Priebeh krajského kola

Krajské kolo 38. ročníka Olympiády v informatike, kategória A, sa koná 17. 1. 2023 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.

A-II-1 Investičný guru

Lukáš predstiera, že je investičný guru, ktorý vie dokonale predvídať trh a investovať len do takých vecí, ktorých cena rastie. V skutočnosti však vlastní len jeden bajtkoin.
Nedávno na verejnosť unikol Lukášov spreadsheet, v ktorom si každý deň zapisoval cenu bajtkoinu. Tá samozrejme nemusela stále len rásť, mohla úplne ľubovoľne kolísať.
Aby Lukáš neprišiel o svoju povesť, začal tvrdiť, že nejde o vývoj ceny jednej komodity, ale že sú tam nejako „na striedačku“ zapisované ceny k rôznych komodít, do ktorých Lukáš investoval a ktoré všetky rastú.

Súťažná úloha

Na vstupe je číslo n a postupnosť **navzájom rôznych** prirodzených čísel $a_1, \ldots, a_n$.
Nájdite najmenšie k, pre ktoré vieme každý prvok postupnosti ofarbiť jednou z k farieb tak, aby každou farbou bola ofarbená rastúca postupnosť. Následne nájdite jedno takéto ofarbenie.

Na výstup vypíšte najskôr riadok obsahujúci optimálnu hodnotu k a potom pre každú farbu jeden riadok a v ňom postupnosť prvkov, ktoré majú dostať túto farbu. (Prvky v každom riadku výstupu musia byť uvedené v tom poradí, v ktorom sú v postupnosti a.)

Obmedzenia a hodnotenie

Môžete predpokladať, že všetky a_i sú navzájom rôzne kladné celé čísla, ktoré sa zmestia do bežných celočíselných premenných.

- Plný počet bodov dostanete za riešenie efektívne pre $n \leq 1\,000\,000$.
- Až 8 bodov môžete získať za riešenie efektívne pre $n \leq 1\,000\,000$, ktoré nájde optimálne k, ale nezostrojí jemu zodpovedajúce ofarbenie.
- Až 6 bodov môžete získať za riešenie efektívne pre $n \leq 1\,000\,000$ za dodatočného predpokladu, že $k \leq 5$. (U tohto riešenia teda stačí, aby bolo efektívne pre vstupy, v ktorých vieme zadanú postupnosť rozdeliť na nanajvýš päť rastúcich podpostupností.)
- Tiež až 6 bodov môžete získať za kompletné riešenie efektívne pre $n \leq 1000$.
- Nanajvýš 3 body sa dajú získať za riešenie efektívne pre $n \leq 7$.

Príklady

vstup

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

výstup

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

vstup

```
4
10 30 20 40
```

výstup

```
2
10 20
30 40
```

vstup

```
4
9 7 3 1
```

výstup

```
4
1
9
3
7
```

V prvom príklade je celá postupnosť rastúca. Stačí teda zobrať jednu farbu a tou ofarbiť všetky jej prvky.

V druhom príklade existuje viacero spôsobov, ako správne použiť dve farby. Jeden vidíte v príklade výstupu. Iné optimálne riešenie je dať 10 a 30 červenou a potom 20 a 40 modrou farbou.
Ďalšie optimálne riešenie: čísla 10, 30, 40 budú červenou a iba číslo 20 modrou.

V príklade 3 musíme každé číslo ofarbiť inou farbou. Lepšie riešenie neexistuje.

A-II-2 Električky

Električková sieť v Blatislave má stromovú topológiu. To znamená, že v meste je n zastávok električky a medzi nimi $n-1$ úsekov koľají. Koľaje sú postavené tak, aby sa po nich dalo dostať z každej zastávky na každú inú.
Peťo bol cez sviatky prvýkrát v Blatislave – na návšteve u tety. Z návštevy si pamätá to, že teta býva k zastávok od Hlavnej stanice.
Z návštevy si Peťo priniesol domov mapu električkovej siete. Doma ale zistil, že mapa je celá od blata a tak z nej nevie prečítať žiadny názov zastávky.

