1. Jada

Selle ülesande lahendus positiivsete testide korral on muidugi triviaalne. Tuleb lihtsalt alustada seisust $i = 1$ ja edasi loendada, mitu korda on vaja omistada $i \leftarrow a_i$, et jõuda uuesti seisu $i = 1$. Ainus potentsiaalne komistuskoht on $a_1 = 1$ õigesti loendamine.

Ülesande põnevam osa on seega negatiivsete juhtude lahendamine. Tegelikult nõuab ka see vaid pisut enam järelemõtlemist. Peaks olema üsna ilmne, et N-elemendilises jadas kindlas järjekorras elemente külastades peame hakkama ennast kordama hiljemalt N sammu järel. Seega, kui me pole N sammu jooksul jõudnud tagasi lähtekohta, siis ei jõua me sinna kunagi. Selline lahendus on failis `jadalah1.pas`.

Alternatiiv on iga elemendi juures meeles pidada, kas me juba oleme seda külastanud või mitte. Kui me jõuame elemendi juurde, mida me oleme juba varem külastanud, oleme tuvastanud tsükli. Kui teist korda külastatav element ei ole a_1, siis ei sisalda tsükkel elementi a_1 ja vastus on eitav. Selline lahendus on failis `jadalah2.pas`. Kuna selle lahenduse korral on vaja algväärtustada N-elemendiline abimassiiv, pole see tegelikult eelmisest efektiivsem.

Natuke raskem on olukord Turbo Pascali (ja muude antiiksete vahendite) kasutajail, kel pole võimalik luua 100 000-elemendilist massiivi. Tuleb tunnistada, et ülesande sisu poolest oleks võinud N ülempiiriks seada ka 10 000 või 15 000. Kõrgema piiri kasuks otsustas žürii puhtalt uuemate vahendite propageerimise eesmärgil. Ülesanne on siiski täielikult lahendatav ka Turbo Pascali abil. Üks võimalik lahendus on failis `jadalah3.pas`.

Testid

1. $N = 1$, $a_1 \to a_1$. Vastus JAH, 1 samm. Minimaalne positiivne test. 2 punkti.
2. $N = 10$, $a_1 \to a_3 \to a_5 \to a_7 \to a_9 \to a_1$. Vastus JAH, 5 sammu. 2 punkti.
3. $N = 100$, juhuslikud üleminekud. Vastus JAH, 50 sammu. 2 punkti.
4. $N = 1\,000$, juhuslikud üleminekud. Vastus JAH, 500 sammu. 2 punkti.
5. $N = 15\,000$, $a_1 \to a_2 \to \ldots \to a_{9\,999} \to a_{10\,000} \to a_1$. Vastus JAH, 10 000 sammu. Maksimaalne 64 KB mälupiiriga lahendusele jõukohane test, kui jada elemente hoida 4-baidiste täisarvudena. 2 punkti.
6. $N = 30\,000$, $a_1 \to a_{25\,000} \to a_{24\,999} \to \ldots \to a_2 \to a_1$. Vastus JAH, 25 000 sammu. Maksimaalne 64 KB mälupiiriga lahendusele jõukohane test, kui märgata, et sellise mälumahu korral võib jada elemente hoida 2-baidiste täisarvudena. 2 punkti.
7. $N = 100\,000$, $a_1 \to a_{50\,001} \to a_2 \to a_{50\,002} \to \ldots \to a_{50\,000} \to a_{100\,000} \to a_1$. Vastus JAH, 100 000 sammu. Maksimaalne positiivne test. 2 punkti.
8. $N = 2$, $a_1 \to a_2 \to a_2$. Vastus EI. Minimaalne negatiivne test. 2 punkti.
9. $N = 1\,000$, juhuslikud üleminekud. Vastus EI. 2 punkti.
10. $N = 100\,000$, juhuslikud üleminekud. Vastus EI. Maksimaalne negatiivne test. 2 punkti.

Kokku 20 punkti.

2. Tabel

Sellel ülesandel on väga palju erinevaid lahendusi.

