Vėl pokeris

Uþdavinys jungia kombinatorikos ir tikimybiø teorijos elementus. Sprendimas gana ilgas, todël já pateiksime dalimis ir kiekvienà dalá nagrinësime atskirai.

Sąlyga prašo surasti tokią paliekamų ir keičiamų kortų kombinaciją, kuriai esant išlošimo vidurkis būtų didžiausias. Pradžioje laikome, kad išlošimo vidurkis lygus nuliui. Tuomet konstruojame visas įmanomas paliekamų ir keičiamų kortų kombinacijas ir skaičiuojame kiekvienos kombinacijos išlošimo vidurkį (kaip jį skaičiuoti panagrinėsime vėliau). Kortų kombinacija, kurios išlošimo vidurkis didžiausias ir bus ieškomas rezultatas.

Žemiau pateikiame svarbiausią procedūrą *spręsk*:

```
const MAX_K = 5;   { duotų kortų skaičius }
      MAX_P = 8;   { išlošiančių padėčių skaičius }
      MAX_R = 6;   { kortų rangų skaičius }
type  korta = record
                ran: longint;  { kortos rangas }
                sp : char      { kortos rūšis }
              end;
      kortos = array [1..MAX_K] of korta;
      išlošimas = array [1..MAX_P] of longint;
                  { išlošiamos sumos }
      log_mas = array [1..MAX_K] of boolean;
                { paliekama korta ar keičiama }
      rangai = array [1..MAX_R] of longint;

procedure spręsk (k: kortos;        { duotosios kortos }
                  išl: išlošimas;   { kiekvienos išlošiančios padėties }
                                    { išlošimo sumos }
                  var opt: log_mas
      { uždavinio rezultatas – paliekamų ir keičiamų kortų sąrašas }) ;
    var pal: log_mas;
        max_nauda: real;  { maksimalus išlošimo vidurkis }
```

```
procedure palikti_keisti (n: longint; pal: log_mas);
   { perrenkami visi įmanomi paliekamų ir keičiamų kortų deriniai }
   { naudojami globalūs kintamieji k, išl, max_nauda ir opt }
   { kintamųjų k ir išl reikšmės nekeičiamos }
   { procedūros rezultatas – kintamojo opt reikšmė }
     var nauda: real;
  begin
     if n = 6 { visos kortos arba paliktos arba nurodytos keisti }
        then begin
              nauda := kokia_nauda (k, išl, pal);
              if nauda > max_nauda
              then begin { rastas naudingesnis kortų pasilikimo variantas }
                     max_nauda := nauda;
                     opt := pal;
                   end
            end
        else begin
               pal[n] := false;
               palikti_keisti (n + 1, pal);
               pal[n] := true;
               palikti_keisti (n + 1, pal);
             end
  end; { palikti_keisti }
begin { spręsk }
  max_nauda := -1;
  palikti_keisti (1, pal);
end; { spręsk }
```

Dabar panagrinėsime kaip rasti *išlošimo vidurkį*, kai žinoma kurios kortos yra paliekamos. Tam reikia suskaičiuoti keliais skirtingais būdais galima sudaryti kiekvieną išlošiamą padėtį bei rasti visų galimų skirtingų kombinacijų skaičių.

Išlošiamas kombinacijas suskirstome į du tipus:

a) *Pora*, *Dvi poros*, *Trejetas*, *Pilni namai*, *Ketvertas.*

b) *Iš eilės*, *Vienos spalvos*, *Pokeris* (šias kombinacijas galima sudaryti tik tada, kai visos paliktos kortos yra skirtingo rango).

Nagrinėjame a) ***Pora, Dvi poros, Trejetas, Pilni namai, Ketvertas.***

Pradėsime nuo šio tipo. Visas įmanomas kombinacijas sudarinėsime rekursiškai. Žinome, kad mums reikia parinkti *liko* kortų bei kiek kiekvieno rango kortų yra palikta. Sudarykime lentelę *vien*, turinčią 4 elementus. Ji rodys, kiek tarp paliktų kortų yra *Porų*, *Trejetų*, *Ketvertų*.

Rekursiją vykdysime pagal kortų rangus. *Devynakių* (arba kortų, kurių rangas yra 1) galime parinkti bet kokį skaičių iš intervalo [0, *min*(liko, p_9)], čia p_9 yra paliktų *devynakių* skaičius. Pavyzdžiui, turėjome tokias kortas: K1, V1, V3, B2, B6

Nusprendėme palikti kryžių devynakę ir vynų valetą, o keisti vynų devynakę, vynų dešimtakę, būgnų tūzą.

