Toto sú návodné úlohy k domácemu kolu 40. ročníka Olympiády v informatike. Ide teda o sadu ľahších úloh, ktoré tematicky súvisia so súťažnými úlohami. Riešenie týchto úloh môže byť dobrou prípravou na riešenie súťažných úloh. Za riešenia týchto návodných úloh nie sú žiadne body do súťaže. O ich riešení môžete ľubovoľne diskutovať – so svojim učiteľom, inými súťažiacimi aj kýmkoľvek iným.

Návodné úlohy k domácemu kolu OI: kategória A

A-I-1 Dejepis

1. Vymyslite pre súťažnú úlohu vstup s piatimi udalosťami, pre ktorý platí, že riešenie preň neexistuje, ale ak vynecháme ľubovoľnú udalosť (a všetky vzťahy, ktoré sa jej týkajú), tak už riešenie existovať bude.

2. Koľko najmenej vzťahov musí byť vo vstupe, ktorý je riešením predchádzajúcej návodnej úlohy? Prečo?

3. V krajine je n miest (očíslovaných od 1 po n). Medzi mestami vedie m jednosmeriek: pre každé i od 1 po m máme jednosmerku z mesta x_i do mesta y_i. Za prechod po ceste i treba zaplatiť $d_i > 0$ eur.

   Monika napísala postupnosť n čísel: $v_1, \ldots, v_n$. Tvrdí o nich, že pre každé j platí, že najlacnejší spôsob, ako sa dostať z vrcholu 1 do vrcholu j, má celkovú cenu presne v_j.

   Napíšte program, ktorý načíta všetky vyššie uvedené čísla (popis krajiny aj Monikinu postupnosť) a čo najefektívnejšie skontroluje, či Monika hovorí pravdu.

   (Správne hodnoty v_j samozrejme vieme pomerne efektívne vypočítať. Viete však požadovanú kontrolu spraviť *ešte efektívnejšie*?)

4. V predchádzajúcej návodnej úlohe vynechajme podmienku $d_i > 0$. Inými slovami, v našej novej situácii môžu niektoré d_i byť aj nulové alebo záporné – čiže prejazd niektorými jednosmerkami môže byť zadarmo, alebo dokonca zaň môžeme dostať aj zaplatené.

   Budeme mať ale predsa len aspoň jeden dodatočný predpoklad: o krajine bude zaručené, že nikde v nej neexistuje cyklus (t.j. postupnosť na seba nadväzujúcich jednosmeriek, ktorá skončí v meste, v ktorom začala) na ktorom je súčet všetkých d_i záporný. Ľudskou rečou, nikde v krajine sa nedá jazdiť dokola a zarábať si tým.

   Pozrite sa na svoj program, ktorý ste navrhli ako riešenie predchádzajúcej návodnej úlohy. Funguje tento program aj v tejto novej, všeobecnejšej situácii? A prečo?

5. Navrhnite program, ktorý načíta popis krajiny (rovnaký ako v predchádzajúcich návodných úlohách, vrátane celých čísel d_i pre jednotlivé jednosmerky) a overí, či táto krajina spĺňa alebo nespĺňa predpoklad z predchádzajúcej návodnej úlohy – teda či v tejto krajine je nejaký alebo nie je žiaden cyklus so záporným súčtom hodnôt d_i.

A-I-2 Obchodovanie

1. Vyriešte súťažnú úlohu za dodatočných predpokladov, že existuje len jeden typ tovaru (tovar číslo 1) a Malcolm smie za celý čas spraviť len jeden nákup a potom jeden predaj. (Udalosti na vstupe môžu byť ľubovoľné, ponuky na nákup a na predaj sa môžu ľubovoľne striedať.)

   Nájdite riešenie, ktorého časová zložitosť je lineárna od počtu udalostí (a nie kvadratická).

2. Upravte svoje riešenie predchádzajúcej návodnej úlohy tak, aby malo *konštantnú* pamäťovú zložitosť. (Takéto riešenie musí vstup spracúvať postupne, nesmie si ho ani len celý načítať a uložiť do poľa. Časová zložitosť musí naďalej zostať lineárna.)

3. Upravte svoje riešenie prvej návodnej úlohy tak, aby fungovalo s ľubovoľnými typmi tovaru – teda oproti súťažnej úlohe už máme navyše len predpoklad, že Malcolm smie za celý čas spraviť len jeden nákup a potom jeden predaj.

