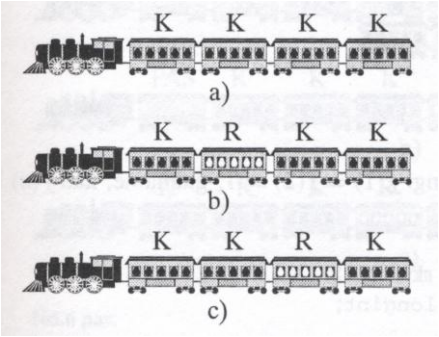
# Traukinys

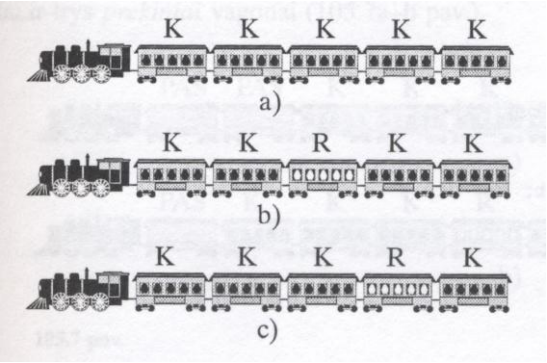
Tarkime, kad mokame suskaičiuoti keliais būdais galima suformuoti sąstatą, jei traukinį sudaro tik *keleiviniai* vagonai ir vagonai-*restoranai*. Jei traukinį sudaro *n* vagonų, tą skaičių pažymėkime *f*(*n*).

Dabar prisiminkime, kad yra dar dviejų rūšių vagonai. *Pašto* vagonų yra ne daugiau dviejų, o *prekinių* – ne daugiau trijų.

Kai turime vieną *pašto* arba *prekinį* vagoną, tai yra du variantai – arba *pašto* vagonas priekyje arba *prekinis* gale (1.a-b pav.).

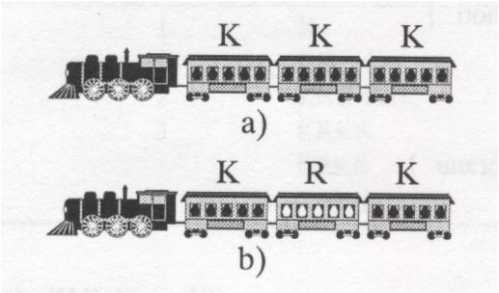



Kai yra du tokie vagonai, turime tris variantus – du *pašto* vagonai priekyje, vienas *pašto* priekyje ir vienas *prekinis* gale arba du *prekiniai* vagonai gale (2.a-c pav.).




Kai turime tris *pašto* ar *prekinius* vagonus yra trys variantai – du *pašto* ir *prekinis*, *pašto* ir du *prekiniai* arba trys *prekiniai*.

Kai yra keturi tokie vagonai, galimi du variantai – du *pašto* ir du *prekiniai* arba vienas *pašto* ir trys *prekiniai* vagonai (3.a-b pav.).




Jei turime penkis *pašto* ir *prekinius* vagonus, yra vienas atvejis – du *pašto* ir trys *prekiniai* vagonai.

Sąstate yra $n$ vagonų ir jame gali būti vienas, du, trys, keturi, penki ar nei vieno *prekinio* ir *pašto* vagono.

Jai sąstate yra vienas toks vagonas, tai jame yra $n$-1 *prekinis* ar *keleivis* vagonas. Skirtingų šių dviejų rūšių vagonų kombinacijų gali būti $f(n-1)$. Kiekvieno tokio sąstato priekyje galima pridėti *pašto* vagoną arba gale prijungti *prekinį* vagoną. Taigi, variantų skaičius lygus $2 \times f(n-1)$.

Analogiškai samprotaudami gauname, kad kai $n \geq 6$ iš viso yra

$f(n) + 2\times f(n-1) + 3\times f(n-2) + 3\times f(n-3) + 2\times(n-4) + f(n-5)$ būdų sudaryti traukinį, čia $f(n)$ rodo keliais variantais galima suformuoti traukinį, jei sąstatą sudaro $n$ vagonų, o šie yra arba *keleiviniai*, arba *restoranai*.

Kai $n$ mažesnis už 6, atmetami negalimi atvejai.

Belieka išsiaiškinti, kaip suskaičiuoti $f(n)$. Vienas paprasčiausių būdų: traukinį formuoti rekursiškai. Tačiau toks algoritmas viršys laiko limitus. Galima pasinaudoti *Dinaminio programavimo* metodu. Susidarysime lentelę iš $n$ elementų, ją pavadinkime $v$. Kiekvienas jos elementas sudarytas iš dviejų laukų. $v[i]$.*kele* rodo keliais būdais galima suformuoti sąstatą iš dviejų rūšių vagonų, jei pirmasis vagonas yra *keleivinis*, $v[i]$.*rest* – jei pirmasis vagonas yra *restoranas*.