Tarp paliktųjų kortų yra viena *devynakė*. Taigi parinkdami kortas, galime neparinkti nei vienos *devynakės*, parinkti, vieną, dvi ar tris. Keturių *devynakių* parinkti negalime todėl, kad reikia parinkti tik tris kortas ir todėl, kad viena *devynakė* jau parinkta. Taigi, *devynakių* skaičių galime parinkti tokiais būdais:

K1, V3, ?, ?, ?	[0][0][0][0]
K1, V3, 9, ?, ?	[0][1][0][0]
K1, V3, 9, 9, ?	[0][0][1][0]
K1, V3, 9, 9, 9	[0][0][0][1]

Parinkdami *Devynakes* pildome ir lentelę *vien*.

Po to sumažiname kortų, kurias reikia parinkti, skaičių ir bandome parinkti *Dešimtakes* ir t. t. Kai parenkame visų rangų kortas, patikriname, ar turimų kortų skaičius lygus 5 ir suradę reikiamą išlošiamą padėtį padidiname tos padėties kombinacijų skaičių bei bendrą kombinacijų skaičių. Bendras kombinacijų skaičius yra didinamas net ir tada, jei jokia išlošiama padėtis nesusidaro.

Atskirai reiktų pakalbėti apie tai, kaip skaičiuojamos kombinacijos. Imkime tą patį pavyzdį, kai pasilikome kortas kryžių devynakę ir vynų valetą. Šią kombinaciją gavome vieninteliu būdu: pasilikdami tas dvi kortas. Rinkdami kortas pagal rangą, pirmiausia renkame *devynakes*. Sakykime, kad parinkome dvi *devynakes*. Galime gauti tokias kombinacijas:

K1, V3, S1, B1, ?
K1, V3, S1, V1, ?
K1, V3, B1, V1, ?

Galime pasakyti, taip: turime *n* (šiuo atveju trys – širdžių devynakė, būgnų devynakė, vynų devynakė) skirtingų kortų aibę. Iš jos reikia išrinkti poaibį iš *k* (šiuo atveju dvi) skirtingų kortų (t. y. negalime parinkti dviejų vynų devynakių). Toks poaibis vadinamas *Deriniu be pasikartojimų iš n po k* elementų. Skirtingų *derinių be pasikartojimų* skaičius lygus:

$C_n^k = n! / k! (n - k)!$

Mūsų atveju $C_3^2 = 3!/(2! (3-1)!) = 3$.

Sakykime, kad parinkę du *devynetus*, nusprendėme parinkti dar vieną dešimtuką. Dešimtuką galime parinkti $C_4^1 = 4!/(1!\times3!) = 4$ būdais. Kombinaciją kryžių devynakė, vynų valetas, dvi devynakės ir viena bet kokia korta, galėjome gauti 3 būdais. Taigi, kombinaciją kryžių devynakė, vynų valetas, dvi devynakės ir dešimtakė galime gauti $3\times4 = 12$ būdų (žr.pav.):

K1, V3, S1, B1, S2	K1, V3, S1, B1, B2	K1, V3, S1, B1, K2	K1, V3, S1, B1, V2
K1, V3, S1, V1, S2	K1, V3, S1, V1, B2	K1, V3, S1, V1, K2	K1, V3, S1, V1, V2
K1, V3, B1, V1, S2	K1, V3, B1, V1, B2	K1, V3, B1, V1, K2	K1, V3, B1, V1, V2

```
function cNpok (n, k: longint): longint;
{ randa derinių be pasikartojimų skaičių: c iš n po k }
  var sk: longint;
      i: longint;
begin
   sk := 1;
   for i := n downto n-k+1 do sk := sk*i;
   for i := 2 to k do sk := sk div i;
   cNpok := sk
end; { cNpok }

function min (a, b: longint): longint;
begin
  if a < b then min := a
           else min := b
end; { min }
```

```
procedure perrinkti (ran, liko: longint; komb: longint);
{ randa Porų, Trejetų, Ketvertų kombinacijų skaičių, }
{ rekursijos gylį nurodo rangas ran, einama padėtis gali susidaryti komb būdais }
{ taip pat randa visų įmanomų kombinacijų skaičių }
{ procedūra naudoja globalius kintamuosius: vien, išviso, kiek }
{ procedūros rezultatai – išviso ir kiek reikšmės }
  var i: longint;
begin
  if ran > MAX_R { perbėgti visi rangai }
  then if liko = 0 { visos 5 kortos parinktos }
       then begin
         if vien[4] > 0 then kiek[7] := kiek[7] + komb
         else if vien[3] > 0
              then if vien[2] > 0
                   then kiek[6] := kiek[6]+komb
                   else kiek[3] := kiek[3]+komb
              else if vien[2] > 0
                   then kiek[vien[2]] := kiek[vien[2]] +
                                         komb;
              išviso := išviso + komb
         end
       else
  else for i := 0 to min (liko, 4-rng[ran]) do
       { imamas visoks įmanomas skaičius ran rango kortų }
         begin
          if i + rng[ran] > 0
          then vien[i+rng[ran]] := vien[i+rng[ran]] + 1;
          perrinkti(ran+1, liko-i, komb*cNpok (4-rng[ran],
i));
          if i + rng[ran] > 0
          then vien[i+rng[ran]] := vien[i+rng[ran]] - 1;
        end;
end;  { perrinkti }
```