   (Ak teda napr. Malcolm kúpi a potom predá tovar typu 7, už si minul svoj jeden nákup a jeden predaj a už nesmie robiť žiadne ďalšie operácie – ani s typom 7 ani s inými typmi tovaru. Opäť by sme v ideálnom prípade chceli mať riešenie s lineárnou časovou zložitosťou.)

4. V rade na polici je n fliaš, v i-tej zľava je p_i ml pomarančového džúsu. Peter môže raz prejsť okolo police zľava doprava a počas toho prechodu nejaké fľaše vypiť. Nesmie však nikdy piť z dvoch po sebe idúcich fliaš – teda z i-tej fľaše ($i > 1$) smie piť len ak nepil z fľaše $i - 1$.

   Napíšte program, ktorý spočíta, koľko najviac džúsu vie Peter dokopy vypiť, ak si správne zvolí, ktoré fľaše vypiť a ktoré nechať nevypité.

   Nájdite riešenie, ktoré bude dostatočne efektívne aj pre vstupy s $n = 1000$ (pričom p_i môžu byť ľubovoľné kladné celé čísla v rozsahu bežných premenných).

5. V rade na polici je n fliaš, v i-tej zľava je buď p_i ml pomarančového džúsu alebo j_i ml jablkového (pre každé i je práve jedno z čísel p_i a j_i kladné a druhé nulové). Peter môže raz prejsť okolo police zľava doprava a počas toho prechodu nejaké fľaše vypiť. Nesmie však nikdy piť dvakrát po sebe ten istý džús, musí striedavo piť pomarančový a jablkový.

   Napíšte program, ktorý spočíta, koľko najviac džúsu vie Peter dokopy vypiť – nezáleží nám na druhu, len na celkovom vypitom objeme.

   Aj tentokrát chceme riešenie, ktoré bude dostatočne efektívne aj pre vstupy s $n = 1000$ (pričom p_i môžu byť ľubovoľné kladné celé čísla v rozsahu bežných premenných).

6. V oboch predchádzajúcich návodných úlohách sa zamyslite, či by vaše riešenie bolo ešte stále efektívne aj pre $n = 500\,000$. Ak nie, viete nájsť také, ktoré bude?

A-I-3 Domy

1. Ako funkciu čísla n vyjadrite presne (vzorcom), akú maximálnu výšku môže mať najvyšší dom po tom, ako podľa pravidiel zo zadania postavíme celú našu ulicu.

2. Napíšte program, ktorý načíta čísla n a p a potom vypočíta odpoveď na nasledujúcu otázku: Ak máme ulicu dĺžky n ako v zadaní a povoleným spôsobom na ňu postavíme domy tak, aby každý mal nanajvýš p poschodí, koľko najviac poschodí *dokopy* môžu tieto domy mať?

   (Dajte si pozor na prípadné okrajové prípady. Ako napr. vyzerá odpoveď pre $n = p = 7$?)

3. Napíšte program, ktorý načíta čísla n, i a p a potom vypočíta odpoveď na nasledujúcu otázku: Ak máme ulicu dĺžky n ako v zadaní, vieme na ňu povoleným spôsobom na ňu postaviť domy tak, aby mal dom i presne p poschodí?

4. Mesto zvažuje pridať ešte jeden zákaz: od budúceho piatku vraj bude zakázané postaviť jednoposchodový dom, ktorého oba susedné domy sú vyššie (teda nutne presne dvojposchodové).

   Janko tvrdí, že nás tento zákaz nemusí trápiť, lebo nijak nezmení *hodnotu* optimálneho riešenia – teda že pre ľubovoľný vstup bude platiť, že maximálny počet domov, ktoré vieme predať zákazníkom, by ostal rovnaký aj keby vstúpil do platnosti tento nový zákaz.

   Má Janko pravdu? Viete jeho tvrdenie dokázať, či prípadne vyvrátiť konkrétnym protipríkladom?

A-I-4 Triedička

Vo všetkých návodných úlohách, v ktorých píšete program pre triedičku, predpokladajte rovnaké obmedzenia ako v súťažných úlohách – teda jadier je presne 2048 a váš program má obsahovať nanajvýš 256 inštrukcií.

1. Vyskúšajte si použiť interpreter – či už na webe na adrese https://oi.sk/apps/triedicka/ alebo pythonovú verziu, ktorú si tam môžeš stiahnuť. Vyskúšajte si ho spustiť na príkladoch zo študijného textu a ubezpečte sa, že rozumiete formátu, v akom vypisuje detaily výstupu.

