Korešpondenčný seminár z programovania

Leták zimnej časti XLIII. ročníka

Korešpondenčný seminár z programovania (KSP) je súťaž programátorov – stredoškolákov a mladších – pripravovaná skupinou študentov Fakulty matematiky, fyziky a informatiky Univerzity Komenského v Bratislave. Naším cieľom je zdokonaliť žiakov v programovaní a v algoritmickom myslení.

Riešením súťažných úloh a štúdiom vzorových riešení sa zlepšíš v programovaní a naučíš sa algoritmicky rozmýšľať. Získané poznatky a skúsenosti využiješ v iných súťažiach v programovaní (napríklad pri riešení Olympiády v informatike), v bežnom živote, počas vysokoškolského štúdia, dokonca aj na prijímacích pohovoroch do zamestnania. Naši riešitelia sa každoročne zúčastňujú a úspešne umiestňujú na medzinárodných olympiádach v informatike (v Austrálii, Taliansku, Kazachstane, Taiwane, …). Mnoho našich bývalých riešiteľov sa tiež bez ťažkostí zamestnalo v špičkových IT spoločnostiach ako Google, Facebook, ESET, …

Ak študuješ na strednej škole a zaujíma ťa programovanie, neváhaj a zapoj sa do KSP:

Ako sa zapojiť do KSP?

- **Prečítaj** si zadania. Nájdeš ich v tomto letáku a na našej stránke https://www.ksp.sk/ulohy. Každý rok máme zimnú a letnú časť, obe majú dve kolá s ôsmimi úlohami.
- Teš sa, aké sú tento rok pekné úlohy.
- **Vyrieš** úlohy. Nemusíš vyriešiť všetky, nemusíš ich vyriešiť najlepšie ako sa dá. Aj za čiastočné riešenia sa dostávajú body, za každú úlohu za dá získať 0 až 20 bodov.
- Na riešenie úloh jedného kola máš približne mesiac a pol a môžeš ich riešiť doma bez toho, aby si niekam cestoval. Termín odovzdania úloh je napísaný aj na našej stránke, aj v PDF zadaniach. Úlohy sa nedajú odovzdávať po termíne, takže si to, prosím, nenechaj na poslednú chvíľu.
- Úlohy rieš samostatne a neprezrádzaj riešenia ostatným riešiteľom. Odpisovanie riešení a prezradenie riešení pred termínom kola je porušením pravidiel KSP. Po skončení kola sa, samozrejme, o riešeniach rozprávať môžeš. :)
- **Odovdzaj** riešenia úloh. Odkaz na odovzdávanie úloh nájdeš pod webovým zadaním každej úlohy alebo na stránke https://www.ksp.sk/odovzdavanie. Na odovzdávanie sa treba prihlásiť, aby sme vedeli, komu máme dať body.
  - Vo väčšine úloh odovzdávaš program a popis.
  - Program je hneď po odovzdaní otestovaný testovačom a hneď vidíš, koľko bodov za program máš. Program môžeš odovzdávať znova a znova, až kým nie si spokojný s výsledkom. Ak nevieš, ako majú vyzerať odovzdané programy, pozri si https://www.ksp.sk/odovzdavanie-programov
  - Do popisu slovne napíšeš, ako tvoje riešenie funguje, prečo funguje a tiež odhad časovej a pamäťovej zložitosti programu. Viac sa dozvieš na stránke https://www.ksp.sk/ako-riesit. Popis opravia a obodujú vedúci KSP po skončení kola.
- Po skončení kola si **prečítaj vzorové riešenia** úloh (veľa sa z toho naučíš), pozri svoje opravené popisy (či ti tam vedúci nenapísali nejaké poučné komentáre), pozri sa do výsledkovky a **teš sa**, koľko máš bodov. Vo výsledkoch sa hodnotí samostatne letná a zimná časť. V každej časti je dôležitý celkový súčet bodov.
- Prečo sa máš tešiť z bodov? Čítaj ďalej.

Čo môžem vyhrať?

- Okrem neoceniteľných vedomostí, skúseností a zručností, ktoré získaš pri riešení semináru, môžeš vyhrať množstvo skvelých vecí.
- Všetci víťazi od nás dostanú **vecné ceny**.
- Pre 36 najlepších riešiteľov organizujeme každoročne dve týždenné **sústredenia**. Sústredenie je niečo ako tábor, na ktorom spoznáš nových priateľov s podobnými záujmami, naučíš sa čosi viac nielen o programovaní a zažiješ kopec zábavy. Sústredenia sú fakt skvelé akcie, najmä, keď ich organizuje Trojsten.

- Aby ste sa mohli pochváliť ostatným, akí ste šikovní, víťazom všetkých levelov udelíme a pošleme **diplomy**.
- Aj keď sa nedostaneš medzi víťazov, stále môžeš byť úspešným riešiteľom. Úspešný riešiteľ je ten, kto získal aspoň polovicu bodov počas celej časti (letnej, či zimnej).

Pravidlá a levely

Počnúc tridsiatym piatym ročníkom rušíme staré kategórie a prechádzame na nový systém *levelov*.

Každý riešiteľ má level, číslo od 1 po 4. Noví riešitelia začínajú na leveli 1 a pokiaľ sa im v riešení darí, level im postupne rastie. Svoj level si môže každý riešiteľ pozrieť na našej stránke. Riešiteľom s levelom L sa započítavajú body len za úlohy s číslami L až 8.

