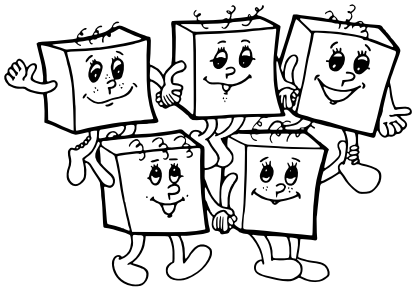
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH

http://oi.sk/



dvadsiaty siedmy ročník
školský rok 2011/2012

zadania krajského kola
kategória B

Priebeh krajského kola

Krajské kolo 27. ročníka Olympiády v informatike, kategória B, sa koná 24. 1. 2012 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.
- 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.

B-II-1 Letenka

Monika opäť raz cestuje kamsi na opačný koniec sveta. Potrebovala by si kúpiť letenku. Samozrejme, čo najlacnejšiu. Avšak prestupovať je otrava, preto je Monika ochotná prestupovať najviac dvakrát.

Súťažná úloha

Napíšte program, ktorý načíta ceny jednotlivých letov a nájde Monike tri možné cesty: najlacnejší priamy let, najlacnejší let s najviac jedným prestupom a najlacnejší let s najviac dvomi prestupmi.

Formát vstupu

Každé letisko má tzv. IATA kód, tvorený tromi veľkými písmenami. Napr. bratislavské letisko je `BTS`, košické je `KSC` a popradské je `TAT`. V prvom riadku vstupného súboru sú dva názvy letísk: odkiaľ a kam Monika cestuje.

V druhom riadku vstupného súboru je číslo ℓ: počet priamych letov, na ktoré si Monika môže kúpiť lístok.

Nasleduje ℓ riadkov, každý z nich popisuje jeden let. Popis letu obsahuje dva kódy letísk (odkiaľ a kam letí) a cenu letenky v eurách (celé číslo od 1 do 10 000).

Lety sú jednosmerné, teda napr. letom „`BTS KSC 47`“ sa nedá dostať z Košíc do Bratislavy. Pre každý let bude platiť, že koncové letisko je rôzne od začiatočného. Pre každú dvojicu letísk môže existovať viacero letov z prvého na druhé z nich. Tieto lety môžu mať navzájom rôzne ceny.

Nech n je počet rôznych letísk, ktoré sa aspoň raz vyskytnú na vstupe. Ako najväčší možný vstup si môžete predstaviť vstup, v ktorom $n = 10\,000$ a $\ell = 1\,000\,000$. Ak to pri svojom riešení potrebujete, môžete teda predpokladať, že rádovo n^2 celých čísel sa do pamäte ešte zmestí.

Formát výstupu

Vypíšte presne tri riadky. V prvom riadku uveďte cenu najlacnejšieho priameho letu, v druhom cenu najlacnejšieho letu s max. 1 prestupom a v treťom cenu najlacnejšieho letu s max. 2 prestupmi.

Ak neexistuje žiadne riešenie, namiesto ceny letu vypíšte reťazec „`neexistuje`“.

Príklady

vstup

```
VIE ZRH
2
VIE MUC 23
MUC ZRH 24
```

výstup

```
neexistuje
47
47
```

Priamy let neexistuje.

Najlacnejší let z Viedne (VIE) do Zurichu (ZRH) s najviac jedným prestupom je cez Mníchov (MUC) za 23+24 = 47.

Tento let je zároveň najlacnejším aj vtedy, ak dovolíme dva prestupy – lebo aj tak nič iné nemáme na výber.

vstup

```
ZRH VIE
8
ZRH VIE 1200
ZRH VIE 470
VIE ZRH 10
ZRH MUC 120
ZRH FRA 220
MUC FRA 30
MUC VIE 220
FRA VIE 110
```

výstup

```
470
330
260
```

B-II-2 Benátky

Hlavnú ulicu v Benátkach tvorí n domov stojacich jeden vedľa druhého v rade. Rôzne domy môžu mať rôzne počty poschodí. Domy v Benátkach sú stavané tak, že sa medzi nimi dá na ľubovoľnom poschodí prechádzať. To aby si domorodci nezamočili nohy, keď idú niekam na opačný koniec ulice. Aj po strechách budov sa dá chodiť.