Arvatavasti kõige lihtsam on järgmine: kuna otsitav element peab leiduma kõigis ridades, peab ta leiduma ka esimeses reas. Vaatleme kordamööda kõiki esimese rea elemente ja kontrollime, kas mõni neist leidub kõigis järgmistes ridades. Lineaarse otsingu korral kulub antud elemendi antud reas leidumise kontrollimiseks kuni V sammu. Iga elemendi korral tuleb kontrollida kuni R rida. Kokku on vaja kontrollida kuni V elementi. Seega kulutame kokku kuni RV^2 võrdlustehet. Selline lahendus on failis `tbllah1.pas`.

Muidugi võib seda lahendust mitut moodi optimeerida. Igas reas otsimisel võiks kokku hoida keskmiselt pooled võrdlused, kui lõpetada otsimine kohe, kui rea elemendid kasvavad otsitavast elemendist suuremaks. Ridade läbivaatuse võib katkestada kohe, kui oleme leidnud rea, milles otsitav element ei esine. Kui esimeses reas on korduvaid elemente, võib järgmisi ridu läbi otsida ainult ühe korra iga erineva väärtuse jaoks. Ja nii edasi...

Esimene põhimõttelisem võimalus kokkuhoiuks lähtub eeldusest, et iga rea elemendid on järjestatud. Seega saame lineaarse otsingu asendada kahendotsinguga ja kulutada ühe elemendi ühes reas otsimisele V operatsiooni asemel $\log V$ operatsiooni. Muud hinnangud on samad nagu eelmise algoritmi korral ja kokku kulutame seega kuni $RV \log V$ võrdlustehet. Selline lahendus on failis `tbllah2.pas`.

Aga saab veel kokkuhoidlikumalt talitada. Nimelt me teame ka seda, et otsitavad (esimese rea) elemendid on mittekahanevas järjekorras. Siis võime iga rea kohta seada sisse loenduri, mis näitab, kui kaugele me jõudsime selles reas otsimisega eelmisel katsel. On selge, et sellest kohast eestpoolt pole mõtet otsida. Nii uurime tabeli igat elementi ainult ühe korra ja kulutame kokku vaid RV võrdlustehet. Selline lahendus on failis `tbllah3.pas`.

Kõik seni vaadeldud lahendused eeldasid siiski kogu tabeli (maksimaalselt 250 000 elementi) korraga mälus hoidmist, sest iga kandidaadi (esimese rea iga elemendi) uurimiseks oli vaja kõigi teiste ridade elemente. Osutub, et ka see on liialdamine, eriti Turbo Pascali (ja muude antiiksete vahendite) kasutajaile, kellele nii suurt hulka mälu vähese vaevaga ei võimaldata.

Tegelikult on meie tabeli näol tegemist R hulgaga (igaühes maksimaalselt V elementi) ja me tahame leida nende kõigi ühisosa. See aga tähendab, et me võime hulki töödelda järjest: kõigepealt leiame kahe esimese ühisosa, siis saadud tulemuse ja järgmise hulga ühisosa jne. Nii on meil korraga vaja mälus hoida vaid paari rea jagu andmeid (hoolika bitinikerdamisega saaks läbi ka vaid ühe reaga, aga näidislahendus seda ei tee, loetavuse huvides). Kuna elemendid kõigis ridades on järjestatud, saame kahe rea ühisosa leida ülimalt V võrdlustehtega ja kulutada kokku maksimaalselt RV sammu. Selline lahendus on failis `tbllah4.pas`.

Veel on võimalik iga tabelis esineva elemendi jaoks lihtsalt loendada, mitmes reas ta esineb. Selleks on otstarbekas võtta kasutusele eraldi loendur iga võimaliku väärtuse (neid on ülesande tingimuste kohaselt 60 001) jaoks. Tuleb vaid olla hoolikas, et loendada ühel real mitu korda esinevat väärtust ikka ainult ühekordselt. Selline lahendus on failis `tbllah5.pas`.

Turbo Pascali sõbrad leiavad 250 000 või 60 001 elemendi hoidmiseks vajaliku mäluhulga hõivamise ja kasutamise näite jadaülesande lahenduste hulgast.

