kako narišemo pravilni mnogokotnik ?

v. batagelj

UDK 681.3:513

FNT, VTO matematika in mehanika Ljubljana

Namen sestavka je,ob reševanju navidez enostavne naloge: narisati pravilni mnogokotnik z vsemi diagonalami, seznaniti bralca z nekaj osnovnimi značilnostmi uporabe risalne naprave (plotterja). V sestavku se srečamo tudi s teorijo grafov (Eulerjev izrek), z uporabo rekurzije pri programiranju in s prevedbo rekurzivne rešitve v iterativno.

HOW TO DRAW A REGULAR POLYGON ? - In the paper some basic characteristics of plotter-programming are illustrated by solving the problem: write a program for drawing the regular polygon with all diagonals. The use of recursion and the transformation of the recursive solution to an iterative one are also included.

Naloga:

> Sestavi program, ki bo na risalni napravi (plotterju) narisal pravilni mnogokotnik z vsemi diagonalami;

kateri je posvečen pričujoči sestavek, sodi v tisto področje programiranja, ki mu radi pravimo "igračkanje na računalniku"; nekateri pa celo: "kako lahko z računalnikom zapravljamo čas (in denar) ?!".

Toda to je le ena plat medalje. Poglejmo še za hip na drugo stran. Programiranje je precej razgibana dejavnost in zahteva od programerjev nenehno učenje. Kajti, če hočemo nekaj uporabljati, moramo to tudi dobro poznati in obvladati. In prav tu se rado pojavi "igračkanje", v katerem se prepletajo:

- preizkušanje in iskanje (za programerja) novih programirnih tehnik in konceptov;
- zanimivost in "čistost" naloge;
- kratkost rešitve (programa) - od nekaj 10 do nekaj 100 vrstic;

... seveda, nove tehnike in koncepte lahko ponavadi veliko hitreje in temeljiteje preizkusimo s posebej v ta namen pripravljenimi testnimi programi ... ali, če povzamemo za [8] : "Čemu zabresti v morje težav, ki se pojavijo pri računanju števila π , recimo, na milijon decimalnih mest natančno ? Nedvomno prvih sto decimalk, znanih že stoletja, zadostuje za reševanje kateregakoli realnega problema, kakršne srečamo v fiziki ali astronomiji. Čemu torej tolikšni napori za nepotrebno natančnost ?" ...

In vendar so računalniki mleli podobne probleme ure in ure. Omenimo le še, da je naš rojak baron Jurij Vega že leta 1794 (brez računalnika) izračunal π na 140 mest natančno.

Večina programerjev, ko se sreča z interaktivnim delom, piše programe za razne igrice; ko pa se srečamo z rekurzijo, skoraj ne moremo mimo naloge o osmih kraljicah [5, 6, 7, 8, 9, 10] .

Toliko v pojasnilo na morebitni ZAKAJ ?

S problemom uporabe risalne naprave sem se srečal, ko sem se odločil, da bom v CLUSE [1] vključil podprogram za risanje dendrogramov (dendrogram = grafični prikaz drevesa združevanja). Preden sem se spoprijel s tem problemom, sem se moral privaditi rutinam, ki omogočajo delo z risalno napravo. Slučajno sem tedaj v neki knjigi iz teorije grafov zagledal sliko polnega grafa (= pravilni mnogokotnik z vsemi diagonalami) in seveda sem se nemudoma vprašal: Ali bi ga znal narisati ?

Preden nadaljujemo, naj za bralca, ki ni domač

z risalno napravo, povem, da je osnovna akcija, ki jo omogočajo rutine za risanje, premik peresa po premici iz točke, v kateri se trenutno nahaja, v izbrano točko. Pri tem je pero ali spuščeno (= risanje) ali pa dvignjeno (= pomik brez risanja).

Za začetek našega ubadanja z nalogo poskusimo odgovoriti na vprašanje: Kako bi nalogo rešili sami z ravnilom, šestilom in kotomerom ?

