Priebeh celoštátneho kola

Celoštátne kolo 38. ročníka Olympiády v informatike, kategórie A, sa koná v dňoch 22.-25. 3. 2023. Na riešenie úloh prvého, teoretického dňa majú súťažiaci 4,5 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.
- 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 prvého (teoretického) dňa

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 úlohy môžu byť uvedené limity na veľkosť premenných. Tieto môžete použiť na odhad toho, ako dobré vaše riešenie je. Na počítači, ktorý vykoná miliardu inštrukcií za sekundu, vyrieši vzorové riešenie ľubovoľný povolený vstup nanajvýš za niekoľko sekúnd.

A-III-1 Konferenčné zákusky

Peťke sa na doktorandskom štúdiu veľmi darí. Spravila niekoľko prelomových objavov, napísala k nim bezchybné vedecké štúdie a teraz je častou návšteníčkou svetových konferencií. Na nich má možnosť vypočuť si zaujímavé prednášky, spoznať nových ľudí, ale najmä ochutnať veľa skvelého jedla. Najlepšie sú koláčiky, ktoré sú k dispozícii počas prestávok medzi prednáškami.
Prestávky sú však krátke a Peťka musí veľmi zvažovať, ktoré koláčiky ochutná. Dá si čokoládovú sušienku, ktorú schrúme za pár sekúnd, alebo oveľa chutnejší cheesecake, ktorého jedenie jej zaberie aspoň polovicu prestávky? A stihne si k tomu dať ovocnú tortičku a muffin? A čo s punčovým rezom? Má si ho zobrať na prednášku?
Vždy cez prestávku zje Peťka niekoľko koláčikov a na jej konci si ešte práve jeden koláč zoberie na ďalšiu prednášku, kde si ho môže v pokoji vychutnať. Stále ju však trápi, či jedla koláčiky tak, aby maximalizovala svoju spokojnosť. Keďže ona počúva prednášky, o pomoc poprosila vás.

Súťažná úloha

Na vstupe dostanete popis n koláčikov, ktoré sú k dispozicii počas konferenčnej prestávky. **Z každého z nich je k dispozícii len jediný kus.**
Koláčik je definovaný dvoma číslami – s_i a t_i. Číslo s_i určuje množstvo spokojnosti, ktoré Peťka získa po jeho zjedení a t_i udáva počet sekúnd, ktoré bude Peťka tento koláč jesť.
Prestávka trvá p sekúnd. Vašou úlohou je určiť, ktoré koláčiky má Peťka zjesť počas prestávky, a ktorý jeden koláč si má zobrať na prednášku tak, aby súčet spokojností s_i všetkých zjedených koláčikov bol čo najväčší. Zároveň musí platiť, že súčet hodnôt t_i pre koláče zjedené počas prestávky nepresahuje hodnotu p.

Formát vstupu a výstupu

Na prvom riadku vstupu dostanete dve celé čísla n a p – počet koláčikov k dispozícii a dĺžku prestávky. Koláčiky sú číslované v poradí od 1 po n.
Druhý riadok obsahuje n celých čísel $s_1, s_2, \ldots, s_n$. Tretí riadok obsahuje n celých čísel $t_1, t_2, \ldots, t_n$.

Na prvý riadok výstupu vypíšte maximálnu spokojnosť, ktorú vie Peťka dosiahnuť jedením koláčov.
Na druhý riadok vypíšte v ľubovoľnom poradí čísla koláčov, ktoré má Peťka zjesť počas prestávky.
Na tretí riadok vypíšte jedno číslo koláča, ktorý si má Peťka zobrať na prednášku.
Ak existuje viacero riešení, ktoré majú rovnakú maximálnu spokojnosť, vypíšte ľubovoľné z nich.

Obmedzenia a hodnotenie

Vaše riešenie bude bodované na základe efektivity podľa nižšie uvedených podmienok. Zároveň však môžete riešiť jednoduchšiu verziu úlohy, v ktorej stačí vypočítať maximálnu Peťkinu spokojnosť a *nie je potrebné* zistiť, ktoré koláče je potrebné zjesť. Stačí teda ak na výstup vypíšete správny prvý riadok. Bodovanie za takéto riešenie je uvedené v zátvorkách.

