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 najneskôr do konca septembra.

Termíny 40. ročníka OI

- Riešenia domáceho kola je potrebné v kategórii A odovzdať najneskôr **15. 11. 2024**, v kategórii B najneskôr **30. 11. 2024**.
- Krajské kolo oboch kategórií prebehne 21. 1. 2025 zvlášť v každom kraji.
- Celoštátne kolo kategórie A sa bude konať v dňoch 19.-22. 3. 2025 (súťaží sa vo štvrtok a piatok).
- Detailnejšie informácie o priebehu súťaže nájdete na webe OI: https://oi.sk/?s=pravidla

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++, Python alebo Java). Presný zoznam podporovaných jazykov vrátane konkrétnych verzií ich kompilátorov a interpreterov nájdete v systéme na odovzdávanie riešení. Taktiež tam nájdete presnejší popis toho, ako sa majú vaše programy správať, napr. detaily realizácie vstupu a výstupu.

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.

Ak nie je v zadaní povedané ináč, riešenia **teoretických úloh** musia obsahovať:

- podstatné časti algoritmu ako program (v ľubovoľnom bežnom prog. jazyku) alebo ekvivalentný pseudokód
- podrobný slovný popis použitého algoritmu
- zdôvodnenie jeho správnosti
- diskusiu o efektivite zvoleného riešenia (odhad časovej a pamäťovej zložitosti)

Ak používate v programe netriviálne algoritmy alebo dátové štruktúry (napr. rôzne súčasti STL v C++), súčasťou slovného popisu algoritmu by mal byť stručný 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 NIVAM v tesnej súčinnosti so Slovenskou komisiou OI.

B-I-1 Byrokracia

Toto je praktická úloha. Riešiť sa dá v ľubovoľnom programovacom jazyku, odovzdávajú sa len správne riešenia pre vopred pripravené testovacie dáta. Detailnejšie pokyny k odovzdávaniu riešení sú uvedené nižšie.

Vybavovanie žiadostí na ministerstve byrokracie je vždy náročný a zdĺhavý proces. Vyzerá napríklad takto:

Kancelária 4: *„Ak sa chcete registrovať do portálu, musíte mať identifikačné číslo, dajú vám ho v kancelárii 8.“*
Kancelária 8: *„Ak chcete identifikačné číslo, doneste potvrdenie o správnosti osobných údajov z kancelárie 2.“*
Kancelária 2: *„Potvrdenie vám vydáme, ale najskôr si podajte žiadosť o overenie údajov v kancelárii 5.“*
Kancelária 5: *„Nie je problém. Aké máte identifikačné číslo? Že nemáte? Pridelia vám ho v kancelárii 8.“*
Kancelária 8: *„Ak chcete identifikačné číslo, doneste potvrdenie o správnosti osobných údajov z kancelárie 2.“*
...

Ministerstvo má n kancelárií, očíslovaných od 1 po n. Človek môže svoju návštevu ministerstva začať v ľubovoľnej z nich. Každá kancelária má jednoznačne určenú jednu ďalšiu kanceláriu, kam posielajú všetkých ľudí: vždy, keď príde človek do kancelárie s číslom i, pošlú ho do kancelárie a_i.

Je zjavné, že na takto fungujúcom ministerstve byrokracie človek nikdy nič nevybaví, len bude do nekonečna putovať po kanceláriách a (pravdepodobne márne) dúfať, že sa nad ním nejaký úradník niekedy zľutuje.

Zistite, koľko je na ministerstve takých kancelárií, pre ktoré sa môže stať, že ich jeden človek počas jednej svojej návštevy ministerstva navštívi aspoň stokrát.

Formát vstupu a výstupu

Na prvom riadku vstupu je jedno celé číslo n – počet kancelárií na ministerstve ($1 \leq n \leq 2\,000\,000$). Na druhom riadku je n medzerou oddelených čísel $a_1 \ldots a_n$, popisujúcich, do ktorej kancelárie nás pošlú z kancelárie číslo i ($1 \leq a_i \leq n$).

Na výstup vypíšte jedno číslo – celkový počet kancelárií, pre ktoré platí, že ich môže jeden človek pri vybavovaní jednej žiadosti navštíviť viac než stokrát.