Recimo, da želimo narisati pravilni n-kotnik. Tedaj narišemo najprej krožnico in jo z n točkami $T_1, T_2, \ldots, T_n$ (oglišča mnogokotnika) razdelimo na n enakih lokov. Nato povežemo vse točke eno z drugo z daljicami (diagonalami). To lahko povzamemo v prvo zamisel rešitve:

1. določi koordinate točk T_i , $i = 1,\ldots,n$
2. za vsak $(i,j) : 1 \leq i,j \leq n$ ponovi
2.1. poveži točki T_i in T_j z daljico

Ta rešitev vsebuje dva spodrsljaja:
- če je $i = j$, poskušamo z diagonalo povezati točko T_i samo s sabo;
- vsako diagonalo rišemo dvakrat,

ki pa ju zlahka odpravimo, če se odločimo, da mora biti indeks j vedno večji od indeksa i. Tako dobimo drugo zamisel:

1. določi koordinate točk T_i , $i = 1,\ldots,n$
2. za vsak $(i,j) : 1 \leq i < j \leq n$ ponovi
2.1. poveži točki T_i in T_j z daljico

Vrstica 2 v tej rešitvi nam pušča proste roke v izbiri vrstnega reda prebora množice $\{(i,j): 1 \leq i < j \leq n\}$. Ena izmed najpreprostejših določitev vrstnega reda je podana v naslednji zamisli rešitve:

1. določi koordinate točk T_i , $i = 1,\ldots,n$
2. za $i = 1,\ldots, n-1$ ponovi
   za $j = i+1,\ldots, n$ ponovi
2.1. prestavi dvignjeno pero v točko T_i
2.2. nariši daljico (od T_i) do točke T_j

Toda v tej rešitvi nismo niti poskusili upoštevati enc od osnovnih zahtev pri programiranju dela risalne naprave:

celotna pot, ki jo pri risanju opravi pero risalne naprave, naj bo kar se da kratka.

To je predvsem zahteva po (časovno) učinkoviti rabi risalne naprave.

Ali lahko pridemo do boljše rešitve ?

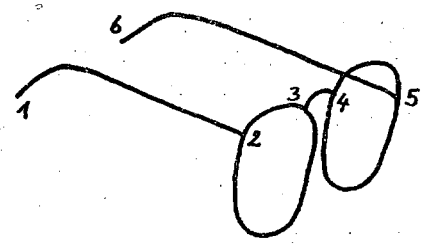
Dolžina vsake poti peresa, katere rezultat je neka izbrana risba, je navzdol omejena z vsoto dolžin dejansko narisanih črt na tej risbi. Potemtakem lahko dobimo boljšo rešitev le tako, da zmanjšamo skupno dolžino premikov dvignjenega peresa. Najugodneje je seveda, če znamo risbo narisati v "eni potezi" - ne da bi dvignili pero.

Tu nam priskoči na pomoč Eulerjev izrek iz teorije grafov [4] . Za razumevanje tega izreka pojasnimo najprej dva pojma: Rekli bomo, da je risba povezana, če lahko iz vsake točke risbe pridemo po risbi v vsako drugo. Točka risbe je liha natanko takrat, ko je ali prosto krajišče črte, ki pripada risbi, ali pa se v njej sreča liho število črt; drugače je soda.

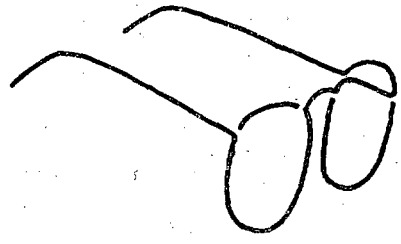
Sedaj že lahko prevedemo Eulerjev izrek v terminologijo našega problema:

Če je na povezani risbi $2k$ lihih točk (število lihih točk je vedno sodo), potem lahko risbo narišemo v k potezah. Vsaka od teh potez se začne in konča v lihi točki. Povezano risbo brez lihih točk lahko narišemo v eni potezi s sklenjeno črto (risanje končamo v točki, v kateri smo ga začeli).

Za primer si oglejmo risbo:



Na njej je šest lihih točk, ki so oštevilčene s številkami od 1 do 6 . Torej jo lahko narišemo v treh potezah. Na primer takole:



Povrnimo se na našo nalogo ! Ali lahko kak (vsak ?) n-kotnik zvsemi diagonalami narišemo v eni potezi ?

Ker so presečišča diagonal sode točke, nam lahko povzročajo težave le oglišča mnogokotnika. Vsako oglišče je povezano z diagonalami z vsemi ostalimi; to pa so tudi vse črte, ki se v njem stikajo. Zato bodo oglišča liha/soda, če bo n sodo/liho število. Torej lahko v eni potezi narišemo le lihe mnogokotnike (in 2-kotnik = daljica).

Toda, KAKO ?

