1. Dominokivid

Üks võimalus selle ülesande lahendamiseks on lugeda sisendist antud kivide hulk ja seejärel kontrollida iga võimaliku N-doomino kivi kohta, kas see kivi sisendis esines või mitte. Selle ülesande väikeste andmemahtude juures võiks kivide hulka hoida lihtsalt loendina ja iga kivi olemasolu kontrollimiseks seda loendit järjest läbi käia, nagu on tehtud lahenduses `domlah0.pas`.

Üldjuhul, kui hulga elementide ja/või päringute arv on suurem, on kasulikum hulga järjestikust läbivaatust vältida. Standardne võte mõõduka suurusega hulga efektiivseks hoidmiseks on teha muutujate kogum, kus igale võimalikule elemendile vastab üks tõeväärtus. Dominokivide puhul on kõige lihtsam kasutada kahemõõtmelist massiivi, kus kivile $x : y$ vastab massiivi element $A_{x,y}$. Lisaks peab veel otsustama, kuidas käsitleda asjaolu, et kivid $x : y$ ja $y : x$ on tegelikult üks kivi. Üks võimalus on sisendist kivi $x : y$ lugemisel mõlemad variandid olemasolevaks märkida, nagu on tehtud failis `domlah1a.pas`. Teine võimalus on kõik viia iga kivi enne edasist töötlemist mingile ühesele kujule, näites $x \leqslant y$, nagu on tethud failis `domlah1b.pas`.

Muidugi, keeltes, kus on olemas standardne hulgatüüp, võib ka seda kasutada. Kui hulgatüüp ei võimalda hoida kirjeid (nagu näiteks Pascalis), siis võib iga doominokivi vaadelda kahekohalise arvuna ja moodustada kivide hulga asemel neid esindavate arvude hulga, nagu on tehtud failides `domlah2a.pas` ja `domlah2b.pas`.

Veel üks võimalus on läheneda ülesandele teisest suunast: konstrueerida alguses kõigi N-doomino kivide hulk ning eemaldada sellest sisendis antud kivid. Lõpuks hulka jäänud kivid ongi otsitav vastus. Sellel ideel põhinevad failides `domlah3a.cpp` ja `domlah3b.cpp` toodud lahendused.

Testid

1. $N = 2$, $K = 3$. $P = 3$. Kõige lihtsam test: kõik kivid kanoonilisel kujul ja leksikograafilises järjekorras. 3 punkti.
2. $N = 2$, $K = 3$. $P = 3$. Osa kive pole kanoonilisel kujul, aga kivid on leksikograafilises järjekorras. 3 punkti.
3. $N = 2$, $K = 3$. $P = 3$. Kivid on kanoonilisel kujul, aga pole leksikograafilises järjekorras. 3 punkti.
4. $N = 2$, $K = 3$. $P = 3$. Kivid pole ei kanoonilisel kujul ega leksikograafilises järjekorras. 3 punkti.
5. $N = 6$, $K = 4$. $P = 24$. Juhuslik test standardse 6-doomino komplektiga. 3 punkti.
6. $N = 6$, $K = 14$. $P = 14$. Juhuslik test standardse 6-doomino komplektiga. 3 punkti.
7. $N = 6$, $K = 24$. $P = 4$. Juhuslik test standardse 6-doomino komplektiga. 3 punkti.
8. $N = 2$, $K = 6$. $P = 0$. Erijuhtum: sisendis on täiskomplekt. 3 punkti.
9. $N = 0$, $K = 0$. $P = 1$. Erijuhtum: sisendis on tühi hulk, minimaalne N. 3 punkti.
10. $N = 9$, $K = 0$. $P = 55$. Erijuhtum: sisendis on tühi hulk, maksimaalne N. 3 punkti.

Kokku 30 punkti.