Vo výsledkových listinách (https://www.ksp.sk/vysledky) sa každému riešiteľovi počíta **5 najlepšie vyriešených úloh**. Celkovo sa dá za časť (dve kolá) získať 200 bodov. Riešitelia, ktorí sa v nejakej výsledkovke umiestnili na jednom z prvých dvoch miest a majú aspoň 150 bodov sú **víťazi**. Najlepších 36 riešiteľov pozývame na sústredenie.

Podrobnejšie pravidlá si môžete prečítať na https://www.ksp.sk/pravidla.

Registrácia

Pred odovzdaním riešenia je potrebné sa zaregistrovať na našej webstránke a vyplniť požadované kontaktné údaje. Odporúčame sa zaregistrovať aspoň pár dní pred odovzdávaním riešenia (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.

Úlohy 2. kola zimnej časti

Termín odoslania riešení tohto kola je **15. decembra 2025**. Doprogramovávanie končí 4. januára 2026.

1. O vrecku čo zradilo ostatné

12 b za popis, 8 b za program

Vedec mal zvláštny deň. Od rána rozmýšľal, ako oklamať nudu — a tak sa rozhodol, že si odváži svoje poklady. V skrini mal n vreciek mincí, ktoré vyzerali úplne rovnako. No jedna vec ho trápila: v jednom vrecku sa ukrývali falošné mince, o niečo ťažšie ako ostatné.

Namiesto toho, aby vážil každé vrecko zvlášť, dostal geniálny nápad. Z každého vrecka odobral nejaký počet mincí, až mal na váhe poriadnu hromadu. Potom sa postavil k váhe, pozrel na číslo a s úsmevom vyhlásil: „Už viem, ktoré vrecko je falošné!“

Úloha

Máme n vreciek mincí. Práve v jednom vrecku sú všetky mince falošné (každá falošná minca váži 55 gramov), vo všetkých ostatných vreckách sú mince reálne (každá reálna minca váži 50 gramov). Pred vážením si môžeme z každého z vreciek odobrať ľubovoľný počet mincí a zložiť ich do jednej spoločnej kôpky. Kôpku potom zvážime na váhe, ktorá nám povie celkovú hmotnosť odobratých mincí v gramoch.

Vrecká sú magické a tak je v každom nekonečne veľa mincí.

Tvojou úlohou je na základe čo najmenej vážení určiť index vrecka, v ktorom sú falošné mince.

Formát vstupu a výstupu

Táto úloha je interaktívna. To znamená, že váš program komunikuje s testovačom. Testovaču pokladáte otázky a on vám na ne bude odpovedať.

Na prvom riadku vstupu je jedno celé číslo n ($1 \leq n \leq 100$) — počet vreciek s mincami. Následne môžete pokladať otázky vo forme `? i1 i2 i3 ... in`, kde `i1`, `i2`, …, `in` sú počty mincí odobraných z jednotlivých vreciek. Z každého vrecka môžete odobrať 0 až 10^9 mincí. Ako odpoveď na každú otázku testovač vypíše jeden riadok s jedným celým číslom - súčtom hmotností všetkých odobraných mincí v gramoch.

Keď už budete vedieť odpoveď, vypíšte riadok vo formáte `! x`, kde `x` je index vrecka s falošnými mincami ($1 \leq x \leq n$). Následne by mal váš program skončiť.

Vstupy sú tvorené viacerými sadami líšiacimi sa v maximálnom počte povolených otázok. Za každú sadu môžete získať 2 body.

Sada	1	2	3	4
Maximálny počet otázok	100	99	20	1

Keďže ide o interaktívnu úlohu, po vypísaní každej otázky je nutné flushovať výstupný buffer. V jazyku C++ to dosiahneme napríklad použitím `std::endl` namiesto `'\n'`, v Pythone parametrom printu : `print("text na vypísanie", flush=True)`.

Ak si tým stále nie ste istí, môžete len doplniť tento template, ktorý celú komunikáciu rieši za vás:

```python
# Vrati sucet vah v kopke. Z vrecka i dame do kopky pocty_minci[i] minci
def odvaz(pocty_minci):
    assert len(pocty_minci) == n
    print("?", *pocty_minci, flush=True)
    result = int(input())
    return result

def odpovedaj(index):
    print("!", index, flush=True)
```

```
    exit(0)

n = int(input())

# V nizsie uvedenom priklade je n = 5
if n == 5:
    # V nizsie uvdedenom priklade dostaneme ako odpoved_1 cislo 250
    odpoved_1 = odvaz([1, 2, 2, 0, 0])

    # Podobne v tomto pripade dostaneme odpoved 385
    odpoved_2 = odvaz([0, 0, 0, 7, 0])

    # Program odpovie 4 a skonci
    odpovedaj(4)

else:
    odpovedaj(42)
```

Príklad komunikácie

```
5
? 1 2 2 0 0
250
? 0 0 0 7 0
385
! 4
```

Máme 5 *vreciek. V prvom meraní zoberieme jednu mincu z prvého vrecka, po dvoch z druhého a tretieho a žiadne z ostatných. Váha nám povie, že dokopy vážia* $250g$. 5 *reálnych mincí váži* $1 \cdot 50 + 2 \cdot 50 + 2 \cdot 50 = 250g$, *takže falošné mince nemôžu byť v prvom, druhom ani treťom vrecku. V druhom meraní zoberieme* 7 *mincí zo štvrtého vrecka, ktoré vážia* $385g$. 7 *reálnych mincí by vážilo* $7 \cdot 50 = 350g$, *takže štvrté vrecko musí obsahovať falošné mince.*

2. Bazalky a čili papričky

12 b za popis, 8 b za program

Kubo je veľký milovník štipľavých jedál. Na poličke v kuchyni má N koreničiek zoradených od najmenej po najviac štipľavú. Každá korenička je označená číslom od 1 po N.
Raz však vyskúšal známu kalifornskú papričku a keď so slzami v očiach odkladal koreničky naspäť, všetky ich poprehadzoval.