Nagrinėjame b) ***Iš eilės, Vienos spalvos, Pokeris*** (šias kombinacijas galima sudaryti tik tada, kai visos paliktos kortos yra skirtingo rango). Patikrinę, ar turimos kortos yra skirtingo rango, imkimės kombinacijų skaičiavimo.

Pradžioje tarkime, kad turime, tik vienos rūšies kortas (pavyzdžiui, *kryžius*). Žinome, kad yra MAX_R skirtingų kortų rangų, ir reikia sudaryti kombinaciją iš MAX_K kortų. Skaičiuodami kombinacijas kartu nagrinėsime pavyzdį, kai Pavyzdžiui, MAX_R = 8 (prisideda *septynakė* ir *aštuonakė*) o reikia parinkti 5 kortas. Skirsime du atvejus:

- *nėra palikta nei vienos kortos*. tuomet, galime sudaryti MAX_R - MAX_K + 1 skirtingą kombinaciją.

Mūsų pavyzdžiu tai galima padaryti 8-5+1 = 4 būdais (žr. pav.)

K7, K8, K1, K2, K3
K8, K1, K2, K3, K4
K1, K2, K3, K4, K5
K2, K3, K4, K5, K6

- *yra palikta keletas kortų*. Tuomet randame mažiausio ir didžiausio rango paliktas kortas. Visas tarp šių rangų esančias kortas, kurios nėra parinktos parenkame vienareikšmiškai. Pavyzdžiui, yra palikta kryžių aštuonakė, kryžių dešimtakė ir kryžių dama.

Trūkstamas kortas kryžių devynakę ir kryžių valetą parenkame vienareikšmiškai (žr. pav.):

K8, K1, K2, K3

Atstumas tarp mažiausio ir didžiausio rangų turi būti mažesnis nei MAX_K. Kitaip negalėsime parinkti nei vienos kombinacijos. Pavyzdžiui, jei palikta kryžių devynakė ir kryžių tūzas, negalėsime sudėti nei vienos kombinacijos.

Sakykime, kad parinkus vidurines kortas vienareikšmiškai dar reikia parinkti *kor* kortų. Trūkstamas kortas galime prijungti arba prie kurio nors vieno, arba prie abiejų kortų eilutės galų. Pavyzdžiui, jei palikta kryžių valetas ir kryžių karalius, vienareikšmiškai parenkame kryžių damą. Likusias dvi kortas galime parinkti taip (žr. pav.):

K1, K2, K3, K4, K5
K2, K3, K4, K5, K6

Iš vienos rūšies kortų galima sudaryti *komb* kombinacijų. Kiekviena nauja kombinacija gaunama keičiant kortų rūšis. Jei keičiant rūšį iš vienos kombinacijos galima gauti *kmb* kombinacijų, tai iš viso bus *komb* × *kmb* kombinacijų.

Rasime skaičių *kmb*. Imkime kurią nors kombinaciją, pavyzdžiui, kryžių devynakę, dešimtakę, valetą, damą, karalių. Keisime parinktų kortų kryžių devynakės, dešimtakės, damos rūšį. Keisdami devynakės rūšį, gausime keturias kombinacijas (žr. pav.):

S1, K2, K3, K4, K5,
B1, K2, K3, K4, K5,
K1, K2, K3, K4, K5,
V1, K2, K3, K4, K5,

Keisdami dešimtakės rūšį iš kiekvienos turimos kombinacijos gausime dar po 4 kombinacijas. Iš viso turėsime 4×4 = 16 kombinacijų. Keisdami damos rūšį vėl iš kiekvienos kombinacijos gausime po keturias naujas kombinacijas. Taigi, kombinacijų skaičius padidės iki 16×4 = 64 kombinacijų.