Mūsų ieškoma reikšmė yra $v[n]$.*kele*, mat sąstatas negali prasidėti *restoranu*. Šios reikšmės ieškosime rekursiškai, tačiau kartu pildysime lentelę ir jei tos pačios reikšmės $v[i]$ prireiks antrą ar trečią kartą, jos nebeieškosime rekursiškai, o pasiimsime iš lentelės.

Jei $n = 4$, lentelė atrodytų taip:

| | lauko *kele* reikšmė | Paaiškinimas | lauko *rest* reikšmė | Paaiškinimas |
|---|---|---|---|---|
| *v*[1] | 1 | K | 0 | Sąstatas negali baigtis *restoranu* |
| *v*[2] | 1 | KK | 1 | RK |
| *v*[3] | 2 | KKK<br>KRK | 1 | RKK |
| *v*[4] | 3 | KKKK<br>KRKK<br>KKRK | 2 | RKKK<br>RKRK |

```
const MAX_N = 43;
type vagonas = (kel, res);
     lentelė = array [1..MAX_N] of record
                                        kele, rest: longint;
                                      end;

function vagonai (n: integer): longint;
  var viso, i: longint;
      v: lentelė;
      { v[i].kele – keliais būdais galima suformuoti sąstatą iš i vagonų, }
      { jei sąstatas prasideda keleiviniu vagonu }
      { ir formuojamas tik iš restoranų ir keleivinių vagonų }

  function form (m: longint; { formuojamo sąstato ilgis }
                 pirm: vagonas { pirmojo vagono tipas }) :
                                                        longint;
  { funkcijos rezultatas – v[m].kele reikšmė, jei pirm = kel ,
  { arba v[i].rest priešingu atveju }
  { šia reikšme funkcija papildo lentelę v }
    var kiek: longint;
  begin
    { gal būt reikiamos reikšmės jau apskaičiuotos }
    if (pirm = kel) and (v[m].kele <> -1)
    then form := v[m].kele
    else if (pirm = res) and (v[m].rest <> -1)
         then form := v[m].kele
         else begin
```

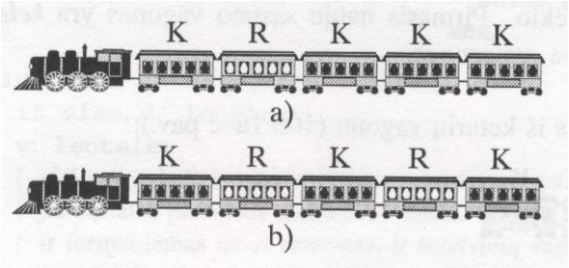
```
                { po keleivinio vagono gali eiti arba keleivinis }
                if v[m-1].kele = -1
                   then kiek := form (m-1, kel)
                   else kiek := v[m-1].kele;
                { arba vagonas-restoranas }
                if (m >= 3) and (pirm = kel)
                then if v[m-1].rest = -1
                     then kiek := kiek + form (m-1, res)
                     else kiek := kiek + v[m-1].rest;
                { papildome lentelę v }
                if pirm = kel
                   then v[m].kele := kiek
                   else v[m].rest := kiek;
                form := kiek;
             end;
  end;  { form }

begin
  for i := 1 to n do
    begin
      v[i].kele := -1;   { sąstatai dar nesuformuoti }
      v[i].rest := -1;
    end;
  v[1].kele := 1;
  v[1].rest := 0;  { sąstatas negali prasidėti restoranu }
  if n > 1
     then viso := form (n, kel)
     else viso := 1;
  if n > 1 then viso := viso + 2 * v[n-1].kele;
  if n > 2 then viso := viso + 3 * v[n-2].kele;
  if n > 3 then viso := viso + 3 * v[n-3].kele;
  if n > 4 then viso := viso + 2 * v[n-4].kele;
  if n > 5 then viso := viso + v[n-5].kele;
 vagonai := viso
end;  { vagonai }
```

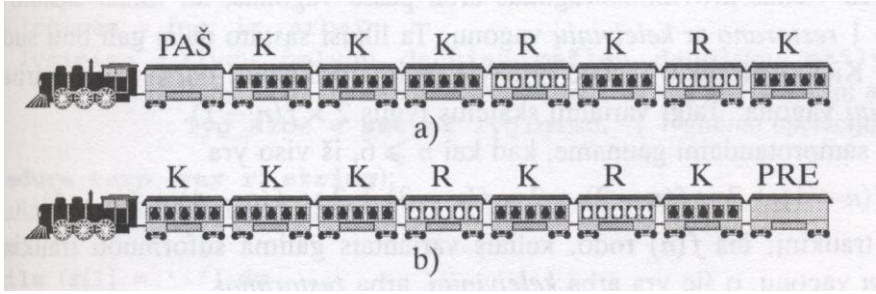
Yra ir kitas, nors matematiškas, tačiau labai įdomus būdas suskaičiuoti $f(n)$. Išvesime rekurentinę formulę kaip skaičiuoti $f(n)$, žinant ankstesnes reikšmes $f(1)$, $f(2)$, ...$f(n\text{-}1)$. Naujus vagonus bandysime prijungti iš priekio. Pirmasis naujo sąstato vagonas yra *keleivinis*. Antrasis gali būti arba *keleivinis* arba *restoranas*.