Keď sa neskôr spamätal a rozhodol sa ich znova upratať, zistil, že nemá dosť síl na veľké presuny. Dokáže vždy vymeniť len dve **susedné** koreničky – teda tie, ktoré stoja na pozíciách A a $A + 1$.

Úloha

Vašou úlohou je zistiť, **na koľko najmenej susedných výmen** vie Kubo dosiahnuť,
aby sa **K najmenej štipľavých koreničiek** (čísel 1 až K) nachádzalo **na prvých** K **pozíciách**,
v **ľubovoľnom poradí**.

Formát vstupu

V prvom riadku vstupu sú dve celé čísla N a K – počet koreničiek a počet najmenej štipľavých koreničiek, ktoré chce Kubo dostať na začiatok.

V druhom riadku je N rôznych celých čísel od 1 po N, oddelených medzerami — aktuálne poradie koreničiek na poličke.

Sada	1	2	3	4
$1 \leq N \leq$	20	10^3	10^6	10^6

vo všetkých sadách platí $1 \leq K \leq N$ ## Formát výstupu

Vypíšte jedno celé číslo — minimálny počet susedných výmen, ktoré musí Kubo vykonať, aby sa najmenších K koreničiek presunulo na prvé K pozície (v ľubovoľnom poradí).

Príklad

vstup

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

výstup

```
1
```

Stačí vymeniť 3-ku s 1-kou

vstup

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

výstup

```
0
```

V tomto vstupe už netreba nič vymieňať

vstup

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

výstup

```
5
```

3. Eww, niečo je tu rozliate

12 b za popis, 8 b za program

KSPáci boli na chate a keďže málo z nich má plne vyvinutú jemnú motoriku, tak sa im na schodoch vždy podarí niečo vyliať. Na polke schodu je lekvár, na druhej zas Kofola™. Na ďalšom zas Nutella™ a kečup…

Filipkovi síce nepadajú veci z rúk ale za to si na chatu zvládol zobrať len jeden pár ponožiek. Chcel by sa teda večer dostať do izby, s čo najmenej zašpinenými ponožkami. Viete mu s tým pomôcť?

Úloha

Chata má `n` schodov. Každý schod má dve polovice. Na každej polovici je rozliate niečo, čo Filipkovi zašpiní ponožky. Pre každý schod teda poznáme dve čísla:

- ľavá strana: koľko špiny spôsobí, keď na ňu stúpi
- pravá strana: to isté, ale pre pravú polovicu

Filipko môže na začiatku začať buď na prvom alebo druhom schode, a ľubovoľnou nohou (ľavou alebo pravou). Potom ide hore po schodoch, pričom sa môže pohnúť buď o jeden schod (napr. zo schodu 2 na 3) alebo dva schody (napr. z 2 na 4). Musí striedať nohy – ak posledný krok spravil ľavou, ďalší musí pravou, a naopak. Musí skončiť presne na poslednom schode (čiže sa nesmie zastaviť skôr ani ho preskočiť).

Vašou úlohou je zistiť, aké najmenšie množstvo zašpinenia ponožiek môže Filipko utrpieť, ak sa chce po zafŕkaných schodoch dostať z dolného poschodia na posledný schod, teda najmenší možný súčet zašpinení na polovicach schodov, na ktoré stúpi.

Formát vstupu

V prvom riadku vstupu je číslo n ($2 \leq n \leq 5 \cdot 10^5$) udávajúce počet schodov na chate.

Potom budú nasledovať dva riadky s n číslami oddelenými medzerou. Prvý pre ľavú stranu a druhý pre pravú. i-te číslo v riadku prezenovať zašpinenie i-teho schodu na danej strane. Pre každé číslo zašpinenia platí: $0 < l_i, p_i \leq 1000$.

V jednotlivých sadách platia nasledujúce obmedzenia:

Sada	1	2	3	4
$1 \leq N \leq$	15	100	5 000	500 000

Formát výstupu

Vypíšte jedno číslo - najnižie možné zašpinenie ponožiek.

Príklady

vstup

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

výstup

```
2
```

Filipko preskočil prvý schod a skočil pravou nohou rovno na druhý s hodnotou 1. Odtiaľ už dočiahol skočiť na posledný schod ľavou nohou s hodnotou 1, teda dokopy 2.

vstup

```
5
1 4 5 1 5
1 7 2 2 3
```

výstup

```
7
```

1. schod ľavá noha(1), 3. schod pravá noha(2), 4. schod ľavá noha(1), 5. schod pravá noha(3): 1 + 2 + 1 + 3 = 7.

