Lošimas disku

Iš pažiūros šis uždavinys nesiskiria nuo kitų lošimo uždavinių, kurie sprendžiami rekursiškai bandant visus ėjimus ir einant „žemyn“ nuo duotosios situacijos link žaidimo pabaigos situacijos. Tačiau sąlygoje prašoma ne tik rasti ėjimą, bet ir paaiškinti žaidimo strategiją. O rekursijos žaidimo strategija nepavadinsi. Be to, vykdymo laikas skirtas ne vienam testui, o 12000 testų. Taigi, šis būdas atpuola.

Norint aprašyti žaidimo strategiją, teks panagrinėti laimimas ir pralaimimas situacijas. Žinome, sukauptą sumą s ir sumą, kurią reikia pasiekti m. Mus domina tik jų skirtumas $liko = s - m$, t. y. suma, kurią dar reikia sukaupti.

Susidarysime laiminčių situacijų lentelę. *liko* reikšmę didinsime nuo 1 iki ims aiškėti žaidimo strategija. Lentelę sudarys dvi skiltys. Vienoje jų bus *liko* reikšmė, kitoje bus surašyti visi skaičiai, kuriems atsiradus viršutiniame disko segmente po disko pasukimo galima laimėti.

Sakykime, kad *liko* = 1. Laimėti galima tik tada, kai pasukus diską viršutiniame disko segmente yra 1. Likusiais atvejais žaidėjas pralaimi.

Jei *liko* = 2, žaidėjas laimi, kai atsisuka 2 arba 1. Jei jis atsisuka 2, tuomet pasiekia sutartą skaičių. Jei atsisuka 1, tuomet *liko* reikšmė sumažėja iki 1 ir disko viršuje lieka 1. Priešininkas laimėti nebegalės, nes diską jis privalės pasukti, o pasukęs niekaip negalės disko viršuje gauti 1 – skaičiaus, kuris turi atsidurti disko viršutiniame segmente, norint laimėti, kai *liko* reikšmė lygi 1.

Taip žiūrėdami į jau išnagrinėtas situacijas, pildome lentelę:

liko reikšmė	skaičius atsiradęs viršutiniame disko segmente, pasukus diską	*liko* reikšmė	skaičius atsiradęs viršutiniame disko segmente, pasukus diską
1	1	15	5, 3, 1
2	2, 1	16	6, 5, 1
3	3	17	5, 3
4	4, 3, 1	18	3, 1
5	5, 4, 1	19	5, 3, 1
6	6, 5, 4, 3, 1	20	5, 3, 1
7	5	21	3, 1
8	5, 1	22	5, 3, 1
9	6, 2, 1	23	5, 3, 1
10	3,	24	5, 3, 1

11	3, 1	25	5, 3, 1
12	5, 1	26	5, 3, 1
13	5, 4, 3, 1	27	5, 3, 1
14	3, 1	28, 29, ...	5, 3, 1

Matome, kad norint laimėti, kai *liko* reikšmės yra didesnės už 21, pasukus diską turi atsisukti arba 1, arba 3, arba 5. Norėdami tai gauti, diską turime sukti į .... bet kurią pusę... Mat jei ėjimo pradžioje disko viršuje buvo vienas iš skaičių 1, 3, 5, į kurią pusę besuktume diską, disko viršuje atsiras 2, 4, arba 6. Iš kitos pusės jei ėjimo pradžioje disko viršuje buvo 2, 4 ar 6, pasukus diską į kairę ar į dešinę, atsisuks 1, 3 ar 5.

Beliko rasti tinkamus ėjimus mažesnėms *liko* reikšmėms ir juos surašyti į funkciją *diskas*.

```
function diskas (n, s, m: integer): integer;
                    { n – skaičius disko viršuje }
                    { s – sukaupta suma }
                    { m – galutinė suma }
  var krit: array [1..6] of set of 1..21;
            { kritinės situacijos, apsprendžiančios laimėjimą }
            { norint laimėti prie šių situacijų, būtina diską sukti į kairę }
      ka,   { koks skaičius bus viršuje diską pasukus į kairę }
      de,   { koks skaičius bus viršuje diską pasukus į dešinę }
      liko: integer;  { kiek dar liko iki sutarto skaičiaus }
begin
  { apskaičiuojame kritines situacijas }
  krit[1] := [2];
  krit[2] := [3, 10, 17];
  krit[3] := [4, 5, 13];
  krit[4] := [5, 7, 8, 12, 16];
  krit[5] := [9, 16];
  krit[6] := [1, 2, 4, 9, 11, 14, 18, 21];
  { rasime skaičius, kurie atsiras disko viršuje pasukus diską }
  if n = 6 then ka := 1
           else ka := n + 1;
  if n = 1 then de := 6
           else de := n - 1;
  liko := m-s;
  if liko > 21
    then diskas := ka
    else if liko in krit[n]
      then diskas := ka
      else diskas := de
end;  { diskas }
```

Sprendimas buvo tikrinamas su 12000 pradinių duomenų variantų: $m = 2000$, s kito nuo 0 iki m-1, n – nuo 1 iki 6.