoradiť do postupnosti $s_0, s_1, \ldots, s_{n-1}$ tak, aby pre každé i platilo, že systém s_i susedí práve s dvoma systémami: s_{i-1} a s_{i+1}. (Oba indexy treba brať modulo n, teda napr. systém s_0 má susediť s s_{n-1} a s_1.)

Podúloha B (3 body).
Navrhnite algoritmus, ktorý zistí, či v Hviezdnom impériu existuje perfektné párovanie. Inými slovami, treba skontrolovať, či sa dá všetky systémy rozdeliť do dvojíc tak, aby každú dvojicu tvorili dva susediace systémy. (Každý systém musí patriť do práve jednej dvojice.)

Podúloha C (5 bodov).
Navrhnite algoritmus, ktorý zistí, či Hviezdne impérium obsahuje Hamiltonovskú kružnicu. Formálne, podobne ako v podúlohe A chceme overiť, či vieme všetky systémy zoradiť do postupnosti $s_0, s_1, \ldots, s_{n-1}$ tvoriacej kružnicu, tentokrát však môže mať každý systém navyše aj ľubovoľne veľa ďalších susedov okrem tých dvoch na kružnici.

V podúlohách A aj C môžete predpokladať, že $n \geq 3$. Pripomíname, že na vašich riešeniach budeme hodnotiť len ich správnosť, korektnosť zdôvodnenia a hodnotu k. Na časovej a pamäťovej zložitosti nezáleží.

Študijný text: Hviezdne impérium

Pred dávnymi rokmi vo vzdialenej galaxii existovalo Hviezdne impérium. Toto impérium bolo tvorené n systémami.

Niektoré dvojice systémov boli k sebe natoľko blízko, že sa medzi nimi dalo lietať a komunikovať bežnými prostriedkami. Takéto dvojice systémov budeme volať *susedné*.

Pre každú dvojicu susedných systémov trvá prenesenie správy z jedného systému do druhého presne jeden rok. Hviezdne impérium je *súvislé* – z ľubovoľného systému vieme postupným preposielaním dostať správu do ľubovoľného iného. Toto však samozrejme môže trvať tisíce rokov, alebo aj viac, keďže Hviezdne impérium je obrovské.

Hviezdne impérium občas potrebuje riešiť rôzne algoritmické problémy. Všetky systémy vlastnia obrovské množstvo veľmi výkonných klasických počítačov, takže nikoho netrápi časová ani pamäťová zložitosť výpočtov, problém však nastáva, ak je pre vyriešenie problému potrebná komunikácia medzi systémami.

V praxi sa preto používa hybridné riešenie, ktoré v sebe spája modernú techniku a tri netradičné zložky: *telepatického cisára, veľmi kvalitných veštcov a možnosť sfarbiť celý vesmír na ružovo*. Funguje to celé nasledovne:

- Cisár Hviezdneho impéria má telepatické schopnosti, vďaka ktorým dokáže okamžite odovzdať informáciu kamkoľvek do celého impéria. Tento kanál je bohužiaľ len jednosmerný, príjemca správy cisárovi na ňu nevie odpovedať.
- Každý systém má svojho veštca. Veštec vie na požiadanie vyveštiť postupnosť bitov zadanej dĺžky. O veštcoch je známe, že naozaj nechcú, aby niekto z nich zomrel.
- Vedci Hviezdneho impéria nedávno objavili techniku, ako pošteklíť samotnú matériu časopriestoru. Ak tak niekto spraví, celý vesmír sa okamžite zafarbí do ružova a zhruba rok v takom stave ostane. Potom ružová farba v priebehu pár dní vybledne. Pošteklíť časopriestor už vedia obyvatelia každého zo systémov v impériu.

Za pomoci týchto nástrojov bol vyvinutý nasledovný postup pre riešenie algoritmických problémov:

1. Cisár každému hviezdnemu systému oznámi ich celkový počet (číslo n) a každému systému taktiež oznámi jeho jednoznačný identifikátor (unikátne celé číslo od 0 po $n-1$).
2. Cisár každému systému oznámi algoritmus, ktorý majú použiť. Tento algoritmus je spoločný pre všetky systémy.
3. Vládca každého systému zájde za svojím veštcom. Veštec mu oznámi nejakú k-bitovú postupnosť R.
4. Na základe svojej lokálnej postupnosti R a predpísaného algoritmu si každý systém vypočíta, aké správy chce poslať svojim susedom, a následne tieto správy pošle.
5. Každý systém rok počká, kým mu prídu správy od jeho susedov.
6. Následne každý systém zo svojej lokálnej postupnosti R a prijatých správ vypočíta, či je jeho lokálna odpoveď „možno“ alebo „nie“.
7. Ak je odpoveď „nie“, dajú popraviť veštca a pošteklia časopriestor.
8. Ešte týždeň všetci počkajú. Ak vesmír stále nie je ružový, znamená to, že nik nemal odpoveď „nie“, a teda všetci uzavrú, že odpoveď je „áno“.

