BAB V

K E S I M P U L A N

Graph adalah suatu model matematika sederhana yang sering digunakan untuk membantu dalam penyelesaian suatu persoalan. Sesudah suatu model matematika mewakili suatu persoalan, maka disusun suatu algoritma untuk menyelesaikan persoalan tersebut.

Suatu struktur data dapat digunakan untuk menyajikan suatu graph, oleh karena itu Tipe Data Abstrak Linked - List dapat digunakan untuk membantu dalam penyelesaian algoritma-algoritma teori graph.

Dengan berdasar pada Tipe Data Abstrak Linked - List yang telah tersusun, maka dapat dibuat suatu program komputer. Karena Tipe Data Abstrak Linked - List adalah tipe yang fleksibel, maka suatu persoalan akan dengan mudah dan cepat dapat diselesaikan dengan menggunakan Tipe Data Abstrak Linked - List yang ditunjang dengan program komputer.

LAMPIRAN 1

Procedure dan Function untuk menyajikan graph menggunakan Tipe Data Abstrak Linked-List.

```
Type   str4       = string[4];
       DafTitik   = ^Titik;
       DafGaris   = ^Garis;
       Titik      = record
                      NamaTitik   : char;
                      NextTitik   : DafTitik;
                      KeGaris     : DafGaris;
                    end;

       Garis      = record
                      NamaGaris   : string[4];
                      Panjang     : byte;
                      KeTitik     : DafTitik;
                      NextGaris   : DafGaris;
                    end;
```

```
Function TitikBaru(X : char) : DafTitik;
var Baru : DafTitik;
Begin
   new(Baru);
   with Baru^ do
   begin
      NamaTitik : X;
      KeGaris   : nil;
      NextTitik : nil;
   end;
   TitikBaru := Baru;
End;

Function GarisBaru(X:str4; Y:byte) : DafGaris;
var Baru : DafGaris;
Begin
   new(Baru);
   with Baru^ do
   begin
      NamaGaris : X;
      Panjang   : Y;
      KeTitik   : nil;
      NextGaris : nil;
   end;
   GarisBaru := Baru;
End;
```

```
Procedure BuatTitik(var GraphAwal : DafTitik,Z : byte);
var Bantu, Vertex : DafTitik;
    Jawab, A      : char;
    i             : byte;
Begin
   new(GraphAwal); GraphAwal := nil;
   write('Banyaknya vertex  : '); readln(Z);
   i := 1;
   while i <= Z do
   begin
      new(Bantu);
      write('Vertex  : ');readln(A);
      Bantu := TitikBaru(A);
      if GraphAwal = nil then
      begin
         GraphAwal := Bantu;
         Vertex    := Bantu;
      end
      else begin
         Vertex^.NextTitik := Bantu;
         Vertex := Bantu;
         if GraphAwal^.NextTitik = nil then
            GraphAwal^.NextTitik := Bantu;
      end;
      Inc(i);
   end;
End;

Procedure BuatData(var GraphAwal : DafTitik,Z : byte);
var Vertex, Bantu  : DafTitik;
    Edge, Bantu1   : DafGaris;
    Jawab, A       : char;
    B              : byte;
    C              : str4;
Begin
   Clrscr;
   BuatTitik(GraphAwal,Z);
   Vertex := GraphAwal;
   while Vertex <> nil do
   begin
      Clrscr;
      write('Vertex ',Vertex^.NamaTitik,' adjacent');
      write(' dengan vertex lain ? (Y/T) ');
      readln(Jawab);
      while upcase(Jawab) = 'Y'; do
      begin
         new(Bantu1);
         write('Dari vertex : ',Vertex^.NamaTitik);
         write(' ke vertex  : '); readln(A);
         write(' Nama garis : '); readln(C);
         write(' Panjang    : '); readln(B);
         Bantu1 := GarisBaru(C,B);
         Bantu  := GraphAwal;
         while (Bantu^.NamaTitik <> A) AND
               (Bantu <> nil) do
               Bantu := Bantu^.NextTitik;
```

```
            if Bantu <> nil then
            begin
               Bantu1^.KeTitik := Bantu;
               if Vertex^.KeGaris = nil then
               begin
                  Vertex^.KeGaris := Bantu1;
                  Edge := Bantu1;
                  if Vertex = GraphAwal then
                     GraphAwal^.KeGaris := Bantu1;
               end
               else begin
                  Edge^.NextGaris := Bantu1;
                  Edge := Bantu1;
               end;
            end;
            write('Masih ada edge lain ? (Y/T) ');
            readln(Jawab);
         end;
         Vertex := Vertex^.NextTitik;
      end;
   End;

   Procedure PrintGraph(GraphAwal : DafTitik);
   var Vertex : DafTitik;
      Edge   : DafGaris;
   Begin
      Clrscr;
      Vertex := GraphAwal;
      while Vertex <> nil do
      begin
         Edge := Vertex^.KeGaris;
         while Edge <> nil do
         begin
            write('Edge ',Edge^.NamaGaris);
            write(' menghubungkan vertex ');
            write(Vertex^.NamaTitik,' ke ');
            write(Edge^.KeTitik^.NamaTitik);
            write(' dengan panjang : ',Edge^.Panjang);
            Edge := Edge^.NextGaris;
         end;
         writeln;
         Vertex := Vertex^.NextTitik;
      end;
   End;
```