4. Delíme sa s jedlom

12 b za popis, 8 b za program

Vedúci KSP sa stretli na chate. Ako tomu už ale býva, každý sa spolieha na to, že ostatní donesú na chatu dostatok jedla, aby sa niečo ušlo aj jemu. Vedúci, ktorý si doniesol jedlo, ho má dosť pre každého na chate, no nie je vždy ochotný podeliť sa s ostatnými. Vyšlo to nakoniec ale tak, že v každej dvojici vedúcich je práve jeden z nich ochotný podeliť sa o svoje jedlo s tým druhým vedúcim z tejto dvojice.

Paulínku zaujímalo, koľko najmenej vedúcich muselo doniesť jedlo a ktorí konkrétni vedúci ho museli doniesť na to, aby nikto na chate netrpel hladom. Pomôžete jej zodpovedať túto otázku?

Úloha

Máme n vedúcich, pričom pre každú dvojicu vedúcich i, j je vždy práve jeden z nich ochotný podeliť sa s tým druhým o jedlo (ak nejaké doniesol). Každý vedúci, ktorý doniesol jedlo, ho doniesol dostatočne veľa na to, aby sa s ním vedel podeliť s kýmkoľvek, komu je ochotný toto jedlo dať, a aby niečo ostalo aj pre neho samotného. Ak však niekto dostane jedlo od niekoho iného, toto jedlo si ponechá a nedá ho už nikomu ďalšiemu, keďže to nie je jeho vlastné jedlo.

Nájdite čo najmenšiu podmnožinu vedúcich takú, že keď všetci vedúci z tejto podmnožiny donesú jedlo, tak pre každého vedúceho platí, že doniesol jedlo alebo že existuje vedúci, ktorý doniesol jedlo a je ochotný podeliť sa s ním (ide o logické alebo, teda môžu nastať aj oba prípady naraz).

Formát vstupu

V prvom riadku vstupu sú dve medzerou oddelené čísla - číslo n ($1 \leq n \leq 1000$) udávajúce počet vedúcich, ktorí prišli na chatu, a číslo s označujúce číslo aktuálnej sady vstupov ($s = 0$ v prípade, že sa jedná o sample, inak $s \in 1, 2, 3, 4$ podľa čísla aktuálnej sady).

V každom z nasledujúcich n riadkov je n čísel, z ktorých každé je buď 0, alebo 1. Označme j-te číslo v i-tom z týchto riadkov ako A_{ij}. Ak $A_{ij} = 1$, znamená to, že vedúci i je ochotný podeliť sa s vedúcim j o svoje jedlo, inak vedúci i nie je ochotný podeliť sa s vedúcim j o svoje jedlo. Špeciálne pre každé i je A_{ii} tiež rovné nule, keďže nedáva zmysel, aby sa vedúci sám so sebou delil o jedlo.

Zároveň pre každú dvojicu rôznych vedúcich i, j platí, že buď vedúci i je ochotný podeliť sa s vedúcim j o svoje jedlo, alebo vedúci j je ochotný podeliť sa s vedúcim i o svoje jedlo (teda nastáva práve jeden z týchto prípadov).

Formát výstupu

Vypíš dva riadky. V prvom z nich vypíš jedno celé číslo k, a to čo najmenší počet vedúcich, ktorí museli na chatu doniesť jedlo, aby sa každý vedúci mohol najesť. Tento počet nemusí byť nutne najnižší možný, ale stačí, že nebude vyšší než nejaký limit $k_{\max}$, ktorý je pre každú sadu rozdielny a v každej ďalšej sade sa postupne znižuje, viď. časť Hodnotenie. V druhom riadku následne v ľubovoľnom poradí vypíš čísla k vedúcich, ktorí museli doniesť na chatu jedlo. Vedúcich číslujeme od 1 do n.

Hodnotenie

Sú 4 sady vstupov, v každej platí $1 \leq n \leq 1000$. Body za jednotlivé sady sú následne udeľované podľa maximálnej hodnoty k, ktorú Tvoj program bude potrebovať spomedzi vstupov pre danú sadu. V jednotlivých sadách tak pre získanie bodov za danú sadu platia nasledujúce obmedzenia:

Sada	1	2	3	4
$k_{\max} \leq$	500	100	20	9

Ak Tvoj program prekročí časový limit v ľubovoľnom vstupe alebo vypíše skupinu vedúcich, ktorá nezaručí to, že každý vedúci bude mať prístup k jedlu, tak Tvoj program skončí chybovou hláškou (`TLE` v prvom prípade a `WA` v druhom prípade) a nedostaneš žiadne body za danú sadu. Podobne ak v nejakom vstupe prekročíš hodnotu $k_{\max}$ danej sady, Tvoj program skončí chybou `WA` a nedostaneš za danú sadu žiadne body.

V popise k tejto úlohe okrem popísania postupu, ktorým Tvoj program vyberá vhodné podmnožiny vedúcich, nezabudni tiež napísať hodnotu k, ktorú tento postup v najhoršom prípade dosahuje, spolu s dôkazom tejto hornej hranice veľkosti vybratej podmnožiny. Okrem toho tiež popíš a zdôvodni časovú a pamäťovú zložitosť svojho algoritmu, hoci sa pri hodnotení na ne nebude klásť až taká váha, ako na veľkosť horného odhadu hodnoty k a na dôkaz toho, že ju popísaný postup nikdy neprekročí.

