Hodometras su akmenukais

Leonardas išrado pirmąjį *hodometrą*: vežimėlį, kuris, besisukant jo ratams, barsto akmenukus ir tokiu būdu matuoja nuvažiuotą atstumą. Suskaičiavęs akmenukus, vartotojas sužino ratų apsisukimų kiekį ir tada gali apskaičiuoti hodometro nukeliautą atstumą. Mes, informatikai, išplėsime hodometro galimybes, sukūrę programinę įrangą jo valdymui. Taigi, užprogramuokite hodometrą pagal žemiau pateiktas taisykles.

Darbinė lentelė

Hodometras juda įsivaizduojamoje kvadratinėje lentelėje, kurią sudaro 256 × 256 vienetinių kvadratėlių. Kiekviename kvadratėlyje telpa 15 akmenukų. Kvadratėlį apibūdina koordinačių pora (eilutė, stulpelis), kur kiekviena koordinatė yra iš intervalo 0, …, 255. Kvadratėliui (i, j) gretimi kvadratėliai yra (jei tokie egzistuoja) (i - 1, j), (i + 1, j), (i, j - 1) ir (i, j + 1). Kvadratėliai, esantys pirmoje ar paskutinėje eilutėje arba pirmame ar paskutiniame stulpelyje, yra vadinami *kraštu*. Pradžioje hodometras yra kvadratėlyje (0, 0) (šiaurės vakarinis kampas), pasisukęs į šiaurę.

Pagrindinės komandos

Hodometras programuojamas šiomis komandomis:

- `left` — pasisukti 90 laipsnių į kairę (priešinga laikrodžio rodyklei kryptimi) ir pasilikti tame pačiame kvadratėlyje (pvz., jei prieš šią komandą hodometras buvo pasisukęs į pietus, tai įvykdęs šią komandą jis pasisuks į rytus);
- `right` — pasisukti 90 laipsnių į dešinę (laikrodžio rodyklės kryptimi) ir pasilikti tame pačiame kvadratėlyje (pvz., jei prieš šią komandą hodometras buvo pasisukęs į vakarus, tai įvykdęs šią komandą jis pasisuks į šiaurę);
- `move` — pajudėti į gretimą kvadratėlį ta kryptimi, kuria hodometras yra pasisukęs. Jei šio kvadratėlio nėra (t.y., hodometras pasiekė kraštą ta kryptimi), tai ši komanda neturi jokio efekto.
- `get` — paimti vieną akmenuką iš kvadratėlio, kuriame stovi hodometras. Jei kvadratėlyje nėra akmenukų, ši komanda neturi jokio efekto.
- `put` — kvadratėlyje, kuriame stovi hodometras, padėti vieną akmenuką. Jei kvadratėlyje jau yra 15 akmenukų, ši komanda neturi jokio efekto. Hodometrui niekada nepritrūks akmenukų.
- `halt` — nutraukti vykdymą.

Hodometras komandas vykdo tokia tvarka, kokia jos yra pateiktos programoje. Kiekvienoje programos eilutėje turi būti ne daugiau kaip viena komanda. Į tuščias eilutes nekreipiama dėmesio. Simbolis # reiškia komentarą: į tekstą, esantį nuo šio simbolio iki eilutės pabaigos, nekreipiama dėmesio. Jei hodometras pasiekia programos failo pabaigą, vykdymas nutraukiamas.

1-as pavyzdys

Nagrinėkime tokią hodometrui pateiktą programą. Ją vykdydamas, hodometras nukeliaują į kvadratėlį (0, 2) ir lieka pasisukęs į rytus. Atkreipkite dėmesį, kad pirmoji `move` komanda yra praleidžiama, nes hodometras yra šiaurės vakariniame kampe pasisukęs į šiaurę.

```
move  # jokio efekto
right
# dabar hodometras pasisukęs į rytus
move
move
```

Žymės, kraštai ir akmenukai

Norėdami pakeisti programos eigą, galite naudoti žymes: simbolių eilutes sudarytas iš ne daugiau kaip 128 simbolių iš intervalų `a`, …, `z`, `A`, …, `Z`, `0`, …, `9`. Atkreipiame dėmesį, kad didžiosios ir mažosios raidės laikomos skirtingomis. Su žymėmis susijusios komandos pateiktos žemiau. Jų aprašymuose *`L`* reiškia bet kurią taisyklingą žymę.