Penyajian graph tak berarah G pada **Gambar : 4.3.** menggunakan Tipe Data Abstrak Linked - List :

INPUT :

```
Banyaknya vertex : 5

Vertex : A
Vertex : B
Vertex : C
Vertex : D
Vertex : E

Vertex A adjacent dengan vertex lain ? (Y/T) Y
Dari vertex A ke vertex B
Panjang    : 2
Nama garis : e1
Masih ada edge dari vertex A ? (Y/T) Y

Dari vertex A ke vertex D
Panjang    : 5
Nama garis : e2
Masih ada edge dari vertex A ? (Y/T) T

Vertex B adjacent dengan vertex lain ? (Y/T) Y
Dari vertex B ke vertex A
Panjang    : 2
Nama garis : e1
Masih ada edge dari vertex B ? (Y/T) Y

Dari vertex B ke vertex C
Panjang    : 4
Nama garis : e3
Masih ada edge dari vertex B ? (Y/T) T

Vertex C adjacent dengan vertex lain ? (Y/T) Y
Dari vertex C ke vertex B
Panjang    : 4
Nama garis : e3
Masih ada edge dari vertex C ? (Y/T) Y

Dari vertex C ke vertex D
Panjang    : 4
Nama garis : e4
Masih ada edge dari vertex C ? (Y/T) Y

Dari vertex C ke vertex E
Panjang    : 6
Nama garis : e5
Masih ada edge dari vertex C ? (Y/T) T

Vertex D adjacent dengan vertex lain ? (Y/T) Y
Dari vertex D ke vertex A
Panjang    : 5
Nama garis : e2
Masih ada edge dari vertex D ? (Y/T) Y
```

```
Dari vertex D ke vertex C
Panjang     : 4
Nama garis : e4
Masih ada edge dari vertex D ? (Y/T) Y

Dari vertex D ke vertex E
Panjang     : 4
Nama garis : e6
Masih ada edge dari vertex D ? (Y/T) T

Vertex E adjacent dengan vertex lain ? (Y/T) Y
Dari vertex E ke vertex C
Panjang     : 6
Nama garis : e5
Masih ada edge dari vertex E ? (Y/T) Y

Dari vertex E ke vertex D
Panjang     : 4
Nama garis : e4
Masih ada edge dari vertex E ? (Y/T) T
```

OUTPUT :

```
Edge e1 menghubungkan vertex A ke B dengan panjang 2
Edge e2 menghubungkan vertex A ke D dengan panjang 5
Edge e1 menghubungkan vertex B ke A dengan panjang 2
Edge e3 menghubungkan vertex B ke C dengan panjang 4
Edge e3 menghubungkan vertex C ke B dengan panjang 4
Edge e4 menghubungkan vertex C ke D dengan panjang 4
Edge e5 menghubungkan vertex C ke E dengan panjang 6
Edge e2 menghubungkan vertex D ke A dengan panjang 5
Edge e4 menghubungkan vertex D ke C dengan panjang 4
Edge e6 menghubungkan vertex D ke E dengan panjang 4
Edge e5 menghubungkan vertex E ke C dengan panjang 6
Edge e6 menghubungkan vertex E ke D dengan panjang 4
```