2. Valguskiir

Selle ülesande lahendamiseks on vaja ära programmeerida kolm tegevust:

1. Kiire sisenemispunkti numbri teisendamine vastava ruudu koordinaatideks

   Selle juures tulevad kasuks tähelepanekud, et:

   - kui $1 \leqslant s \leqslant v$, siis siseneb kiir ülemisest servast ja liigub allapoole; veel tuleb tähele panna, et punktide numbrid kasvavad vasakult paremale;
   - kui $v + 1 \leqslant s \leqslant v + r$, siis siseneb kiir paremast servast ja liigub vasakule; veel tuleb tähele panna, et punktide numbrid kasvavad ülalt alla;
   - kui $v + r + 1 \leqslant s \leqslant v + r + v$, siis siseneb kiir alumisest servast ja liigub ülespoole; veel tuleb tähele panna, et punktide numbrid kasvavad paremalt vasakule;
   - kui $v + r + v + 1 \leqslant s \leqslant v + r + v + r$, siis siseneb kiir vasakust servast ja liigub paremale; veel tuleb tähele panna, et punktide numbrid kasvavad alt üles.

2. Kiire peegeldumiste modelleerimine

   Selle juures on vaja läbi mõelda kokku kaheksa varianti (kiir võib liikuda neljas suunas ja peegel võib olla kahes asendis). Osutub, et kokkuhoidliku programmeerimise korral taanduvad need kõigest kaheks eraldi käsitlemist nõudvaks juhuks.

3. Kiire väljumispunkti numbri leidmine vastava ruudu koordinaatidest

   Selle juures tulevad muidugi kasuks esimese punkti tähelepanekud.

Testid

1. $R = 5$, $V = 3$. Erijuhtum: peegleid polegi. Kiir siseneb ülalt ja väljub alt. 1 punkt.
2. $R = 5$, $V = 3$. Erijuhtum: peegleid polegi. Kiir siseneb alt ja väljub ülalt. 1 punkt.
3. $R = 5$, $V = 3$. Erijuhtum: peegleid polegi. Kiir siseneb paremalt ja väljub vasakult. 1 punkt.
4. $R = 5$, $V = 3$. Erijuhtum: peegleid polegi. Kiir siseneb vasakult ja väljub paremalt. 1 punkt.
5. $R = 1$, $V = 1$. Minimaalne test, ainult üks peegeldus. Kiir siseneb ülalt ja väljub vasakult. 1 punkt.
6. $R = 1$, $V = 1$. Minimaalne test, ainult üks peegeldus. Kiir siseneb paremalt ja väljub ülalt. 1 punkt.
7. $R = 1$, $V = 1$. Minimaalne test, ainult üks peegeldus. Kiir siseneb alt ja väljub paremalt. 1 punkt.
8. $R = 1$, $V = 1$. Minimaalne test, ainult üks peegeldus. Kiir siseneb vasakult ja väljub alt. 1 punkt.
9. $R = 3$, $V = 5$. Väike lihtne test, ainult üks peegeldus. Kiir siseneb ülalt ja väljub paremalt. 1 punkt.
10. $R = 3$, $V = 5$. Väike lihtne test, ainult üks peegeldus. Kiir siseneb paremalt ja väljub alt. 1 punkt.
11. $R = 3$, $V = 5$. Väike lihtne test, ainult üks peegeldus. Kiir siseneb alt ja väljub vasakult. 1 punkt.
12. $R = 3$, $V = 5$. Väike lihtne test, ainult üks peegeldus. Kiir siseneb vasakult ja väljub ülalt. 1 punkt.
13. $R = 10$, $V = 10$. Kiir läbib mõnda ruutu korduvalt. 4 punkti.
14. $R = 10$, $V = 10$. Kiir peegeldub mõnest peeglist kahelt poolt. 4 punkti.
15. $R = 8$, $V = 6$. Ääretingimuste täpsuse kontroll: peeglid külgede lähedal. 4 punkti.
16. $R = 6$, $V = 8$. Veel üks ääretingimuste täpsuse kontroll: peeglid nii külgede lähedal kui ka nurkades. 4 punkti.
17. $R = 20$, $V = 20$. Keskmise suurusega juhuslik test. 4 punkti.
18. $R = 30$, $V = 30$. Keskmise suurusega juhuslik test. 4 punkti.
19. $R = 50$, $V = 50$. Maksimaalse suurusega karp, maksimumilähedane peegelduste arv. 4 punkti.

