Nameliai

Pirmas algoritmo žingsnis turbūt akivaizdus visiems: pagaliukai surikiuojami jų ilgių didėjimo (ar mažėjimo) tvarka. Tuomet atrenkami namelių sienoms tinkami pagaliukai, t. y. vienodo ilgio pagaliukų ketvertai ir sunumeruojami. „Sienos numeris“ skamba labai keistai, todėl vartosime terminą „namelio numeris“.

Tada taikome grįžimo metodą – perrenkame visus įmanomus variantus. Rekursijos parametras – namelio numeris *nr*. Didžiausias namelių skaičius randamas taip:

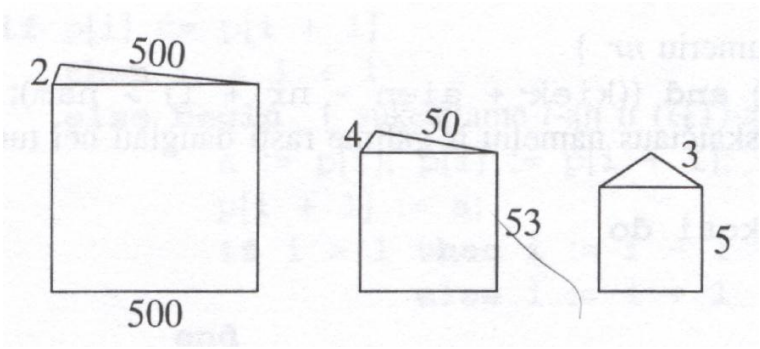
Suskaičiuojame kiek daugiausia namelių galime sukonstruoti, jei nestatysime namelio su numeriu *nr*. Taip pat suskaičiuojame kiek namelių galime pastatyti, jei šį namelį statysime. Skaičiuojant šį atvejį būtina išbandomi visi jau pastatytų namelių stogams nesunaudoti pagaliukai, t. y. visi galimi namelio stogo variantai.

```
const MAX_P = 500;  { maksimalus pagaliukų skaičius }
type  pagaliukai = array [1..MAX_P] of integer;
        lentelė = array [1..MAX_P] of boolean;

procedure rikiuoti (n: integer; var p: pagaliukai);
{ surikiuoja pagaliukus ilgių mažėjimo tvarka }
{ rikiuojama burbuliuko metodu }
   var i, a: integer;
begin
  i := 1;
  while i < n do
    if p[i] <= p[i + 1]
       then i := i + 1
       else begin   { sukeičiame i-tąjį ir (i+1)-mąjį elementą }
              a := p[i];
              p[i] := p[i + 1];
              p[i + 1] := a;
              if i > 1 then i := i - 1
                       else i := i + 1;
            end
end;  { rikiuoti }
```

```
function min (x, y: integer): integer;
begin
  if x < y
     then min := x
     else min := y;
end; { min }

function max (x, y: integer): integer;
begin
  if x > y
     then max := x
     else max := y;
end; { max }

function trik (a, b, c: integer): boolean;
{ ar galima iš atkarpų a, b, c sudaryti trikampį }
begin
  trik := ((a + b) > c) and ((a + c) > b) and
          ((b + c) > a)
end; { trik }

function namel (n: integer; p: pagaliukai): integer;
  var sien, nam, daug, i: integer;
      s: pagaliukai;
      nau: lentelė;

  function konstr (nr, kiek: integer): integer;
  { rekursinė funkcija, konstruojanti namelius }
    var viso, pag, i, j: integer;
        laisva, ieškoti: boolean;
  begin
    if nr > sien
    then begin
           if kiek > nam
              then nam := kiek;
           konstr := kiek;
         end
    else begin
           { galime šio namelio nekonstruoti }
           viso := konstr(nr+1, kiek);
           { galime konstruoti namelį su numeriu nr }
           { tikriname, ar nesunaudoti sienoms skirti pagaliukai }
           laisva := true;
           pag := s[nr];
```

```
for i := 0 to 3 do
  if nau[pag + i]
     then laisva := false;
if not laisva { jei pagaliukai sunaudoti }
   then begin { daugiau nieko negalime padaryti }
          konstr := viso;
          exit;
        end;
{ sunaudosime sienoms skirtus pagaliukus }
for i := 0 to 3 do
  nau[pag + i] := true;
{ rasime stogą nameliui su numeriu nr }
ieškoti := (nam < daug) and
           ((kiek + sien-nr+1) > nam);
{ jei neradome maksimalaus skaičiaus namelių ir }
{ galime rasti daugiau nei turime }
i := 1;
while (i <= n) and ieškoti do
  begin
    j := i + 1;
    while (j <= n) and ieškoti do
      begin
        if not nau[i] and not nau[j] and
           trik (p[i], p[j], p[s[nr]])
        then begin { radome stogą }
               nau[i] := true;
               nau[j] := true;
               viso :=
                 max(viso,konstr(nr+1,kiek+1));
               nau[i] := false;
               nau[j] := false;
               ieškoti := (nam < daug) and
                     ((kiek + sien-nr) > nam);
             end;
        { praleidžiami vienodo ilgio pagaliukai }
        while (j <= n-1) and (p[j] = p[j+1]) do
           j := j + 1;
        j := j + 1;
        if p[i] + p[s[nr]] <= p[j]
        { jei j-asis pagaliukas jau per ilgas }
           then j := n + 1;
      end;
    i := i + 1;
  end;
```

```
            { atstatome buvusias reikšmes }
            for i := 0 to 3 do
              nau[pag + i] := false;
            konstr := viso
          end;
  end;  { konstr }

begin
  { pagaliukai surikiuojami ilgių didėjimo tvarka }
  rikiuoti (n, p);
  { surasime visus galimas namelių sienas }
  sien := 0;
  for i := 1 to n-3 do
    if p[i] = p[i+3]
       then begin  { radome naują sieną }
              sien := sien + 1;
              s[sien] := i;  { įsimename vietą }
            end;
  daug := min (sien, n div 6);  { daugiau namelių sudaryti negalima }
  nam := 0;  { didžiausias sukonstruotų namelių skaičius }
  { nei vienas pagaliukas dar nepanaudotas }
  for i := 1 to n do
    nau[i] := false;
  if daug = 0
     then namel := 0
     else namel := konstr(1, 0);
end;  { namel }
```

Manantiems, kad uždavinio sprendimui tinka *Šykštusis* algoritmas siūlome panagrinėti pavyzdį, kai duota 18 pagaliukų: 2, 3, 3, 4, 5, 5, 5, 5, 50, 53, 53, 53, 53, 500, 500, 500, 500, 500.

Iš jų galima sudaryti tris namelius (žr. pav.):