Problém je ale v tom, že Benátky sa potápajú. Historické zdroje uvádzajú čísla medzi 0.4 a 2.4 mm/rok. Jeden však nikdy nevie, či zajtra nezačnú klesať rádovo rýchlejšie. No a časom to povedie k tomu, že niektoré domy skončia celé pod vodou a Hlavná ulica sa rozpadne na niekoľko ostrovov. A to bude raj pre gondolierov, ktorí budú potom všetkých obyvateľov medzi ostrovami prevážať.

Súťažná úloha

Na vstupe dostanete počet domov n a pre každý dom počet poschodí, ktoré má (celé číslo od 1 po n). Napíšte čo najefektívnejší program, ktorý zistí všetky nasledujúce údaje:

- Na koľko ostrovov bude rozdelená ulica, keď voda zatopí prvé poschodia všetkých domov?
- Na koľko ostrovov bude rozdelená ulica, keď voda zatopí druhé poschodia všetkých domov?
- ...
- Na koľko ostrovov bude rozdelená ulica, keď voda zatopí n-té poschodia všetkých domov?

(Ak voda práve zatopila k poschodí každého domu, strechy k-poschodových domov sú ešte suché.)

Pozor! Existujú rozlične efektívne riešenia tejto úlohy. Prvé, ktoré nájdete, nemusí nutne byť to najlepšie.

Formát vstupu

V prvom riadku vstupu je celé číslo n. V druhom riadku sú medzerami oddelené počty poschodí v domoch na Hlavnej ulici, v poradí v akom po sebe na ulici nasledujú. Výška každej budovy je celé číslo od 1 po n.

Formát výstupu

Na výstup vypíšte n čísel, pričom i-te z nich má byť počet ostrovov vo chvíli, kedy voda zatopí spodných i poschodí každého domu.

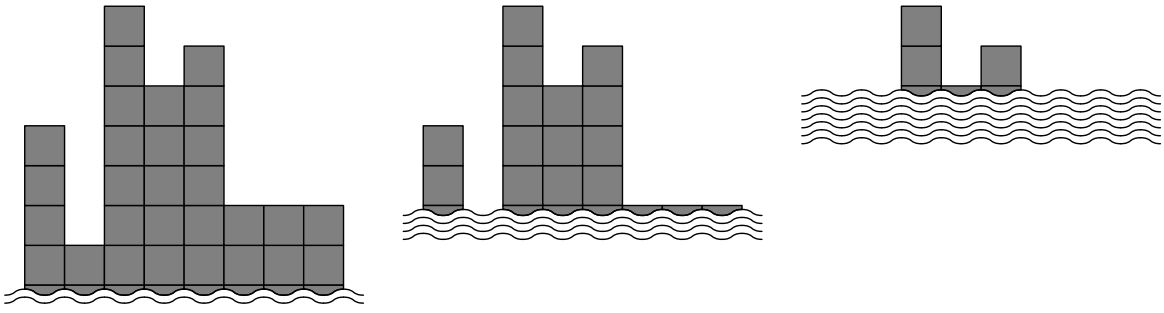
Príklad

vstup

```
8
4 1 7 5 6 2 2 2
```

výstup

```
1 2 2 2 1 2 1 0
```



Na obrázku vľavo je situácia v súčasnosti, ktorú popisujú vstupné dáta z príkladu.
Celá ulica je zatiaľ jedným veľkým ostrovom.

Na obrázku uprostred je situácia, keď už voda zatopila dve poschodia.
Vidíme, že ulicu teraz tvoria 2 ostrovy, preto je druhé číslo vo výstupe 2.

