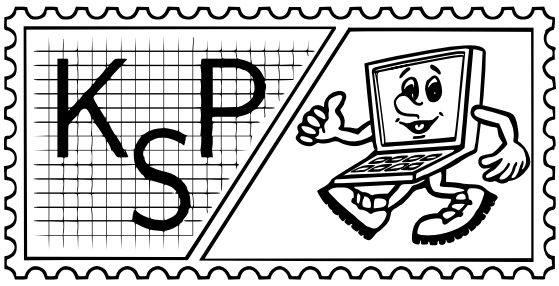


Korešpondenčný seminár z programovania

Leták zimnej časti XXXI. ročníka

Korešpondenčný seminár z programovania (KSP) je súťaž programátorov – stredoškolákov a mladších – pripravovaná skupinou študentov a doktorandov FMFI UK. Jej cieľom je zdokonaliť žiakov v programovaní a v algoritmickom myslení.

Ak študuješ na strednej škole a vieš aspoň trochu programovať, neváhaj a zapoj sa do našej súťaže, má to množstvo výhod:

- Riešením súťažných úloh a štúdiom našich vzorových riešení sa môžeš naučiť mnoho nového. Získané poznatky a skúsenosti sa ti iste budú hodiť v iných súťažiach v programovaní (napríklad pri riešení Olympiády v informatike), počas vysokoškolského štúdia, či pri prijímacích pohovoroch do zamestnania. (Mnoho našich bývalých riešiteľov sa bez ťažkostí zamestnalo v špičkových IT spoločnostiach ako Google, Facebook, ESET, ...)
- Na riešenie úloh máš dosť času a môžeš ich riešiť doma bez toho, aby si niekam cestoval.
- Medzi zadaniami sa nachádzajú ľahšie aj ťažšie. Každý si môže vybrať tie, ktoré vie riešiť a ktoré považuje za zaujímavé.
- Pre najlepších riešiteľov organizujeme každoročne dve týždenné sústredenia. Sústredenie je jedinečnou príležitosťou ako spoznať nových priateľov s podobnými záujmami, naučiť sa čosi viac nielen o programovaní a zažiť kopec zábavy.

Ako KSP prebieha?

Počas školského roka prebehnú dve samostatné časti súťaže: zimná a letná. Každá časť sa skladá z dvoch kôl, každé kolo obsahuje desať súťažných úloh. Najlepších riešiteľov zimnej časti pozývame na jarné sústredenie; najlepších riešiteľov letnej časti zase na to jesenné.

Súťažiť sa dá v troch kategóriách: **Z** (pre začínajúcich riešiteľov, obsahuje úlohy 1–5), **O** (skúsenejší riešitelia, úlohy 4–8) a **T** (špeciálna kategória pre náročných, päť samostatných úloh). Každá kategória má svoju vlastnú výsledkovú listinu. Na sústredenia pozývame riešiteľov na základe výsledkov v kategóriách Z a O.

Vaše riešenia úloh môžete odovzdávať na stránke `http://www.ksp.sk/eRiesenie`, kým neuplynie termín určený v zadaniach kola. Po tom, čo riešenia opravíme, nájdete na tomto mieste aj naše komentáre k nim a počet bodov získaný za jednotlivé úlohy.

Ako má vyzerať riešenie a za čo dostanem body?

Vašou úlohou je vytvoriť program, ktorý rieši zadanú úlohu. V prvom rade sa snažte, aby bol korektný, t.j. aby dal pre každý vstup správnu odpoveď, v druhom rade aby bol čo najrýchlejší a mal čo najmenšie pamäťové nároky.

Riešenie úlohy pozostáva z programu a popisu použitého algoritmu. V zadaní vždy uvedieme, koľko bodov sa dá získať za program a koľko za popis; výsledný počet bodov za úlohu je súčtom týchto dvoch hodnotení.

Váš program hneď po odovzdaní automaticky otestujeme na viacerých vopred pripravených vstupoch. Body za neho vám pridelíme podľa toho, na koľkých vstupoch dá správnu odpoveď v časovom limite. Len čo sa program dotestuje, dozviete sa výsledok. Ak ste nezískali plný počet bodov, môžete program vylepšiť a odovzdať ho znova. Podrobnejšie informácie o odovzdávaní programov nájdete na našej webstránke.

Popis algoritmu by mal byť natoľko podrobný a zrozumiteľný, aby bolo možné podľa neho napísať program rovnako efektívny, ako ten váš. Ďalej vyžadujeme odhad časovej a pamäťovej zložitosti a zdôvodnenie (ak je to potrebné, aj dôkaz) správnosti algoritmu.

Ak vo svojom riešení používate zložitejšie dátové štruktúry (napríklad haldu, nie obyčajné pole), musíte popísať aj ich implementáciu. To platí aj v prípade, že ich váš programovací jazyk už obsahuje a vy ste ich neimplementovali. Ak si nie ste istí, či niečo môžete použiť bez popisu, radšej to popíšte, prípadne sa spýtajte vo fóre na stránke.

Popis odovzdávajte vo formáte PDF alebo ako plain text. Hodnotíme hlavne korektnosť algoritmu a v druhom rade jeho efektivitu. Získaný počet bodov sa dozviete, keď vaše riešenie po termíne odovzdania opravíme.

Zopár ukážkovo vyriešených starších úloh nájdete na stránke `http://www.ksp.sk/wiki/Seminar/Riesenie`. Na tomto mieste sa tiež môžete dočítať, čo je vlastne časová a pamäťová zložitosť (ak vám tieto pojmy veľa nehovoria).