Kokku 40 punkti.

3. Pank

Lahenduse idee:

1. Loeme sisse iga kassa poolt pakutavate teenuste andmed ning säilitame need selliselt, et oleks lihte leida, kas kassa k pakub teenust t.
2. Säilitame iga kassa kohta selle vabanemise kellaaega (algväärtus 0).
3. Töötleme järjest sisenevaid kliente:
   (a) Kontrollime iga kassa kohta, kas seal pakutakse soovitud teenust.
   (b) Kui jah, leiame esimese kellaaja, millal seal saaks klienti teenindada. (NB! klienti ei saa teenindada enne tema saabumist, isegi kui kassa oli enne vaba!)
   (c) Leiame võimalike teenindamise aegade miinimumi, mitme miinimumi korral vähima numbriga kassa.
   (d) Nihutame leitud kassa vabanemisaega edasi.

Veel tasub tähele panna, et kuigi klientide saabumise ajad ei ületa 10 000, võivad teenindamise ajad järjekordade tõttu väljuda 16-bitise muutuja väärtusvarust.

Testid

1. 3 kassat, igaühes üks teenus, järjekorda pole. 3 punkti.
2. 5 kassat, igaühes üks teneus, tekib järjekord. 3 punkti.
3. 9 kassat, 2 teenust, järjekorda ei teki, kõik kassad kasutuses. 4 punkti.
4. 20 kassat, 3 teenust, tekib järjekord. 4 punkti.
5. 5 kassat, 5 teenust, 1000 klienti, järjekordi pole. 4 punkti.
6. 4 kassat, 10 teenust, 5000 klienti, väike järjekord. 4 punkti.
7. 15 kassat, 6 teenust, 8000 klienti, järjekorrad. Suured arvud väljundis. 4 punkti.
8. 25 kassat, 25 teenust, pikad järjekorrad, 10 000 klienti. Suured arvud väljundis. 4 punkti.

Kokku 30 punkti.

4. Prefikspakkimine

Üldiselt pole selle ülesande lahendamiseks paremat varianti kui sõnastik lihtsalt ära sorteerida ja siis seda järjest läbi vaadates pakitud tulemus väljastada.

Kui sõnastiku sorteerimiseks kasutada ebaefektiivset, $O(N^2)$ keerukusega algoritmi, nagu on tehtud failis `preflah0.pas`, siis jääb see lahendus suuremates testides ajahätta.

Efektiivsema, $O(N \log N)$ keerukusega sorteerimisalgoritmi kasutamisel, nagu on tehtud failides `preflah1a.pas` (sorteerimine kuhjameetodil) ja `preflah1b.pas` (sorteerimine randomiseeritud kiirmeetodil), töötab pakkimine kõigis testides piisavalt kiiresti.

Keeltes, mille standardteegis on sorteerimisalgoritm olemas, on ka see piisavalt efektiivne. Failis `preflah2a.c` toodud lahendus kasutab sõnede võrdlemiseks C standardteegi funktsiooni `strcmp()` ja peab seetõttu sõnede ümberjärjestamiseks terveid sõnesid mälus ühest kohast teise kopeerima. Failis `preflah2b.c` toodud lahendus aga sorteerib sõnede endi asemel nende aadresse ja töötab tänu sellele veel veidi kiiremini. Failis `preflah2c.cpp` toodud lahendus kasutab C++ standardteeki kuuluvat sorteerimisalgoritmi, mis omakorda kasutab C++ standardteeki kuuluvat sõnede võrdlemise operaatorit.