Vo všetkých vstupoch platí $n \leq 10\,000$ a $p \leq 10\,000$. Vo všetkých vstupoch pre všetky i platí $1 \leq s_i, t_i \leq 10^9$.
Plný počet bodov (nanajvýš 8 bodov) môžete získať za riešenie, ktoré je efektívne pre tieto obmedzenia.
Nanajvýš 6 bodov (nanajvýš 5 bodov) môžete získať za riešenie, ktoré je efektívne za dodatočného predpokladu, že $s_i = t_i$ pre všetky i.
Nanajvýš 4 body (nanajvýš 3 body) môžete získať za riešenie, ktoré je efektívne pre $n \leq 100$.

Príklady

vstup	výstup
4 6 3 4 1 5 5 3 2 4	10 3 4 2

Ďalšie vyhovujúce riešenie je také, kde Peťka zje koláče 2 a 3 počas prestávky, a koláč 4 si zoberie na prednášku.

vstup	výstup
3 4 2 2 1 5 3 100	4 2 1

Tretí koláč Peťka nestihne zjesť cez prestávku. Vziať si ho na prednášku by mohla, ale neoplatí sa to – vyššiu spokojnosť dosiahne zo zjedenia zvyšných dvoch koláčov.

vstup	výstup
3 20 1 4 6 5 3 8	11 2 3 1

V tomto príklade by Peťka stihla zjesť všetky tri koláče už počas prestávky, no my musíme nájsť a vypísať také riešenie, v ktorom si práve jeden koláč vezme na prednášku.

vstup	výstup
1 20 1 21	1 1

Počas prestávky Peťka nebude jesť žiadne koláče, druhý riadok výstupu teda ostane prázdny.

A-III-2 Lezenie

Športové lezenie je mladým olympijským športom – prvýkrát sa v ňom súťažilo na nedávnych OH v Tokiu. Pri výbere súťažiacich na budúce OH v Paríži však došlo k zábavnej zámene: namiesto známeho českého olympionika Adama Ondru pozvali nášho Adama, bývalého riešiteľa Olympiády v informatike. Adam is takúto príležitosť samozrejme nechce nechať ujsť. Rozhodol sa preto poctivo trénovať a čo najlepšie sa pripraviť na OH.
Súťaž na OH v Paríži bude pozostávať z dvoch častí: bouldering (pri ktorom lezci skúšajú zdolať malé ale technicky náročné lezecké problémy) a lead (po našom lezenie na obtiažnosť – pri ňom sú lezci istení lanom a snažia sa dostať na čo najvyššie miesto cesty po umelej stene).
Adamova schopnosť v oboch disciplínach je momentálne rovná nule. Aby si na OH nespravil hanbu, potreboval by mať aspoň schopnosť β v boulderingu a λ v leade. Poraďte mu, ako najefektívnejšie trénovať.

Súťažná úloha

V lezeckom centre je n rôznych stien, kde sa dá trénovať – cesty na lead, problémy na boulder a rôzne iné. Tieto steny sú očíslované od 0 po $n-1$. O každej stene i vieme, že za každú hodinu strávenú tréningom na nej stúpne Adamova schopnosť v boulderingu o b_i a zároveň stúpne jeho schopnosť v leade o ℓ_i. (Adam môže trénovať aj ľubovoľnú necelú časť hodiny. Jeho schopnosti potom stúpnu o príslušný zlomok toho, čo by získal za celú hodinu.)
Napíšte program, ktorý zistí, ako najrýchlejšie sa vie Adam vytrénovať na aspoň požadované úrovne schopností.

Formát vstupu a výstupu

V prvom riadku vstupu sú tri kladné celé čísla n, β a λ. V druhom riadku sú kladné celé čísla $b_0, \ldots, b_{n-1}$. V treťom riadku sú kladné celé čísla $\ell_0, \ldots, \ell_{n-1}$.