Súťažná úloha

Na vstupe dostanete slepú mapu električkovej siete, teda popis nejakého n-vrcholového stromu. Jeho vrcholy sú očíslované od 1 po n. Niektorý vrchol zodpovedá Hlavnej stanici a niektorý iný zastávke, kde býva teta – neviete ale, ktoré to sú. Na vstupe tiež dostanete číslo k: vzdialenosť medzi nimi.

V prvom riadku vstupu je číslo n, v druhom je číslo k. Zvyšok vstupu tvorí $n-1$ riadkov, v každom z nich sú dve celé čísla – čísla dvoch zastávok spojených priamym úsekom koľají.
Napíšte program, ktorý v danom strome spočíta, koľko (neusporiadaných) dvojíc vrcholov má vzdialenosť presne k. Tento počet vypíšte na výstup.

Obmedzenia a hodnotenie

Plný počet bodov môžete získať za riešenie, ktoré je efektívne pre $n \leq 200\,000$ a ľubovoľné k.
Takisto plný počet bodov môžete dostať aj za riešenie, ktoré je efektívne pre $n \leq 200\,000$ za dodatočného predpokladu $k \leq 100$.
Nanajvýš 8 bodov môže získať riešenie, ktoré je efektívne pre $n \leq 200\,000$ a $k \leq 100$ za dodatočného predpokladu, že každá zastávka priamo susedí s nanajvýš piatimi inými.
Nanajvýš 5 bodov môžete získať za riešenie efektívne pre $n \leq 5\,000$.
Nanajvýš 3 body môžete získať za riešenie efektívne pre $n \leq 400$.

Príklad

vstup

```
4
2
1 2
1 3
1 4
```

výstup

```
3
```

Pre túto mapu vieme vydedukovať, že stanica musí byť v jednom z vrcholov 2, 3, 4 a teta potom musí bývať v inom z týchto troch vrcholov.
Sú teda tri možnosti: (2, 3), (2, 4) a (3, 4).

vstup

```
4
2
1 2
2 3
3 4
```

výstup

```
2
```

Stanica a teta sú buď vo vrcholoch (1, 3) alebo vo vrcholoch (2, 4).

A-II-3 Čokoláda s hrozienkami

Máš čokoládu s n hrozienkami. Čokoláda je obdĺžnik rozdelený na dieliky, tých je r riadkov a s stĺpcov. Každé hrozienko je na inom dieliku čokolády. Jedno z hrozienok je viditeľne väčšie ako všetky ostatné.
Chceš túto čokoládu rozdeliť medzi seba a svojich $n-1$ kamarátov, a to tak, aby každý dostal jeden obdĺžnik čokolády a niekde na ňom jedno hrozienko. Ty si samozrejme necháš ten dielik, na ktorom je veľké hrozienko.
Každý obdĺžnik musí byť tvorený niekoľkými celými dielikmi čokolády. Čokoládu budeš na dieliky krájať nožom, môžeš si teda zvoliť aj také rozdelenie na obdĺžniky, ktoré sa nedá dosiahnuť jednoduchým lámaním.

Súťažná úloha

Napíš program, ktorý zistí, či sa dá danú čokoládu rozdeliť vyššie popísaným spôsobom. Ak áno, tvoj program by tiež mal zistiť, koľko najviac dielikov čokolády môžeš dostať – teda akú najväčšiu veľkosť môže mať pri platnom rozdelení čokolády dielik obsahujúci veľké hrozienko.

V prvom riadku vstupu sú rozmery čokolády: čísla r a s. V druhom riadku je číslo n udávajúce počet hrozienok (a zároveň želaných dielikov po rozdelení). Zvyšok vstupu tvorí n riadkov, z ktorých každý obsahuje súradnice jedného z hrozienok. Riadky aj stĺpce číslujeme od 1. Prvé hrozienko na vstupe je to veľké.