LAMPIRAN 2

Procedure dan Function untuk mengetahui Komponen dan Keterhubungan suatu graph

```
Type    Derajat    = ^Maximal;
        Komponen   = ^Simpul;
        Maximal    = record
                        Nm         : char;
                        Adjacent   : Komponen;
                        Next       : Derajat;
                     end;
        Simpul     = record
                        Nama       : char;
                        Berikut    : Komponen;
                     end;

Function SudahAda(Awal : Derajat;
                  A  :  char)       :  Boolean;
var Ada      : Boolean;
    SBantu1 : Derajat;
    SBantu2 : Komponen;
Begin
   Ada := false;
   SBantu1 := Awal;
   while SBantu1 <>  nil do
   begin
     if SBantu1^.Nm = A then  Ada  :=  true
     else begin
        SBantu2  : SBantu1^.Adjacent;
        while SBantu2 <> nil do
        begin
           if SBantu2^.Nama = A then Ada := true;
           SBantu2 := SBantu2^.Berikut;
        end;
     end;
     SBantu1 := SBantu1^.Next;
   end;
   SudahAda := Ada;
End;

Function DerajatMax(GraphAwal : DafTitik;
                    Awal : Derajat)  :   DafTitik;
var MBantu, MBantu1 : DafTitik;
    MBantu2         : DafGaris;
    MAda            : Boolean;
    D, z            : byte;
Begin
  MBantu1 := GraphAwal;
  D := 0; z := 0;
  while MBantu1 <> nil do
  begin
    MAda := SudahAda(Awal,MBantu1^.NamaTitik);
    if MAda = false then
    begin
      MBantu2 := MBantu1^.KeGaris;
      while MBantu2 <> nil do
      begin
        z := z + 1;
```

```
          MBantu2 := MBantu2^.NextGaris;
        end;
        if z > D then
        begin
          D := z;
          MBantu := MBantu1;
        end;
        z := 0;
      end;
      MBantu1 := MBantu1^.NextTitik;
    end;
    DerajatMax := MBantu;
End;

Function KomponenBaru(A : char) : Derajat;
var KBantu : Derajat;
Begin
  new(KBantu);
  with KBantu^ do
  begin
    Nm       := A;
    Adjacent := nil;
    Next     := nil;
  end;
  KomponenBaru := KBantu;
End;

Function AnggotaBaru(A : char) : Komponen;
var ABantu  : Komponen;
Begin
  New(ABantu);
  with ABantu^ do
  begin
    Nama    := A;
    Berikut := nil;
  end;
  AnggotaBaru := ABantu;
End;

Procedure KompHub(GraphAwal : DafTitik);
var M, i, j                  : byte;
    Vertex, TitikDMax        : DafTitik;
    Awal,Kawan,Terakhir      : Derajat;
    Edge                     : DafGaris;
    Benar                    : Boolean;
    Anggota, Akhir, Bantu1   : Komponen;
Begin
  Clrscr;
  write('Banyaknya vertex  : ');readln(M);writeln;
  i := 0; j := 0;
  new(Awal); Awal := nil;

  Repeat
    j := j + 1;
    Vertex := DerajatMax(GraphAwal,Awal);
    TitikDMax := Vertex;
    Kawan := KomponenBaru(TitikDMax^.NamaTitik);
    if Awal = nil then
```

```
begin
  new(Awal);
  Awal := nil;
  Terakhir := Kawan;
  Awal := Kawan;
  i := i + 1;
end
else begin
  Terakhir^.Next := Kawan;
  Terakhir := Kawan;
  if Awal^.Next = nil then Awal^.Next := Kawan;
  i := i + 1;
end;
Edge := TitikDMax^.KeGaris;
While Edge <> nil do
begin
  Vertex := Edge^.KeTitik;
  Benar := SudahAda(Awal,Vertex^.NamaTitik);
  if Benar = false then
  begin
    Anggota := AnggotaBaru(Vertex^.NamaTitik);
    if Terakhir^.Adjacent = nil then
    begin
      Akhir := Anggota;
      Terakhir^.Adjacent := Akhir;
      if Awal^.Adjacent = nil then
         Awal^.Adjacent := Akhir;
      i := i + 1;
    end
    else begin
      Akhir^.Berikut := Anggota;
      Akhir := Anggota;
      i := i + 1;
    end;
  end;
  Edge := Edge^.NextGaris;
end;
Bantu1 := Terakhir^.Adjacent;
while Bantu1 <> nil do
begin
  Vertex := GraphAwal;
  while Vertex^.NamaTitik <> Bantu1^.Nama do
        Vertex  := Vertex^.NextTitik;
  Edge := Vertex^.KeGaris;
  while Edge <> nil do
  begin
    Benar := SudahAda(Awal,
                      Edge^.KeTitik^.NamaTitik);

if Benar = false then
begin
    Anggota := AnggotaBaru(Edge^.KeTitik^.NamaTitik);
    Akhir^.Berikut := Anggota;
    Akhir := Anggota;
    i := i + 1;
end;
Edge := Edge^.NextGaris;
end;
```

```
        Bantu1 := Bantu1^.Berikut;
      end;
  Until i = M;
  If j = 1 then
  begin
      write('Graph adalah graph terhubung ');
      writeln('yang terdiri dari vertex-vertex  :');
      write('-----------------------------------');
      writeln('-----------------------------------');
      Terakhir := Awal;
      write(Terakhir^.Nm);
      Akhir := Terakhir^.Adjacent;
      while Akhir <> nil do
      begin
          write('  -  ',Akhir^.Nama);
          Akhir := Akhir^.Berikut;
      end;
      writeln;
  end
  else begin
     write('Graph tidak terhubung dan ');
     writeln('terdiri dari ',j,' komponen.');
     write('---------------------------');
     writeln('---------------------------');
     i := 0;
     Terakhir := Awal;
     while Terakhir <> nil do
     begin
         i := i + 1;
         writeln;write('* Komponen ke - ',i);
         writeln(' terdiri dari vertex - vertex  :');
         write('-------------------------');
         writeln('-------------------------');
         write(Terakhir^.Nm);
         Akhir := Terakhir^.Adjacent;
         while Akhir <> nil do
         begin
             write('  -  ',Akhir^.Nama);
             Akhir := Akhir^.Berikut;
         end;
         writeln;writeln;
         Terakhir := Terakhir^.Next;
     end;
  end;
End;
```