Obmedzenia a hodnotenie

Táto úloha má 10 hodnotených vstupov. Za každý správne vyriešený vstup dostanete jeden bod.

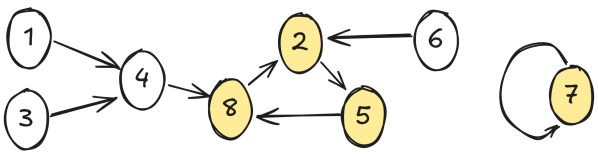
Počet kancelárií vo vstupoch sa postupne zvyšuje od 20 až po 2 000 000.

Vo vstupoch 1, 3, 5 a 7 zaručujeme, že existuje len jeden cyklus – bez ohľadu na to, v ktorej kancelárii začne človek riešiť svoj problém, kancelárie, ktoré eventuálne navštívi stokrát, budú vždy presne tie isté.

Príklad

vstup

```
8
4 5 4 8 8 2 7 2
```



výstup

```
4
```

Na obrázku vľavo môžeme vidieť schematický nákres ministerstva. Vyfarbené kancelárie sú tie, ktoré máme spočítať.
Človek, ktorý začne v kancelárii 4, eventuálne navštívi stokrát každú z kancelárií 8, 2 a 5. Človek, ktorý začne v kancelárii 7, ju eventuálne navštívi stokrát.

Odovzdávanie riešení

Zo stránky https://oi.sk/?d=rocnik si stiahnite ZIP archív obsahujúci 10 testovacích vstupov.
Vstupy sú nazvané `01.txt` až `10.txt`.
Vyrobte k čo najviac vstupom správne výstupy a uložte ich do súborov `sol01.txt` až `sol10.txt`.
Odovzdajte ZIP archív obsahujúci **zdrojový kód vášho programu** a tieto výstupné súbory.
Za každý správny výstupný súbor získate 1 bod.

B-I-2 Činka

Toto je praktická úloha. Riešiť sa dá v ľubovoľnom programovacom jazyku, odovzdávajú sa len správne riešenia pre vopred pripravené testovacie dáta. Detailnejšie pokyny k odovzdávaniu riešení sú uvedené nižšie.

Nachádzate sa v malej posilňovni. Majú tu práve jednu činku. Základom činky je tyč, na ktorej dva konce sa umiestňujú kotúče rôznych hmotností. Tyč váži 20 kilogramov. Okrem nej tu ešte majú $2n$ kotúčov: dva kotúče každej z hmotností $1, 2, \ldots, n$ kilogramov.

Činka je *správne postavená* ak sú na nej kotúče umiestnené symetricky a sú usporiadané podľa veľkosti – od najťažších pri strede po najľahšie pri kraji.

Príklad správne postavenej činky s celkovou hmotnosťou 104 kg.

Na efektívny tréning potrebujete správne postavenú činku hmotnosti presne v. Žiaľ, človek pred vami si po sebe neupratal. A tak namiesto toho, aby ste pred sebou mali prázdnu tyč a sadu kotúčov, máte pred sebou jeho činku, ktorá je síce správne postavená, ale pravdepodobne má nesprávnu hmotnosť.

Budete teda musieť nejaké kotúče pridať resp. odobrať, aby ste činku prispôsobili svojím potrebám. Samozrejme, kotúče môžete pridávať aj odoberať iba od krajov.

Aby vám zostalo čo najviac síl na samotný tréning, chcete minimalizovať úsilie potrebné na prestavanie činky – teda celkovú hmotnosť presunutých kotúčov.

Formát vstupu a výstupu

Na prvom riadku vstupu sú dve medzerou odelené čísla n a v: počet rôznych hmotností kotúčov a požadovaná hmotnosť činky v kilogramoch. Platí $1 \leq n \leq 50$ a $0 \leq v \leq 10^9$.

