Pokeris

Šis uždavinys yra tolesniame etape pateikto uždavinio „Vėl pokeris“ palengvintas variantas, todėl jam tiktų to uždavinio sprendimo idėja: tereikia ją supaprastinti. Tačiau tame sprendime yra nemažai kombinatorikos elementų, kurių gali nežinoti jaunesni moksleiviai. Todėl pateikiame paprastesnį algoritmą.

Bandysime keisti visas kortas iš eilės.

Imame pirmąją kortą ir dedame atgal į kaladę. Dabar kaladėje yra dvidešimt kortų. Traukiame iš kaladės kortą ir dedame vietoj pakeistosios. Turime penkių kortų kombinaciją. Reikia patikrinti, ar ši kombinacija yra išlošiama ir jei taip – nustatome kuri iš aštuonių išlošiamų kombinacijų bei padidiname šios kombinacijos variantų skaičių. Taip vietoj trūkstamos dedame visas kaladėje likusias kortas. Išbandę visas kortas, gauname, tokią lentelę (ji atitinka pavyzdį sąlygoje):

Keičiamos kortos numeris – 1								
Išlošiamos kombinacijos nr.	1	2	3	4	5	6	7	8
Keliais būdais galima sudaryti tą kombinaciją	12	6	2	0	0	0	0	0

Iš lentelės matome, kad pakeitus pirmąją kortą didžiausia tikimybė yra sudaryti *Porą*. Ši tikimybė lygi 12/20 = 0.6. Atkreipkite dėmesį, kad lentelėje tikimybės neskaičiuojamos. Mat tikimybė sudaryti kažkurią išlošiamą kombinaciją bus lygi variantų skaičiui, padalintam iš dvidešimties. Taigi, kuo didesnis variantų skaičius, tuo didesnė ir tikimybė.

Užpildytoji lentelę parodo išlošimo galimybes pakeitus pirmąją kortą. Tokia pat lentelė sudaroma dar keturis kartus – keičiant antrąją, trečiąją ir t. t. kortas. Ieškomą tikimybę bus galima apskaičiuoti iš visų penkių lentelių išrinkus didžiausią variantų skaičių ir jį padalinus iš 20.

```
const K_SK = 5;  { duotų kortų skaičius }
      IŠL = 8;   { išlošiamų kombinacijų skaičius }
      GAL_KOMB = 20;  { keičiamą kortą galime ištraukti tiek būdų }
type rangas = (dev, deš, val, dam, kar, tūz);
     rūšis = (čir, būg, kry, vyn);
     korta = record
               rū: rūšis;   { kortos rūšis }
               ra: rangas { kortos rangas }
```

```
            end;
      kortos = array [1..K_SK] of korta;
      išlošimai = array [1..IŠL] of integer;

function buvo (kr: kortos; { turimos kortos }
               i: integer; { keičiamos kortos numeris }
               k: korta{ naujai įdedamos kortos numeris}): boolean;
{ tikrina, ar nauja korta nesutampa su kuria nors nekeičiama korta }
  var j: integer;
begin
  buvo := false;
  for j := 1 to K_SK do
    if (i <> j) and
       (k.rū = kr[j].rū) and (k.ra = kr[j].ra)
       then buvo := true
end; { buvo }

procedure išlošimas (kr: kortos;
                     var nr,   { keičiamos kortos numeris }
                         komb: integer; { išlošiama kombinacija }
                     var tik: real);
  var kiek: išlošimai;
      { kelios kombinacijos sudarys i-tąją išlošiamą situaciją }
      keič, k: korta;
      kom_sk, { didžiausias kombinacijų skaičius }
      i, j: integer;
begin
  kom_sk := 0; { nesudarėme nei vienos išlošiamos kombinacijos }
  for i := 1 to K_SK do
    begin { keisime i-tąją kortą }
      for j := 1 to IŠL do
        kiek[j] := 0;
        keič := kr[i]; { įsimename keičiamą kortą }
        { vietoj keičiamos kortos įstatome visas likusias kortas }
        for k.rū := čir to vyn do
          for k.ra := dev to tūz do
            if not buvo(kr, i, k)
               { jei pakeista korta nedubliuoja kurios nors iš paliktų }
               then begin
                      kr[i] := k;
                      laimi (kr, kiek);
                    end;
        { patikrinsime, gal radome išlošiamą situaciją, kurią sudaryti yra }
        { dar didesnė tikimybė }
        for j := 1 to IŠL do
```

```
            if kiek[j] > kom_sk
               then begin
                       kom_sk := kiek[j];
                       nr := i;
                       komb := j;
                    end;
         kr[i] := keič; { atstatome pakeistą kortą }
     end;
   { beliko apskaičiuoti tikimybę }
   tik := kom_sk/GAl_KOMB;
end; { išlošimas }
```

Žemiau pateikta trūkstama procedūra *laimi*, kuri pagal turimas penkias kortas nustato ar jos sudaro kurią nors išlošiamą kombinaciją ir jei taip, tai kokią. Procedūra gan aiški, todėl jos daug nekomentuosime. Svarbiausia – suskaičiuoti kiek yra kiekvieno rango ir kiekvienos rūšies kortų. Jei pavyko sudaryti aukštesnio prioriteto kombinaciją (pavyzdžiui, *Ketvertą*), žemesnio prioriteto išlošiamos kombinacijos (pavyzdžiui *Trejetas*) nebenagrinėjamos.

```
procedure laimi (kr: kortos; var kiek: išlošimai);
{ tikrina, kurią laimimą situaciją sudaro turima penkių kortų kombinacija }
  var i: rangas;
      j: rūšis;
      ran: array [dev..tūz] of integer;
      rūš: array [čir..vyn] of integer;
      galima: array [1..IŠL] of boolean;
      { ar galima sudaryti i-tąją išlošiamą kombinaciją }
      t: integer;
begin
  for i := dev to tūz do
    ran[i] := 0;
  for j := čir to vyn do
    rūš[j] := 0;
  { suskaičiuosime kiek kokių kortų turime }
  for t := 1 to K_SK do
    begin
      ran[kr[t].ra] := ran[kr[t].ra] + 1;
      rūš[kr[t].rū] := rūš[kr[t].rū] + 1;
    end;
  for t := 1 to IŠL do
    galima[t] := false;
  { tikrinsime pirmąsias keturias kombinacijas }
  for i := dev to tūz do
    if ran[i] = 2
       then if not galima[1]
```

```
                then galima[1] := true { Pora }
                else galima[2] := true { Dvi poros }
        else if ran[i] = 3
                then galima[3] := true { Trejetas }
                else if ran[i] = 4
                        then galima[4] := true; { Ketvertas }
  { tikrinsime kombinaciją Pilni namai }
  galima[5] := galima[3] and galima[1];
  { tikrinsime kombinaciją Visos kortos vienos rūšies }
  galima[6] := false;
  for j := čir to vyn do
    if rūš[j] = K_SK
       then galima[6] := true;
  { tikrinsime kombinaciją Visos kortos iš eilės }
  galima[7] := true;
  for i := dev to tūz do
    if (ran[i]>1) or (ran[i]=0) and not (i in [dev,tūz])
       then galima[7] := false;
  { tikrinsime kombinaciją Pokeris }
  galima[8] := galima[6] and galima[7];
  { įsiminsime didžiausią sudarytą kombinaciją }
  t := IŠL;
  while (t > 0) and not galima[t] do
    t := t - 1;
  if t > 0
     then kiek[t] := kiek[t] + 1;
end; { laimi }
```