Menentukan Komponen dan Keterhubungan graph tak berarah G pada **Gambar : 4.4.** menggunakan Tipe Data Abstrak Linked - List :

INPUT :

```
Banyaknya vertex : 9

Vertex : A
Vertex : B
Vertex : C
Vertex : D
Vertex : E
Vertex : F
Vertex : G
Vertex : H
Vertex : I

Vertex A adjacent dengan vertex lain ? (Y/T) Y
Dari vertex A ke vertex B
Panjang     : 1
Nama garis : e1
Masih ada edge dari vertex A ? (Y/T) Y

Dari vertex A ke vertex C
Panjang     : 1
Nama garis : e2
Masih ada edge dari vertex A ? (Y/T) Y

Dari vertex A ke vertex D
Panjang     : 1
Nama garis : e3
Masih ada edge dari vertex A ? (Y/T) T

Vertex B adjacent dengan vertex lain ? (Y/T) Y
Dari vertex B ke vertex A
Panjang     : 1
Nama garis : e1
Masih ada edge dari vertex B ? (Y/T) Y

Dari vertex B ke vertex C
Panjang     : 1
Nama garis : e4
Masih ada edge dari vertex B ? (Y/T) Y

Dari vertex B ke vertex E
Panjang     : 1
Nama garis : e5
Masih ada edge dari vertex A ? (Y/T) T

Vertex C adjacent dengan vertex lain ? (Y/T) Y
Dari vertex C ke vertex A
Panjang     : 1
Nama garis : e2
Masih ada edge dari vertex C ? (Y/T) Y
```

```
Dari vertex C ke vertex B
Panjang     : 1
Nama garis : e4
Masih ada edge dari vertex C ? (Y/T) Y

Dari vertex C ke vertex D
Panjang     : 1
Nama garis : e6
Masih ada edge dari vertex C ? (Y/T) Y

Dari vertex C ke vertex E
Panjang     : 1
Nama garis : e7
Masih ada edge dari vertex C ? (Y/T) T

Vertex D adjacent dengan vertex lain ? (Y/T) Y
Dari vertex D ke vertex A
Panjang     : 1
Nama garis : e3
Masih ada edge dari vertex D ? (Y/T) Y

Dari vertex D ke vertex C
Panjang     : 1
Nama garis : e6
Masih ada edge dari vertex D ? (Y/T) Y

Dari vertex D ke vertex E
Panjang     : 1
Nama garis : e8
Masih ada edge dari vertex D ? (Y/T) T

Vertex E adjacent dengan vertex lain ? (Y/T) Y
Dari vertex E ke vertex B
Panjang     : 1
Nama garis : e5
Masih ada edge dari vertex E ? (Y/T) Y

Dari vertex E ke vertex C
Panjang     : 1
Nama garis : e7
Masih ada edge dari vertex A ? (Y/T) Y

Dari vertex E ke vertex D
Panjang     : 1
Nama garis : e8
Masih ada edge dari vertex E ? (Y/T) T

Vertex F adjacent dengan vertex lain ? (Y/T) Y
Dari vertex F ke vertex G
Panjang     : 1
Nama garis : e9
Masih ada edge dari vertex F ? (Y/T) T

Vertex G adjacent dengan vertex lain ? (Y/T) Y
Dari vertex G ke vertex F
Panjang     : 1
Nama garis : e9
Masih ada edge dari vertex G ? (Y/T) Y
```

```
Dari vertex G ke vertex H
Panjang     : 1
Nama garis : e10
Masih ada edge dari vertex G ? (Y/T) Y

Dari vertex G ke vertex I
Panjang     : 1
Nama garis : e11
Masih ada edge dari vertex G ? (Y/T) T

Vertex H adjacent dengan vertex lain ? (Y/T) Y
Dari vertex H ke vertex G
Panjang     : 1
Nama garis : e10
Masih ada edge dari vertex H ? (Y/T) Y

Dari vertex H ke vertex I
Panjang     : 1
Nama garis : e12
Masih ada edge dari vertex H ? (Y/T) T

Vertex I adjacent dengan vertex lain ? (Y/T) Y
Dari vertex I ke vertex G
Panjang     : 1
Nama garis : e11
Masih ada edge dari vertex H ? (Y/T) Y

Dari vertex I ke vertex H
Panjang     : 1
Nama garis : e12
Masih ada edge dari vertex I ? (Y/T) T
```