Zvyšok vstupu popisuje počiatočný stav činky. Na druhom riadku vstupu je číslo k ($0 \leq k \leq n$) – počet kotúčov na jednej strane počiatočnej činky. Na treťom riadku je k čísel $v_1, v_2, \ldots, v_k$ – hmotnosti kotúčov (v kilogramoch) na počiatočnej činke v poradí od stredu ku kraju. Môžete predpokladať, že činka je správne postavená – teda obe strany sú naložené symetricky a platí $v_1 > v_2 > \cdots > v_k$.

Ak na činke nie je možné dosiahnuť požadovanú hmotnosť v, na výstup vypíšte jeden riadok s číslom -1. Ak ju je možné dosiahnuť, vypíšte na prvý riadok výstupu dve medzerou oddelené celé čísla: počet operácií ℓ, ktoré chcete vykonať, a súčet m hmotností kotúčov, ktoré pri jednotlivých operáciách premiestnite.

Na nasledujúcich ℓ riadkoch potom vypíšte jednotlivé operácie v nasledujúcom formáte:

- „`L -`“ ak chcete odobrať najkrajnejší kotúč z ľavej strany tyče, „`R -`“ ak z pravej
- „`L +` x“ ak chcete pridať kotúč hmotnosti x na ľavú stranu tyče, „`R +` x“ ak na pravú.

Ak nie je povedané ináč (viď časť „Obmedzenia a hodnotenie“), hodnota m vo vašom riešení musí byť **najmenšia možná**. Ak existuje viacero optimálnych riešení, môžete vypísať ľubovoľné z nich. (Rôzne optimálne riešenia môžu mať rôznu hodnotu ℓ.)

Upozorňujeme, že ak vo svojom riešení viackrát manipulujete s tým istým kotúčom (napr. ho najprv z tyče dáte dole a neskôr ho na ňu pridáte späť), pri každej manipulácii treba znovu zarátať jeho hmotnosť do m.

Odovzdávanie riešení

Zo stránky `https://oi.sk/?d=rocnik` si stiahnite ZIP archív obsahujúci 10 testovacích vstupov.
Vstupy sú nazvané `01.txt` až `10.txt`.
Vyrobte k čo najviac vstupom správne výstupy a uložte ich do súborov `sol01.txt` až `sol10.txt`.
Odovzdajte ZIP archív obsahujúci **zdrojový kód vášho programu** a tieto výstupné súbory.
Za každý správny výstupný súbor získate 1 bod.

Obmedzenia a hodnotenie

Táto úloha má 10 hodnotených vstupov. Za každý správne vyriešený vstup dostanete jeden bod.

Niektoré vstupy sú zvolené tak, aby boli ľahšie riešiteľné. Pre niektoré vstupy vám dokonca dáme body aj za čiastočne správne riešenia. Presnejšie:

- Vo všetkých vstupoch platí, že ak je správny výstup -1, body dostanete len za výstup -1.
- Vo vstupoch 1 a 2 začínate s prázdnou tyčou – na druhom riadku vstupu je číslo 0 a tretí je prázdny.
- Vo vstupoch 3 a 4 netreba optimalizovať hmotnosť presunutých kotúčov. Pre tieto vstupy uznáme ako správnu ľubovoľnú postupnosť $\ell \leq 1000$ korektných operácii, ktorej výsledkom bude správne postavená činka s hmotnosťou v. Hodnota m vo vašom výstupe musí zodpovedať vašej postupnosti operácií, len nemusí byť najmenšia možná.
- Vo vstupoch 5 a 6 uznáme ako správne ľubovoľné riešenie, ktoré správne zistí najmenšiu potrebnú celkovú presunutú hmotnosť. Stačí teda vypísať ľubovoľnú hodnotu ℓ ($0 \leq \ell \leq 1000$) a za ňou správnu hodnotu m. Na zvyšku výstupu nezáleží – bod dostanete, aj ak vypíšete nesprávnu alebo žiadnu postupnosť operácií.