Gauname, kad $kmb = liko^4$, čia *liko* – parenkamų kortų skaičius.

Bendras kombinacijų, sudarančių išlošiamą situaciją *Iš eilės* skaičius lygus

$$komb \times liko^4$$

Į šį skaičių įtrauktos ir išlošiamos kombinacijos *Vienos spalvos* bei *Pokeris*. Todėl radus jų skaičių, jį reikės atimti iš išlošiamos padėties *Iš eilės* kombinacijų skaičiaus.

Rasime išlošiamos padėties *Pokeris* kombinacijų skaičių. Jei palikta nors viena korta, kombinacijų skaičius lygus anksčiau apskaičiuotam *komb*, mat spalvos keisti negalima. Jei nepalikta nei vienos kortos, jis lygus: *komb* × 4.

```
function kelk (a, n : longint): longint;
{ a kelia n-tuoju laipsniu }
  var i: longint;
      sk: longint;
begin
   sk := 1;
   for i := 1 to n do
      sk := sk * a;
   kelk := sk
end; { kelk }

function vienodos_spalvos (k: kortos; pal: log_mas):
                                                boolean;
{ patikrina, ar paliktos kortos yra vienodos spalvos }
  var sp: char;
      i: longint;
```

```
begin
  vienodos_spalvos := true;
  sp := ' ';          { nežinome pirmosios paliktos kortos spalvos }
  for i := 1 to 5 do   { patikrinama ar paliktos kortos yra }
    if pal[i] then     { vienodos spalvos }
       if sp = ' '   { jei tai pirma palikta korta }
          then sp := k[i].sp
          else if k[i].sp <> sp { yra nevienodos spalvos kortų }
                  then vienodos_spalvos := false
end; { vienodos_spalvos }

procedure iš_eilės_komb (liko: longint; pal: log_mas;
                         k: kortos;
                         var kiek: išlosimas);
{ suskaičiuoja kiek yra galimų kombinacijų, kuriose kortos yra }
{ išdėliotos iš eiles; t. y. skaičiuoja keliais variantais sudaroma }
{ ketvirta ir aštunta išlošianti kombinacija }
  var komb, min, max, i, kr, kair, deš: longint;
begin
  { pradžioje skaičiuosime kombinacijas, tarsi turėtume tik }
  { vienos rūšies kortas }
  if liko = MAX_K then komb := MAX_R - MAX_K + 1
  else begin
        max := 1; min := MAX_R; { randamos mažiausio ir didžiausio }
        for i := 1 to MAX_K do { rango paliktos kortos }
          if pal[i]
            then begin
                  if k[i].ran > max then max := k[i].ran;
                  if k[i].ran < min then min := k[i].ran
                end;
        if max - min >= MAX_K { jei atstumas tarp rangų per didelis }
           then komb := 0
           else begin
                  { tiek kortų reikia pridėti iš galų }
                  kr := MAX_K - (max - min + 1);
                  komb := 0;
                  for kair := 0 to kr do
                    begin
                      { tiek kortų pridėsime iš dešinio galo }
                      deš := kr - kair;
                      if (min - kair >= 1) and
                         (max + deš <= MAX_R)
                          then komb := komb + 1;
                    end;
```

```
                end;
        end;
  { rasime kombinacijų skaičių, kai kortos yra keturių rūšių }
  kiek[4] := komb * kelk (4,liko); { kiekviena iš parenkamų kortų }
                                   { gali būti vienos iš keturių spalvų }
  if vienodos_spalvos (k, pal) { Pokeris }
     then begin
            if liko = 5
               then kiek[8] := komb*4 { tinka bet kuri spalva }
               else kiek[8] := komb;  { tinka paliktų kortų spalva }
            { šios kombinacijos jau buvo priskaičiuotos prie ketvirtos }
            { išlošiamosios padėties }
             kiek[4] := kiek[4] - kiek[8]
          end
end; { iš_eilės_komb }
```

Beliko apskaičiuoti *Vienos spalvos* kombinacijų skaičių. Samprotaujama analogiškai, kaip ir skaičiuojant išlošiamų padėčių su keliomis vienodo rango kortomis (*Pora, Dvi poros* ir t. t.) kombinacijų skaičių.