Tarkime, kad $n = 5$.

Imame visas galimas kombinacija iš keturių vagonų (4.a-b pav.):



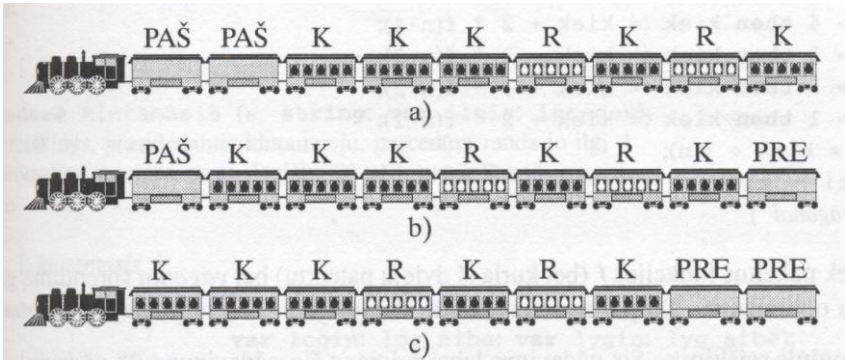

Prie kiekvieno tokio sąstato priekyje galime prijungti *keleivinį* vagoną ir gausime naują sąstatą iš *n* vagonų (5 pav.).



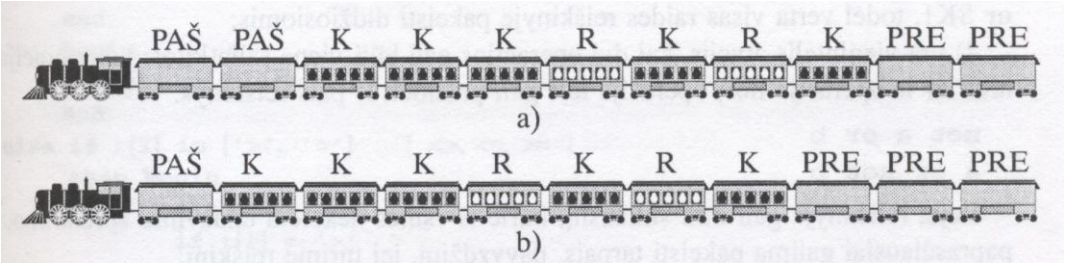

Taip gavome visas įmanomas kombinacijas, kurių antrasis vagonas yra *keleivinis*.

Taigi, $f(n) = f(n\text{-}1) + ?$;

Norint gauti visas kombinacijas, kur sąstato antrasis vagonas yra *restoranas*, reikia imti visus įmanomus sąstatus, sudarytus iš *n*-2 vagonų, prasidedančius *keleiviniu* vagonu. Tokių sąstatų yra $f(n\text{-}2)$. Imkime mūsų pavyzdį (6 pav.):



Prie jų prijungiame po du vagonus (7 pav.):




Taigi, $f(n) = f(n\text{-}1) + f(n\text{-}2)$. Kadangi $f(1) = f(2) = 1$, gauname, kad $f(n)$ lygus $n$-ąjąm Fibonačio skaičiui.

```
const MAX_N = 43; { maksimalus vagonų skaičius }
type lentelė = array [1..MAX_N] of longint;

function vagonai (n: longint): longint;

  procedure fibonači (n: longint; var fibo: lentelė);
  { sudaro fibonačio skaičių nuo 1 iki n lentelę }
    var i: longint;
  begin
    fibo[1] := 1; fibo[2] := 1;
    for i := 3 to n do
      fibo[i] := fibo[i-1] + fibo[i-2];
  end; { fibonači }

  var kiek: longint;
      fibo: lentelė;
  begin
  fibonači (n, fibo);
  kiek := 0;
  if n > 5 then kiek := kiek + fibo[n-5];
  if n > 4 then kiek := kiek + 2 * fibo[n-4];
  if n > 3 then kiek := kiek + 3 * fibo[n-3];
  if n > 2 then kiek := kiek + 3 * fibo[n-2];
  if n > 1 then kiek := kiek + 2 * fibo[n-1];
  kiek := kiek + fibo[n];
  vagonai := kiek
end; { vagonai }
```