Príklady

vstup

```
4 20
0
```

výstup

```
0 0
```

Keďže tyč je na začiatku prázdna a sama osebe váži požadovaných 20 kg, nemusíme nič presúvať.

vstup

```
4 34
1
4
```

výstup

```
4 6
L + 2
L + 1
R + 2
R + 1
```

Počiatočná tyč má na každej strane kotúč s hmotnosťou 4 kg, váži preto $20 + 4 + 4 = 28$ kilogramov. Na každú stranu preto potrebujeme ešte pridať tri kilogramy. To môžeme dosiahnuť napríklad tak, že postupne na každú stranu najprv pridáme kotúč hmotnosti 2 a potom kotúč hmotnosti 1.
Riešenie, ktoré spraví tie isté štyri operácie ale v poradí „`L + 2`“, „`R + 2`“, „`R + 1`“, „`L + 1`“ je tiež korektné a na konci vyrobí tú istú činku.
Ešte ďalšie optimálne riešenie dostaneme, keď na každú stranu činky zo vstupu pridáme kotúč hmotnosti 3. Toto riešenie má tiež celkovú presunutú hmotnosť 6, aj keď vyrobí ináč vyzerajúcu činku.

vstup

```
4 35
2
4 3
```

výstup

```
-1
```

Správne postavená činka nemôže vážiť 35 kilogramov.

vstup

```
50 130
1
5
```

Kotúče hmotnosti 5 raz odoberieme a raz pridáme. Do presunutej hmotnosti sa teda rátaju dvakrát.

výstup

```
6 120
L -
R -
L + 50
R + 50
L + 5
R + 5
```

B-I-3 Ideme na túru

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte ***súbor vo formáte ZIP****. Jeho požadovaný obsah je popísaný v časti Odovzdávanie riešení.*

Paulinka sa cez prázdniny rozhodla vybrať na hory. Pozrela sa do máp, vypichla si desať zaujímavo vyzerajúcich miest, nahodila do navigačky poradie a vybrala sa na cestu. Keď sa o dvanásť hodín s boľavými nohami konečne vrátila domov, hovorila si – naozaj som si vybrala najlepšiu možnú trasu?

Paulinka má zoznam 10 zaujímavých miest očíslovaných od 1 po 10. Pre každú dvojicu miest i a j zistila, či z miesta i na miesto j vedie priama cesta, a ak áno, koľko trvá jej prejdenie. Tieto údaje si zapísala do tabuľky 10×10, kde v i-tom riadku a j-tom stĺpci je číslo $t_{i,j}$ udávajúce počet minút potrebných na prechod z miesta i do miesta j. V prípade, že sa z miesta i nevieme priamo dostať do j, je v tabuľke $t_{i,j} = -1$. A pre úplnosť, pre každé i je na hlavnej uhlopriečke tabuľky hodnota $t_{i,i} = 0$.

Dostať sa z miesta i do miesta j netrvá nutne rovnako dlho ako opačným smerom,[1] takže hodnota $t_{i,j}$ sa môže líšiť od hodnoty $t_{j,i}$. Dokonca sa môže stať aj to, že práve jedna z tejto dvojice hodnôt je rovná -1.

Paulinka chce spraviť jeden *okruh*. Presnejšie, chce **práve raz** navštíviť každé z miest a nakoniec sa vrátiť na miesto, kde začínala. Miesto, na ktorom okruh začne (a teda aj skončí), si Paulinka môže zvoliť ľubovoľne.

Pomôžte jej nájsť najkratší takýto okruh cez všetkých 10 miest.

Súťažná úloha

Niektoré informatické úlohy sú už zo svojej podstaty veľmi ťažké. Toto je jedna taká úloha. Pri takých úlohách nám preto často neostáva nič iné ako vyskúšať (takmer) všetky možnosti. Aj to je však dôležitá schopnosť a takéto skúšanie sa dá robiť rôzne dobre.

Na vstupe dostanete vyššie popísanú tabuľku s rozmermi 10×10. Vašou úlohou je pomocou skúšania rôznych možností nájsť taký okruh začínajúci a končiaci na tom istom mieste, ktorý sa dá prejsť za najmenší celkový čas (resp. podať správu, ak žiaden hľadaný okruh neexistuje).

Hodnotenie

Vaše riešenie tejto úlohy sa bude skladať z dvoch častí:

- **Slovného popisu** algoritmu riešiaceho súťažnú úlohu. V popise vysvetlite, ako a prečo váš algoritmus funguje (hlavne akým spôsobom skúšate rôzne okruhy) a ako efektívny je.
- **Implementácie** vášho algoritmu (v ľubovoľnom bežnom programovacom jazyku).