- *`L`*`:` (t.y. *`L`* ir dvitaškis '`:`') — aprašo žymės *`L`* vietą programoje. Visos aprašytos žymės turi būti skirtingos. Žymės aprašymas neturi jokios įtakos hodometrui.
- `jump` *`L`* — tęsti vykdymą besąlygiškai peršokant į eilutę, kurioje yra žymė *`L`*.
- `border` *`L`* — tęsti vykdymą peršokant į eilutę, kurioje yra žymė *`L`*, jei hodometras yra krašte ir pasisukęs kryptimi, kuria nėra gretimo kvadratėlio, (t.y., `move` komanda neturėtų jokio efekto); priešingu atveju, ši komanda neturi jokio efekto ir vykdymas tęsiamas nuo tolesnės komandos.
- `pebble` *`L`* — tęsti vykdymą peršokus į eilutę, kurioje yra žymė *`L`*, jei kvadratėlyje, kur jis dabar stovi, yra bent vienas akmenukas; priešingu atveju, ši komanda neturi jokio efekto ir vykdymas tęsiamas nuo tolesnės komandos.

2-as pavyzdys

Pateikta programa suranda pirmą (labiausiai nutolusį į vakarus) akmenuką eilutėje 0, hodometras sustoja tame lagelyje ir baigia darbą; jei eilutėje 0 nėra akmenukų, hodometras sustoja krašte, eilutės pabaigoje. Ji naudoja dvi žymes: `leonardo` ir `davinci`.

```
right
leonardo:
pebble davinci  # rastas akmenukas
border davinci  # eilutės pabaiga
move
jump leonardo
davinci:
halt
```

Pradžioje hodometras pasisuka dešinėn. Ciklas prasideda žymės apibrėžimu `leonardo:` ir baigiasi komanda `jump leonardo`. Cikle hodometras tikrina, ar yra akmenukas ir ar pasiekė eilutės pabaigą; jei ne, hodometras atlieka `move` komandą iš esamojo kvadratėlio (0, j) į gretimą kvadratėlį (0, j + 1), kadangi pastarasis yra lentelėje. (Komanda `halt` čia nėra būtinai reikalinga nes programa bet kuriuo atveju sustos.)

Užduotis

Pateikite programą hodometro programavimo kalba (aprašyta aukščiau), kurią vykdydamas hodometras elgsis, kaip tikimasi. Kiekviena užduotis (žr. žemiau) aprašo elgseną, kurios tikimasi iš hodometro, ir ribojimus, kuriuos pateikti sprendimai turi atitikti. Ribojimai yra dviejų tipų:

- *Programos dydis* — programa turi būti pakankamai trumpa. Programos dydis lygus joje esančių komandų skaičiui. Žymių apibrėžimai, komentarai ir tuščios eilutės *neskaičiuojamos*.
- *Vykdymo trukmė* — programa turi sustoti pakankamai greitai. Vykdymo trukmė lygi atliktų *žingsnių* skaičiui: kiekvienas komandos įvykdymas lygus vienam žingsniui, nepriklausomai nuo to, ar komanda turėjo kokį nors efektą, ar ne; žymių apibrėžimai, komentarai ir tuščios eilutės nėra žingsniai, todėl *neskaičiuojamos*.

Pirmame pavyzdyje ir programos dydis, ir vykdymo trukmė yra 4. Antrame pavyzdyje programos dydis yra 6, o vykdymo laikas, įvykdžius lentelėje su akmenuku kvadratėlyje (0, 10), yra 43 žingsniai: `right`, 10 ciklo iteracijų, trunkančių po 4 žingsnius (`pebble davinci`; `border davinci`; `move`; `jump leonardo`), ir, galiausiai, `pebble davinci` bei `halt`.

1 užduotis [9 taškai]

Pradžioje kvadratėlyje (0, 0) yra x akmenukų, kvadratėlyje (0, 1) yra y akmenukų, o visi kiti kvadratėliai yra tušti. Primename, kad kvadratėlyje telpa 15 akmenukų. Parašykite programą, kuriai sustojus hodometras būtų kvadratėlyje (0, 0), jei $x \leq y$, arba kvadratėlyje (0, 1), priešingu atveju. Hodometro kryptis programos pabaigoje yra nesvarbi; akmenukų kiekis ir pozicijos programos pabaigoje taip pat nesvarbios.

Ribojimai: programos dydis ≤ 100, vykdymo trukmė ≤ 1 000.

2 užduotis [12 taškų]

Taip pat kaip ir pirmoje užduotyje, tik programos pabaigoje kvadratėlyje (0, 0) turi būti lygiai x akmenukų, o kvadratėlyje (0, 1) — lygiai y.

Ribojimai: programos dydis ≤ 200, vykdymo trukmė ≤ 2 000.

3 užduotis [19 taškų]

Eilutėje 0 yra lygiai du akmenukai: vienas yra kvadratėlyje (0, x), o kitas — kvadratėlyje (0, y); x ir y yra skirtingi, o x + y yra lyginis. Parašykite programą, kuriai pasibaigus hodometras bus kvadratėlyje (0, (x + y) / 2), t.y. lygiai per vidurį tarp kvadratėlių su akmenukais. Lentelės būsena pasibaigus programai yra nesvarbi.

Ribojimai: programos dydis ≤ 100, vykdymo trukmė ≤ 200 000.