2. Každé jadro dostane na vstupe jeden člen postupnosti: celé číslo od 0 po 10^9. Napíšte program, ktorý zistí, či sú v tejto postupnosti samé nuly. Presnejšie, jadro s ID=0 má na výstup dať 1 ak áno a 0 ak nie.

3. Každé jadro dostane na vstupe jeden člen postupnosti: celé číslo od 0 po 10^9. Napíšte program, ktorý zistí, či je táto postupnosť usporiadaná (od najmenšej hodnoty po najväčšiu). Presnejšie, jadro s ID=0 má na výstup dať 1 ak áno a 0 ak nie.

   (Viete pri tom využiť riešenie predchádzajúcej návodnej úlohy?)

4. Každé jadro dostane na vstupe jeden člen postupnosti: celé číslo od 0 po 10^9. Vstup čísla i označíme a_i. Každé jadro má dať na výstup číslo, ktoré je vstupom jadra o štyri pozície napravo (cyklicky) – teda ak je jadier n, jadro i má dať na výstup hodnotu $a_{(i+4) \bmod n}$. Napíšte program, ktorý toto spraví.

5. Každé jadro dostane na vstupe jeden člen postupnosti: celé číslo od 0 po 10^9. Každé jadro má dať na výstup hodnotu, ktorú dostalo na vstupe jadro 0. Napíšte program, ktorý toto spraví.

Návodné úlohy k domácemu kolu OI: kategória B

B-I-1 Byrokracia

Postupnosť kancelárií $k_1, k_2, \ldots k_p$, ktoré na seba postupne odkazujú tak, že kancelária k_1 posiela ľudí do k_2, tá do k_3 a tak ďalej až kancelária k_p posiela ľudí do k_1, budeme nazývať *cyklus*.

1. Koľko najmenej a koľko najviac cyklov môže mať ministerstvo s n kanceláriami?

2. Zdôvodnite, že v každom ministerstve musí existovať aspoň jeden cyklus.

3. Zdôvodnite, že nemôže existovať také ministerstvo, v ktorom by jedna kancelária bola súčasťou dvoch rôznych cyklov.

4. Nech všetky kancelárie ministerstva tvoria jeden veľký cyklus, v ktorom kancelária 1 posiela do kancelárie 2, tá do kancelárie 3 až kancelária n posiela do 1.

   Ak človek začne v kancelárii 26, ktorú kanceláriu navštívi stýkrát ako prvú?

5. Nech je o ministerstve zaručené, že obsahuje buď jeden alebo dva cykly. (Nie všetky kancelárie musia byť súčasťou týchto dvoch cyklov.) Navrhnite algoritmus, ktorý zistí, ktorá z týchto možností nastala.

6. Nižšie je uvedená časť kódu v jazyku Python a C++, ktorá simuluje prechádzanie cez kancelárie. Kancelária `i` odkazuje na kanceláriu `A[i]`. Upravte tento program tak, aby zastal, akonáhle sa ocitne v kancelárii, v ktorej už niekedy predtým bol.

   #### Listing programu (Python)

   ```python
   aktualna_kancelaria = 1
   while True:
       aktualna_kancelaria = A[aktualna_kancelaria]
   ```

   #### Listing programu (C++)

   ```cpp
   int aktualna_kancelaria = 1;
   while(true) {
       aktualna_kancelaria = A[aktualna_kancelaria];
   }
   ```

B-I-2 Činka

Keďže na začiatku aj konci musí byť činka symetrická, môžeme sa pri riešení zamerať iba na jednu stranu činky. V týchto návodných úlohách sa teda budeme pýtať iba na nakladanie váhy na jednej strane činky.

1. Ak máme kotúče 1 až n, akú najľahšiu a akú najťažšiu jednu stranu činky vieme naložiť?

2. Vieme na činku vyskladať všetky možné hmotnosti v intervale udanom riešením predchádzajúcej úlohy?

3. Ak začíname s prázdnou tyčou, môže existovať viac spôsobov ako na jednu jej stranu naložiť váhu v. Môžu sa tieto možnosti líšiť v celkovej hmotnosti presunutých kotúčov?

4. Rôzne spôsoby, ako na jednu stranu prázdnej tyče naložiť celkovú váhu v, sa môžu líšiť v počte použitých kotúčov. Aký postup nakladania by sme zvolili, ak by sme chceli pri nakladaní váhy v zaručene naložiť *najmenší možný počet* kotúčov?