Príklady

vstup

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

výstup

```
1
1
```

Vedúci 1 *je ochotný podeliť sa o jedlo s obomi zvyšnými vedúcimi, preto stačí, aby doniesol jedlo iba on.*

vstup

```
4 0
0 1 0 0
0 0 1 0
1 0 0 1
1 1 0 0
```

výstup

```
2
2 4
```

Vedúci $1, 2, 3, 4, 1$ *v tomto poradí tvoria cyklus toho, kto je komu ochotný dať jedlo, preto stačí vybrať napríklad každého druhého vedúceho v tomto cykle. Ak by doniesol len jeden vedúci jedlo, nestačilo by to na nasýtenie každého.*

5. Už nie je viac jedla

12 b za popis, 8 b za program

Je to tak, už ste všetko zjedli. Je načase ísť pracovať. Dnes sa na programovaní naučíte pracovať s Unixovým terminálom. Ako prvý sa naučíte príkaz `echo` s veľmi príznačným menom, ktorý ako ozvena vypíše svoje parametre na štandardný výstup.

Potom však prídete k príkazom, ktoré začnú mať príliš krátke a nezrozumiteľné mená. Začne to úplne nebadane, príkazmi `cp`, `mv` a `rm` na kopírovanie, presúvanie a mazanie súborov, ktorých mená vznikli jednoducho z CoPy, MoVe a ReMove. Neskôr sa to začne zhoršovať (aj si hovoríte že ste mali jest pomalšie ale už je neskoro). Potom prídete napríklad k príkazom `chmod` (change mode), `tar` (tape archiver), `tr` (transform) či `chgrp` (change group). Vy sa snažíte sústrediť ale jediné čo vám chodí po rozume je jedlo a to že kedy bude dalšie.

Nič si nezapamätáte, ale zato večeru už máte premyslenú. Idete radšej programovať v jazyku `C`. Vôbec si však nepomôžete, práve naopak. Tu sú mená funkcií nielen prikrátke, ale priam až kryptické. Napríklad `strrchr`. Po dlhých minútach zamyslenia (rozptyľuje vás predstava blížiacej sa večere) zistíte že to `str` znamená `string`, `r` znamená `reverse` a `chr` znamená `character` a funkcia teda hľadá výskyt znaku v reťazci odzadu. Keďže vás to premýšľanie začalo bolieť, rozhodli ste sa už druhýkrát zmeniť predmet svojho záujmu. Vybrali ste si jazyk SNOBOL. Ale toto vám už nedalo. Prečo práve SNOBOL? Ako k tomu došli? Vraj StriNg Oriented symBOlic Language. To nijak nedáva zmysel. Napokon ste došli k tomu, že vo všeobecnosti skratka vzniká tak, že len jednoducho vyberieme niektoré znaky názvu, pričom zachováme ich poradie a zapíšeme ich za seba (kiežby ste aj k tej večery už konečne došli). Z každého slova musíte vybrať aspoň jedno písmeno. A žiadne ďalšie pravidlá zrejme neplatia. Vy by ste si chceli vedieť overovať, či je daná skratka platná ale chcete to automatizovať aby sa pri tom dalo jesť. Tak šup šup, už začína večera ale vy nemôžte ísť kým nevylúštite zvyšok.

Úloha

Vašou úlohou je spočítať počet možností, ako mohla vzniknúť skratka S z názvu T.

Aby skratka S mohla vzniknúť z textu T, musí platiť, že písmenám v S vieme priradiť písmená v T, pričom musíme zachovať poradie (teda ak priradíme písmenko z S nejakému písmenku z T, ďalšiemu môžeme priradiť len nejaké napravo v T) a z každého slova T musíme použiť aspoň jedno písmeno.

Keďže počet možností je v niektorých prípadoch naozaj veľký, vypíšte len jeho zvyšok po delení $10^9 + 7$.

Formát vstupu

Na prvom riadku sa nachádza číslo w – počet slov textu, z ktorého má byť utvorená skratka.

Na druhom riadku sa nachádza skratka - reťazec S a na treťom riadku sa nachádza text T, ktorého skratka to má byť. Oba reťazce na vstupe pozostávajú len z malých písmen anglickej abecedy a medzier. Jednotlivé slová T sú oddelené práve jednou medzerou (medzier je teda $w - 1$). Skratka žiadne medzery neobsahuje.

Formát výstupu

Vypíšte jedno číslo – počet možností, ako možno priradiť písmenám skratky S písmená textu T tak, aby tvorili platnú skratku, modulo $10^9 + 7$. Ak to nie je možné, vypíšte 0.

Hodnotenie

Nech M je dĺžka skratky a N dĺžka textu. Sú 4 sady vstupov. V jednotlivých sadách platia nasledovné obmedzenia:

1. $M \leq 5$, $N \leq 12$
2. $M \leq 12$, $N \leq 50$
3. $M \leq 120$, $N \leq 200$
4. $M \leq 200$, $N \leq 10000$

Príklady

vstup

```
2
tar
tape archiver
```

výstup

```
4
```

Možnosti sú TApe aRchiver, TApe archiveR, Tape ARchiver a Tape ArchiveR.

vstup

```
4
snobol
string oriented symbolic language
```

výstup

```
1
```

vstup

```
3
strrchr
string reverse char
```

výstup

```
3
```

vstup

```
4
radar
radio detection and ranging
```

výstup

```
1
```

Jediná možnosť je RAdio Detection And Ranging. RADio detection And Ranging nie je platná možnosť, pretože zo slova detection sme nepoužili žiadne písmeno.

vstup

```
4
acm
academy of computer makers
```

výstup

```
0
```

Skratka je jednak prikrátka, a jednak neobsahuje žiadne z písmen slova of.