Ako sme už spomínali, veštci naozaj nechcú, aby niekto z nich zomrel. Ak existuje taká sada veštieb, ktorá všetkým zachráni život, je zaručené, že jednu takú sadu naozaj vyveštia. Ak teda bolo pred veštením možné, že odpoveď bude „áno“, tak je zaručené, že po veštení sa tak naozaj stane.

Príklad 1: tri tímy

Cisára zaujíma, či sa dajú všetky systémy v impériu rozdeliť do troch tímov tak, aby žiadne dva susediace systémy neboli v tom istom tíme.

Riešenie: Vládca každého systému si dá vyveštiť dva bity. Ak dostane 00, rovno odpovie „nie“ a dá veštca popraviť. Ak dostane 01, 10 alebo 11, prečíta to ako číslo svojho tímu v dvojkovej sústave (1, 2 alebo 3). Následne každý systém pošle všetkým svojim susedom svoje vyveštené číslo tímu. Ak o rok od niektorého suseda dostane systém rovnaké číslo, odpovie „nie“. Ak nik neodpovedal „nie“, tak vieme, že každý systém má číslo tímu iné od všetkých svojich susedov, a teda je naozaj odpoveď „áno“.

Ak existuje aspoň jedno platné rozdelenie do troch tímov, veštci si vedia jedno také rozdelenie zvoliť a vyveštiť jemu zodpovedajúce čísla. Ako sme zdôvodnili vyššie, povedie to k želanej odpovedi „áno“. Ak rozdelenie neexistuje, buď aspoň jeden veštec vyveští 00, alebo všetci veštci vyveštia platné čísla – potom ale nutne niektorí dvaja susedia dostanú to isté číslo a obaja to následne odhalia a vyhlásia „nie“.

Popísaná stratégia používa $k = 2$ (veštia sa len dva bity).

Príklad 2: čokoľvek

Vyššie popísaným protokolom vieme za niečo vyše roka vyriešiť ľubovoľnú rozumnú algoritmickú úlohu takéhoto typu. Vždy totiž môžeme postupovať nasledovne:

V každom systéme si necháme od veštca vyveštiť mapu celého Hviezdneho impéria, presnejšie, jeho maticu súvislosti. To je n^2 bitov: po riadkoch vypísaná tabuľka rozmerov $n \times n$, v ktorej riadku i a stĺpci j je hodnota 1 alebo 0 podľa toho, či systémy i a j susedia.

Následne každý systém pošle všetkým susedom celú vyveštenú mapu a svoj identifikátor. Po roku, keď každý systém dostane správy od susedov, tak spraví nasledovné:

- Skontrolujeme, či všetci susedia dostali vyveštenú tú istú mapu ako my. Ak nie, rovno odpovieme „nie“.
- Skontrolujeme, či sada identifikátorov, ktoré nám prišli, presne zodpovedá tomu, koho máme mať za susedov podľa mapy. Opäť, ak to nesedí, rovno odpovieme „nie“.

Ak žiaden systém zatiaľ neodpovedal „nie“, znamená to, že naozaj všetky dostali vyveštenú správnu mapu impéria. No a teraz si už každý systém môže na svojich počítačoch (v zanedbateľnom čase) celý problém vyriešiť a podľa toho odpovedať „áno“ alebo „nie“.

Hodnotenie riešení

Ako sme už uviedli, čas a pamäť potrebné na klasické algoritmické výpočty budeme považovať za zanedbateľné. Vo svojich popisoch riešení ich ani nemusíte odhadovať.

Naopak, veštci za svoje služby pýtajú značné sumy a veštenie každého bitu je preto veľmi drahé. Snažíme sa teda nájsť také riešenie, ktoré má čo najmenšie k – teda čo najmenej vyveštených bitov v každom systéme.

Príklad 2 ukazuje, ako v podstate ľubovoľnú úlohu vyriešiť s n^2 vyveštenými bitmi. Tento počet vieme dokonca ľahkou úpravou znížiť na $n(n-1)/2$. Preto očakávajte, že body dostanete len za riešenia, ktoré potrebujú vyveštených bitov rádovo menej ako n^2.

Ako riešenie odovzdajte popis vášho algoritmu a zdôvodnenie jeho správnosti. Nezabudnite explicitne uviesť hodnotu k pre váš algoritmus.

Algoritmus môžete detailne slovne popísať, uviesť ho ako pseudokód, alebo ho naprogramovať v ľubovoľnom bežnom jazyku. V algoritme môžete používať:

- Read-only premenné `N` a `ID` obsahujúce celkový počet systémov a identifikátor aktuálneho systému.
- Read-only premennú `stupen` obsahujúcu počet susedných systémov.
- Funkcie `vyvesti_bit()`, `vyvesti_bity(b)` a `vyvesti_cislo(x)`, ktoré vieme používať na získanie veštby.

