Raidžių stačiakampis

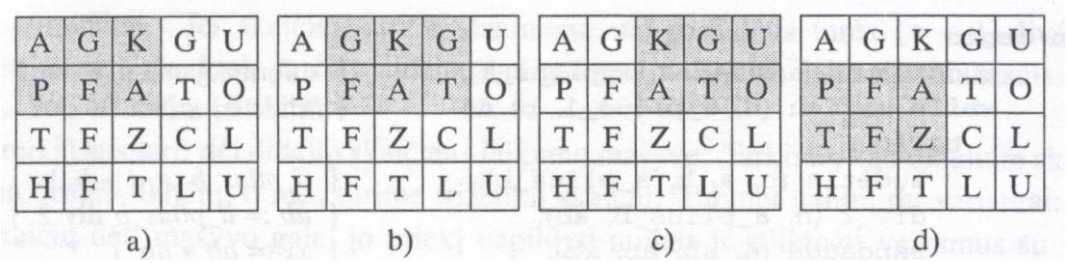
Tai tipiškas perrinkimo uždavinys. Vienintelė problema – optimizuoti perrinkimą, kad programos vykdymo laikas neviršytų duotojo.

Lotyniškoje abėcėlėje yra 26 raidės. Taigi, ieškomas stačiakampis negali turėti daugiau kaip 26 langelius. Naudinga suskaičiuoti, kiek yra skirtingų raidžių duotajame stačiakampyje: jų skaičiaus taip pat negali viršyti ieškomo stačiakampio dydis.

Perrenkama tokiu būdu. Sakykim, reikia surasti stačiakampį, kurio dydis yra 26. Peržiūrimi to skaičiaus dalikliai ir randamos galimos stačiakampio aukščio ir pločio kombinacijos. Šiuo atveju tai būtų (1, 26), (2, 13), (13, 2), (26, 1). Tuomet lape bandoma rasti kiekvieno dydžio stačiakampį.

Fiksuotų matavimų stačiakampio ieškoma taip. Imamas lapas ir laikoma, kad ieškomo stačiakampio kairiojo viršutiniojo kampo koordinatės yra (1, 1). Tuomet tikrinama, ar pažymėtame stačiakampyje visos raidės skirtingos. Jei ne, imama jo dalis su skirtingomis raidėmis ir žiūrima, ar ji nėra didesnė už iki šiol rastą didžiausią stačiakampį. Neradus ieškomo dydžio stačiakampio, viršutiniojo kampo koordinatės paslenkamos per vieną langelį į (1, 2) ir taip kartojami veiksmai, kol peržiūrimas visas lapas.

Pavyzdžiui, lapo dydis yra 4×5. Jame ieškome 2×3 dydžio stačiakampio:



