1. Lukud

Vaatleme koopa kaarti graafina, milles on tipp iga läbipääsu tähistava kaardiruudu kohta ja servadega on ühendatud naaberruutudele vastavad tipud. Ülesande lahendamiseks on kasulik teha selle graafi kohta järgmised tähelepanekud:

- kui graafi igast tipust on mingi tee väljapääsuni, siis on tegemist sidusa graafiga;
- kui graaf on sidus, siis on mingi tee ka aardeotsija lähtekohast mistahes tipuni;
- kui graafi igast tipust on väljapääsuni ülimalt üks lihttee, siis on tegemist tsükliteta graafiga;
- kui graaf on tsükliteta, siis on ka aardeotsija lähtekohast mistahes tipuni ülimalt üks lihttee;
- sidus tsükliteta graaf on definitsiooni järgi puu; vaatleme seda kui juurega puud, mille juureks on aardeotsija lähtekoht S; vaatleme alamgraafe, mis tekivad tipu S ja kõigi sellega seotud servade eemaldamisel; osutub, et:
  - need alamgraafid on omakorda puud;
  - kahe samasse alampuusse kuuluva tipu vaheline lihttee ei läbi juurtippu;
  - kahe erinevatesse alampuudesse kuuluva tipu vaheline tee läbib juurtippu;

(Kõigi nende väidete kehtivuse põhjendamine jääb lugejale koduseks ülesandeks.)

Järgmiseks osutub, et lühima tee leidmiseks on kasulik koguda kokku kõigepealt kõik võtmed ühest alampuust ja siis kõik võtmed teisest alampuust, mitte alampuude vahel korduvalt edasi-tagasi liikuda. (Miks? Ka seda väidet on vaja põhjendada!)

Siis aga ei sõltu ühes alampuus asuvate võtmete kokkukogumine (tipust S alustades ja tippu S tagasi jõudes) sellest, milliseid teisi alampuid ja millises järjekorras me enne või pärast selle alampuu töötlemist läbime. Sellest omakorda järeldub, et me võime lühima teega võtmete kogumise ülesannet lahendada iga alampuu jaoks eraldi ja pärast need teed suvalises järjekorras kokku kleepida. Välja arvatud alampuu, milles asub ka väljapääs — selle töötlemine on kasulik jätta viimaseks, sest pärast kõigi võtmetega väljapääsu juurde jõudmist pole enam vaja lähtekohta tagasi pöörduda.

Seega on meil puu igas tipus sellest algavate alampuude optimaalseks töötlemiseks vaja teada, millistes alampuudes on võtmeid ja millises (kui üldse) on väljapääs. Selle informatsiooni saame kokku koguda puu ühekordse lõppjärjestuses läbimisega ja pärast seda võime lahenduse väljastada puu eesjärjestuses läbimisega.

Ettevaatliku programmeerimise korral pole tegelikult vaja koopa kaardist puud konstrueerida. Failis `lukudlah1.pas` toodud lahendus töötab otse kaardil, kasutades vaid paari abimassiivi, et hoida iga kaardiruudu jaoks temast algava alampuu võtmete ja väljapääsude arvu.

Testid

1. $N = 4$, $M = 4$. Võtmeid: 0. Samme: 5. Vastus ühene. 10 punkti.
2. $N = 5$, $M = 5$. Võtmeid: 1. Samme: 9. Vastus ühene. 10 punkti.
3. $N = 8$, $M = 8$. Võtmeid: 8. Samme: 49. Vastus ühene. 10 punkti.
4. $N = 10$, $M = 10$. Võtmeid: 10. Samme: 58. Vastus pole ühene. Lähtekoht väljapääsu ja võtmete vahel. 10 punkti.
5. $N = 12$, $M = 12$. Võtmeid: 12. Samme: 73. Vastus pole ühene. Väljapääs lähtekoha ja võtmete vahel. 10 punkti.
6. $N = 20$, $M = 20$. Võtmeid: 50. Samme: 300. Vastus pole ühene. Variantide läbivaatus jääb tõenäoliselt ajahätta. 10 punkti.
7. $N = 40$, $M = 40$. Võtmeid: 200. Samme: 1196. Vastus pole ühene. Variantide läbivaatus jääb kindlasti ajahätta. 10 punkti.
8. $N = 60$, $M = 60$. Võtmeid: 500. Samme: 2674. Vastus pole ühene. 10 punkti.
9. $N = 80$, $M = 80$. Võtmeid: 800. Samme: 4750. Vastus pole ühene. 10 punkti.
10. $N = 100$, $M = 100$. Võtmeid: 1000. Samme: 6973. Vastus pole ühene. 10 punkti.

Kokku 100 punkti.

2. Mündipoks

Selle mängu strateegia uurimisel võiks kõigepealt panna tähele, et kui meil õnnestub mõne oma käiguga saavutada seis, kus meie münt on igal ribal vahetult vastase oma kõrval, on vastane selle partii sisuliselt juba kaotanud — tal pole muud võimalust kui mõne oma mündiga taganeda, aga me võime enda omaga kohe järgneda ja sundida ta järgmisel käigul uuesti taganema. Lõpuks pole tal enam kuhugi minna...