4 užduotis [iki 32 taškų]

Lentelėje yra daugiausiai 15 akmenukų, tačiau visi akmenukai yra skirtinguose kvadratėliuose. Parašykite programą, kuri surinktų juos į šiaurės vakarinį kampą; tiksliau, jei pradžioje lentelėje buvo x akmenukų, tai pabaigoje kvadratėlyje (0, 0) turi būti lygiai x akmenukų, o visur kitur turi būti tuščia.

Šios užduoties įvertinimas priklauso nuo pateiktos programos vykdymo trukmės. Jei L yra didžiausia testų vykdymo trukmė, įvertinimas bus lygus:

- 32 taškai, jei $L \leq 200\,000$;
- $32 - 32 \log_{10} (L / 200\,000)$ taškų, jei $200\,000 < L < 2\,000\,000$;
- 0 taškų, jei $L \geq 2\,000\,000$.

Ribojimai: programos dydis ≤ 200.

5 užduotis [iki 28 taškų]

Kiekviename langelyje gali būti bet koks akmenukų kiekis (nuo 0 iki 15). Parašykite programą, kuri surastų mažiausią akmenukų kiekį langelyje, t.y., pasibaigus programai, hodometras būtų tokiame kvadratėlyje (i, j), kad kiekvienas kitas kvadratėlis turėtų ne mažiau akmenukų nei (i, j). Programos pabaigoje kiekviename kvadratėlyje turi būti tiek pat akmenukų kaip ir pradžioje.

Šios užduoties įvertinimas priklauso nuo pateiktos programos dydžio P. Įvertinimas bus lygus:

- 28 taškai, jei $P \leq 444$;
- $28 - 28 \log_{10} (P / 444)$ taškų, jei $444 < P < 4\,440$;
- 0 taškų, jei $P \geq 4\,440$.

Ribojimai: vykdymo trukmė ≤ 44 400 000.

Realizacija

Jums reikia pateikti po vieną failą kiekvienai užduočiai, parašytą pagal aukščiau išdėstytas sintaksės taisykles. Didžiausias pateikiamo failo dydis gali būti 5 MiB. Kiekvienai užduočiai, jūsų hodometro

programa bus testuojama su keliais testais ir jūs gausite tam tikrą grįžtamąjį ryšį apie jūsų programos naudojamus resursus. Jei programa nėra sintaksiškai taisyklinga, gausite informacijos apie konkrečią sintaksės klaidą.

Jūsų sprendime neprivalo būti po programą kiekvienai užduočiai. Jei sprendime nėra X užduoties programos, paskutinė jūsų X užduoties programa bus automatiškai pridėta; jei tokios programos nėra, tos užduoties įvertinimas bus lygus 0.

Pateikto sprendimo įvertinimas lygus visų užduočių įvertinimų sumai, o galutinis uždavinio įvertinimas yra didžiausias iš visų *vertinimui* pateiktų sprendimų ir paskutinio sprendimo.

Simuliatorius

Testavimui galite naudotis pateiktu hodometro simuliatoriumi, kuriam galite pateikti savo programas ir pradines lenteles.

Lentelės aprašymas turi būti pateiktas tokiu formatu: kiekvienoje failo eilutėje turi būti po tris sveikuosius skaičius R, C ir P, kurie reiškia, kad kvadratėlis (R, C) turi P akmenukų. Nepaminėti kvadratėliai neturi akmenukų. Pavyzdžiui:

```
0 10 3
4 5 12
```

Aprašytoji lentelė turi 15 akmenukų: 3 kvadratėlyje (0, 10) ir 12 kvadratėlyje (4, 5).

Testų simuliatorių galite iškviesti paleidę programą `simulator.py`, esančią užduoties kataloge, pateikdami programos failą kaip argumentą. Simuliatoriaus programa priima tokias komandinės eilutės nuostatas:

- `-h` pateiks trumpą galimų nuostatų apžvalgą;
- `-g GRID_FILE` perskaitys lentelės būseną iš failo `GRID_FILE` (jei nepateikta: tuščia lentelė);
- `-s GRID_SIDE` nustatys lentelės dydį į `GRID_SIDE x GRID_SIDE` (numatytoji parinktis: 256); mažesnės lentelės gali būti patogios derinant programą;
- `-m STEPS` apribos simuliacijos vykdymo trukmę iki `STEPS` žingsnių;
- `-c` įjungs kompiliavimo režimą; šiame režime simuliatorius pateiks tuos pačius rezultatus, tačiau simuliaciją vykdys ne su Python, o sugeneruos ir sukompiliuos mažą C programą. Simuliatoriaus pradžia užtruks, tačiau veiks greičiau. Rekomenduojame naudoti šį režimą, kai tikitės atlikti daugiau nei 10 000 000 žingsnių.

Sprendimų kiekis

Šiai užduočiai leidžiama pateikti 128 sprendimus.