6. Žeby závod?

12 b za popis, 8 b za program

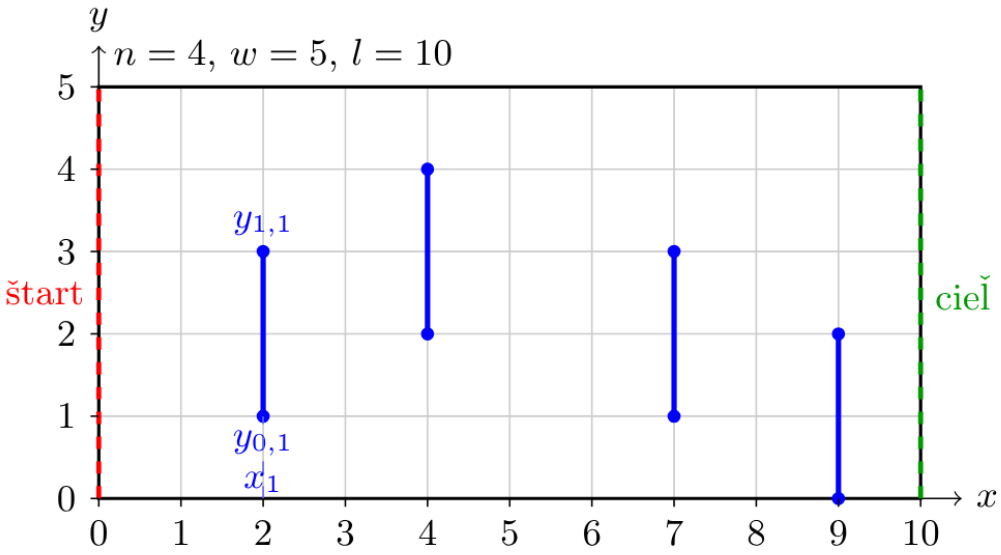
Vedúci zistili, že nemajú dostatok chutných lasagní pre všetkých a niektorí vedúci budú musieť jesť nie až tak oblúbenú dusenú zeleninu. Takmer okamžite sa strhla hádka o to, kto bude jesť lasagne, a kto nie. Aby sa vedúci o lasagne nepobili, tak sa dohodli, že si o ne spravia závod na elektrokolobežkách.

Závod sa uskutoční na parkovisku širokom w metrov a dlhom l metrov. Na tomto parkovisku je taktiež n nabíjacích staníc pre kolobežky, ktoré pri prechode cez ne instantne nabijú kolobežku na 100%. Keďže baterky sú ťažké, tak by každý vedúci rád vedel, akú najmenšiu baterku potrebuje, aby prešiel celú trasu. Pomôžte im to zistiť.

Úloha

Máme parkovisko široké w metrov a dlhé l metrov. Na tomto parkovisku sa nachádza n nabíjacích staníc. Nabíjacia stanica je zvislý pruh, i-ta nabíjacia stanica sa nachádza v stĺpci x_i, na rozpätí riadkov $y_{0,i}$ až $y_{1,i}$. Ak kolobežka prejde cez nabíjaciu stanicu (počíta sa aj jej okraj), tak sa okamžite nabije na plnú kapacitu.

Máme $w+1$ možných štartovacích pozícií pre kolobežky, $y \in [0, w]$, a pre každú z nich by sme radi vedeli akú najmenšiu batériu potrebuje mať kolobežka závodiaca na danom riadku, aby zvládla prejsť celú trasu. Vieme, že za jednu jednotku nabitia prejde kolobežka jeden meter.



Formát vstupu

V prvom riadku vstupu sú čísla n, w, l ($1 \leq n, w, l \leq 2 \times 10^5$) udávajúce počet nabíjacích staníc, šírku a dĺžku parkoviska.

Na nasledujúcich n riadkoch sa nachádzajú popisy nabíjacích staníc. Na i-tom riadku sú čísla $x_i, y_{0,i}, y_{1,i}$, ($0 < x_i < l, 0 \leq y_{0,i} < y_{1,i} \leq w$).

V jednotlivých sadách platia nasledujúce obmedzenia:

Sada	1	2	3,4
$1 \leq n \leq$	1 000	1 000	2×10^5
$1 \leq w \leq$	1 000	1 000	2×10^5
$1 \leq l \leq$	1 000	2×10^6	2×10^6

Je garantované, že sa nabíjacie stanice nepretínajú, a ani nedotýkajú.

Formát výstupu

Vypíš $w+1$ riadkov, na i-tom z nich vypíš minimálnu potrebnú kapacitu pre kolobežku na i-tom riadku.

Príklad

vstup

```
4 5 10
2 1 3
4 2 4
7 1 3
9 0 2
```

výstup

```
9
5
3
3
6
10
```

7. Jedenie v polceste

12 b za popis, 8 b za program

Kašľať na programovanie, AIčky a počítače. Treba sa vrátiť ku koreňom, dotknúť sa trávy. Preto ste zanevreli na Korešpondenčný Seminár z Programovania a otvorili ste si radšej agentúru Komfortné Spojenie s Prírodou, ktorá ponúka caterované túry s horským vodcom. Vymysleli ste si ponuku n túr a na každej by ste chceli ponúkať turistom teplý obed.