Každá časť bude hodnotená približne **polovicou bodov**. Keďže ide o teoretickú úlohu, odovzdaný program nebude testovaný na žiadnych vstupoch. Bude hodnotený len opravovateľom, pričom dôraz bude kladený na implementáciu hlavných myšlienok.

Majme 10 miest takých, že z každého existuje priama cesta do každého iného. Pre takúto sadu miest existuje presne $10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$ rôznych okruhov, z ktorých si Paulinka môže vybrať.[2] Máme totiž 10 možností, kde začať, potom 9 možností, kam sa odtiaľ vybrať, a tak ďalej. Počet možností postupne klesá, keďže každé miesto môžeme navštíviť len raz.

Riešenie, ktoré vyskúša prinajhoršom všetkých 10! možných okruhov vie získať nanajvýš 5 bodov.

Vzorové riešenie vyskúša v najhoršom prípade 9! okruhov a za takto dobré riešenie viete získať 8 bodov.

Zvyšné 2 body viete získať za také vylepšenia programu, ktoré mu umožnia *niekedy* (pre vhodné hodnoty na vstupe) vyskúšať menší počet okruhov – aj keď v najhoršom prípade bude stále potrebné vyskúšať 9!, resp. 10! možností.

(Podotkneme ešte, že existujú aj pokročilé algoritmy, ktoré *vždy* potrebujú prezrieť menej ako 9! možností. Takéto riešenie, ak ho niekto odovzdá, bude tiež hodnotené 10 bodmi.)

[1] Napríklad ísť do kopca je väčšinou pomalšie ako ísť z kopca.

[2] Súčin čísel od 1 po n označujeme $n!$ a nazývame „n faktoriál“.

Odovzdávanie riešení

Odovzdajte ZIP archív obsahujúci **dva** (ľubovoľne pomenované) **súbory**:

- Súbor vo formáte PDF obsahujúci slovný popis vášho riešenia úlohy.
- Súbor v ľubovoľnom príslušnom formáte obsahujúci implementáciu vášho riešenia úlohy.

Nápomocné knižničné funkcie

Väčšina moderných programovacích jazykov obsahuje knižničné funkcie, ktoré vám vedia pomôcť so skúšaním všetkých možností. V tejto časti uvádzame takúto funkciu pre jazyky Python a C++.[3]

Pozor, uvedené funkcie vám vedia pomôcť pri získavaní 8 bodov za túto úlohu, pre získanie plného počtu 10 bodov však budú nepostačujúce. Sú však veľmi dobrým odrazovým mostíkom (a 8 bodov vôbec nie je málo).

Predstavme si, že máme zadaných niekoľko čísel a chceli by sme vypísať všetky možné spôsoby ako tieto čísla zoradiť. Každý jeden spôsob zoradenia vytvára nejakú postupnosť zadaných čísel.

Dve postupnosti k čísel $A_1, \ldots, A_k$ a $B_1, \ldots, B_k$ vieme medzi sebou porovnať. Konkrétne, povieme že postupnosť A **je väčšia** ako postupnosť B ak je na *prvom* mieste (zľava), v ktorom sa tieto postupnosti líšia väčšia hodnota v postupnosti A. Presnejšie, nech i je najmenšie také číslo, že $A_i \neq B_i$. Potom postupnosť A je väčšia ako B práve vtedy, ak platí $A_i > B_i$.

Vďaka tomu, že rôzne postupnosti vieme porovnávať, vieme ich zoradiť od najmenšej po najväčšiu. No a knižničné funkcie v Pythone aj C++ nám umožňujú prechádzať týmto poradím.

Listing programu (Python)

```
from itertools import permutations

postupnost = [4, 8, 9, 11, 14]

for permutacia in permutations(postupnost):
    print(permutacia)
```

Prejsť všetkými postupnosťami zadaných čísel (permutáciami) vieme v Pythone pomocou funkcie `permutations()` z knižnice `itertools`. Výhodou je, že prejdeme cez všetky možnosti bez ohľadu na to, ktorou postupnosťou začneme (na rozdiel od C++). Nevýhodou ale je, že ak naša postupnosť obsahuje viacero rovnakých čísel, niektoré možnosti sa objavia viackrát.