Na výstup vypíšte nezáporné reálne číslo: minimálny čas t v hodinách, za ktorý sa Adam vie vytrénovať na OH. Formálne, čas t má byť najmenšie reálne číslo, pre ktoré existujú nezáporné reálne čísla $t_0, \ldots, t_{n-1}$ také, že:

- súčet všetkých t_i je t,
- súčet súčinov $t_i b_i$ je aspoň β,
- súčet súčinov $t_i \ell_i$ je aspoň λ.

(Slovne: ak Adam strávi tréningom na každej stene i práve t_i hodín, bude mať dokopy natrénované dosť aj v boulderingu aj v leade.)
Keďže ide o teoretickú úlohu, nemusíte sa zaoberať zaokrúhľovacími chybami pri výpočte t.

Príklady

vstup

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

výstup

```
2.0
```

Adam napr. strávi hodinu na stene 0 a potom hodinu na stene 1.

vstup

```
2 4 3
6 2
2 4
```

výstup

```
1.0
```

Adam strávi na každej stene pol hodiny.

vstup

```
3 6 6
1 3 5
5 3 1
```

výstup

```
2.0
```

Tu existuje veľa optimálnych možností tréningu. Pri jednej z nich strávi Adam na každej stene 40 minút.

vstup

```
2 3 3
1 100
1 6
```

výstup

```
0.5
```

Adam strávi pol hodiny na stene číslo 1. Jeho výsledná schopnosť v boulderingu bude 50 a jeho výsledná schopnosť v leade bude 3.

Obmedzenia a hodnotenie

Vo všetkých vstupoch platí $1 \leq n \leq 10^5$ a $1 \leq \beta, \lambda \leq 10^9$.
Vo všetkých vstupoch pre všetky i platí $1 \leq b_i, \ell_i \leq 10^9$.
Môžete tiež predpokladať, že **žiadne dve steny nemajú presne rovnaké parametre**:
usporiadané dvojice (b_i, ℓ_i) sú všetky navzájom rôzne.

Za riešenie efektívne pre vyššie uvedené obmedzenia môžete dostať plných 10 bodov.
Za riešenie efektívne pre $n \leq 10^5$ za predpokladu, že všetky b_i aj ℓ_i sú nanajvýš rovné 1000, môžete dostať 7 bodov.
Za riešenie efektívne pre $n \leq 1000$ (a ľubovoľné b_i a ℓ_i) môžete dostať 5 bodov.
Za riešenie fungujúce pre $n = 2$ môžete dostať dva body.

A-III-3 Hviezdne impérium a hypercestovanie

K tejto úlohe patrí študijný text uvedený na nasledujúcich stranách. Je identický s tým, ktorý ste už videli v zadaniach predchádzajúcich kôl.

Podúloha A (4 body). Cisár chce vybudovať systém hyperdiaľnic. Tento systém má spĺňať nasledujúce podmienky:

- Každá hyperdiaľnica musí spájať niektoré dva *susedné* systémy (teda také, ktoré spolu vedia komunikovať).
- Celé Hviezdne impérium musí byť prepojené hyperdiaľnicami – z každého systému sa musí dať po nich docestovať do každého iného.
- Z každého systému smú vychádzať nanajvýš tri hyperdiaľnice.

Navrhnite algoritmus, ktorý odpovie `ANO` práve vtedy, ak existuje nejaká vyhovujúca sada hyperdiaľnic.

Podúloha B (6 bodov). Ako alternatívne riešenie zvažuje cisár ešte aj jednoduchšiu možnosť: medzi každými dvoma susednými systémami postaviť hyperchodník. Takéto riešenie má len jeden potenciálny problém: v rozpočte má Hviezdne impérium dosť zdrojov len na m hyperchodníkov.

Navrhnite algoritmus, ktorý odpovie `ANO` práve vtedy, ak je celkový počet dvojíc susediacich systémov (v celom Hviezdnom impériu dokopy) menší alebo rovný m. Pri písaní algoritmu môžete predpokladať, že každý systém má prístup k (všade tej istej) testovanej hodnote m v premennej `m`.

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. **(Riešenia, ktorých hodnota k sa líši len o malú konštantu, budú hodnotené rovnakým maximálnym počtom bodov.)** 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 hviedznemu 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, Jakub Šimo
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2023