Testid

1. $R = 1$, $V = 1$. Vastus JAH, 1. Minimaalne positiivne test. 2 punkti.
2. $R = 4$, $V = 4$. Vastus JAH, 4. Üheski reas pole korduvaid väärtusi. 2 punkti.
3. $R = 5$, $V = 5$. Vastus JAH, 6. Igas reas korduvad väärtused. 2 punkti.
4. $R = 6$, $V = 7$. Vastus JAH, {1, 2, 3, 4, 5}. Vastus pole ühene. 2 punkti.
5. $R = 50$, $V = 100$. Vastus JAH, -3. Keskmine juhuslik test, väga suured ja väga väikesed arvud. 2 punkti.
6. $R = 400$, $V = 300$. Vastus JAH, {-24, 42}. Suur juhuslik test, väga suured ja väga väikesed arvud. 2 punkti.
7. $R = 500$, $V = 500$. Vastus JAH, 400. Maksimaalne positiivne test. 2 punkti.
8. $R = 2$, $V = 1$. Vastus EI. Minimaalne negatiivne test. 2 punkti.
9. $R = 200$, $V = 50$. Vastus EI. Keskmise suurusega negatiivne test. 2 punkti.
10. $R = 500$, $V = 500$. Vastus EI. Maksimaalne negatiivne test. 2 punkti.

Kokku 20 punkti.

3. Sugulased

Pole ilmselt raske näha, et kirjeldatud sugulusseos jagab kõik külalised "suguvõsadesse" nii, et igas suguvõsas on kõik inimesed omavahel sugulased ja ka vastupidi, kui kaks inimest on sugulased, siis on nad kindlasti samas suguvõsas. Selliseid seoseid nimetatakse matemaatikas ekvivalentsiseosteks ja tekkivaid suguvõsasid ekvivalentsiklassideks.

Standardlahendus ekvivalentsiklasside hoidmiseks on kujutada iga klassi puuna, milles iga tipp osutab oma ülemusele. Selline esitus võimaldab vaadelda iga puu juurtippu vastava klassi esindajana. Selleks, et kontrollida, kas kaks elementi on samas klassis, tuleb leida kummalegi elemendile vastava klassi esindaja (liikudes viitasid mööda edasi, kuni jõuame juurtipuni) ja siis vaadata, kas esindajad langevad kokku.

Kahe klassi ühendamiseks tuleb neist ühe esindaja määrata teise esindaja alluvaks ja edaspidi leiavad mõlema klassi elemendid esindaja otsimisel automaatselt uue ühise juure. Lõpuks väljastame otsitava grupi liikmetena parajasti iga klassi esindaja, see tähendab kõigi puude juurtipud.

Selles algoritmis tuleb $2K$ korda leida klassi liikme järgi selle klassi esindaja. Halvimal juhul võib esindaja leidmiseks kuluda N sammu, siis on selle algoritmi tööaeg suurusjärku NK. Halvimat juhtu on võimalik vähese vaevaga vältida, aga failides `suglah1.pas` ja `suglah1.java` toodud lahendused seda lihtsuse huvides ei tee.

Teine võimalus on vaadelda sisendandmeid graafina, mille tippudeks on külalised ja servadeks sugulusseosed. Sellisel juhul on iga suguvõsa graafi sidususkomponent.

Graafi sidususkomponentide leidmiseks võib kasutada graafi läbimist laiuti:

```
alustame seisust, kus kõik graafi tipud on märkimata;
seni kuni veel leidub märkimata tippe:
    valime neist ühe;
    märgime selle ja paneme järjekorda;
    seni kuni järjekord pole tühi:
        võtame järjekorra esimese elemendi;
        vaatleme selle kõiki veel märkimata naabreid:
            märgime naabri ja paneme järjekorra lõppu;
```

Kordus "seni kuni järjekord pole tühi" märgib kõik valitud esindajaga samas sidususkomponendis olevad tipud, seega kõik pärast seda märkimata tipud on mingites teistes komponentides. Kordus "seni kuni veel leidub märkimata tippe" märgib ükshaaval kõik komponendid. Pole raske näha, et see algoritm vaatleb graafi iga serva kaks korda — ühe korra kummagi otstipu poolt — ja kulutab seega suurusjärku K operatsiooni. Selline lahendus on failis `suglah2.pas`.