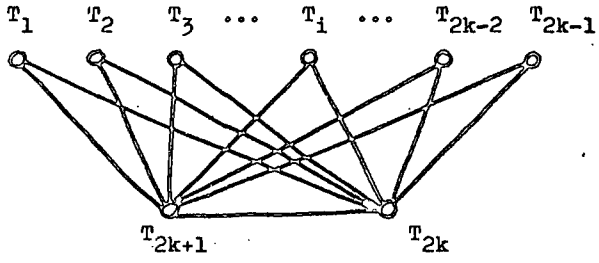
Recimo, da želimo narisati 2k+1 - kotnik. Kakor vemo, ga je mogoče narisati s sklenjeno črto v eni potezi. To velja tudi za 2k-1 - kotnik. Ali bi znali narisati 2k+1 - kotnik, če že vemo, kako narišemo 2k-1 kotnik ? Če to znamo, potem znamo narisati poljuben lihi mnogokotnik; saj, ker znamo narisati trikotnik, bi znali narisati tudi petkotnik; ker znamo tega, bi znali tudi sedemkotnik; ...

Poglejmo, kam nas pripelje ta zamisel postopka risanja 2k+1 - kotnika:

1. nariši 2k-1 - kotnik
2. nariši ostanek: diagonale, ki imajo eno krajišče v točki T_{2k} ali točki T_{2k+1}

Kako narisati ostanek ?

Tudi vse točke ostanka so sode. Torej ga lahko narišemo s sklenjeno črto v eni potezi.



Ponuja nam se nekaj "smiselnih" risanj ostanka; toda paziti moramo še, da bosta risanje 2k-1 - kotnika in risanje ostanka vsklajena: točka, v kateri začnemo (in zaradi sklenjenosti tudi končamo) risati 2k-1 - kotnik, mora biti začetek (in konec) risanja ostanka. Najbrž je najugodneje, če izberemo za to točko kar točko T_1. Tedaj lahko narišemo ostanek recimo takole (na začetku risanja se nahajamo v točki T_1):

```
2.    "nariši ostanek"
2.1.    za i = 2, 3, ..., 2k ponovi
2.1.1.      če je i sod potem
2.1.1.1.        nariši daljico do T2k+1
            drugače
2.1.1.2.        nariši daljico do T2k
2.1.2.      nariši daljico do Ti
2.2.    nariši daljico do T1
```

Če jezik, v katerega programiramo, pozna rekurzivne podprograme, smo s tem risanje lihega mnogokotnika že ugnali.

Recimo, da je dana procedura crtado(i) , ki nariše daljico iz točke, v kateri se pero trenutno nahaja, do oglišča T_i , potem bi risanje lihega mnogokotnika opisali v Pascalu z naslednjo rekurzivno proceduro:

```
procedure risilih( m: integer );
  { m - red mnogokotnika; liho število }
var  i: integer;
begin
    if  m > 1  then begin
        risilih( m-2 );
        for  i := 2  to  m-1  do begin
            crtado( m - i mod 2 ); crtado( i )
        end;
        crtado( 1 )
    end
end { risilih } ;
```

Kaj pa, če jezik, v katerega programiramo, ne pozna rekurzije ? Običajno se moramo tedaj zateči k uporabi sklada; vendar to v našem primeru ni potrebno, ker postopek vsebuje samo en rekurzivni klic. Take procedure ne zahtevajo "drevesnega mehanizma" in jih zato lahko pogosto brez posebnih težav prepišemo v iterativno obliko. Proceduro risilih bi tako zapisali v Structranu [3] :

```
SUBROUTINE RISI LIH ( M )
N = 3
WHILE ( N .LE. M ) DO
   NM = N - 1
   FOR ( I = 2, NM ) DO
      CALL CRTA DO ( N - MOD(I,2) )
      CALL CRTA DO ( I )
   ENDFOR
   CALL CRTA DO ( 1 )
   N = N + 2
ENDWHILE
RETURN
END
```

Tako, liho mnogokotnike znamo narisati v eni potezi. Kako pa je s sodimi ?

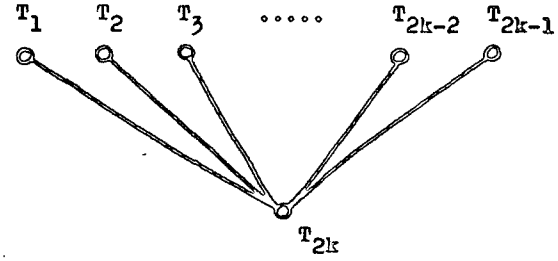
Recimo, da želimo narisati 2k - kotnik. Potem nam Eulerjev izrek pove, da bomo morali vsaj k krat pomakniti pero "po zraku" v drugo točko; pa tudi, da to zadostuje. Podobno kot prej, lahko problem razbijemo na dva podproblema:

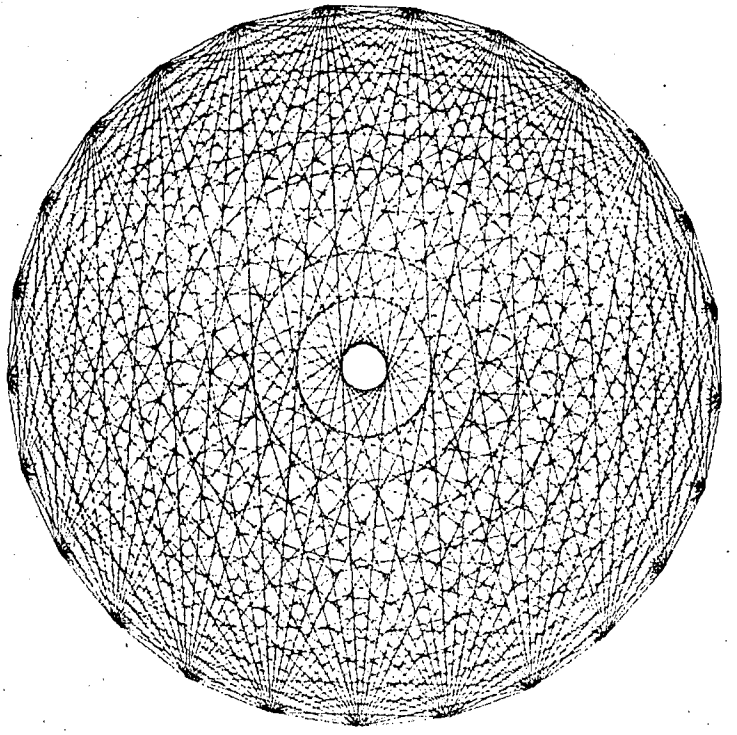
- nariši 2k-1 - kotnik

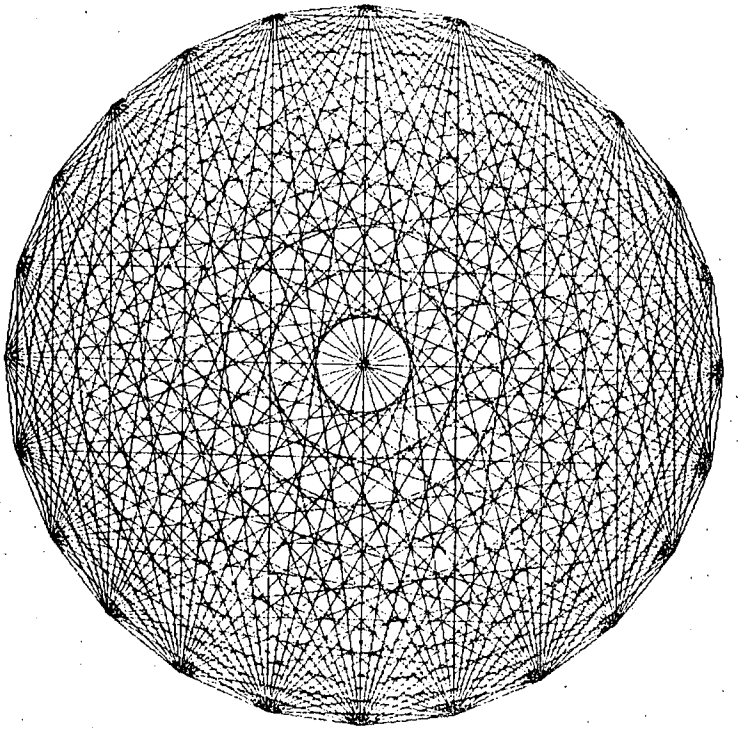
in

- nariši ostanek: diagonale, ki imajo eno krajišče v točki T_{2k} .

Prvi podproblem je naš stari znanec - risanje lihega mnogokotnika. Torej moramo premisliti le še risanje ostanka. Ta sedaj izgleda takole:







Ker mora biti risanje 2k-1 - kotnika in ostanka vsklajeno, je najenostavneje začeti risanje v točki T_1. Tako dobimo naslednji postopek:

1. pomakni se v točko T_1 in nariši 2k-1 - kotnik
2. za i = 1, 2, ..., k-1 ponovi
2.1. nariši daljico do T_{2k},
2.2. nariši daljico do T_{2i},
2.3. pomakni se v točko T_{2i+1}
3. nariši daljico do T_{2k}