Sellest tähelepanekust järeldub, et tegelikult on seisu analüüsimisel oluline ainult meie ja vastase müntide vahele jäävate vabade positsioonide arv igal ribal. 2 ribaga mängus järeldub sellest omakorda, et võitmiseks on meil vaja hoida need vahed mõlemal ribal võrdsed: kui vastane astub meile ühel ribal lähemale, astume meie teisel ribal samapalju lähemale ja viime seisu uuesti tasakaalu, aga juba lähemal vahetult vastamisi asetusele. Kui vastane taganeb, siis järgneme talle samal ribal samapalju. See ei vii meid otsesele vastasseisule lähemale, aga samas ei saa vastane ka lõpmatult taganeda. Sellest järeldub, et kui enne meie käiku on müntide vahed erinevad, siis on see meile võiduseis, kui aga võrdsed, on meil lootus võita ainult vastase vea korral.

Eelmises lõigus toodud arutluskäiku võib üldistada ka rohkemate ribadega mängule. 3 ribaga mängus on võiduseisud need, kus enne meie käiku on kahel ribal vahed võrdsed ja kolmandal ribal ükskõik milline positiivne vahe — siis võime oma käiguga selle kolmanda vahe nulliks muuta ja edasi mängida ülejäänud kahel ribal eelmises lõigus tuletatud strateegia järgi. 4 ribaga mängus on võiduseisud need, kus võime ribad jagada kahte paari nii, et ühes paaris on vahed võrdsed ja teises erinevad — siis võime 2 riba strateegiat rakendada kummalegi paarile eraldi. Selline "näppude peal" tuletatud strateegia jätab lahtiseks, mida teha, kui 3 või 4 ribaga mängus on kõigil ribadel vahed erinevad. Parema idee puudumisel teeb failis `pokslah1.cpp` toodud lahendus selles olukorras juhusliku käigu. See pole muidugi optimaalne, aga garanteerib siiski peaaegu alati võidu failis `pokslah0.cpp` toodud juhusliku mängija vastu.

Osutub, et 2 ribaga mängu strateegia saab siiski üldistada suvalise ribade arvu jaoks. Selleks on kasulik vaadelda müntide vahele jäävate vabade pesade arvu igal ribal kahendsüsteemis. Sellisel juhul kehtib väide: kaotusseisud on parajasti need, mille puhul igas järgus on numbri 1 esinemiste arv paarisarv. Vaatleme näiteks 3 ribaga mängu seisu, kus esimesel ribal on müntide vahel $65 = 1000001_2$, teisel ribal $66 = 1000010_2$ ja kolmandal $67 = 1000011_2$ vaba kohta. Kui me kirjutame need arvud kohakuti (nagu kirjalikus liitmises), siis näeme, et nii üheliste kui kaheliste veerus on kummaski 2, aga 64-liste veerus 3 ühte, järelikult on tegu käigul oleva mängija jaoks võiduseisuga ja selle edu hoidmiseks tuleks ükskõik millisel ribal astuda vastasele $1000000_2 = 64$ sammu lähemale. Failis `pokslah2.cpp` toodud lahendus kasutabki seda teooriat ning võib optimaalselt ja efektiivselt mängida praktiliselt igasuguse suurusega laual — rohkem kui 4 riba jaoks tuleb ainult suurendada massiive `b` ja `f`.

Eelmises lõigus toodud väite kehtivus järeldub teisest väitest: mistahes seisus, kus vabade kohtade kahendkujudest moodustatud tabeli igas veerus on numbrit 1 paarisarv, on mistahes käigule võimalik vastata käiguga, mis taastab paarsuse ja ei suurenda vabade kohtade koguarvu võrreldes esialgse seisuga. Selle väite põhjendamine jääb koduseks järelemõtlemiseks, aga hoiatan kohe, et see on veidi keerulisem kui esmapilgul tundub (vaatleme näiteks olukorda, kus alguses on ühel ribal 67, teisel 64 ja kolmandal 3 vaba pesa ning vastane astub esimesel ribal 65 sammu meie suunas).

Lõpetuseks olgu veel öeldud, et selle mängu kuju, kus käia toihib ainult edasi (mis, nagu eelpool näha, mängu sisuliselt ei lihtsusta), on kombinatoorikas laialt tuntud nimede Nim ja Tactix all. Nim'i võidustrateegia leidis ja avaldas Charles L. Bouton juba aastal 1902. Umbes 30 aastat hiljem üldistasid Roland Percival Sprague ja Patric Michael Grundy selle kõigile mängudele, kus mõlemal mängijal on igas seisus võrdsed käiguvõimalused (see tulemus on tänapäeval tuntud Sprague-Grundy teooria nime all).