Graafi sidususkomponente võib leida ka sügavuti läbimise abil. Algoritm erineb laiuti läbimisest ainult selle poolest, et järjekorra asemel kasutame magasini. Kuna endiselt vaatleme graafi iga serva täpselt kaks korda, on tööaja hinnang sama, mis eelmisel juhul. Selline lahendus on failis `suglah3.pas`.

Sügavuti otsimise võime realiseerida ka rekursiooni abil, kasutades ilmutatud andmemagasini asemel programmeerimissüsteemi kutsemagasinis hoitavaid lokaalseid muutujaid. Programi tööaja hinnangut see muidugi ei muuda, aga mõne koodirea hoiame kokku. Selline lahendus on failis `suglah4.pas`.

Testid

1. $N = 1$, $K = 0$. Minimaalne test. Vastus 1, (1). 1 punkt.
2. $N = 5$, $K = 0$. Väike tühigraaf. Vastus 5, (1, 2, 3, 4, 5). 1 punkt.
3. $N = 5$, $K = 1$. Väike ühe servaga graaf. Vastus 4, ({1, 5}, 2, 3, 4). 1 punkt.
4. $N = 7$, $K = 6$. Seosed 1-2-3-4-5-6-7, seega on kõik omavahel sugulased. Vastus 1, ({1, 2, 3, 4, 5, 6, 7}). 2 punkti.
5. $N = 6$, $K = 6$. Graaf koosneb kahest 3-tipulisest klikist. Vastus 2, ({1, 2, 3}, {4, 5, 6}). 2 punkti.
6. $N = 6$, $K = 7$. Graaf koosneb kahest 3-tipulisest klikist, lisaks on kummastki üks tipp teisega seotud, seega on kõik omavahel sugulased. Vastus 1, ({1, 2, 3, 4, 5, 6}). 2 punkti.
7. $N = 7$, $K = 5$. Korduvalt antud servad, üksikud tipud. Vastus 4, (1, {2, 3}, {4, 5, 6}, 7). 3 punkti.
8. $N = 40$, $K = 40$. Keskmise suurusega hõre juhuslik graaf. Vastus 7. 3 punkti.
9. $N = 40$, $K = 900$. Keskmise suurusega tihe juhuslik graaf. Vastus 1. 3 punkti.
10. $N = 80$, $K = 70$. Üsna suur hõre juhuslik graaf. Vastus 17. 3 punkti.
11. $N = 80$, $K = 2\,200$. Üsna suur tihe juhuslik graaf. Vastus 3. 3 punkti.
12. $N = 90$, $K = 1\,200$. Üsna suur juhuslik graaf. Vastus 3. 3 punkti.
13. $N = 100$, $K = 4$. Maksimaalse suurusega väga hõre graaf, korduvalt antud seos 1-1. Vastus 100. 4 punkti.
14. $N = 100$, $K = 4\,950$. Maksimaalse suurusega väga tihe graaf. Vastus 1. 4 punkti.
15. $N = 100$, $K = 98$. Seoste struktuur meenutab ühildusmeetodit: kõigepealt on elemendid seotud paarikaupa (1-2, 3-4, ..., 99-100), siis paaride esindajad paarikaupa (2-4, 6-8, ..., 98-100), siis nelikute esindajad paarikaupa jne. Ainus puuduv seos on 1-2, mistõttu lõpuks on kõik ülejäänud omavahel sugulased ja 1 eraldi. Üsna ebamugav test lahendusele, mis ilmutatult ekvivalentsiklasse hoiab. Vastus 2. 5 punkti.

Kokku 40 punkti.

4. Tutvumine

Nagu iga objektide ja nendevaheliste seostega tegelev ülesanne, on ka see tõlgendatav graafiülesandena. Tegemist on klassikalise salakavala ülesandega, mida tuntakse üldiselt graafi värvimise ülesande nime all.