S tem smo obdelali obe možnosti (lihe in sode mnogokotnike). Preostane nam le še, da ju združimo v enoten program za risanje poljubnega mnogokotnika:

```
      PROGRAM TETIVE( INPUT, TAPE14=INPUT )
      COMMON / TOCKE / X(50), Y(50)
      DATA   PI / 3.14159 26536 /
      CALL ODPRI

C PREBEREMO RED  M  IN POLMER  R  MNOGOKOTNIKA

      READ( 14, * ) M, R

C DOLOČIMO KOORDINATE OGLIŠČ MNOGOKOTNIKA

      FI = 2 * PI / M
      FOR ( I = 1, M ) DO
         X(I) = R * COS(I*FI)
         Y(I) = R * SIN(I*FI)
      ENDFOR

C NARIŠEMO MNOGOKOTNIK

      CALL POMIK V ( 1 )
      IF ( MOD(M,2) .EQ. 1 ) THEN
         CALL RISI LIH ( M )
      ELSE
         MM = M - 1
         CALL RISI LIH ( MM )
         FOR ( I = 2, MM, 2 ) DO
            CALL CRTA DO ( M )
            CALL CRTA DO ( I )
            CALL POMIK V ( I+1 )
         ENDFOR
         CALL CRTA DO ( M )
      ENDIF
      CALL ZAPRI
      END
```

V programu smo poleg podprograma risilih, ki ga že poznamo, uporabili nekaj podprogramov, ki omogočajo delo z risalno napravo. Ti so odvisni od osnovnih rutin za risanje, ki se od računalnika do računalnika spreminjajo. Za računalnik Cyber na RRC v Ljubljani izgledajo takole [11]:

```
      SUBROUTINE CRTA DO ( I )
      COMMON / TOCKE / X(50), Y(50)
      CALL PLINE ( X(I), Y(I), 1 )
      RETURN
      END

      SUBROUTINE POMIK V ( I )
      COMMON / TOCKE / X(50), Y( 50 )
      CALL PLINE ( X(I), Y(I), 0 )
      RETURN
      END

      SUBROUTINE ODPRI
      CALL POPEN ( 11, 6LTETIVE )
      CALL PSCALE ( 11, 0, 9, 11, 11,
                    -1, 1, -1, 1 )
      RETURN
      END

      SUBROUTINE ZAPRI
      CALL PCLOSE
      RETURN
      END
```

Opomba: Bralec, ki ima dostop do Cybera, lahko zahteva izpis priročnikov za delo z risalnimi rutinami in za Structran s Scope karticama:

```
BEGIN,MINI.
MANUAL,PLOTTER,STRUCTRAN.
```

Na koncu naj omenim soroden in nekoliko "uporabnejši" (vsaj za tiste, ki se ukvarjajo z grafi) problem, pri reševanju katerega bi nam nabrane izkušnje precej koristile. To je: sestavi program, ki bo na podoben način, kot smo ga uporabili pri risanju mnogokotnikov, narisal dani neusmerjeni graf, v katerem poljubni dve točki veže največ ena povezava. Nekaj koristnih idej v tej smeri je najti na zadnjih straneh sestavka [2]. Problem si lahko dodatno zabelimo z zahtevo, naj bodo točke grafa razvrščene po krožnici tako, da se bodo povezave sekale čim manjkrat. To pa je že zelo težek problem.

LITERATURA

[1] V.Batagelj: CLUSE - program za hierarhično razvrščanje v skupine; priročnik, Ljubljana, 1977

[2] V.Batagelj, T.Pisanski: Delno usmerjeni Eulerjevi grafi; (rokopis) seminar za računalniško in numerično matematiko, Ljubljana, 1977

[3] V.Batagelj, E.Zakrajšek: Structran, Informatica 75, Bled, 1975

[4] D.Cvetković, M.Milić: Teorija grafova i njene primene; Naučna knjiga, Beograd, 1977

[5] O.J.Dahl, E.W.Dijkstra, C.A.R.Hoare: Structured programming; Academic Press, London, New York, 1972

[6] F.Gruenberger, G.Jaffray: Problems for Computer Solution; John Wiley & Sons, New York, 1965

[7] R.B.Kieburtz: Structured programming and problem solving with PL/1; Prentice-Hall, Englewood Cliffs, New Yersey, 1977

[8] J.Nievergelt, J.C.Farrar, E.M.Reingold: Computer approaches to mathematical problems, Prentice-Hall, Eng. Cliffs, N.Yersey, 1974

[9] M.B.Wells: Elements of Combinatorial Computing; Pergamon Press, Oxford, 1971

[10] N.Wirth: Algorithms+Data Strctures=Programs; Prentice-Hall, Eng.Cliffs, N.Yersey, 1976

[11] E.Zakrajšek: Uporaba risalne naprave v RRC; priročnik, Ljubljana, 1975