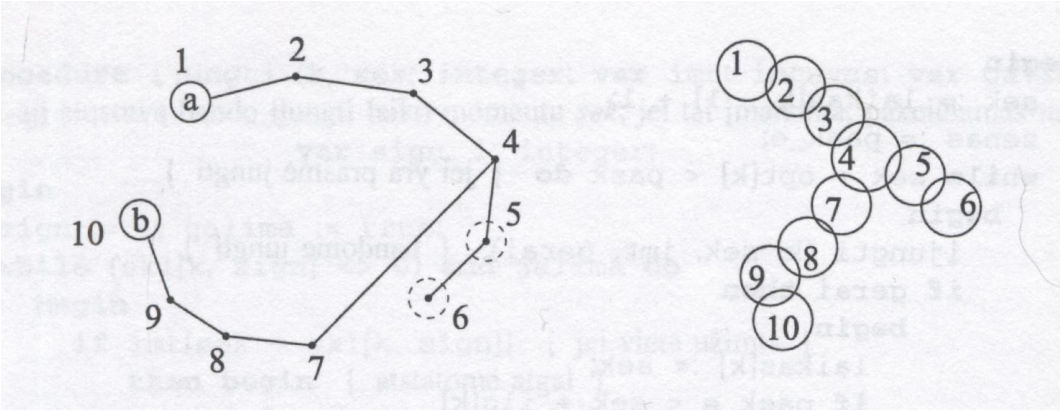
Sukabinti žiedai

Sumodeliuokime situaciją naudodamiesi grafais. Grafo viršūnėmis laikykime žiedus. Viršūnės *a* ir *b* sujungtos briauna, jei žiedai *a* ir *b* sukabinti tarpusavyje. Gavome neorientuotą grafą.

Imkime bet kurią vientisą, be išsišakojimų grandinę, jungiančią žiedus *a* ir *b*. Tai kelias grafe, jungiantis viršūnes *a* ir *b* ir neturintis ciklų. Jei kelio ilgis yra *l*, tai tame kelyje sutinkama *l*+1 viršūnė (žiedas). Jei iš viso turime *n* žiedų, galime pašalinti *n* - (*l*+1) žiedą.

Paveiksle nubrėžtas grafas, atitinkantis pradinių duomenų pavyzdį sąlygoje:



Sąlygą prašo rasti kiek *daugiausia* žiedų galima pašalinti. Kuo mažiau žiedų bus grandinėje, jungiančioje žiedus *a* ir *b*, tuo daugiau žiedų bus galima pašalinti.

Randame trumpiausio kelio, jungiančio žiedus *a* ir *b* ilgį *min_l*. Grandinėje, atitinkančioje šį kelią yra *min_l* + 1 žiedas. Vadinasi, daugiausia galime pašalinti *n* - *min_l* + 1 žiedą.

Trumpiausio kelio grafe tarp dviejų viršūnių galima ieškoti įvairiais būdais. Mes pasirinkome *Paieškos į plotį* algoritmą (žr. 83 užd).

Pats grafas uždavinyje vaizduojamas neįprastai – porų masyvų. Mat kvadratinė lentelė, aprašyta tokiu duomenų tipu:

```
type grafas = array [1..300, 1..300] of boolean;
```

užimtų apie 88K, tuo tarpu, kai Turbo Paskalis statiniams duomenims leidžia išskirti ne daugiau 64K.

```
const MAX_N = 300;   { maksimalus žiedų skaičius }
      MAX_K = 1000;  { maksimalus sukabinimų skaičius }
type  grafas = array [1..MAX_K] of record
                                     v1, v2: integer
                                   end;
      viršūnės = array [1..MAX_N] of
                         record
                           sp: (balta, pilka);
                           d: integer { trumpiausias atstumas }
                         end;          { iki a }

      eilė = array [1..MAX_N] of integer;
             { į eilę įtraukiami tik viršūnių numeriai }

procedure įtraukti_į_eilę (x: integer;  var e: eilė;
                           var uodega: integer);
{ įtraukia į eilės e galą viršūnę x }
begin
  e[uodega] := x;
  if uodega = MAX_N
     then uodega := 1
     else uodega := uodega + 1
end; { įtraukti_į_eilę }

procedure pašalinti_iš_eilės (var galva: integer);
{ pašalina iš eilės e pirmąją viršūnę }
begin
  if galva = MAX_N
     then galva := 1
     else galva := galva + 1
end; { pašalinti_iš_eilės }

function žiedai(a, b, k, n: integer; g: grafas): integer;
  var galva, uodega, { eilės atributai }
      u, v, i: integer;
      atr: viršūnės; { visų viršūnių atributai }
      e: eilė;       { viršūnių eilė }
begin
  { inicializuojame viršūnes }
  for i := 1 to n do
    begin
      atr[i].sp := balta;
      atr[i].d := maxint;
    end;
```

```
  atr[a].sp := pilka;    { žinome trumpiausią atstumą iki a }
  atr[a].d := 0;
  galva := 1;
  uodega := 1;
  įtraukti_į_eilę (a, e, uodega);
  while galva <> uodega do     { kol eilė netuščia }
    begin
      u := e[galva];
      for i := 1 to k do
        if (g[i].v1 = u) and (atr[g[i].v2].sp = balta) or
           (g[i].v2 = u) and (atr[g[i].v1].sp = balta)
           { randame dar nenagrinėtą viršūnę, gretimą viršūnei u }
           then begin
                  if g[i].v1 = u { imame reikiamą viršūnę iš poros }
                     then v := g[i].v2
                     else v := g[i].v1;
                  atr[v].sp := pilka;
                  atr[v].d := atr[u].d + 1;
                  įtraukti_į_eilę (v, e, uodega)
                end;
      pašalinti_iš_eilės (galva);
    end;
  { rasime pašalinamų žiedų skaičių }
  žiedai := n - (atr[b].d + 1)
end; { žiedai }
```