Олександр Рудик

Ейлерові шляхи й цикли

Означення 1. *Ейлеровим шляхом у неорієнтованому графі називають довільний шлях, що проходить кожне ребро графа один раз. Якщо такий шлях замкнений, то його називають ейлеровим циклом.*

Теорема 1 (Ейлер, 1736). *Ейлерів шлях у зв'язаному графі існує тоді й лише тоді, коли він містить не більше, ніж дві вершини непарного степеня.*

Доведення. Доведемо спочатку *необхідність*. Якщо вершина, відмінна від початку та кінця ейлерового шляху, зустрічається у ньому j разів, то степінь цієї вершини дорівнює $2j$. Якщо початок і кінець ейлерового шляху збігаються, то степінь цієї вершини також парний.

Доведення *достатності* подамо алгоритмом знаходження ейлерового шляху. Не обмежуючи загальності міркувань, розглянемо зв'язний граф, у якому всі вершини мають парний степінь. Якщо це не так, долучимо до графа ще одну вершину і сполучимо її ребрами з двома вершинами, що мають непарний степінь.

1. Будуємо довільний цикл, вилучаючи з подальшого розгляду кожне пройдене ребро. У результаті степінь кожної вершини залишиться парним, хоча граф може втратити зв'язність.
2. Поки не вилучено усі ребра, робимо таке:
    - серед вершин попередньо побудованого циклу C вибираємо ту, що є кінцем невилученого ребра. Відсутність такої вершини означає, що всі ребра уже пройдено або граф не є зв'язним;
    - будуємо довільний цикл, що містить вибрану вершину, вилучаючи з подальшого розгляду кожне пройдене ребро. Розпочатий шлях завжди можна продовжити до циклу, бо парність степенів кожної вершини означає: якщо ми потрапили у якусь вершину, то також існує шлях з неї. У результаті степінь кожної вершини залишиться парним, а кількість зв'язних компонент графа може зрости;
    - з попередньо побудованого і новоствореного циклу утворюємо один, «розірвавши» кожний цикл у вибраній вершині та «склеївши» їх відповідним чином.

Зауваження 1. *Якщо у зв'язаному графі всі вершини мають парний степінь, то довільний ейлерів шлях є циклом.*

У програмній реалізації з динамічним розподілом пам'яті потрібно уважно й детально виписати алгоритм перетворення структури даних при вилученні ребер і склеюванні побудованих циклів. Подамо приклад програми