OUTPUT :

```
Graph tidak terhubung dan terdiri dari 2 komponen
-------------------------------------------------

* Komponen ke - 1 terdiri dari vertex - vertex  :
-------------------------------------------------
C  -  A  -  B  -  D  -  E

* Komponen ke - 2 terdiri dari vertex - vertex  :
-------------------------------------------------
G  -  F  -  H  -  I
```

LAMPIRAN 3

Procedure dan Function untuk mengetahui lintasan terpendek menggunakan Tipe Data Abstrak Linked-List.

```
Type  List       = ^LabelMin;
      LabelMin   = record
                     Nama       : str4;
                     BesarLabel : byte;
                     Next       : List;
                   end;

Function BerLabel(HList : List;A : char) : Boolean;
var BBantu  : List;
    Sudah   : Boolean;
Begin
  Sudah := false; BBantu  :=  HList;
  while BBantu <> nil do
  begin
    if BBantu^.Nama = A then Sudah := true;
    BBantu := BBantu^.Next;
  end;
  BerLabel := Sudah;
End;

Function  ListBaru(A : char; B : byte)  :  List;
var LBantu  :  List;
Begin
  new(LBantu);
  with LBantu^ do
  begin
    Nama       := A;
    BesarLabel := B;
    Next       := nil;
  end;
  ListBaru := LBantu;
End;

Procedure LintasanTerpendek(GraphAwal : DafTitik;
                            Z         : byte);
var Start, Finish, Jawab                : char;
    HList, Akhir, Baru, Bantu1, Bantu2  : List;
    Vertex                              : DafTitik;
    Edge                                : DafGaris;
    L, M, X, Y, i                       : byte;
    Ada                                 : Boolean;
Begin
  WRITE('Mencari Lintasan Terpendek ? Y / T ');
  readln(Jawab);
  WHILE upcase(Jawab) <> 'T' do
  begin
   write('Awal Lintasan   :  ');readln(Start);
   write('Akhir Lintasan  :  ');readln(Finish);
   writeln;
```

```
HList := ListBaru(Start,X);
Akhir := HList;
i := 1;
repeat
  Bantu1 := HList;
  while Bantu1 <> nil do
  begin
    Vertex := GraphAwal;
    while (Vertex^.NamaTitik <> Bantu1^.Nama)
           AND (Vertex <> nil) do
           Vertex := Vertex^.NextTitik;
     if Vertex <> nil then
     begin
      Edge := Vertex^.KeGaris;
      while Edge <> nil do
      begin
       Ada := BerLabel(HList,Edge^.KeTitik^.NamaTitik);
       if Ada = false then
       begin
        new(Baru);
        X := Bantu1^.BesarLabel + Edge^.Panjang;
        Baru := ListBaru(Edge^.KeTitik^.NamaTitik,X);
        Akhir^.Next := Baru;
        Akhir := Baru;
        if HList^.Next = nil then HList^.Next := Baru;
       end
       else begin
         Bantu2 := HList;
         while Bantu2^.Nama<>Edge^.KeTitik^.NamaTitik do
               Bantu2 := Bantu2^.Next;
         L := Bantu2^.BesarLabel;
         M := Bantu1^.BesarLabel + Edge^.Panjang;
         if L > M then Bantu2^.BesarLabel := M;
       end;
       Edge := Edge^.NextGaris;
      end;
     end;
     Bantu1 := Bantu1^.Next;
  end;
  Inc(i);
until i > Z;
Bantu1 := HList;
while Bantu1^.Nama <> Finish do
      Bantu1 := Bantu1^.Next;
X := Bantu1^.BesarLabel;
Y := X;
write('Lintasan Terpendek dari ',Start,);
write(' hingga ',Finish,);
writeln(' ( jika ditarik mundur ) adalah : ');
while Start <> Finish do
begin
   write(Finish,'  -  ');
   Vertex := GraphAwal;
   while Vertex <> nil do
   begin
    Edge := Vertex^.KeGaris;
    while Edge <> nil do
    begin
```

```
            if Edge^.KeTitik^.NamaTitik = Finish then
            begin
              Bantu1 := HList;
              while Vertex^.NamaTitik <> Bantu1^.Nama do
                    Bantu1 := Bantu1^.Next;
              L := Edge^.Panjang;
              M := Bantu1^.BesarLabel;
              if M = X - L then
              begin
                write(Edge^.NamaGaris,' - ');
                Finish := Vertex^.NamaTitik;
                X := M;
                Vertex := nil;
                Edge := nil;
              end;
            end;
            if Edge <> nil then Edge := Edge^.NextGaris;
          end;
          if Vertex <> nil then
            Vertex := Vertex^.NextTitik;
        end;
    end;
    writeln(Finish);
    writeln('Dengan Panjang Lintasan : ',Y);
    writeln;
    WRITE('Masih akan mencari Lintasan Terpendek ? Y/T');
    readln(Jawab);
    Dispose(HList);
  end;
End;
```