Esmapilgul võib tunduda, et selle ülesande lahendamiseks sobib "ahne algoritm": vaatleme külalisi mingis suvalises (näiteks nende numbrite) järjekorras ja määrame igaühe neist esimesse (vähima numbriga) gruppi, kus pole veel ees ühtki tema tuttavat. Ülesande salakavalus väljendub selles, et ilma spetsiaalselt pingutamata pole väga kerge koostada näidet, mida see algoritm valesti lahendaks. Siiski on selliseid näiteid piisavalt, mida tõestab näiteks asjaolu, et ahne algoritmi alusel koostatud programm failist `tutlah0.pas` teenib umbes pooled punktid.

Osutub, et graafi värvimise ülesanne kuulub niinimetatud "raskete" ülesannete hulka, mille täpseks lahendamiseks pole teada variantide läbivaatamisest paremat algoritmi. On olemas hulk efektiivseid algoritme, mis annavad üsna hea lahenduse, aga seni pole teada ühtegi, mis oleks efektiivne ja annaks alati parima võimaliku lahenduse. Arvestades "raskete" ülesannete uurimisel seni tehtud jõupingutusi, pole kuigi tõenäoline, et selline algoritm üldse olemas on. Siiski pole keegi ka tõestanud, et sellist algoritmi olemas olla ei saa. Tegemist on ühe tähtsama lahendamata küsimusega algoritmiteoorias.

Variantide läbivaatamisega algoritmides on oluline määrata, milliseid variante me läbi vaatame. Sageli pole see sugugi lihtne. Näiteks käesolevas ülesandes oleks variantide läbivaatus üsna lihtne, kui me teaks ette, mitmeks grupiks me külalised jagama peame, ja siis otsiks ainult sobiva gruppide arvuga jaotust.

Üks võimalus gruppide arvu määramiseks on lihtsalt hakata proovima variante väiksematest alates: proovime algul paigutada kõik külalised ühte gruppi; kui see ei õnnestu, siis jagada nad kahe grupi vahel; kui ka see ei õnnestu, siis kolme grupi vahel jne. Niipea kui me oleme leidnud ülesande tingimusi rahuldava jaotuse mingi gruppide arvu jaoks, võime otsingu lõpetada, suurema gruppide arvuga jaotusi pole enam mõtet uurida. Selline lahendus on failis `tutlah1.pas`.

Selle algoritmi põhiidee sarnaneb iteratiivse süvendamisega, kuigi antud juhul oleks täpsem nimetada seda iteratiivseks laiendamiseks, sest me ei suurenda sammukaupa mitte läbivaatuse sügavust (näiteks maleseisu analüüsimisel tähendab sügavus seda, mitu käiku ette me vaatame), vaid hargnevust (näiteks maleseisu analüüsimisel tähendab hargnevus seda, mitut võimalikku käiku me igal seisul vaatleme).

Veel üks võimalus on proovida variante juhuslikult: määrame iga külalise gruppi, mille valime juhuslikult nende hulgast, kuhu ta veel tohib minna. Kui mõnele külalisele kohta ei leidu, saadame kõik grupid laiali ja alustame otsast. Kui leiame mingi gruppide arvuga jaotuse, proovime edasi väiksema gruppide arvuga. Kordame seda seni, kuni aeg otsa saab ja väljastame siis parima leitud tulemuse. Muidugi ei anna selline lähenemine garanteeritult õiget vastust, aga praktikas on seda tüüpi algoritmid sageli väga tõhusad "piisavalt heade" vastuste leidmisel. Selline lahendus on failis `tutlah2.pas`.