Ktoré kategórie môžem riešiť?

Kategóriu Z môžu riešiť:

- tretiaci a štvrtáci[1], ak do začiatku príslušného polroka neboli na sústredení KSP,
- druháci a mladší, ak do začiatku príslušného polroka boli najviac na jednom sústredení KSP.

Kategórií O a T sa môžu zúčastniť všetci stredoškoláci (a mladší) bez obmedzenia. Ak vám to pravidlá dovoľujú, môžete riešiť aj viacero kategórií naraz.

Čo je na kategórii T iné?

Keď sa vám zdá, že príkladov je málo, chcete sa naučiť viac, osvojiť si nové finty a vyskúšať si naprogramovať niečo, čo ste doteraz neskúšali, táto kategória je akurát pre vás.

Úlohou tejto kategórie je aj celoročná príprava riešiteľov na medzinárodné súťaže. Jej víťaza čaká večná sláva a navyše **body z kategórie T budú mierne zohľadnené pri výbere reprezentácie na Medzinárodnú olympiádu v informatike**. (Tento výber má formu týždňového sústredenia, na ktoré sú pozvaní najlepší riešitelia celoštátneho kola Olympiády v informatike, kat. A.)

Na rozdiel od kategórií Z a O, **v kategórii T nemusíte písať popis riešenia, odovzdávate len program. Zadania kategórie T už nenájdete v papierovej podobe, ale len na internetovej stránke** `http://www.ksp.sk/wiki/Zadania/Zadania`**.**

Na našej stránke sa dozviete aj viac o tejto kategórii.

Registrácia

Pred odovzdaním elektronického riešenia je potrebné zaregistrovať sa na našej webstránke a vyplniť požadované kontaktné údaje. Odporúčame sa zaregistrovať pár dní pred dňom, kedy chcete odovzdať vaše riešenie (pre prípad, že by ste mali počas registrácie nejaké problémy).

Účasťou v KSP nám dávate súhlas spracovať a archivovať údaje, ktoré nám poskytnete pri registrácii, ako aj zverejniť vaše meno, školu, ročník a získané body vo výsledkovej listine.

[1] Za štvrtákov považujeme študentov, ktorí maturujú v tomto školskom roku; tretiaci sú tí, ktorí budú maturovať budúci školský rok; ostatné ročníky analogicky.

Korešpondenčný seminár z programovania
XXXI. ročník, 2013/14
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Úlohy 1. kola zimnej časti

Termín odoslania riešení tejto série je pondelok **21. októbra 2013**.

1. Zavinuté hady

kat. Z; 2 b za popis, 8 b za program

Hady sa veľmi rady vyhrievajú na slnku a s obľubou sa zatáčajú do všelijakých tvarov.

Kráľ hadov však zaviedol nové pravidlá, podľa ktorých sa hady musia tvarovať. Všetky hady musia byť uložené do štvorca a v závislosti od svojho druhu musia mať jeden zo štyroch vzorov.

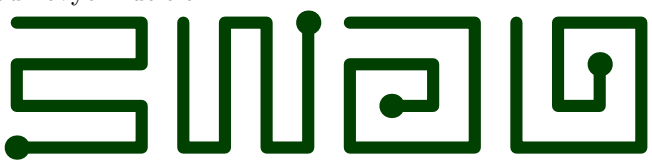
V každom haďom meste sa dokonca zriadil úrad, ktorý na požiadanie vykreslí každému hadovi jeho vzor.

Vašou úlohou je napísať program pre tieto úrady.

Úloha

Pre daný typ a veľkosť hada vykreslite správny vzor. Vzory sú štyri:

1. vpravo, vľavo, vpravo...
2. dole, hore, dole...
3. špirála v smere hodinových ručičiek,
4. špirála proti smeru hodinových ručičiek.



Chvost hada je **vždy** v ľavom hornom rohu (na začiatku prvého riadka výstupu).

Hada vypisujeme na výstup pomocou znakov `>` (väčší), `<` (menší), `v` (malé v), `^` (strieška) a `+` (plus). „Zobák“ ukazuje vždy na ďalší kus hada (bližší k hlave) a znak `+` predstavuje hlavu hada. (Pozrite si ukážkový vstup a výstup.)

Formát vstupu

V prvom riadku vstupu sú dve čísla t a n ($1 \leq t \leq 4$, $1 \leq n \leq 50$) udávajúce typ vzoru a rozmer štvorca. (Dĺžka hada je $n \times n$.)

Formát výstupu

Vykreslite hada pomocou znakov `><v^+` tak, ako je to vysvetlené v úlohe.

Príklady

vstup

```
1 4
```

výstup

```
>>>v
v<<<
>>>v
+<<<
```

Prvý obrázok v zadaní.

vstup

```
2 1
```

výstup

```
+
```

vstup

```
3 5
```

výstup

```
>>>>v
>>>vv
^>+vv
^^<<v
^<<<<
```

2. Znudený Jožko

kat. Z; 6 b za popis, 4 b za program

Jožko Kockáč sa začal doma nudiť. Domáce úlohy má už hotové, vonku prší, takže ani von sa nedá ísť, v televízii dávajú samé hlúposti a dokonca ani tie počítačové hry nie sú to, čo bývali. Preto sa Jožko rozhodol, že sa dá na ručné práce. Niekde v skrini vyhrabal nejaké drevo, zobral nožík a išiel niečo pekné vyrezať. Lenže čo by to malo byť? Rozhodol sa, že na začiatok bude vyrezávať kocky. Lenže aby to neboli len také obyčajné kocky, rozhodol sa, že to budú tzv. *magické* kocky. *Magické* kocky sa vyznačujú tým, že majú celočíselnú dĺžku hrany a špeciálny objem. Konkrétne taký, ktorého desiatkový zápis sa končí trojčíslím 888.