Menentukan Lintasan Terpendek graph berarah G pada **Gambar : 4.5.** menggunakan Tipe Data Abstrak Linked-List :

INPUT :

```
Banyaknya vertex : 7

Vertex : S
Vertex : A
Vertex : B
Vertex : C
Vertex : D
Vertex : E
Vertex : F

Vertex S adjacent dengan vertex lain ? (Y/T) Y
Dari vertex S ke vertex A
Panjang    : 4
Nama garis : e1
Masih ada edge dari vertex S ? (Y/T) Y


Dari vertex S ke vertex B
Panjang    : 2
Nama garis : e2
```

```
Masih ada edge dari vertex S ? (Y/T) Y

Dari vertex S ke vertex C
Panjang     : 3
Nama garis : e3
Masih ada edge dari vertex S ? (Y/T) T

Vertex A adjacent dengan vertex lain ? (Y/T) Y
Dari vertex A ke vertex C
Panjang     : 1
Nama garis : e4
Masih ada edge dari vertex A ? (Y/T) Y

Dari vertex A ke vertex D
Panjang     : 6
Nama garis : e5
Masih ada edge dari vertex A ? (Y/T) T

Vertex B adjacent dengan vertex lain ? (Y/T) Y
Dari vertex B ke vertex E
Panjang     : 7
Nama garis : e6
Masih ada edge dari vertex B ? (Y/T) T

Vertex C adjacent dengan vertex lain ? (Y/T) Y
Dari vertex C ke vertex D
Panjang     : 6
Nama garis : e7
Masih ada edge dari vertex C ? (Y/T) Y

Dari vertex C ke vertex E
Panjang     : 8
Nama garis : e8
Masih ada edge dari vertex C ? (Y/T) Y

Dari vertex C ke vertex F
Panjang     : 12
Nama garis : e9
Masih ada edge dari vertex C ? (Y/T) T

Vertex D adjacent dengan vertex lain ? (Y/T) Y
Dari vertex D ke vertex F
Panjang     : 4
Nama garis : e10
Masih ada edge dari vertex D ? (Y/T) T

Vertex E adjacent dengan vertex lain ? (Y/T) Y
Dari vertex E ke vertex F
Panjang     : 3
Nama garis : e11
Masih ada edge dari vertex E ? (Y/T) T

Vertex F adjacent dengan vertex lain ? (Y/T) T
```

Mencari Lintasan Terpendek ? Y/T Y
Awal Lintasan : S
Akhir Lintasan : F

<u>OUTPUT</u> :

Lintasan Terpedek dari S hingga F (jika ditarik mundur) adalah :