Na obrázku vpravo je situácia po zatopení piatich poschodí.
Opäť máme len 1 ostrov, preto je piate číslo vo výstupe 1.

B-II-3 Aničkine darčeky

Anička mala nedávno narodeniny. Keďže Miško uhádol presný dátum, zorganizoval jej veľkú párty a pozval všetkých jej kamarátov. Tí sú všetci slušne vychovaní a nezabudli so sebou priniesť každý po jednom darčeku.

Keď už svitalo a odišiel aj posledný hosť, dojatá Anička si ešte raz poobzerala všetky tie nádherné darčeky. Každému z nich určila hodnotu a podľa nej ich usporiadala do dlhého radu, ktorý končil až kdesi v kuchyni.

Keď sa ale potom uložila spať, prisnilo sa jej, že príde veľká bieda. Aničku ale len tak niečo nezlomí. Rozhodla sa, že ak ozaj nastanú zlé časy a bude potrebovať určitý obnos peňazí, vyberie si jeden darček, ktorý predá. Zároveň jej ale nesmie byť ľúto, že stačilo predať aj niečo menej hodnotné.

Súťažná úloha

Podúloha a) *(6 bodov)*

V pamäti už je načítané pole celých čísel A, v ktorom sú uložené ceny všetkých n darčekov. Toto pole je *usporiadané* od najmenšej ceny po najväčšiu.

Vašou úlohou je napísať *čo najefektívnejšiu funkciu*, ktorá dostane ako parametre pole A, počet darčekov n a sumu p, ktorú Anička potrebuje. Táto funkcia musí nájsť *najlacnejší* darček, ktorého cena je *väčšia alebo rovná* ako p. Návratovou hodnotou vašej funkcie by mala byť cena dotyčného darčeka, prípadne číslo -1, ak žiaden darček nie je dostatočne drahý.

Dobrá rada: Anička peniaze potrebuje čo najrýchlejšie. Keby vaša funkcia musela prechádzať celé pole, asi by dovtedy Anička umrela od hladu. Snažte sa nájsť riešenie, ktoré odpoveď vypočíta rádovo rýchlejšie.

Upozornenie: Naozaj uveďte presnú implementáciu v nejakom vami zvolenom programovacom jazyku. Riešenia obsahujúce len slovný popis algoritmu budú hodnotené veľmi prísne!

Príklad: Majme $n = 12$ darčekov, ich ceny nech sú $A = (2, 3, 3, 3, 5, 8, 9, 10, 23, 32, 47, 99)$.

- Pre hodnotu $p = 4$ by vaša funkcia mala vrátiť hodnotu 5.
- Pre hodnotu $p = 5$ by vaša funkcia mala tiež vrátiť hodnotu 5.
- Pre hodnotu $p = 17$ by vaša funkcia mala vrátiť hodnotu 23.
- No a pre hodnotu $p = 100$ by vaša funkcia mala vrátiť hodnotu -1 (nemáme darček s cenou ≥ 100)

Podúloha b) *(1 bod)*

Na koľko políčok poľa A sa (v najhoršom možnom prípade) pozrie tvoj algoritmus riešiaci podúlohu a)? Nepotrebujeme presné číslo, stačí nám vedieť, ako rádovo závisí tento počet od čísla n.

Podúloha c) *(3 body)*

Na koľko políčok poľa A sa (v najhoršom možnom prípade) musí pozrieť úplne optimálny algoritmus? Svoje tvrdenie dokážte.