Testid

1. $N = 20$. Prefiksite pikkused ühekohalised arvud. 3 punkti.
2. $N = 150$. Prefiksite pikkused 9–10–11. 3 punkti.
3. $N = 150$. Prefiksite pikkused $1 \ldots 29$. 3 punkti.
4. $N = 450$. Korduvad sõnad, prefiksite pikkused 9–10–11, 19–20–21, 30. 3 punkti.
5. $N = 1500$. Juhuslik sõnastik. 3 punkti.
6. $N = 2000$. Juhuslik sõnastik, järjestatud tagurpidi. 3 punkti.
7. $N = 4000$. Juhuslik sõnastik, korduvad sõnad. 3 punkti.
8. $N = 1000$. Juhuslik sõnastik, järjestatud õigetpidi. 3 punkti.
9. $N = 8000$. Juhuslik sõnastik. 3 punkti.
10. $N = 10\,000$. Juhuslik sõnastik. 3 punkti.

Kokku 30 puntki.

5. Palindroomarvud

Muidugi võib seda ülesannet lahendada proovimise teel: kontrollime, kas $N, N \pm 1, N \pm 2, \ldots$ on palindroomid ja väljastame kummaski suunas liikudes esimese leitud palindroomi. Sellel ideel baseerub näiteks failis `pallah0.pas` toodud lahendus, mis on põhimõtteliselt õige, aga jääb suuremates testides ajahätta, sest N ja talle lähima palindroomi vahe võib olla kuni $\sqrt{N}$.

Parema lahenduse leidmiseks tuleb märgata, et L-kohaline palindroom on täielikult määratud oma $\lceil L/2 \rceil$ esimese numbriga, sest ülejäänud numbrid peavad olema samad numbrid peegelpildis. Sellest järeldub, et kui arvu N esimesed $\lceil L/2 \rceil$ numbrit moodustavad arvu M, on mõtet väljastamiseks kaaluda vaid palindroome, mille esimese poole moodustavad $M-1$, M ja $M+1$. Lihtsaim lahendus ongi genereerida need kolm palindroomarvu, võrrelda neid siis algse arvuga ja väljastada neist tingimustele vastavad. Sellised lahendused on toodud failides `pallah1.c` ja `pallah1.pas`. Mõlemas põhjustab täiendavaid komplikatsioone asjaolu, et $M-1$ ja $M+1$ ei tarvitse olla sama pikad kui M.

Testid

1. $N = 0$. Minimaalne test. N on ise palindroom. 2 punkti.
2. $N = 111$. N on ise palindroom, pikkus paaritu arv. 2 punkti.
3. $N = 9999$. N on ise palindroom, pikkus paarisarv. 2 punkti.
4. $N = 132$. Keskmine number suureneb, N pikkus paaritu arv. 2 punkti.
5. $N = 1332$. "Keskmine" number suureneb, N pikkus paarisarv. 2 punkti.
6. $N = 251$. Keskmine number väheneb, N pikkus paaritu arv. 2 punkti.
7. $N = 2551$. "Keskmine" number väheneb, N pikkus paarisarv. 2 punkti.
8. $N = 10\,000$. N_1 lühem kui N, N pikkus paaritu arv. 2 punkti.
9. $N = 100\,000$. N_1 lühem kui N, N pikkus paarisarv. 2 punkti.
10. $N = 73\,461\,643\,734\,616\,436$. Suur test. Keskmine number väheneb, N pikkus paaritu arv. 4 punkti.
11. $N = 734\,616\,437\,734\,616\,438$. Suur test. "Keskmine" number suureneb, N pikkus paarisarv. 4 punkti.
12. $N = 9\,223\,372\,035\,302\,733\,229$. Peaaegu maksimaalne test. N on ise palindroom. 4 punkti.

Kokku 30 punkti.