5. Ak máme algoritmus, ktorý vie naložiť prázdnu činku váhou v, ako ho vieme využiť na všeobecné riešenie za predpokladu, že nemusíme minimalizovať celkovú presunutú váhu?

B-I-3 Ideme na túru

Permutácia čísel 1 až n je ľubovoľná postupnosť dĺžky n, v ktorej sa každé číslo 1 až n nachádza práve raz.

1. Koľko existuje permutácií čísel 1 až 3? Vypíšte ich všetky.

   Koľko existuje permutácií čísel 1 až 4? Vypíšte ich všetky.

2. Koľko existuje okruhov medzi 4 miestami, ak predpokladáme, že medzi každou dvojicou miest vedie cesta? Vypíšte ich.

   Sú všetky tieto okruhy **rôzne** pre potreby nášho riešenia? V čom sa líšia od permutácií?

   Ktoré z okruhov skúšame zbytočne? Vieme určiť, že sú zbytočné už *počas* ich generovania?

3. Vo Vysokých Tatrách je n lokalít (očíslovaných od 1 po n) a medzi nimi vedie m obojsmerných turistických chodníkov (očíslovaných od 1 po m). Pre každý chodník i sú zadané čísla x_i, y_i dvoch lokalít, ktoré spája. Lokalita 1 je Štrbské Pleso.

   Eliška by si chcela spraviť výlet, ktorý vyzerá tak, že začne na Štrbskom Plese a potom sa prejde **dvoma na seba naväzujúcimi** chodníkmi. (Ak napr. existujú chodníky $1-7$ a $3-7$, tak jeden možný výlet je $1 \to 7 \to 3$ a druhý $1 \to 7 \to 1$.)

   Napíšte program, ktorý načíta popis Vysokých Tatier, vhodne si ho uloží a potom vygeneruje všetky možné výlety pre Elišku.

4. Chodníkov v Tatrách nie je nejak veľa: ich počet m je rádovo menší ako n^2, z každej lokality ich vychádza len zopár. Zamyslite sa, či v takejto situácii vaše riešenie predchádzajúcej úlohy neskúša zbytočne veľa možností. Ak áno, dá sa ho nejak vylepšiť? Ideálne by sme chceli, aby počet prezretých možností bol (ak nie presne, tak aspoň približne) rovný počtu nájdených a vypísaných výletov.

5. Jakub by si chcel spraviť podobný výlet ako Eliška, ale tvorený až **tromi** po sebe idúcimi chodníkmi. Aj jemu by sme chceli napísať program, ktorý mu vygeneruje všetky možné výlety.

   Predstavme si, že sme už vyriešili úlohu pre Elišku a naprogramovali sme si funkciu, ktorej zadáme číslo štartovej lokality (Eliška by jej zadala 1 pre Štrbské Pleso) a funkcia nám vygeneruje všetky možné výlety. Dá sa takúto funkciu nejak šikovne využiť pri implementácii programu pre Jakuba?

B-I-4 Svetielkové kódovanie

1. Koľkými rôznymi spôsobmi vieme rozsvietiť doštičku 2×2? Ktoré to sú?

   Koľkými rôznymi spôsobmi vieme rozsvietiť doštičku 3×3?

2. V bežnom živote sa stretávame s veľkým množstvom kódov – obrázky, symboly či zvuky *kódujú* rôzne významy na základe vopred dohodnutých (a často verejných) pravidiel. Tieto pravidlá môžu, ale nemusia, mať vnútornú logiku.

   Napríklad je jasné, prečo päť zazvonení na veži signalizuje päť hodín. Ale pri Morseovej abecede je písmeno `D` kódované ako „dlhá, krátka, krátka" len preto, že sme sa na tom tak dohodli.

   Aké kódy poznáte? Skúste nájsť aspoň tri, s ktorými sa bežne stretávate.

3. Pri komunikácii počítačov sa často používa ASCII kódovanie, ktoré kóduje znaky (písmená, čísla, diakritiku, ...) do binárnej sústavy. Nájdite na internete ASCII tabuľku a prečítajte si niečo o jej využití.

4. Čísla kreditných kariet či rodné čísla (a mnoho ďalších) vedia rozoznávať chybné údaje. Ak sa pri písaní svojho rodného čísla pomýlite v jednej cifre, nemôžete dostať platné rodné číslo iného človeka.

   Každé platné číslo totiž spĺňa nejakú vlastnosť, ktorá sa zmenou cifry poruší. Aká podobná vlastnosť doštičky sa môže zmeniť pri chybe jedného svetielka?