Predstavte si, že si úplne všetky možné *magické* kocky postavíme do radu podľa veľkosti a očíslujeme, pričom najmenšia kocka dostane číslo 1, ďalšia v poradí 2, atď. Jožko chce vyrezať len niektoré magické kocky (napríklad takú štyridsiatu siedmu). Pomôžte mu zistiť, akú dĺžku hrany má mať kocka, ktorú ide práve vyrezávať.

Úloha

Na vstupe bude číslo n. Vašou úlohou je nájsť dĺžku strany n-tej najmenšej *magickej* kocky.

Formát vstupu

Vstup bude pozostávať z jediného čísla n ($1 \leq n \leq 10^{18}$).

Poznámka: Premenné vo väčšine programovacích jazykov (C++, Pascal, Java…) majú obmedzenú veľkosť, tak si na to dávajte pozor pri písaní programu.

Formát výstupu

Vypíšte dĺžku hrany n-tej *magickej* kocky.

Príklady

vstup	výstup
1	192

Objem tejto kocky je $192^3 = 7\,077\,888$.

vstup	výstup
6	1442

3. Zemfírina veštba

kat. Z; 8 b za popis, 4 b za program

Keď sa Bobovi už sedemnástykrát po sebe stalo, že prišiel do jedálne práve v čase najväčšieho návalu obeduchtivých spolužiakov, rozhodol sa, že je čas na zmenu. Preto vyhľadal pomoc známej veštice Zemfíry. Tá po dvadsiatich eurách, desiatich minútach hľadenia do krištáľovej gule a troch minútach čmárania na papier podala Bobovi usporiadaný zoznam okamihov (udávaných v nanosekundách od desiatej hodiny ráno) a riekla: „Keď v niektorom z týchto okamihov do jedálne vstúpiš, rýchlo obed dostaneš!“

Bob však má nabitý rozvrh a na obed si môže odskočiť len v niektorých konkrétnych okamihoch. Aj tých zoznam (v rovnakom formáte a tiež usporiadaný) má Bob k dispozícii. Teraz mu ku šťastiu chýba už len zistiť, ktoré z okamihov v jeho zozname sa nachádzajú aj v Zemfírinom zozname, a tiež koľké v poradí v Zemfírinom zozname sú.

Úloha

Na vstupe dostanete Zemfírin usporiadaný n-prvkový zoznam vhodných okamihov $z_1, \ldots, z_n$ a tiež Bobov usporiadaný m-prvkový zoznam okamihov $b_1, \ldots, b_m$ kedy si Bob naozaj môže odskočiť na obed.

Pre každé číslo z Bobovho zoznamu zistite, či a kde sa nachádza v Zemfírinom zozname.

Formát vstupu

V prvom riadku vstupu sú čísla n a m ($1 \leq n, m \leq 200\,000$) oddelené medzerou. V druhom riadku je usporiadaný zoznam čísel $z_1, \ldots, z_n$, v treťom riadku je usporiadaný zoznam čísel $b_1, \ldots, b_m$. Platí $1 \leq z_1 < z_2 < \cdots < z_n \leq 10^9$ a $1 \leq b_1 < b_2 < \cdots < b_m \leq 10^9$.

Formát výstupu

Pre každé b_i vypíšte na výstup jeden riadok s jedným číslom: ak pre nejaké z_j platí $b_i = z_j$, vypíšte j, inak vypíšte -1.

Príklad

vstup

```
3 5
4 9 201
1 4 5 8 201
```

výstup

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

4. Žravce

kat. Z a O; 10 b za popis, 5 b za program

Žravce sa znova premnožili, nie však nadlho. Totiž akonáhle zožrali žravce všetko okolo seba, začali sa žrať navzájom.

Všetky žravce stoja v rade za sebou a každú sekundu všetky žravce naraz skontrolujú, či náhodou pred nimi nestojí menší jedinec. Ak áno, zožerú ho. A to aj v prípade, že ten menší žravec tiež práve niekoho žerie. (Všetky žravce sa žerú naraz.)

Zoológov by zaujímalo, po akom čase sa situácia ustáli, teda kedy sa žravce prestanú žrať.

Úloha

Na vstupe dostanete veľkosti žravcov tak ako stoja v rade, od posledného po prvého žravca (pred ktorým už nestojí nikto). Všetky žravce sú navzájom rôzne veľké. Zistite, koľko sekúnd bude trvať, kým sa žravce prestanú žrať, teda kým už pred každým žravcom bude stáť buď väčší žravec alebo nikto.

Pri popise programu *poriadne* zdôvodnite správnosť a časovú zložitosť vášho algoritmu.

Formát vstupu

V prvom riadku vstupu je celé číslo n udávajúce počet žravcov, $(1 \leq n \leq 200\,000)$.

V druhom riadku je n čísel v rozsahu od 1 po n (každé práve raz). Tieto čísla udávajú veľkosti žravcov v poradí od *konca* radu.