Listing programu (C++)

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int main() {
    vector<int> postupnost = {4, 8, 9, 11, 14};

    do {
        for(int i = 0; i < 5; i++) cout << postupnost[i] << "␣";
        cout << endl;
    } while(next_permutation(postupnost.begin(), postupnost.end()));
}
```

V C++ vieme prejsť cez všetky postupnosti týmto spôsobom. Funkcia `next_permutation` zadané pole preusporiada tak, aby utvorila nasledujúcu permutáciu, a následne vráti true, ak sa jej to podarilo, resp. false, ak už žiadna ďalšia permutácia neexistuje. Treba si dať pozor, že musíme začať najmenšou možnou postupnosťou – takou, kde sú čísla usporiadané od najmenšieho po najväčšie. Výhodou ale je, že ak postupnosť obsahuje rovnaké čísla, každá možnosť sa vo výsledku objaví iba raz.

Bez ohľadu na to, ktorú funkciu sa rozhodnete použiť, odporúčame sa trochu s ňou pohrať a vyskúšať si, ako sa správa jej výstup pre rôzne začiatočné postupnosti.

[3] Pre iné jazyky odporúčame hľadať v ich dokumentácii generovanie permutácií. Ak to nepomôže a budete potrebovať poradiť, opýtajte na discorde Trojstenu https://discord.com/invite/F9HZP9b v kanáli `#inf-sutaze` alebo napíšte na misof@algo.sk.

Príklady

Nižšie uvedené príklady vstupu všetky začínajú riadkom, v ktorom je uvedený počet miest, ktoré daný vstup obsahuje. Niektoré príklady majú pre názornosť a lepšiu čitateľnosť miest menej ako 10. Vo vašich programoch môžete, ale nemusíte vedieť spracovať aj takéto vstupy. Presnejšie, pri implementácii riešenia môžete predpokladať, že **miest je presne desať**.

vstup

```
2
0 30
20 0
```

výstup

```
50
1 2
```

V tomto prípade zaujímajú Paulinku len dve miesta. Má na výber dva okruhy: buď začať na prvom mieste, následne ísť do druhého a vrátiť sa, alebo začať na druhom mieste, ísť sa pozrieť na prvé, a vrátiť sa. Oba okruhy jej zaberú 50 minút.

vstup

```
3
0 1 -1
1 0 1
-1 1 0
```

výstup

```
ziadny okruh neexistuje
```

Pre tento vstup neexistuje žiaden platný okruh. Ak by sme napr. začali na mieste 1, môžeme odtiaľ ísť len na miesto 2, potom na jediné nenavštívené miesto 3, ale odtiaľ sa už nevieme dostať priamo naspäť na miesto 1.

vstup

```
3
0 1 98
1 0 1
99 1 0
```

výstup

```
100
2 1 3
```

Okruh $1 \to 2 \to 3 \to 1$ *by Paulinke trval* $1 + 1 + 99 = 101$ *minút. Okruh* $2 \to 1 \to 3 \to 2$ *je lepší, trvá len* $1 + 98 + 1 = 100$ *minút. Tento okruh je jedným z optimálnych riešení – neexistuje žiaden rýchlejšie prejditeľný okruh, ktorý každé miesto navštívi práve raz. (Cesta* $2 \to 1 \to 2 \to 3 \to 2$ *síce trvá len 4 minúty, nie je to ale platný okruh – miesto* 2 *navštívi priveľakrát.)*

vstup

```
10
0 65 60 -1 55 -1 20 95 25 -1
85 0 75 35 15 65 70 40 25 55
-1 15 0 45 25 -1 20 15 35 5
10 75 30 0 65 85 85 5 60 50
20 90 -1 90 0 65 85 5 45 65
35 90 60 -1 55 0 40 -1 70 40
50 -1 45 85 20 45 0 60 -1 90
50 5 30 45 25 -1 90 0 -1 45
-1 60 60 5 65 -1 10 30 0 35
50 50 -1 80 25 50 85 60 95 0
```

výstup