F - e11 - E - e6 - B - e2 - S

Dengan Panjang Lintasan : 12

Masih akan mencari Lintasan Terpendek ? Y/T T

LAMPIRAN 4

Procedure dan Function untuk mengetahui Spanning Tree Minimal menggunakan Tipe Data Abstrak Linked-List.

```
Type  ListTree   = ^Terpilih;
      ListSisa   = ^TakTerpilih;
      Terpilih   = record
                      Nama    : str4;
                      Panjang : byte;
                      Lanjut  : ListTree;
                   end;
      TakTerpilh = record
                      Nama  : str4;
                      Next  : ListSisa;
                   end;

Function Uji1 (X : ListTree; A : str4) : Boolean;
Var Ada    : Boolean;
    Bantu  : ListTree;
Begin
    Ada := false;
    Bantu := X;
    while Bantu <> nil do
    begin
        if Bantu^.Nama = A then Ada := true;
        Bantu := Bantu^.Lanjut;
    end;
    Uji1 := Ada;
End;

Function Uji2 (Y : ListSisa; A : str4) : Boolean;
Var Ada    : Boolean;
    Bantu  : ListSisa;
Begin
    Ada := false;
    Bantu := Y;
    while Bantu <> nil do
    begin
        if Bantu^.Nama = A then Ada := true;
        Bantu := Bantu^.Next;
    end;
    Uji2 := Ada;
End;

Function PanjangMin (GraphAwal : DafTitik;
                     X : ListTree;
                     Y : ListSisa) : DafGaris;
Var Bantu          : DafTitik;
    Bantu1, Bantu2 : DafGaris;
    Ada1, Ada2     : Boolean;
    M              : byte;
Begin
    Bantu := GraphAwal;
    M := 255;
```

```
    begin
        Bantu1 := Bantu^.KeGaris;
        while Bantu1 <> nil do
        begin
            Ada1 := Uji1(X,Bantu1^.NamaGaris);
            Ada2 := Uji2(Y,Bantu1^.NamaGaris);
            if (Ada1 = false) AND (Ada2 = false) then
            begin
                if Bantu1^.Panjang < M then
                begin
                    Bantu2 := Bantu1;
                    M := Bantu1^.Panjang;
                end;
            end;
            Bantu1 := Bantu1^.NextGaris;
        end;
        Bantu := Bantu^.NextTitik;
    end;
    PanjangMin := Bantu2;
End;

Procedure SpanningTreeMinimal (GraphAwal : DafTitik);
Var AwalTree, AkhirTree, Bantu  : ListTree;
    AwalSisa, AkhirSisa, Bantu1 : ListSisa;
    Edge, Bantu2                : DafGaris;
    M, i                        : byte;
    Jawab                       : char;
Begin
    Clrscr;
    write('Banyaknya vertex  : ');readln(M);
    i := 0;
    new(AwalTree); AwalTree := nil;
    new(AwalSisa); AwalSisa := nil;
    Repeat
        Edge := PanjangMin(GraphAwal,AwalTree,AwalSisa);
        write(Edge^.NamaGaris,' terpilih ? (Y/T) ');
        readln(Jawab);
        if upcase(Jawab) = 'Y' then
        begin
            new(Bantu);
            with Bantu^ do
            begin
                Nama    := Edge^.NamaGaris;
                Panjang := Edge^.Panjang;
                Lanjut  := nil;
            end;
            if AwalTree = nil then
            begin
                AwalTree  := Bantu;
                AkhirTree := Bantu;
            end
            else begin
                AkhirTree^.Lanjut := Bantu;
                AkhirTree := Bantu;
                if AwalTree^.Lanjut = nil then
                    AwalTree^.Lanjut := Bantu;
            end;
```

```
                Inc(i);
            end
            else begin
                new(Bantu1);
                with Bantu1^ do
                begin
                    Nama := Edge^.NamaGaris;
                    Next := nil;
                end;
                if AwalSisa = nil then
                begin
                    AwalSisa  := Bantu1;
                    AkhirSisa := Bantu1;
                end
                else begin
                    AkhirSisa^.Next := Bantu1;
                    AkhirSisa := Bantu1;
                    if AwalSisa^.Next = nil then
                       AwalSisa^.Next := Bantu1;
                end;
            end;
        Until i = M - 1;
        M := 0;
        Clrscr;
        Bantu := AwalTree;
        write('Spanning Tree Minimal ');
        writeln(' dibentuk oleh edge - edge : ');
        write('-----------------------');
        writeln('----------------------------- ');
        while Bantu <> nil do
        begin
            write(Bantu^.Nama,' , ');
            M := M + Bantu^.Panjang;
            Bantu := Bantu^.Lanjut;
        end;
        writeln;writeln('Dengan panjang  : ',M);
End;
```