Formát výstupu

Vypíšte jedno číslo: koľko sekúnd bude trvať, kým sa žravce prestanú žrať.

Príklady

vstup

```
8
8 2 7 6 5 1 3 4
```

výstup

```
3
```

Počas prvej sekundy žravec veľkosti 8 zožerie žravca veľkosti 2. Zároveň s tým žravec veľkosti 7 žerie žravca veľkosti 6, ktorý žerie žravca veľkosti 5, ktorý zase žerie žravca veľkosti 1. Z tejto skupinky teda po prvej sekunde zostane len žravec veľkosti 7. Celý priebeh žrania: $(8, 2, 7, 6, 5, 1, 3, 4) \to (8, 7, 3, 4) \to (8, 4) \to (8)$.

vstup

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

výstup

```
0
```

Žravce sa nebudú žrať vôbec, lebo pred každým (okrem posledného) stojí väčší.

5. Ono to rieši samo! (diel 1: SAT solver)

kat. Z a O; 16 b za popis, 0 b za program

V tomto ročníku vás piate úlohy zavedú do sveta deklaratívneho programovania. O čo pôjde? Pri klasickom (tzv. imperatívnom) programovaní počítaču vysvetľujeme, *ako* má niečo spraviť: pre každý prvok tohto poľa spočítaj toto, to následne porovnaj s týmto, a tak ďalej. V niektorých situáciách je ale pohodlnejšie (a občas dokonca aj efektívnejšie) počítaču len popísať, *čo* od neho očakávame a nechať na ňom, nech nájde najlepší spôsob, ako to dosiahnuť.

Oukej, to zrejme znie ako scifi. Rozhodne ešte nie sme na úrovni, aby sme počítaču povedali „naprogramuj mi nejakú vesmírnu strieľačku“ a on to urobí. Ale pokiaľ si pod „popísať, čo od neho očakávame“ predstavíme exaktný matematický popis, už začína mať počítač celkom rozumnú šancu.

O probléme SAT

V informatike existuje veľa problémov, ktorých algoritmické riešenie je ťažké – nepoznáme pre ne žiaden efektívny algoritmus[2] a domnievame sa, že takýto algoritmus ani neexistuje. Medzi týmito problémami existuje skupina tzv. NP-úplných problémov, ktoré sú všetky približne rovnako ťažké: keby niekto vymyslel, ako efektívne riešiť jeden z nich, zrazu by sme vedeli efektívne riešiť všetky ostatné.

Teraz si predstavíme jeden konkrétny spomedzi takýchto problémov: problém SAT. Názov „SAT“ pochádza z anglického slova satisfiability – splniteľnosť. Úloha je na prvý pohľad jednoduchá. Na vstupe dostaneme nejaký logický výraz, napr.[3] $(a \wedge (b \vee \overline{c})) \rightarrow \overline{a}$. Našou úlohou je zistiť, či je splniteľný – teda či existuje nejaké ohodnotenie premenných, pre ktoré je výsledný výrok pravdivý. Náš výraz z príkladu je splniteľný – pravdivý výrok dostaneme napríklad ak sú všetky tri premenné nepravdivé. Logický výraz $a \wedge \overline{a}$ splniteľný nie je.

Splniteľnosť výrazu vieme ľahko rozhodnúť hrubou silou: ak sa vo výraze vyskytuje n rôznych premenných, tak máme 2^n možností, ako im priradiť pravdivostné hodnoty. Stačí teda všetky tieto možnosti vyskúšať a zistiť, či niektorá z nich vyhovuje.

Smutnejšie je, že principiálne inak ako hrubou silou splniteľnosť výrazu rozhodovať nevieme. A že ide o dôležitý problém, ktorý často riešiť treba, sústredila sa pozornosť výskumníkov na to, ako spraviť program, ktorý tú hrubú silu bude využívať čo najlepšie – rýchlo spozná, ktoré možnosti k riešeniu určite nevedú, a tak sa čo najviac vyhne zbytočnému skúšaniu.

V tejto úlohe nás ale nebude zaujímať, ako tieto programy (nazývané SAT solvery) fungujú. My takýto program budeme len chcieť použiť – na riešenie iných problémov.

Zápis v CNF

Väčšina SAT solverov očakáva svoj vstup zapísaný v špeciálnom tvare, tzv. *konjunktívnej normálnej forme (CNF)*: výraz musí byť zapísaný ako logický *and* niekoľkých podmienok, a navyše každá podmienka musí byť zapísaná ako logický *or* niekoľkých premenných a ich negácií.[4]

Rozmyslite si, že v takomto tvare vieme zapísať ľubovoľný logický výraz. Napríklad výraz „a implikuje b“ (inými slovami, „ak a, tak aj b“) môžeme zapísať aj v tvare „neplatí a, alebo platí b“, teda „$\overline{a}$ alebo b“.

Tu je príklad zložitejšieho výrazu zapísaného v CNF: $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_3 \vee \overline{x_1}) \wedge (x_4)$.

Skoro všetky SAT solvery používajú ten istý formát vstupu. Ten si teraz stručne popíšeme. Predchádzajúci výraz by sme zapísali nasledovne:

```
p cnf 4 3
1 -3 0
2 3 -1 0
4 0
```

Vysvetlenie: Prvý riadok má tvar „`p cnf` n p“, kde n je počet premenných v našom výraze a p je počet podmienok. Následne pre každú podmienku sú uvedené čísla premenných, ktoré obsahuje. Čísla premenných sú od 1 po n, mínus reprezentuje negáciu. Za každou podmienkou je nula. Medzi jednotlivými číslami môžu byť ľubovoľné biele znaky, my však budeme (a aj vám odporúčame) pre prehľadnosť vždy písať každú podmienku ako jeden riadok.