```
200
2 9 4 3 10 6 1 7 5 8
```

Tento okruh mohla Paulinka prejsť za 200 minút. Všimnite si, že tento okruh by Paulinka nevedela ísť v opačnom smere: z miesta 9 *na miesto* 2 *sa nedá priamo dostať.*

B-I-4 Svetielkové kódovanie

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF obsahujúci slovný popis vášho riešenia.

Najlepšie kamarátky Alica a Hanka navštevujú síce tú istú školu, chodia však do iných tried. Kvôli rozdielnemu rozvrhu sa často stane, že sa celý deň nevidia. Chceli by sa preto stretnúť aspoň po škole, Alica však až ráno zistí, kedy jej začína futbalový tréning, nevie sa preto s Hankou dohodnúť na čase dopredu.

Majú však plán. Počas dňa majú obe vyučovanie v počítačovej učebni. Nie naraz, ale vedia, že najprv má informatiku Alica a až niekedy po nej Hanka. Dohodli sa preto, že Alica nechá Hanke v počítačovej učebni odkaz s časom stretnutia, ktorý si potom Hanka prečíta a budú sa môcť stretnúť. Odkaz však musí byť nenápadný, aby ho náhodou niekto nezničil či nevyhodil.

Na informatike pracujú tento rok s `micro:bit`om. Je to malý počítač, ku ktorému sa dajú pripojiť rôzne senzory, tlačidlá či svetielka, a ktorému sa dá jednoduchým spôsobom naprogramovať správanie. Na hodinách často pracujú s rôzne veľkými *štvorcovými* doštičkami, na ktorých sú malé LED svetielka. Doštičku s n riadkami a n stĺpcami si viete predstaviť ako mriežku $n \times n$, pričom v každom políčku mriežky je jedno svetielko.

Tieto doštičky sa dajú programovať tak, aby sa ľubovoľne rozsvietili – každé svetielko na doštičke môže byť zapnuté alebo vypnuté nezávisle od ostatných svetielok.

Alica na konci hodiny jednu doštičku naprogramuje tak, aby sa rozsvietila špecifickým spôsobom. Hanka si ju potom na začiatku svojej hodiny zapne, pozrie sa na Alicin odkaz a určí z neho, kedy sa stretnú.

Príklad

Predstavte si, že sa Alica s Hankou sa chcú stretnúť o nejakej presnej minúte medzi 16:00 a 16:20. V takomto prípade by sa mohli vopred dohodnúť napríklad tak, že Alica použije doštičku 5×5 a naprogramuje ju tak, aby sa na nej rozsvietilo x svetielok, kde x je minúta, o ktorej sa majú stretnúť. Hanka potom po zapnutí doštičky jednoducho spočíta, koľko svetielok na doštičke svieti, a podľa toho určí čas stretnutia.

Všimnite si, že väčšinu časov vie Alica zakódovať veľa rôznymi spôsobmi, keďže pri vyššie popísanom kódovaní vôbec nezáleží na tom, ktoré svetielka sú zapnuté, len na ich celkovom počte. Napr. pre zakódovanie času stretnutia 16:12 by si mohla vybrať ľubovoľných 12 políčok, ktoré by sa rozsvietili.

Iný spôsob kódovania by mohol vyzerať tak, že si dievčatá svetielka na doštičke očíslujú od 0 po 24. (Toto vedia spraviť napr. po riadkoch zhora dole a v každom riadku zľava doprava – teda prvý riadok dostane čísla 0-4, druhý 5-9, a tak ďalej.)

Na zakódovanie minúty x by potom Alica rozsvietila práve jedno svetielko – to, ktoré má priradené číslo x. Napr. v prípade času 16:10 by musela rozsvietiť svetielko číslo 10, teda najľavejšie v treťom riadku.

Všimnite si ešte, že obe vyššie popísané kódovania umožňujú Alici poslať aj niekoľko časov za koncom požadovaného rozsahu. Prvým spôsobom by si dievčatá vedeli poslať časy po 16:25, druhým len po 16:24.

Súťažná úloha

