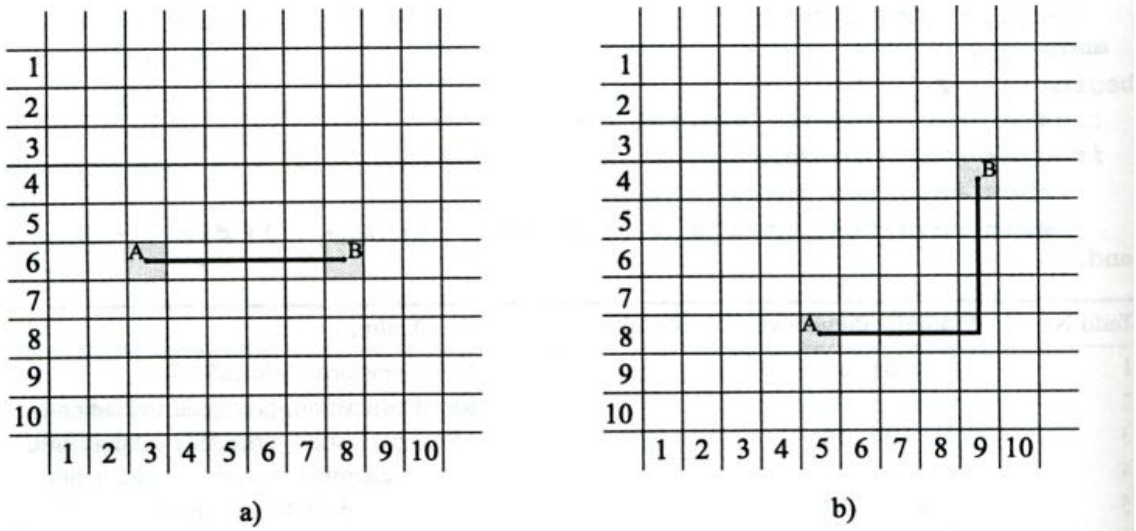
Paprastas kelias

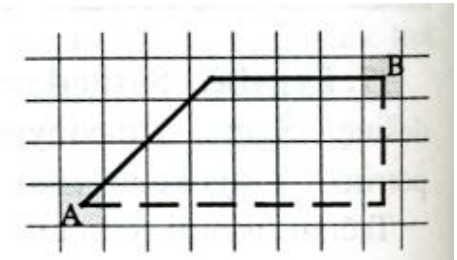
Prieš sudarydami šio uždavinio algoritmą pirmiausia šiek tiek pamastykime. Kaip eitume tarp dviejų lentos langelių? Jei eitume tik langeliais, kurie liečiasi kraštinėmis (t.y. eitume tik horizontaliai ar vertikaliai), tai trumpiausias kelias būtų dviejų atkarpų laužtė (atskiru atveju – viena, kai langeliai A ir B toje pačioje tiesėje).



18.1 pav.

Sąlygoje leidžiama eiti ir įstrižai (t. y. langeliais, kurių kampai liečiasi). Tuomet ankstesnis kelias nebus trumpiausias. Reikia prisiminti akivaizdžią taisyklę: kelias įžambine visuomet trumpesnis negu statiniais.

Taisyklę taikome kombinuotai: einame įžambine, kol susilyginame su langeliu B, likusią kelio dalį einame horizontaliai arba vertikaliai



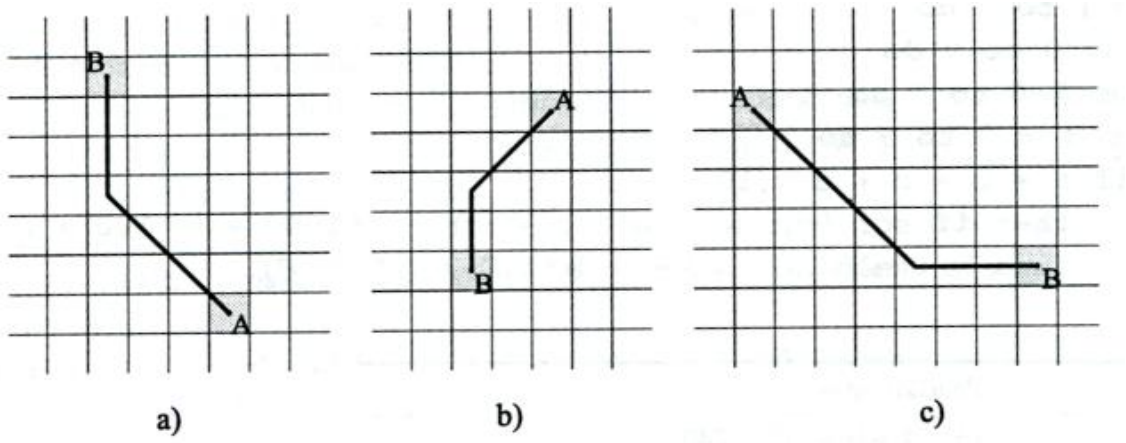
18.2 pav.

Visiškai nesvarbu, kur yra langelis B: prieš langelį A, po jo, virš ar žemiau jo. Visais atvejais elgiamės analogiškai: pirmiausia einame įstrižai, po to – tiesiai į B

Vadinasi, pirmiausia reikia rasti langelių A ir B koordinačių skirtumus (absoliutiniu dydžiu):

$dx = |xa - xb|$
$dy = |ya - yb|$

Mažesnysis iš dydžių *dx* ir *dy* rodo, kiek langelių reikės eiti įstrižai. Pastebėsime, kad kai langeliai A ir B yra viename stulpelyje arba vienoje eilutėje, tuomet vienas iš skaičių *dx* arba *dy* lygus 0, o tai reiškia, kad įstrižai nereikės eiti.



Paėjus įstrižai, atsiduriama langelyje, kuris yra vienoje tiesėje su langeliu B. Dabar iš didesniojo iš dydžių *dx* ir *dy* atėmus mažesnįjį sužinosime, kiek langelių teks eiti tiese.

Taigi trumpiausias kelias lygus:

min (*dx*, *dy*) + max (*dx*, *dy*) - min (*dx*, *dy*) + 1 = max (*dx*, *dy*) + 1

Vienetas pridedamas todėl, kad langelis A taip pat priklauso ieškomam keliui.

```
type langelis = record
                  x, y: 1..100
                end;

function kelias (A, B: langelis): integer;
  var dx, dy: integer;
begin
  dx := abs (A.x - B.x);
  dy := abs (A.y - B.y);
  if dx > dy
     then kelias := dx + 1
     else kelias := dy + 1
end;
```