Riešenie iných problémov

Na príklade si teraz ukážeme, ako pomocou SAT solveru vyriešiť úplne inú úlohu: problém n dám na šachovnici. V tomto probléme je úlohou rozmiestniť na šachovnicu $n \times n$ práve n dám tak, aby sa žiadne dve neohrozovali – teda neležali v tom istom riadku, v tom istom stĺpci, ani na tej istej uhlopriečke.

Ako by ste takúto úlohu riešili v klasickom programovacom jazyku? Úplne hrubou silou by sa úlohu dalo riešiť pomocou *skúšania všetkých permutácií*: v optimálnom riešení zjavne musí byť v každom riadku aj stĺpci práve jedna dáma, každé rozmiestnenie dám teda zodpovedá nejakej permutácii stĺpcov. Lenže už pre $n = 15$ začína byť všetkých možných permutácií nepríjemne veľa.

O niečo lepší prístup je použiť *backtracking* (prehľadávanie s návratom): skúsim umiestniť prvú dámu, potom druhú niekam kde ju tá prvá neohrozuje, potom tretiu... až pri trinástej dáme zistím, že už ju nemám kam dať. Vtedy sa vrátim k predchádzajúcej dáme, skúsim ju umiestniť inam a s touto jej polohou zase pokračovať ďalej. A tak dokola, až kým nenájdeme riešenie alebo nevyskúšame všetky „rozumné“ možnosti. Ak ste takýto program ešte nikdy nepísali, skúste si to! (V tejto úlohe za to síce body nebudú, ale v budúcnosti sa vám to oplatí.)

[2] T.j. napríklad algoritmus s polynomiálnou časovou zložitosťou.

[3] Symboly $\wedge$, $\vee$, $\rightarrow$ a $^-$ predstavujú logický and, logický or, implikáciu a negáciu premennej.

[4] Pre zaujímavosť: nášmu logickému výrazu sa odborne hovorí „formula“, podmienkam „klauzuly“ a premenné a ich negácie dokopy označujeme slovom „literály“.

My však sme leniví riešenie tejto úlohy programovať. Radšej len popíšeme, ako prípustné riešenie vyzerá, a necháme SAT solver, nech nám jedno nájde.

Ako na to? Budeme mať n^2 premenných: pre každé políčko šachovnice jednu. Hodnota premennej nám bude hovoriť, či na danom políčku je alebo nie je dáma. No a teraz jediné, čo potrebujeme, je zapísať správnu sadu podmienok. Slovne si ich môžeme sformulovať nasledovne: 1. v každom riadku je nejaká dáma; a 2. ak sú dve políčka v tom istom riadku/stĺpci/uhlopriečke, nemôžu obe obsahovať dámu.

Teraz teda potrebujeme vyrobiť súbor obsahujúci tieto podmienky vo vyššie popísanom formáte. A keďže sme leniví (a chceme postupne skúšať rôzne n), tento súbor nebudeme písať ručne – vygenerujeme si ho:

Listing programu (Python 3)

```
N = int(input())
podmienky = []

def premenna(r,c): return N*r+c+1

def vypis_podmienku(p):
    for x in p: print(x,end=' ')
    print(0)

# v každom riadku je aspoň jedna dáma
for r in range(N): podmienky.append(  [ premenna(r,c) for c in range(N) ]  )

# pomocná funkcia, vracia či sa dve políčka ohrozujú
def ohrozi(r1,c1,r2,c2): return r1==r2 or c1==c2 or r1+c1==r2+c2 or r1-c1==r2-c2

# žiadne dve ohrozujúce sa políčka nemajú naraz dámu
for r1 in range(N):
    for c1 in range(N):
        for r2 in range(r1,N):
            for c2 in range(N):
                if (r1,c1)<(r2,c2) and ohrozi(r1,c1,r2,c2): podmienky.append( [-premenna(r1,c1),-premenna(r2,c2)] )

print('p cnf {} {}'.format(N**2,len(podmienky)))
for p in podmienky: vypis_podmienku(p)
```

Napríklad pre $n = 4$ tento program vygeneruje nasledovný „SAT program“:

```
p cnf 16 156
1 2 3 4 0
5 6 7 8 0
9 10 11 12 0
13 14 15 16 0
-1 -2 0
-1 -3 0
-1 -4 0
-1 -5 0
... 70 vynechaných riadkov ...
-14 -16 0
-15 -16 0
```

(Za hlavičkou nasledujú najskôr 4 podmienky hovoriace „v 1. riadku je dáma“ až „v 4. riadku je dáma“, a následne tam máme kopu podmienok hovoriacich, že z danej dvojice políčok musí aspoň jedno byť prázdne.)