Kostra riešenia podúlohy a)

V Pascale by riešenie podúlohy a) malo vyzerať približne nasledovne:

```
function najdi_darcek(var A : array of longint; n : longint; p : longint) : longint;
var odpoved : longint;
    { sem napis deklaracie ostatnych premennych }
begin
    { ceny darcekov su A[0] az A[n-1] }
    { sem napis program, ktory vypocita spravnu cenu darceka do premennej odpoved }
    najdi_darcek := odpoved;
end;
```

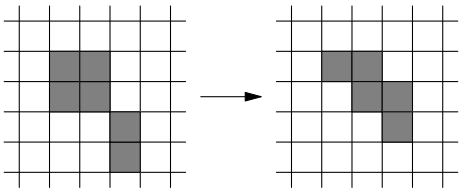
A v C zase napríklad nasledovne:

```
int najdi_darcek(int *A, int n, int p) {
    // ceny darcekov su A[0] az A[n-1]
    // sem napis program, ktory vypocita spravnu cenu darceka do premennej odpoved
    return odpoved;
}
```

B-II-4 Čierne štvorce sú tu opäť

Rovnako ako v domácom kole, aj túto sadu úloh ukončíme jednou o prefarbovaní štvorcov. Najskôr si ale pripomenieme, ako naše prefarbovanie vlastne funguje.

Predstavme si nekonečnú štvorcovú sieť. Na začiatku sú skoro všetky štvorce, ktoré ju tvoria, biele, len n ich je čiernych. Každú minútu pre každý štvorec vypočítame jeho novú farbu: Pozrieme sa na staré farby jeho, štvorca naľavo od neho a štvorca pod ním. Ako jeho novú farbu vyberieme tú, ktorú vidíme aspoň dvakrát. Tieto zmeny sa udejú naraz pre všetky štvorce. Na obrázku je príklad toho, ako sa po jednej minúte zmení situácia.



Súťažná úloha

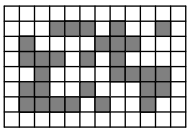
Podúloha a) *(2 body za hociktoré jedno rozmiestnenie / 3 body za obe)*

Nájdite jedno rozmiestnenie presne 7 čiernych štvorcov, pre ktoré platí: O 3 minúty ešte bude aspoň jeden štvorec čierny, ale o 4 minúty už budú úplne všetky štvorce v celej rovine biele.

Nájdite iné rozmiestnenie presne 7 čiernych štvorcov, pre ktoré platí: O 6 minút bude práve jeden štvorec v celej rovine čierny. Tento štvorec nesmie ležať na žiadnej z pozícií, kde boli čierne štvorce na začiatku.

Podúloha b) *(2 body)*

Marika nemá nekonečnú štvorcovú sieť, preto si túto hru skúša na papieri, kde si nakreslila obdĺžnikovú oblasť rozmerov 8×12. V tej si na začiatku vyfarbila nejaké čierne štvorce tak, aby žiaden z nich neležal na okraji. Príklad takéhoto vyfarbenia:



Môže sa Marike niekedy stať, že o pár minút bude niektorý zo štvorcov na okraji jej oblasti čierny? (Marika počíta nové farby štvorcov na okraji tak, ako keby okolo jej papiera boli ešte ďalšie biele štvorce.)

Podúloha c) *(3 body)*

Vie Marika zafarbiť na čierno niektoré štvorce vo vnútri svojej oblasti tak, aby aj po 100 minútach menenia farieb mala vo svojej oblasti aspoň jeden čierny štvorec? Ak áno, nájdite jedno také začiatočné zafarbenie, ak nie, zdôvodnite, prečo to nejde.

Podúloha d) *(2 body)*

Na záver sa vráťme k nekonečnej štvorcovej sieti. Ak ste úspešne vyriešili všetky doterajšie podúlohy, ste pripravení na záverečnú otázku.

Nájdite nejaké (nie nutne najmenšie) prirodzené číslo k také, aby platilo: Ak na začiatku zafarbíme na čierno ľubovoľných 47 štvorcov, tak po k minútach už zaručene budú všetky štvorce v celej rovine biele.

Samozrejme, svoje tvrdenie patrične dokážte.

DVADSIATY SIEDMY ROČNÍK OLYMPIÁDY V INFORMATIKE

Autori úloh: Michal Forišek, Ján Hozza, Mária Mrocková
Recenzent: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2012