Testid

Lahendusi testiti turniiril, kus iga programm mängis iga teisega kokku kuus partiid: kolmest erinevast algseisust, igast seisust korra esimesena ja korra teisena käijana. Seisud olid:

- 2 ribaga laud, vastavalt 42 ja 43 pesaga; see on alustajale võiduseis;
- 3 ribaga laud, vastavalt 67, 68 ja 69 pesaga; see on alustajale võiduseis;
- 4 ribaga laud, vastavalt 92, 93, 94 ja 95 pesaga; see on alustajale kaotusseis.

3. Lukud

Selle ülesande lahendamiseks on kasulik kõigepealt panna tähele, et ruudust (i, j) juustuni viivate teede koguarvu võime avaldada kujul $a_{i,j} = a_{i,j+1} + a_{i+1,j}$, kus $a_{i,j+1}$ on `E`- ja $a_{i+1,j}$ `S`-sammuga algavate teede arv. Selle tähelepaneku alusel võime välja arvutada igast ruudust juustutükini viivate teede arvud. Kõige efektiivsem on seda teha dünaamilise planeerimise meetodiga, alustades paremast alumisest nurgast ja liikudes edasi üles ja vasakule. Mitteläbitavate ruutude teede arvuks omistame muidugi 0.

Kui meil on kõigi teede arvude tabel käes, pole ka antud järjekorranumbriga tee konstrueerimine enam keeruline: alustame ruudust $(1, 1)$; sealt on võimalik liikuda paremale $a_{1,2}$ ja alla $a_{2,1}$ viisil. Kuna kõik `E`-sammuga algavad teed on tähestiku järjekorras eespool `S`-sammuga algavatest, tähendab see, et $T \leqslant a_{1,2}$ korral peab otsitav tee algama `E`-sammuga ja edasi otsime samal viisil alates ruudust $(1, 2)$. Vastupidisel juhul peab otsitav tee algama `S`-sammuga, aga ruudust $(2, 1)$ algavate hulgast peame otsima teed järjekorranumbriga $T - a_{1,2}$. Samamoodi arutledes võime leida ka järgmised sammud.

Jääb veel üks komplikatsioon. Tekst lubab, et otsitava tee järjekorranumber T ei ületa 1 000 000 000, aga ei ütle midagi teede koguarvu kohta. Osutub, et halvimal juhul (tühja 30×30 labürindi korral) võib see olla $\mathrm{C}(58, 29) = 30\,067\,266\,499\,541\,040 \approx 2^{54,7}$, mis muidugi ei mahu 32-bitise täisarvu sisse ära.

Kõige lihtsam on sellest probleemist üle saada 64-bitiste täisarvude kasutamisega, nagu on tehtud failis `rottlah1.pas` toodud lahenduses. Teine võimalus on panna tähele, et meie ülesande lahendamise seisukohalt on kõik T-st suuremad arvud võrdsed, ja kasutada nende tegelike väärtuste asemel näiteks $T + 1$ — siis ei ületa ühegi summa $a_{i,j+1} + a_{i+1,j}$ väärtus tema arvutamise hetkel $2 \cdot T + 2$, mis mahub ka maksimaalse lubatud T korral 32-bitise täisarvu määramispiirkonda. Selline lahendus on toodud failis `rottlah2.pas`.

Kolmas võimalus on panna tähele, et kui mingist ruudust paremale viivate teede arv on juba piisavalt suur, siis pole meil sellest ruudust alla viivate teede arvu üldse vaja, ja vältida ületäitumise võimalust sellega, et me liiga suuri arve lihtsalt välja ei arvuta. Selline lahendus on toodud failis `rottlah3.pas`.

Testid

1. $N = 1$, $M = 10$, $T = 1$. Tühi labürint, üksainus võimalik tee. 10 punkti.
2. $N = 7$, $M = 11$, $T = 18$. Väike labürint, ka läbivaatus saab hakkama. 10 punkti.
3. $N = 23$, $M = 7$, $T = 345\,940$. Läbivaatus saab tõenäoliselt hakkama. Otsitav tee on loetelus viimane. 10 punkti.
4. $N = 16$, $M = 16$, $T = 10\,000\,000$. Läbivaatus jääb tõenäoliselt ajahätta. 10 punkti.
5. $N = 20$, $M = 20$, $T = 20\,000\,000$. 10 punkti.
6. $N = 23$, $M = 23$, $T = 883\,860\,144$. Läbivaatus jääb kindlasti ajahätta. Otsitav tee on loetelus viimane. 10 punkti.
7. $N = 26$, $M = 26$, $T = 1\,000\,000\,000$. 10 punkti.
8. $N = 15$, $M = 30$, $T = 543\,212\,345$. 10 punkti.
9. $N = 15$, $M = 30$, $T = 544\,689\,437$. 10 punkti.
10. $N = 30$, $M = 30$, $T = 1\,000\,000\,000$. Teede koguarv ei mahu 32-bitisesse täisarvu. 10 punkti.

Kokku 100 punkti.