Testid

1. $N = 1$, $K = 0$. Minimaalne test. Üks tipp, servi pole. 1 grupp. 2 punkti.
2. $N = 3$, $K = 2$. Tutvusseosed moodustavad ahela. 2 gruppi. 2 punkti.
3. $N = 4$, $K = 4$. Tutvusseosed moodustavad paaristsükli. 2 gruppi. 2 punkti.
4. $N = 5$, $K = 5$. Tutvusseosed moodustavad paaritu tsükli. 3 gruppi. 2 punkti.
5. $N = 4$, $K = 6$. Tutvusseosed moodustavad diagonaalidega ruudu. 4 gruppi. 2 punkti.
6. $N = 6$, $K = 5$. Tutvusseosed moodustavad 5-harulise tähe. 2 gruppi. 2 punkti.
7. $N = 10$, $K = 11$. Paaristsükli diagonaal tekitab kaks paaritut tsüklit. 3 gruppi. 2 punkti.
8. $N = 8$, $K = 13$. Üsna tihe paarisgraaf. 2 gruppi. 2 punkti.
9. $N = 50$, $K = 50$. Pikk paaristsükkel. 2 gruppi. 2 punkti.
10. $N = 40$, $K = 59$. Pikk paaritu tsükkel pluss üks tipp, mis on seotud paljude tsüklis olevate tippudega. 3 gruppi. 2 punkti.
11. $N = 8$, $K = 26$. 8-tipuline klikk miinus kaks mittelõikuvat serva. 6 gruppi. 4 punkti.
12. $N = 9$, $K = 31$. 9-tipuline klikk miinus 5-servaline ahel. 6 gruppi. 4 punkti.
13. $N = 18$, $K = 40$. Mittesidus graaf. Üks komponent on eelmise testi sisend, teine 9-tipuline tsükkel. 6 gruppi. 4 punkti.
14. $N = 20$, $K = 50$. Viis 4-tipulist klikki, mida seovad neli 5-tipulist tsüklit nii, et igas tsüklis on üks tipp igast klikist. 4 gruppi. 4 punkti.
15. $N = 100$, $K = 100$. Maksimaalne test. 25 tippu on omavahel seotud: kümme 5-tipulist klikki, mis lõikuvad nii, et iga tipp kuulub täpselt 2 klikki. Ülejäänud 75 tippu on kõik üksikud. 5 gruppi. 4 punkti.

Kokku 40 punkti.

5. Fail `crontab`

See on kahtlemata kogu komplekti kõige loomingulisem, mittestandardsem ülesanne.

Üldisi tähelepanekuid:

- Sisendandmete vaatlus näitab: seitse testi kümnest mahuvad ühe aasta sisse; kuues testis kümnest on minutite väli alati 0. Järelikult ei pea programmid olema eriti efektiivsed.
- Lubatud on mitu programmi, kasvõi igale testile oma.

Lähenemisvõimalusi:

- Teisendaja logifail → `crontab`.

  Pluss: annab otseselt lahenduse.

  Miinus: keeruline realiseerida.

  Idee: kirjutame algul triviaalse lahenduse ja püüame hiljem kas programmi ennast või selle väljundit konkreetsest testist lähtuvalt "järele aidata".

- Teisendaja `crontab` → logifail (pöördülesande lahendus).

  Pluss: võimaldab väljundfailide õigsust kontrollida.

  Miinus: ei lahenda väljundfailide saamise küsimust.

Testid

1. Lihtne test, väike sisend. Žürii lahendus 5 rida. 4 punkti.
2. Lihtne test, väike sisend, käskudel argumendid. Žürii lahendus 5 rida. 4 punkti.
3. Väike sisend, aasta 2000 (liigaasta). Žürii lahendus 2 rida. 4 punkti.
4. Suur sisend, töö nädalapäevadega. Žürii lahendus 3 rida. 4 punkti.
5. Üsna väike sisend, töö hulkadega. Žürii lahendus 4 rida. 4 punkti.
6. Keskmiselt suur sisend, 30.10.2004 vahelejätmine. Žürii lahendus 1 rida. 4 punkti.
7. Keskmiselt suur sisend, töö samaaegsete käskude järjekorraga, ühe käsu kohta `crontab`is 32 rida. Žürii lahendus 63 rida. 4 punkti.
8. Suur sisend, aasta 2040 (mis võib valmistada raskusi mõnede süsteemide kuupäeva- ja ajafunktsioonidele). Žürii lahendus 6 rida. 4 punkti.
9. Üsna suur sisend, aasta 1960 (mis võib valmistada raskusi mõnede süsteemide kuupäeva- ja ajafunktsioonidele), töö nädalapäevadega. Žürii lahendus 4 rida. 4 punkti.
10. Üsna suur sisend, suhteliselt lihtne ja realistlik `crontab`. Žürii lahendus 6 rida. 4 punkti.

Kokku 40 punkti.

Kõik testid kasutavad standardaega — suve- ja talveajale pole vaja tähelepanu pöörata.