Ak čokoládu nevieme želaným spôsobom rozdeliť, tvoj program by o tom mal podať správu. Ak riešenie existuje, tvoj program by mal vypísať $n+1$ riadkov: najskôr riadok s najväčšou možnou veľkosťou tvojho kúsku čokolády a potom n riadkov popisujúcich obdĺžniky v jednom možnom optimálnom rozdelení čokolády.
Popis každého obdĺžnika tvoria štyri čísla: súradnice jeho ľavého horného a potom pravého dolného rohu.

Obmedzenia a hodnotenie

Plný počet bodov môžu získať riešenia efektívne pre $n \leq 10^6$ a $r, s \leq 10^4$.
Najviac 8 bodov môžu získať riešenia efektívne pre $r, s \leq 10^3$.
Najviac 6 bodov môžu získať riešenia efektívne pre $r, s \leq 200$.
Najviac 5 bodov môžu získať riešenia efektívne pre $r, s \leq 35$.
Najviac 3 body môžeš získať za riešenie, ktoré len rozhodne, či existuje platné rozdelenie čokolády, a ak áno, tak ľubovoľné jedno takéto rozdelenie vyrobí. (Vo vypísanom rozdelení teda nemusí platiť, že dostaneš najväčší možný kúsok.) Ak chceš riešiť túto verziu úlohy, nezabudni to prosím v riešení uviesť.

Príklady

vstup

```
5 7
1
3 5
```

výstup

```
1 1 5 7
```

Celá čokoláda je tvoja.

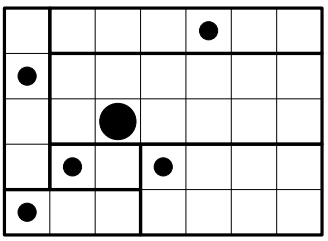
vstup

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

výstup

```
12
2 2 3 7
4 4 5 7
1 1 4 1
1 2 1 7
5 1 5 3
4 2 4 3
```

Vstup aj jedno jeho optimálne riešenie sa nachádza na obrázku vpravo.



A-II-4 Hviezdne impérium, čokoláda a strom

K tejto úlohe patrí študijný text uvedený na nasledujúcich stranách. Študijný text je identický s tým, ktorý ste už videli v zadaniach domáceho kola.

Podúloha A (5 bodov). V tejto podúlohe je zaručené, že Hviezdne impérium tvorí cestu. Inými slovami, všetky systémy v ňom vieme zoradiť do postupnosti $s_0, s_1, ..., s_{n-1}$ tak, že každý systém s_i fyzicky susedí práve so systémami s_{i-1} a s_{i+1} (ak existujú).
V niektorých systémoch poznajú čokoládu, v iných nie. Navrhnite algoritmus, ktorý zistí, či existuje aspoň $n/3$ systémov, v ktorých majú čokoládu.
Pri písaní algoritmu môžete predpokladať, že každý systém má svoju premennú `coko`, ktorá má hodnotu true alebo false podľa toho, či v tom systéme poznajú čokoládu.

Podúloha B (5 bodov). Navrhnite algoritmus, ktorý rozhodne, či je Hviezdne impérium strom.
Pripomíname, že je zaručené, že Hviezdne impérium je súvislé. Sú teda len dve možnosti: ak Hviezdne impérium obsahuje ľubovoľnú kružnicu, odpoveď je NIE, ak žiadnu kružnicu neobsahuje, odpoveď je ÁNO.
Formálne, kružnica je postupnosť $k \geq 3$ navzájom rôznych systémov $s_0, s_1, ..., s_{k-1}$ taká, že každý systém s_i susedí aj so systémom $s_{(i-1) \bmod k}$ aj so systémom $s_{(i+1) \bmod k}$. (Systémy na kružnici môžu samozrejme mať aj ľubovoľných ďalších susedov.)

Pripomenieme ešte, že pri hodnotení tejto úlohy vôbec nezáleží na časovej a pamäťovej zložitosti tvojho algoritmu. Dôležité je len, aby bol správny a aby hodnota k popisujúca, ako veľa vyveštených bitov potrebuješ, bola čo najmenšia. Nezabudni na zdôvodnenie správnosti!

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: NIVAM – Národný inštitút vzdelávania a mládeže, Bratislava 2023