Visos parenkamos kortos bus skirtingo rango ir tos pačios spalvos. Reikia parinkti *liko* kortų. Jau yra parinkta MAX_K - *liko* kortų. Kadangi iš viso yra MAX_R skirtingo rango kortų, galima rinkti iš MAX_R - (MAX_K - *liko*) kortų. Vėl gauname *Derinius be pasikartojimų*, kur *n* = MAX_R - (MAX_K - *liko*), o k = *liko*.

```
procedure vien_spalvos_komb (liko: longint;
                             var kiek: išlošimas);
{ randa kiek yra vienodos spalvos kombinacijų }
begin
  kiek[5] := cNpok (MAX_R-(5-liko), liko);
  if liko = 5 then kiek[5] := kiek[5]*4; { tinka bet kuri spalva }
  { atimamos Pokerio kombinacijos }
  kiek[5] :=  kiek[5] - kiek[8]
end; { vien_spalvos_komb }
```

Procedūras *perrinkti*, *iš_eilės_komb* ir *vien_spalvos_komb* sujungiame funkcija *kokia_nauda*.

```
function kokia_nauda (k: kortos; išl: išlošimas;
                      pal: log_mas { kiek atitinkamo rango kortų }
                 { yra palikta }: real;
{ skaičiuoja, koks išlosimo vidurkis pakeitus nurodytas kortas }
```

```
  var vien: array [1..4] of longint;  { vien[i] > 0, jei yra i }
                                      { vienodų kortų }
      rng: rangai;  { kiek tarp paliktų kortų yra šio rango kortų }
      kiek: išlošimas;  { kelias kombinacijas galima sudaryti, kad }
                        { gautume i-tąją išlošiančiąją padėtį }
      išviso,  { galimų skirtingų kombinacijų skaičius, keičiant }
               { nurodytas kortas }
      liko,   { kiek kortų reikia parinkti }
      vid, i: longint;
      skirt: boolean;
  procedure analizuok_paliktas(k: kortos;  { duotosios kortos }
                               pal: log_mas;  { paliktos kortos }
                               var rng: rangai;
                               var liko: longint);
  { suskaičiuoja kiek iš kortų reikia parinkti }
  { ir kiek kiekvieno rango kortų palikta }
    var i: longint;
  begin
    for i := 1 to  MAX_R do
      rng[i] := 0;
    liko := MAX_K;
    for i := 1 to MAX_K do
      if pal[i] then begin
                       liko := liko - 1;
                       rng[k[i].ran] := rng[k[i].ran] + 1
                     end;
  end;  { analizuok_paliktas }
begin
  { suskaičiuojama kiek kortų reikia parinkti  ir kiek kokio rango kortų liko }
  analizuok_paliktas (k, pal, rng, liko);
  { kol kas nesudarėme nei vienos išlošiamos padėties }
  for i := 1 to MAX_P do
    kiek[i] := 0;
  for i := 1 to 4 do
    vien[i] := 0;
  išviso := 0;  { nesudarėme nei vienos kombinacijos }
  { rasime visų galimų kombinacijų skaičių bei kombinacijų skaičių, kai }
  { susidaro išlošiančios padėtys: Pora, Dvi poros, Trejetas, Ketvertas }
  { į likusias kombinacijas nekreipiama dėmesio; procedūros rezultatas – }
  { masyvas kiek }
  perrinkti (1, liko, 1);
  { bandysime sudaryti likusias kombinacijas }
  { patikrinsime, ar paliktos kortos yra skirtingo rango }
  skirt := true;
```

```
  for i := 1 to MAX_R do
    skirt := skirt and (rng[i] <= 1);
  if skirt { visos kortos skirtingos }
     then begin { ieškosime 4-tos, 5-tos ir 8-tos išlošiančių }
            iš_eilės_komb (liko, pal, k, kiek);   { padėčių }
            if vienodos_spalvos (k, pal)
               then vien_spalvos_komb (liko, kiek)
          end;
  { apskaičiuojamas išlošimo vidurkis }
  vid := 0;
  for i := 1 to MAX_P do
    vid := vid + kiek[i] * išl[i];
  kokia_nauda := vid/išviso;
end; { kokia_nauda }
```