Keď sme na tento SAT program spustili SAT solver, vypísal na výstup kopu textu. Väčšina riadkov začínala `c` a teda obsahovala komentáre. Nás však zaujímali hlavne nasledovné dva riadky:

```
s SATISFIABLE
v -1 2 -3 -4 -5 -6 -7 8 9 -10 -11 -12 -13 -14 15 -16 0
```

Prvý z nich nám hovorí, že riešenie existuje. Druhý riadok obsahuje jedno konkrétne ohodnotenie premenných, pre ktoré je náš výraz pravdivý. Vidíme, že pravdivé premenné sú 2, 8, 9 a 15. Tomu zodpovedajú dámy rozmiestnené nasledovne:

```
. * . .
. . . *
* . . .
. . * .
```

Pre $n = 3$ by sme dostali jediný zaujímavý riadok výstupu: „`s UNSATISFIABLE`“ – riešenie neexistuje. Pre $n = 100$ sme na našom počítači na riešenie čakali necelé dve minúty: asi 45 sekúnd sa SAT program generoval a asi minútu preň solver hľadal riešenie. Slušné, nie? Obzvlášť keď si minútu porovnáme s časom behu programu, ktorý skúša všetkých $n!$ permutácií, či nebodaj všetkých 2^{n^2} možných ohodnotení našich n^2 premenných.

Kde zohnať SAT solver?

Veľa SAT solverov je voľne dostupných. Vyberte si nejaký, čo vám bude vyhovovať.

- Vaša obľúbená distribúcia Linuxu možno už má nejaké SAT solvery ako balíčky. V Ubuntu napríklad nájdete balíček `picosat`.

- Celkom šikovný je Glucose, ktorý vlani vyhral v jednej kategórii súťaže SAT solverov. Treba si stiahnuť jeho zdrojový kód z `https://www.lri.fr/~simon/downloads/glucose2.2.tgz` a skompilovať si ho.
- Tiež fajn je Cryptominisat. Buď si ho môžete z `https://gforge.inria.fr/frs/?group_id=1992` stiahnuť a skompilovať, alebo na tej istej stránke nájdete aj binárky staršej verzie (2.9.6) pre Linux aj Windows. Tie sú pre naše potreby tiež bez problémov použiteľné.
- Ďalšou možnosťou je Relsat, ktorý si môžete stiahnuť z `http://relsat.googlecode.com/files/relsat_2.02.tar`. Treba ho tiež kompilovať, ale kompiluje sa (aj pod Windows) ľahko a príjemne. Rieši pomalšie, ale má milú fičúriu: vie vygenerovať aj viacero riešení a prestať po predpísanom počte.

A ak by toto všetko zlyhalo, napíšte nám na fórum, vysvetlite v čom je problém a určite vám niekto pomôže.

Úlohy

Z adresy `http://www.ksp.sk/wiki/uploads/Zadania/ksp31-1-5-vstupy.zip` si stiahnite súbor s testovacími vstupmi pre podúlohy B a C.

Podúloha A (2 body).
Nájdite a pošlite nám nejaké rozmiestnenie 100 dám na šachovnici rozmerov 100×100 také, že sa žiadne dve dámy neohrozujú. Pre každú dámu uveďte jej súradnice (r, c), pričom $0 \leq r, c < 100$. Jedna z dám musí mať súradnice $(47, 47)$.

Podúloha B (7 bodov).
V každom zo súborov nazvaných `ham??.in` je popis jednej krajiny: počet miest n, počet obojsmerných ciest m, a následne zoznam m dvojíc miest spojených cestami. Mestá majú čísla od 0 po $n-1$. Môžete predpokladať, že v každej z týchto krajín existuje tzv. Hamiltonovská kružnica – teda okružná cesta začínajúca a končiaca v tom istom meste a prechádzajúca každým mestom práve raz. Vašou úlohou je pre čo najviac krajín jednu takúto kružnicu nájsť. Ako výstup uveďte poradie, v akom treba mestá navštíviť.

Podúloha C (7 bodov).
V každom zo súborov nazvaných `col???.in` je opäť popis jednej krajiny v takom istom formáte. Chceli by sme každé mesto vyzdobiť stuhami nejakej farby. Aby bola krajina pestrá, musí platiť, že ak dve mestá susedia (t.j. sú prepojené priamou cestou), tak nesmú použiť tú istú farbu. Pre čo najviac krajín zistite, aký *najmenší* počet farieb nám stačí na ofarbenie celej krajiny.

Inštrukcie k odovzdávaniu riešení.
Vaše riešenia budeme bodovať ručne. Ku každej úlohe by ste mali pripojiť aj slovný popis toho, ako ste pri jej riešení využili SAT solver. Ku každému spusteniu SAT solveru skúste uviesť približnú dĺžku jeho výpočtu. Priložte tiež zdrojové kódy pomocných programov, ak ste si nejaké napísali. Pokojne z toho všetkého vyrobte jedno veľké PDF a odovzdajte to.

6. Otrasná nuda

kat. O; 13 b za popis, 7 b za program

Ani by ste neverili, aké veci sa dejú v KSP-áckej miestnosti T2. Napríklad pri poslednom upratovaní sme sa dozvedeli niečo o budúcnosti KSP. Aj o niekoľko storočí neskôr KSP naďalej funguje, dokonca vlastní časopriestorový informačný stream. To znamená, že sa dokážu pripojiť na počítač v minulosti a posielať naň informácie. Už to raz spravili, keď poslali správu našim zakladateľom aby zachránili vzácny IOI plagát.

Minule sedel Jano v T2 a strašne sa nudil. Napísal teda na tabuľu „Nudím sa. 19.9.2013 2:32am Nezotierať!“. Hneď na to sa na jednom z terminálov otvoril časopriestorový stream. Presne ako chcel.