  Prvá funkcia vyveští a vráti jeden bit. Druhá vyveští rovno b bitov a vráti ich ako pole. Tretia funkcia vyveští $\lceil \log_2 x \rceil$ bitov a vráti ich ako nezáporné celé číslo z rozsahu od 0 po aspoň $x - 1$.
- Pole `outbox` obsahujúce `stupen` záznamov ľubovoľného typu. Do každého políčka môžete zapísať správu, ktorú chcete odoslať jednému zo susedných systémov.
- Read-only pole `inbox` obsahujúce `stupen` záznamov toho istého typu. V každom políčku je správa, ktorú ste dostali od príslušného suseda.
- Funkciu `ruzovy_vesmir()`, ktorá vráti logickú hodnotu pravda alebo nepravda podľa toho, či je vesmír ružový.

Indexy do `outbox` a `inbox` si zodpovedajú: do `inbox[i]` dostanete odpoveď od toho suseda, ktorému ste poslali správu cez `outbox[i]`. Pole `inbox` smiete začať čítať až po ukončení zápisu do poľa `outbox`.

Váš algoritmus musí vždy skončiť, a to tým, že explicitne vráti odpoveď `ANO` alebo `NIE`.

Príklad 3: cesta

Chceme zistiť, či je celé Hviezdne impérium jedna veľká cesta – inými slovami, či sa dá všetky systémy zoradiť do poradia $s_0, s_1, \ldots, s_{n-1}$ tak, aby každý systém fyzicky susedil práve s tými systémami, s ktorými susedí v postupnosti.

Riešenie: Ak má ľubovoľný systém stupeň väčší ako 2 (teda viac ako dvoch susedov), odpoveď je zjavne `NIE`.

V opačnom prípade sú len dve možnosti: buď je celé impérium cesta, alebo je to kružnica. (Pripomíname, že impérium je súvislé a všetci to o ňom vedia.)

Každý systém si preto dá vyveštiť dve veci: naše poradové číslo p na ceste a jeden bit navyše: náš lokálny index toho suseda, ktorý je na ceste pred nami. (Ak máme len jedného suseda, tento vyveštený bit bude 0 ak je sused pred nami, resp. 1, ak je za nami.)

Následne potrebujeme overiť, či nám veštci vyveštili všetko správne. Toto vieme spraviť napríklad tak, že každému susedovi pošleme číslo, o ktorom si myslíme, že je to jeho poradové číslo. Ak dostaneme od ľubovoľného suseda číslo iné ako p, odpovieme `NIE`, ak všetci dostanú všetko správne, všetci odpovedia `ANO`.

Ak Hviezdne impérium tvorí cestu, veštci si môžu vybrať jeden smer po nej, podľa neho vyveštiť poradové čísla a smery a tak zabezpečiť, že bude odpoveď `ANO`. Naopak, ak impérium tvorí kružnicu, bez ohľadu na to, aké bity veštci vyveštia, niektorý systém dostane vyveštenú najmenšiu hodnotu p zo všetkých. Tento systém potom jednému zo svojich dvoch susedov pošle hodnotu $p - 1$ a ten sused následne nutne odpovie `NIE`.

Toto riešenie potrebuje vyveštiť $k = \lceil \log_2 n \rceil + 1$ bitov.

Príklad implementácie

Nižšie uvádzame ukážkovú implementáciu riešenia z príkladu 3. Všimnite si, že kvôli stručnosti zápisu neuvádzame čakanie ako samostatnú inštrukciu. (Implicitne je jasné, že pred prvým prístupom do poľa `inbox` počkáme rok, aby stihli prísť správy od susedov, a pred záverečnou kontrolou `ruzovy_vesmir()` počkáme týždeň, aby už všetky systémy mali dopočítané.) Takto môžete písať aj svoje riešenia súťažných úloh.

```
# ak sme jediny system v celom imperiu, odpoved je ano
if N == 1: return ANO

# ak mame privelky stupen, urcite to nie je cesta
if stupen > 2: return NIE

# vyvestime si nase poradove cislo na ceste a skontrolujeme, ze je z rozsahu od 0 po N-1
p = vyvesti_cislo(N)
if p >= N: return NIE

# vyvestime si index suseda, ktory je na ceste pred nami: 0 alebo 1
pred = vyvesti_bit()

if stupen == 2:
    # mame dvoch susedov, obom posleme prislusne spravy
    outbox[pred] = p-1
    outbox[1-pred] = p+1
    # a o rok na to skontrolujeme, ci od oboch prislo spravne cislo
    if inbox[0] != p: return NIE
    if inbox[1] != p: return NIE
else:
    # mame len jedneho suseda, bud pred nami alebo za nami
    if pred == 0:
        outbox[0] = p-1
    else:
        outbox[0] = p+1
    # a od neho o rok ma prist nase spravne cislo
    if inbox[0] != p: return NIE

# pockame a ak vesmir nie je ruzovy, nik nemal odpoved NIE, a teda vsetci vratime odpoved ANO
if ruzovy_vesmir(): return NIE
return ANO
```

TRIDSIATY ÔSMY ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Michal Forišek
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2022