Vašou úlohou bude vymyslieť spôsob kódovania, ktoré môžu Alica a Hanka použiť na zakódovanie ľubovoľnej presnej minúty z rozsahu od 14:00 po 18:00 vrátane. Toto kódovanie musí byť jednoznačné – pre každý možný čas z tohto intervalu musí Alica vedieť, ktoré svetielka má rozsvietiť a Hanka musí z rozsvietených svetielok vedieť jednoznačne identifikovať, v ktorom čase sa majú stretnúť.

Všimnite si, že na kódovaní sa Alica s Hankou dohadujú dopredu, obe preto budú poznať zvolený spôsob. Navyše majú obe naozaj dobrú pamäť a vedia si zapamätať ľubovoľne komplikované pravidlá, výnimky a špecifikácie. Jediná vec, ktorú nepoznajú dopredu, je konkrétny čas, ktorý budú kódovať – vedia len, že bude aspoň 14:00 a nanajvýš 18:00.

Vaše riešenie bude hodnotené podľa toho, ako veľkú **štvorcovú doštičku** budú musieť Alica s Hankou na kódovanie využiť. Čím menšiu doštičku bude váš navrhnutý spôsob potrebovať, tým viac bodov môžete získať.

a) (4 body) Navrhnite kódovanie, ktoré dokáže s presnosťou na minútu zakódovať ľubovoľný čas od 14:00 po 18:00. Ak teda Alica zistí, že sa môžu s Hankou stretnúť o 15:47, musí vedieť, ktoré svetielka rozsvietiť, aby si Hanka prečítala v správe práve tento čas.
Ľubovoľné správne riešenie získa aspoň 2 body. Plný počet bodov môžete získať za riešenie, ktorému stačí doštička 3×3.

b) (2 body) Odôvodnite, prečo v predchádzajúcej podúlohe **nemôže** existovať správne riešenie, ktoré použije doštičku 2×2.
Túto podúlohu môžete riešiť nezávisle od toho, či ste vyriešili podúlohu a).

c) (4 body) Doštičky `micro:bit`, ktoré majú v škole, sa občas trochu pokazia. Presnejšie, môže sa stať, že keď Hanka doštičku zapne, **práve jedno svetielko** sa nebude správať podľa Alicinho programu ale presne opačne – teda ak Alica chcela, aby bolo zhasnuté, tak bude Hanke svietiť, a naopak. Toto dievčatám vcelku komplikuje situáciu.

Pomôžte im a navrhnite, ako si majú zakódovať správu tak, aby platilo:

- ak celá doštička funguje, tak Hanka správne zistí čas stretnutia
- ak sa niektoré svetielko pokazí, tak Hanka správne zistí, že sa niektoré svetielko pokazilo (pričom môže, ale nemusí, aj zistiť správny čas stretnutia)

V príklade, v ktorom sme si svetielka očíslovali od 0 po 24, by Alica mohla naďalej používať to isté kódovanie. Ak Hanka uvidí len jedno svetielko, určí podľa neho správny čas stretnutia. Ak uvidí dve svetielka, bude vedieť, že musela nastať chyba – okrem Alicinho sa rozsvietilo ešte nejaké iné. A rovnako bude vedieť, že nastala chyba, ak nebude svietiť žiadne svetielko – vtedy musí byť pokazené práve to, ktoré Alica rozsvietila.

Aj v tejto podúlohe chcú Alica s Hankou vedieť zakódovať ľubovoľnú minútu od 14:00 po 18:00. Ľubovoľné správne riešenie získa aspoň 2 body. Plný počet bodov môžete aj tentokrát získať za riešenie, ktorému stačí doštička 3×3.

V riešení podúloh a) aj c) nezabudnite popísať:

- Jednoznačný spôsob, akým Alica zo zadaného času určí, ktoré svetielka sa majú rozsvietiť.
- Jednoznačný spôsob, akým Hanka zo svetielok zistí, aký čas ukazujú (prípadne, že sa správa pokazila).

ŠTYRIDSIATY ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Truc Lam Bui, Ján Hozza, Paulína Smolárová
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: NIVAM – Národný inštitút vzdelávania a mládeže, Bratislava 2024