Imkime atvejį d). Sakykime, kad jau peržiūrėjome pajuodintas raides ir sustojome ties antrąją F. Nors neradome stačiakampio, kurio dydis būtų lygus 6, radome du mažesnius stačiakampius. Vieno jų dydis lygus dviem, kito – trim. Jei iki šiol buvome radę mažesnį stačiakampį, įsimename didesnįjį iš rastųjų.

```
const MAX = 100;  { maksimalus lapo kraštinės dydis }
      RSK = 26;  { raidžių skaičius abėcėlėje }
type lapas = array [1..MAX, 1..MAX] of char;

{ žemiau surašyti globalieji kintamieji }
var s: lapas;          { duotasis lapas }
    m, n: integer;  { jo išmatavimai }

function skirt: integer;
{ randa skirtingų raidžių skaičių stačiakampyje s }
  var aibė: set of char;
      i, j, kiek: integer;
      c: char;
begin
  { visas stačiakampio raides surašysime į aibę }
  aibė := [];
  for i := 1 to m do
    for j := 1 to n do
      aibė := aibė + [s[i, j]];
  { rasime skirtingų raidžių skaičiaus }
  kiek := 0;
  for c := 'A' to 'Z' do
    if c in aibė
       then kiek := kiek + 1;
  skirt := kiek
end;  { skirt }

function daliklis (sk, x: integer): integer;
{ randa mažiausią sk daliklį didesnį už x }
begin
  x := x + 1;
  while  sk mod x <> 0 do
    x := x + 1;
  daliklis := x;
end;  { daliklis }

procedure ieškoti (au, pl: integer;
                   var aukštis, plotis,
                       eil, st, dydis: integer);
                     { jei randamas stačiakampis įsimenami jo duomenys }
{ tikrina, ar lape yra stačiakampis su skirtingomis raidėmis, }
{ kurio aukštis au, o plotis – pl }
  var i, j, i1, j1: integer;
      nerasta, skirtingos: boolean;
```

```
      aibė: set of char;
begin
  i := 1;  nerasta := true;
  while (i <= m-au+1) and nerasta do
    begin { kairysis viršutinis kampas yra eilutėje i }
      j := 1;
      while (j <= n-pl+1) and nerasta do
        begin    { ir stulpelyje j }
          { tikrinama, ar stačiakampis su tokiais parametrais }
          { sudarytas iš skirtingų raidžių }
          aibė := [];  skirtingos := true;
          i1 := i;
          while (i1 <= i+au-1) and skirtingos do
            begin
              j1 := j;
              while (j1 <= j+pl-1) and skirtingos do
                if s[i1, j1] in aibė
                   then skirtingos := false
                   else begin
                          aibė := aibė + [s[i1, j1]];
                          j1 := j1 + 1;
                        end;
               if skirtingos
                  then i1 := i1 + 1;
            end; { while }
          if skirtingos   { radome ko ieškojome }
             then begin  nerasta := false;
                         aukštis := au; plotis := pl;
                         eil := i; st := j;
                         dydis := au * pl;
                  end
             else begin  { radome ne tokį didelį stačiakampį kokio }
                         { ieškojome, bet gal didesnį nei iki šiol rastą }
                    { rasime paskutinės skirtingos raidės koordinatę }
                    if j1 = 1
                       then begin i1 := i1 - 1;
                                  j1 := j+pl-1
                            end
                       else j1 := j1 - 1;
                    if (i1-i+1) * (j1-j+1) > dydis
                    then begin
                           aukštis := i1-i+1;
                           plotis := j1-j+1;
                           eil := i; st := j;
                           dydis := aukštis * plotis;
                         end
```

```
                    else if (i1-i)*pl > dydis
                      then begin
                         aukštis := i1-i; plotis := pl;
                         eil := i; st := j;
                         dydis := aukštis * plotis;
                       end
                  end;
            j := j + 1;
          end;
        i := i + 1;
      end;
end;  { ieškoti }

procedure rasti (var aukštis, plotis, eil, st: integer);
                  { ieškomo stačiakampio išmatavimai ir kairiojo }
                  { viršutinio kampo koordinatės }
{ randa didžiausią stačiakampį su skirtingomis raidėmis }
  var  didž, dydis, au, pl, i, j: integer;
begin
  { rasime skirtingų raidžių skaičių stačiakampyje }
  didž := skirt;          { didesnio stačiakampio negali būti }
  dydis := 1;             { didžiausio iki šiol rasto stačiakampio dydis }
  eil := 1;  st := 1;     { ir kiti duomenys }
  aukštis := 1;   plotis := 1;
  { atliksime perrinkimą }
  while dydis < didž do       { ieškome stačiakampio, kurio plotas – didž }
    begin
      pl := 1;
      while pl <= didž do    { tokio stačiakampio plotis yra pl }
        begin
          au := didž div pl;
          if (au <= m) and (pl <= n) then
            ieškoti(au,pl,aukštis,plotis,eil,st,dydis);
          { randame tolesnę pl reikšmę }
          if pl < didž
             then pl := daliklis (didž, pl)
             else pl := pl + 1;  { ciklas baigiasi }
        end;
      didž := didž - 1;
    end;
end;  { rasti }
```