Nosiť jedlo zo sebou zaberá priveľa miesta (a ešte by vychladlo), preto ste sa radšej rozhodli otvoriť poľné kuchyne na miestach, kde sa nachádza prestávka na obed. Okrem toho, že zo sebou nemusíte nosiť jedlo, majú poľné kuchyne tú výhodu, že vedia ponúkať obed pre viac než jednu túru.

Keďže treba maximalizovať zisk, chceli by ste postaviť čo najmenej poľných kuchýň. Ale keďže sľubujete zákazníkom Komfort, obed by mal byť na správnom mieste – a to čo najbližšie k polceste túry. Avšak, vaše biznis plány začali byť príliš ambiciózne, a možno sa budete na ten počítač obrátiť ešte raz, aby ste tento logistický problém vyriešili.

Úloha

Existuje n zaujímavých miest, po ktorých môžete viesť turistov. Medzi $n-1$ pármi z nich vedú turistické chodníky, pričom sa dá z každého miesta dostať na každé iné nejakou postupnosťou ciest (čiže hory majú tvar *stromu*).

Agentúra Komfortné Spojenie s Prírodou ide ponúkať m túr. Trasa každej túry je najkratšia cesta medzi dvoma zaujímavými miestami (dá sa ukázať, že pre každú dvojicu miest existuje presne jedna najkratšia cesta). Obed by mal byť ponúkaný presne v polceste túry, pričom dĺžka cesty je počet zaujímavých miest na nej. Poľnú kuchyňu nechceme otvárať v strede turistického chodíka, a preto v prípade, že na túry navštívime párny počet zaujímavých miest, $a_1, \ldots, a_{2k}$, obed môže byť na ktoromkoľvek z dvoch miest najbližšie ku polceste, čiže buď v a_k alebo a_{k+1}.

Koľko najmenej kuchýň musíte otvoriť?

Formát vstupu

V prvom riadku vstupu sú dva čísla n a m ($1 \leq n, m \leq 10^5$) – počet zaujímavých miest a počet ponúkaných túr, oddelené medzerou.

Na ďalších $n-1$ riadkoch nasleduje popis turistických chodníkov. Popis každého chodníka pozostáva z dvoch čísel a a b ($1 \leq a, b \leq n$) oddelených medzerou, označujúcich, že medzi miestami a a b existuje priamy turistický chodník.

Je garantované, že sa dá nejakou postupnosťou turistických chodníkov presunúť medzi každou dvojicou zaujímavých miest.

Na ďalších m riadkoch nasleduje popis túr: popis každej túry pozostáva z dvoch čísel a a b ($1 \leq a, b, \leq n$) oddelených medzeru, označujúcich, že Komfortné Spojenie s Prírodou ponúka túru, ktorá začína na mieste a a končí na mieste b.

Formát výstupu

Na prvom riadku vstupu vypíšte jedno celé číslo k – najmenší počet kuchýň ktoré musíte otvoriť.

Na druhom riadky vstupu vypíšte k medzerou oddelených čísel – miesta na ktorých by ste mali otvoriť poľné kuchyne, tak aby pokryli potreby všetkých ponúkaných túr. Ak existuje viacero riešení, môžete vypísať ľubovoľné z nich.

Hodnotenie

Úloha má osem testovacích sád. V jednotlivých sadách platia nasledujúce obmedzenia:

Sada	1	2, 3, 4	5, 6, 7, 8
$1 \leq n, m \leq$	20	1 000	10^5

V sadách 4 a 5 navyše platí, že každá túra obsahuje presne *nepárny* počet miest.

V sadách 2 a 6 navyše platí, že i-ty turistický chodník spája miesta číslo i a $i+1$, čiže chodníky tvoria jednu dlhú cestu.

Príklady

vstup

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

výstup

```
1
3
```

Tento príklad by sa mohol nachádzať v druhej alebo šiestej sade. Prvá aj druhá túra musia mať obe obed na treťom mieste. Tretia túra môže mať obed buď na druhom, alebo na treťom mieste. Ak si vyberieme tretie, stačí nám na pokrytie všetkých túr len jedna poľná kuchyňa. Všimnite si, že druhá túra začína aj končí tom istom mieste – na tom mieste teda musí byť aj obed.

vstup

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

výstup

```
3
1 7 5
```

Prvá túra potrebuje poľnú kuchyňu buď v prvom alebo treťom mieste. Pre druhú túru potrebujeme otvoriť poľnú kuchyňu v druhom, alebo prvom mieste. Zabezpečiť catering pre tieto dve túry teda vieme tak, že otvoríme kuchyňu v prvom mieste. Hoci sa tretia túra čiastnočne prekrýva s prvou, a štvrtá s treťou, pri týchto túrach nemáme na výber: poľné kuchyne musíme postaviť v piatom a siedmom mieste.

8. Escargot

12 b za popis, 8 b za program

Niet času na vysvetlenie. Musíte hneď TERAZ zjesť slimáka. Ak to nedokážete, tak sa asi stane niečo zlé. Neviem, som len zadanie úlohy číslo 8 v KSP, aj tak ma nikto nečíta.