6. Tsirkus

Mängulauda võib vaadelda suunatud graafina, kus ruudud on tippudeks ning servad lähevad igast tipust nendesse, kuhu mängija satub pärast täringuviset. Näiteks kui ruudult 9 läheb tunnel ruudule 56, ruudult 11 ruudule 24 ja ruutudelt 7, 8, 10, 12 ei välju ühtegi tunnelit, siis väljub ruutu 6 kujutavast tipust 6 serva ning need suunduvad tippudesse 7, 8, 56, 10, 24 ja 12. Eraldi struktuuri graafi jaoks ei ole vaja. Piisab sellest, et teada ruutude arvu ning igalt ruudult väljuva tunneli sihtpunkti. Seejuures võib teha lihtsustuse, lugedes, et ruutudelt, kust tunnelit ei välju, läheb tunnel iseendale. Nii saame, et igalt ruudult pääseb neile ruutudele, mis on järgnevalt 6 ruudult väljuvate tunnelite lõpp-punktideks. Eraldi tuleb jälgida, et me lõppruudust mööda ei läheks.

Ülesandeks on nüüd leida lühim tee mööda graafi servi ühest tipust teiseni. Selleks võib kasutada graafi laiuti läbivaatust, millega saame lühima tee ruudust 1 igale ruudule. Et me peame pärast ka tee algusest lõppu väljastama, jätame iga ruudu juures meelde eelmise ruudu numbri ning täringuviske, mis meid sellele ruudule tõi. Laiuti läbivaatuse juures on vaja realiseerida järjekord. Failis `tslah1.cpp` on kasutatud STL-i järjekorda, failis `tslah2.cpp` aga on lihtne järjekord ise realiseeritud, kasutades piisavalt pikka massiivi ning pidades meeles indekseid, kust järjekord algab ja kus ta lõpeb. Kuna me lisame järjekorda igat graafi tippu täpselt ühe korra, piisab valida järjekorra pikkuseks N.

Ei tööta "ahned" strateegiad, mis püüavad iga kord kasutada parimat ettejuhtuvat pikka tunnelit. On võimalik konstrueerida ka selliseid teste, kus tuleb kasutada tagasi viivaid tunneleid.

Testid

1. 1 ruut, 0 tunnelit, vastus 0. Minimaalne test. 4 punkti.
2. 120 ruutu, 24 tunnelit, vastus 5. Test põhineb reaalsel lauamängul. 4 punkti.
3. 39 ruutu, 9 tunnelit, vastus 3. Esimesel ja ka viimasel käigul tuleb ignoreerida edasi viivat tunnelit. 4 punkti.
4. 49 ruutu, 12 tunnelit, vastus 6. Lühim tee sisaldab tagasi viivat tunnelit. 4 punkti.
5. 300 ruutu, 30 tunnelit, vastus 21. Lühim tee sisaldab tagasi viivat tunnelit. Lõppruutu jõutakse tunneli kaudu. 4 punkti.
6. 467 ruutu, 38 tunnelit, vastus 42. Lühim tee sisaldab palju tagasi viivaid tunneleid, liikumine toimub vaheldumisi laua alumise ja ülemise poole vahel. 4 punkti.
7. 678 ruutu, 60 tunnelit, vastus 102. Tuleb vältida edasi viivaid tunneleid, mis kõik (peale viimase) viivad "lõksu", kust lõppu ei pääse. 4 punkti.
8. 850 ruutu, 295 tunnelit, vastus 43. Juhuslik test, kuhu on käsitisi lisatud mõned huvitavad elemendid. Algusest lõppu pääsemiseks peab igal juhul ühe korra tagurpidi liikuma (ja teepikuse optimeerimiseks mõned korrad veel). 4 punkti.
9. 999 ruutu, 300 tunnelit, vastus 121. Juhuslik test. 4 punkti.
10. 1000 ruutu, 500 tunnelit, vastus 167. Maksimaalne test. Peaaegu kõik tunnelid on tagurpidi ja neist pole kasu, nii et maksimaalne on ka vajalike käikude arv. 4 punkti.

Kokku 40 puntki.