Úloha

Dozvedel sa, že v budúcnosti vedúci akurát balia veci na sústredenie a chcú sa s Janom zahrať nasledovnú hru. Na sústredenie chcú zobrať n vecí, pričom každá z nich má nejaký (kladný celočíselný) objem v_i. Na odnesenie vecí majú len jedno jediné vrece s objemom c. Je možné, že sa všetky veci naraz do vreca nezmestia, preto chcú vedúci vybrať takú množinu vecí, že súčet ich objemov je najviac c a spomedzi všetkých takých objemov je najväčší možný. (Nezáleží teda na počte vecí, ktoré vezmú, len na tom, aby ich celkový objem neprekročil c a bol k nemu čo najbližšie.)

Samozrejme, keby Janovi len tak oznámili, aké objemy majú jednotlivé predmety, tak by si ten vypočítal optimálne riešenie a vôbec by sa s ním nezahrali. Preto mu vedúci objemy predmetov nepovedia. Na začiatku mu len prezradia hodnoty n a c.

Vedúci v budúcnosti si na začiatku zvolili nejakú (Janovi neznámu) podmnožinu všetkých vecí. Tú si nazveme A. V každom kole hry potom vedúci vyrobia z aktuálnej množiny A novú množinu B, ktorá sa od A

bude líšiť práve v jednom predmete. Táto zmena prebehne nasledovne: Vedúci v budúcnosti si (rovnomerne) náhodne vyberú jeden z n predmetov. Ak tento predmet bol v A, tak v B nebude, a naopak. Následne vedúci povedia Janovi súčet objemov predmetov v B a Jano sa môže rozhodnúť, s ktorou množinou z A a B sa bude pokračovať ďalej. Tú množinu, ktorú si vybral, vedúci v budúcnosti premenujú na A a odznova sa začne celý vyššie popísaný proces.

Namiesto toho, aby si Jano vybral jednu z množín A a B, má počas hry Jano ešte tretiu možnosť: môže povedať stop. Keď Jano povie stop, znamená to, že práve ponúkaná množina B je tou pravou, ktorú treba zobrať na sústredenie. Týmto hra končí. Jano vyhráva vtedy, keď je B naozaj hľadanou množinou vecí, (Ak existuje viacero optimálnych riešení, môže Jano nájsť ľubovoľné z nich.)

V každom kole musí teda Jano na tabuľu napísať jednu z možných odpovedí:

- **stop** – je to záverečná akcia. Znamená, že B obsahuje optimálne riešenie.
- **accept** – prijíma riešenie B a to je nakopírované do A.
- **decline** – odmieta riešenie B, to je zahodené a hodnota A zostane nezmenená.

Ak odpoveď nie je **stop**, nasleduje ďalšie kolo. Jano smie absolvovať najviac 1000 kôl. Ak tento počet prekročí, prehral.

Interakcia

Komunikácia prebieha prostredníctvom štandardného vstupu a výstupu.

Váš program má začať načítaním troch hodnôt – n, c a objemu množiny B ponúkanej v prvom kole. Potom musí váš program vypísať odpoveď na danú ponuku B, znovu načítať novú hodnotu objemu B a takto pokračovať až kým nevypíše poslednú akciu – **stop**. Po poslednej akcii má váš program skončiť.

Každú odpoveď zakončite znakom nového riadka. Aby ste sa navyše vyhli problémom s bufferovaním výstupu, flushujte („spláchnite“) ho po každom výpise. Napr. v C++ použite funkciu `fflush(stdout)`.

Je zaručené, že náš program vyberá predmety, ktoré vymení, naozaj náhodne, používajúc pseudonáhodný generátor. Vzorové riešenie sa príliš nepotrebuje spoliehať na šťastie – aj keby ste ho spustili tisíckrát na každom testovacom vstupe, sme si takmer istí, že všetko správne vyrieši.

Formát vstupu

V prvom riadku sa nachádzajú tri čísla n ($1 \leq n \leq 20$), c ($0 \leq c \leq 10^9$) a objem riešenia B pre prvé kolo. Naviac pre všetky i je zaručené, že $1 \leq v_i \leq 10^8$. Všetky ďalšie riadky budú obsahovať jedno celé číslo: celkový objem B v príslušnom kole.

Formát výstupu

Štandardný výstup predstavuje sériu vašich odpovedí. Každá odpoveď je reprezentovaná jedným riadkom s jedným slovom: **accept**, **decline** alebo **stop**.

Príklad

vstup

```
2 5 3
2
5
```

výstup

```
decline
accept
stop
```

Vedúci majú dva predmety s objemami 2 a 3 (čo ale Jano v tej chvíli nevie). Na začiatku je A prázdne (to Jano tiež nevie).

*V prvom kole vedúci pridajú do A druhý predmet. Janovi je teda ponúknutá množina B s objemom 3. Odmietne ju (**decline**). Stále je teda A prázdna.*

*V druhom kole vedúci pridajú do A prvý predmet. Ponúkanú množinu B s objemom 2 Jano prijme (**accept**). V tejto chvíli si už Jano môže byť istý, že predmety majú objemy 2 a 3 a že v aktuálnej množine je predmet veľkosti 2. Jano síce nevie, či na úplnom začiatku bola A prázdna alebo obsahovala oba prvky, to ho ale netrápi.*

*Keďže už Jano presne všetko pozná, vie určiť, že optimálnym riešením je zobrať na sústredenie oba predmety. Keď mu je teda následne ponúknutá množina s objemom 5 (čo je presne rovné c), Jano program ukončí príkazom **stop**.*

7. Obyčajný tender

kat. O; 12 b za popis, 8 b za program

Institute of Black Magic (IBM) sa chystá vstúpiť do tendra pre vybudovanie internetovej siete v Absurdistane. Požiadavky na túto sieť sú pomerne jednoduché. Treba, aby sa z každého mesta dala poslať správa úplne všade (možno nie priamo, ale cez iné mestá), a navyše aby to celé stálo čo najmenej. Avšak je dosť možné, že úradníci prídu s požiadavkou, aby nejaká dvojica miest bola spojená priamou linkou. Na to sa v IBM potrebujú pripraviť a táto úloha padla na vás.