Samozrejme, nie sme barbari. Slimák sa servíruje ako escargot, čo po Francúzsky niečo znamená.

Slimáka si vieme predstaviť ako dvojicu disjunktných útvarov (nie nutne súvislých!) pozostávajúcich z políčok v nekonečnej štvorčekovej mriežke[1], teda každý obsahuje aspoň jedno políčko a žiadne políčko sa nenachádza v oboch z nich. Jeden z týchto útvarov predstavuje svalnatú nohu slimáka, a druhý jeho ulitu. Samozrejme, ulitu jesť nechceme, pretože máme vkus a česť[2]. Takže najprv musíme ulitu od svalnatej nohy oddeliť.

Na to existuje špeciálny exkluzívny escargot príbor[3], ktorým dokážeme presunúť buď celú ulitu, alebo celú svalnatú nohu o jedno políčko v jednom zo štyroch kardinálnych smerov (všetky políčka jedného z útvarov sa posunú o jedno políčko hore, dole, doprava alebo doľava). Samozrejme, v žiadnom momente sa nesmie na tom istom mieste nachádzať políčko svalnatej nohy aj ulity zároveň, pretože by nastalo verejné pobúrenie, pokles populácie potočákov[4], a v neposlednej rade aj kolaps Pauliho kvantových princípov a s nimi reality ako ju poznáme.

Samozrejme, vašou úlohou je zistiť, či je možné takýmito manévrami ulitu od svalnatej nohy oddeliť, a ak áno, ako. Konkrétne, treba umiestniť oba útvary do takej polohy, že v aspoň jednom zo štyroch kardinálnych smerov bude mať každé políčko jedného z nich vyššiu súradnicu ako každé políčko druhého (inak povedané bude existovať priamka rovnobežná s horizontálnou alebo vertikálnou osou, ktorá tieto útvary od seba oddeľuje).

[1] Samozrejme, v praxi nekonečná štvorčeková mriežka na jedenie slimákov neexistuje. Pokiaľ vám to robí problém, jednoducho si predstavte, že viem, kde bývate, a osobne k vám o tretej ráno prídem, pokiaľ sa budete ďalej sťažovať.

[2] Opäť pre účely úlohy predpokladajte, že to je tak.

[3] https://pl.wikipedia.org/wiki/%C5%81y%C5%BCkowidelec

[4] https://en.wikipedia.org/wiki/Salmo_trutta_fario

Reklamná vsuvka

Hej. Hej! Pst! Ty! Počúvaj. Teraz ti prezradím prastaré kozmické tajomstvo. Tvoj program pravdepodobne beží príliš pomaly, pretože slimačí sliz ti zalepil procesory a grafiky. Ak chceš bežať rýchlejšie, budeš potrebovať legendárny algoritmus na násobenie polynómov menom Fast Fourier Transform[5]. Klikni SEM[6] a obdrž doživotnú zásobu.

Úloha

Zistite, či je možné od seba dva útvary v nekonečnej štvorečkovej mriežke oddeliť (teda umiestniť ich tak, aby existovala priamka rovnobežná s vodorovnou alebo zvislou osou taká, že oba útvary úplne oddeľuje) iba pohybmi o jedno políčko v kardinálnych smeroch, a ak áno, zistite ako. Vaše riešenie musí pozostávať z najviac 10^7 pohybov. To, že takéto riešenie existuje, je garantované.

Táto úloha sa môže testovať dlho, buďte trpezliví a milí k testovaču. Zároveň si buďte vedomí, že táto úloha má jedinú sadu a nedajú sa získať čiastkové body, teda nemá zmysel odovzdávať pomalé riešenie hrubou silou.

Formát vstupu

V prvom riadku vstupu sú čísla m, n, udávajúce veľkosť oblasti mriežky, ktorá obsahuje oba útvary.

Na každom z m nasledujúcich riadkov je reťazec n znakov určujúci do ktorého útvaru dané políčko riadku patrí: `.` ak nepatrí do ani jedného, zatiaľ čo `@` a `#` reprezentujú oba útvary.

V jedinej sade vstupov platí $1 \leq m, n \leq 1000$.

Formát výstupu

Ak sa útvary nedajú oddeliť vypíšte na jediný riadok výstupu číslo `-1`.

Ak sa oddeliť dajú, vypíšte na prvý riadok výstupu počet pohybov $k \leq 10^7$, ktorými to dosiahnete (nemusí byť optimálny). Na nasledujúcich k riadkoch vypíšte vaše kroky vo formáte `c d`, kde `c` je znak `@` alebo `#`, podľa toho, ktorým útvarom hýbete, a `d` je veľké písmenko `N`, `E`, `S` alebo `W`, podľa kardinálneho smeru v ktorom množinou hýbete v angličtine (North je hore, South je dole, East je doprava, West je doľava).

Príklady

vstup

```
6 4
.#..
...#
#@#.
##..
#...
.@..
```

výstup

```
5
@ N
@ E
# S
@ E
@ E
```

Po vykonaní týchto pohybov sa každé políčko @ nachádza napravo od každého políčka #.

vstup

```
4 4
##..
@...
#@.#
..##
```

výstup

```
-1
```

[5] Rýchlosť tohto algoritmu spočíva v tom, že má v názve slovo fast.
[6] https://cp-algorithms.com/algebra/fft.html