мовою Turbo Pascal 7.0 з коментарями щодо структури вхідних і вихідних файлів.

```
{$I+}                                {Верхня межа}
const nv_max=16000;                {кількості вершин}
type  pointe=^edge;
      edge=record                          {Ребро:}
      v1,v2: word;                {суміжні вершини}
      n1,n2,                       {наступні ребра}
      p1,p2: pointe end;          {попередні ребра,
       відповідно інцендентні вершинам v1, v2}
   pointer=array[0..nv_max] of pointe;
     words=array[0..nv_max] of word;
   pointv =^vertix;
     vertix=record                  {Вершина циклу:}
     v: word;                        {номер вершини}
     n: pointv end;                {наступна вершина}
                                     {Вказівники на:}
var n:^pointer;                   {інцендентне ребро}
    e: pointe;                   {новостворене ребро}
   ne:^words;              {к-сть інцендентних ребер}
   nv,                              {Кількість вершин}
    v,                                 {Номер вершини}
j1,j2,j3,j: word;              {Допоміжні лічильники}
         o: text;                        {Файл даних}
v0,v1,v2,u0,u1: pointv;              {Вершини циклів}
stop: boolean; {Умова припинення repeat until}

PROCEDURE circle;      {Побудова циклу} BEGIN
       new(v1);    v1   :=v0;
REPEAT new(v2);    v1^.n:=v2;    e:=n^[v1^.v];
 {Переадресація}    if e^.v1=v1^.v then begin
 v2^.v:=e^.v2;
 if e^.p1<>nil      then                 begin
 if e^.p1^.v1=v1^.v then e^.p1^.n1:=nil
                    else e^.p1^.n2:=nil   end;
 if e^.p2<>nil      then                 begin
 if e^.p2^.v2=v2^.v then e^.p2^.n2:=e^.n2
                    else e^.p2^.n1:=e^.n2 end;
 if e^.n2<>nil      then                 begin
 if e^.n2^.v2=v2^.v then e^.n2^.p2:=e^.p2
                    else e^.n2^.p1:=e^.p2 end
                                           end
                   {if e^.v1=v2^.v}else begin
 v2^.v:=e^.v1;
 if e^.p1<>nil      then                 begin
 if e^.p1^.v1=v2^.v then e^.p1^.n1:=e^.n1
                    else e^.p1^.n2:=e^.n1 end;
 if e^.n1<>nil      then                 begin
 if e^.n1^.v1=v2^.v then e^.n1^.p1:=e^.p1
                    else e^.n1^.p2:=e^.p1 end;
 if e^.p2<>nil      then                 begin
 if e^.p2^.v2=v1^.v then e^.p2^.n2:=e^.n2
```

```
                  else e^.p2^.n1:=e^.n2 end;
                                          end;
 dec(ne^[v1^.v]);      {Облік вилучення ребра}
 dec(ne^[v2^.v]);
 if   e^.v1=v1^.v   then begin
 if   n^[v2^.v] =e
 then n^[v2^.v]:=e^.p2;
      n^[v1^.v]:=e^.p1     end     else begin
                  if   n^[v2^.v] =e
                  then n^[v2^.v]:=e^.p1;
                       n^[v1^.v]:=e^.p2 end;
 dispose(e);           {Вивільнення пам'яті}
    stop:=(v2^.v = v0^.v);
 if stop then v1^.n:=v0      {Замикання циклу}
         else v1   :=v2;
UNTIL stop;
      stop:=false                          END;
                                        BEGIN
nv:=0;  new(n);  new(ne);
for v:=0 to nv_max do begin
              n^[v]:=nil;
             ne^[v]:=0 end;
assign(o,'EULER.DAT');   reset(o);
            {Зчитування рядків вхідного файлу,
    кожний з яких містить номери кінців ребра}
REPEAT readln(o,j1,j2);
  if j1>nv then nv:=j1;
  if j2>nv then nv:=j2;
  new(e);  e^.v1:=j1;  e^.n1:=nil;
           e^.v2:=j2;  e^.n2:=nil;
                {Опрацювання даних вершини j1}
  if n^[j1]=nil then e^.p1:=nil
                else              begin
    if n^[j1]^.v1=j1 then n^[j1]^.n1:=e
                     else n^[j1]^.n2:=e;
    e^.p1:=n^[j1]                   end;
                {Опрацювання даних вершини j2}
  if n^[j2]=nil then e^.p2:=nil
                else              begin
    if n^[j2]^.v1=j2 then n^[j2]^.n1:=e
                     else n^[j2]^.n2:=e;
    e^.p2:=n^[j2]                   end;
  inc(ne^[j1]);   n^[j1]:=e;
  inc(ne^[j2]);   n^[j2]:=e;
UNTIL seekeof(o);  close(o);

assign (o,'EULER.RES');   rewrite(o);
j1:=0;
j2:=0;             {Перевірка умови існування}
j3:=0;                      {Ейлерового шляху}
 v:=0;
repeat inc(v); if ne^[v] mod 2 = 1 then begin
       if j1=0 then j1:=v else
```

```
       if j2=0 then j2:=v else j3:=v      end
 until (j3>0) or (v=nv);
              {Якщо немає Ейлерового шляху...}
    if (j3>0) then                        begin
                  writeln(o,'No solution!');
                    close(o);   halt    end;

              {Якщо немає Ейлерового циклу...}
if j1>0 then                            begin
{Долучення вершини 0 і 2-х інциндентних ребер}
    new(e);  e^.v1:=j1;   e^.n1:=nil;
             e^.v2:=0;    e^.n2:=nil;
                {Опрацювання даних вершини 0}
     n^[0]:=e;
    e^.p1:=nil;
                {Опрацювання даних вершини j1}
    if n^[j1]^.v1=j1 then n^[j1]^.n1:=e
                     else n^[j1]^.n2:=e;
    e^.p2:=n^[j1];
    n^[j1]:=e;
    inc(ne^[j1]);

    new(e);  e^.v1:=0;    e^.n1:=nil;
             e^.v2:=j2;   e^.n2:=nil;
                {Опрацювання даних вершини 0}
    n^[0]^.n2:=e;
    e^.p1:=n^[0];
                {Опрацювання даних вершини j2}
    if n^[j2]^.v2=j2 then n^[j2]^.n2:=e
                     else n^[j2]^.n1:=e;
    e^.p2:=n^[j2];
    inc(ne^[j2]);   n^[j2]:=e;
    ne^[0]:=2;      n^[0] :=e         end;

new(v0);      {Вибір першої вершини циклу}
if j1>0 then v0^.v:=0
        else v0^.v:=1;

circle;               {Побудова першого циклу}

REPEAT                            {Пошук вершини,
              з якої виходять невилучні ребра}
  u0:=v0;           if ne^[v0^.v]=0 then begin
    v0:=v0^.n;
    while (ne^[v0^.v]=0) and (v0<>u0) do
                                 v0:=v0^.n   end;
  if ne^[v0^.v]=0 then stop:=true  else begin
      u0:=v0;
      new(v0);
      v0^.v:=u0^.v;
      circle;      {Побудова наступного циклу}
      u1:=v0^.n;          {"Склеювання" циклів}
          v0^.n:=u0^.n;
```

```
                u0^.n:=u1                end
UNTIL  stop;
    {Запис вершин у порядку обходу ейлерового
     шляху. Для циклу перша й остання вершини}
                                    {збігаються}
if j1>0 then while u0^.v<>0 do u0:=u0^.n;
                                u1:=u0;
if j1=0 then       write(o,u1^.v) else  begin
       u1:=u1^.n;  write(o,u1^.v)         end;
repeat u1:=u1^.n;  write(o,' ',u1^.v)
 until (u1=u0) or (j1>0) and (u1^.n^.v=0);
                   writeln(o);  close(o) END.
```