Úloha

V Absurdistane je n miest. Každé dve z nich sa dajú spojiť káblom, pričom kábel medzi mestami i a j stojí c_{ij} (platí $c_{ij} = c_{ji}$). Pre každú dvojicu miest i, j nájdite najlacnejšiu množinu káblov X, pre ktorú platí:

- Je v nej použitý kábel, ktorý vedie medzi mestami i, j.
- Medzi každou dvojicou miest sa dá poslať správa len s použitím káblov v množine X.

Formát vstupu

Prvý riadok vstupu obsahuje číslo n ($2 \leq n \leq 800$). Každý z nasledujúcich n riadkov obsahuje n celých čísel, pričom v i-tom riadku je j-te číslo c_{ij} – t.j. cena za vybudovanie kábla medzi mestami i, j. Platí $c_{ii} = 0, c_{ij} = c_{ji}, 0 \leq c_{ij} \leq 1\,000\,000$.

Formát výstupu

Vypíšte n riadkov, kde v každom je n čísel. j-te číslo v i-tom riadku má obsahovať cenu najlacnejšie množiny, ktorá spĺňa podmienky vyššie a obsahuje kábel medzi mestami i, j. V prípade, že $i = j$, vypíšte 0.

Príklad

vstup

```
3
0 42 47
42 0 23
47 23 0
```

výstup

```
0 65 70
65 0 65
70 65 0
```

8. Ochrana pred povodňami

kat. O; 15 b za popis, 10 b za program

Obyvateľov obce Nalomená Trieska už roky trápia správy o povodniach z celého Slovenska. Cez obec síce nepreteká žiadna rieka, no čo ak sa jedného dňa vyleje ich rybník, nešťastne umiestnený priamo v strede námestia? Aby takémuto hroznému osudu predišli, obyvatelia spoločne vybudovali jednoduchý protipovodňový systém.

Hladina rybníka má tvar kruhu. Na jeho obvode teraz stojí n zátarás rôznych dĺžok (každá vyzerá zhora ako kružnicový oblúk). Dokopy tieto zátarasy pokrývajú celý breh rybníka, pričom niektoré z nich sa môžu aj prekrývať. Formálne: zjednotenie kružnicových oblúkov (vrátane ich krajných bodov) tvorí celú kružnicu.

Také množstvo zátarás však zaberá väčšinu námestia a to nepôsobí priaznivo na turistický ruch. Starosta dal preto príkaz odstrániť čo najviac zátarás tak, aby zvyšné stále pokrývali celý obvod rybníka a tak chránili Nalomenú Triesku.

Úloha

Obvod rybníka si rozdelíme na m rovnako dlhých úsekov, ktoré zaradom očíslujeme $0, 1, 2, \ldots, m-1$. O každej zátarase dostanete zadané dve hodnoty: číslo prvého pokrytého úseku x_i a dĺžku zátarasy ℓ_i. Teda i-ta zátarasa pokrýva úseky $x_i, (x_i + 1) \bmod m, (x_i + 2) \bmod m, \ldots, (x_i + \ell_i - 1) \bmod m$.

Vašou úlohou je nájsť čo najmenšiu množinu zátarás, ktorá pokrýva každý z m úsekov obvodu rybníka. Budeme akceptovať ľubovoľné riešenie používajúce minimálny počet zátarás.

Ak sa váš algoritmus opiera o netriviálne tvrdenia, nezabudnite ich dokázať. Takisto si dajte záležať na odhade časovej zložitosti (ak nie je zrejmý).

Formát vstupu

Prvý riadok vstupu obsahuje dve čísla m a n oddelené medzerou – počet úsekov obvodu rybníka a celkový počet zátarás ($1 \leq m \leq 10^9, 1 \leq n \leq 10^5$).

Nasledujúcich n riadkov popisuje jednotlivé zátarasy. Každý z nich obsahuje dve čísla x_i a ℓ_i oddelené medzerou – prvý pokrytý úsek a dĺžku i-tej zátarasy ($0 \leq x_i < m, 1 \leq \ell_i \leq m$). Každý úsek obvodu je pokrytý aspoň jednou zo zadaných zátarás.

Formát výstupu

Na prvý riadok výstupu vypíšte počet zátarás použitých vo vašom riešení. Na druhý riadok vypíšte ich čísla (podľa poradia na vstupe, začínajúc číslovať od 1) v rastúcom poradí.

Príklad

vstup

```
5 3
0 2
3 3
1 3
```

výstup

```
2
2 3
```

Zadania kategórie T

Zadania kat. T sa čoskoro objavia na našej stránke `http://www.ksp.sk/wiki/Zadania/Zadania`. Nezabudnite sledovať novinky, čaká na vás ďalších **päť** zaujímavých úloh rôznych obtiažností.