Menentukan Spanning Tree Minimal graph tak berarah G pada **Gambar : 4.6.** menggunakan Tipe Data Abstrak Linked-List :

INPUT :

```
Banyaknya vertex : 8

Vertex : A
Vertex : B
Vertex : C
Vertex : D
Vertex : E
Vertex : F
Vertex : G
Vertex : H

Vertex A adjacent dengan vertex lain ? (Y/T) Y
Dari vertex A ke vertex B
Panjang     : 6
```

```
Nama garis : e1
Masih ada edge dari vertex A ? (Y/T) Y

Dari vertex A ke vertex D
Panjang    : 5
Nama garis : e2
Masih ada edge dari vertex A ? (Y/T) Y

Dari vertex A ke vertex E
Panjang    : 4
Nama garis : e3
Masih ada edge dari vertex A ? (Y/T) T

Vertex B adjacent dengan vertex lain ? (Y/T) Y
Dari vertex B ke vertex A
Panjang    : 6
Nama garis : e1
Masih ada edge dari vertex B ? (Y/T) Y

Dari vertex B ke vertex C
Panjang    : 8
Nama garis : e4
Masih ada edge dari vertex B ? (Y/T) Y

Dari vertex B ke vertex F
Panjang    : 3
Nama garis : e5
Masih ada edge dari vertex B ? (Y/T) T

Vertex C adjacent dengan vertex lain ? (Y/T) Y
Dari vertex C ke vertex B
Panjang    : 8
Nama garis : e4
Masih ada edge dari vertex C ? (Y/T) Y

Dari vertex C ke vertex D
Panjang    : 7
Nama garis : e6
Masih ada edge dari vertex C ? (Y/T) Y

Dari vertex C ke vertex G
Panjang    : 5
Nama garis : e11
Masih ada edge dari vertex C ? (Y/T) T

Vertex D adjacent dengan vertex lain ? (Y/T) Y
Dari vertex D ke vertex A
Panjang    : 5
Nama garis : e2
Masih ada edge dari vertex D ? (Y/T) Y

Dari vertex D ke vertex C
Panjang    : 7
Nama garis : e6
Masih ada edge dari vertex D ? (Y/T) Y
Dari vertex D ke vertex H
Panjang    : 2
Nama garis : e7
Masih ada edge dari vertex D ? (Y/T) T
```

```
Vertex E adjacent dengan vertex lain ? (Y/T) Y
Dari vertex E ke vertex A
Panjang     : 4
Nama garis : e3
Masih ada edge dari vertex E ? (Y/T) Y

Dari vertex E ke vertex F
Panjang     : 9
Nama garis : e8
Masih ada edge dari vertex E ? (Y/T) Y

Dari vertex E ke vertex H
Panjang     : 6
Nama garis : e9
Masih ada edge dari vertex E ? (Y/T) T

Vertex F adjacent dengan vertex lain ? (Y/T) Y
Dari vertex F ke vertex B
Panjang     : 3
Nama garis : e5
Masih ada edge dari vertex F ? (Y/T) Y

Dari vertex F ke vertex E
Panjang     : 9
Nama garis : e8
Masih ada edge dari vertex F ? (Y/T) Y

Dari vertex F ke vertex G
Panjang     : 10
Nama garis : e10
Masih ada edge dari vertex F ? (Y/T) T

Vertex G adjacent dengan vertex lain ? (Y/T) Y
Dari vertex G ke vertex C
Panjang     : 5
Nama garis : e11
Masih ada edge dari vertex G ? (Y/T) Y

Dari vertex G ke vertex F
Panjang     : 10
Nama garis : e10
Masih ada edge dari vertex G ? (Y/T) Y

Dari vertex G ke vertex H
Panjang     : 8
Nama garis : e12
Masih ada edge dari vertex G ? (Y/T) T

Vertex H adjacent dengan vertex lain ? (Y/T) Y
Dari vertex H ke vertex D
Panjang     : 2
Nama garis : e7
Masih ada edge dari vertex H ? (Y/T) Y
Dari vertex H ke vertex E
Panjang     : 6
Nama garis : e9
Masih ada edge dari vertex H ? (Y/T) Y
```

```
Dari vertex H ke vertex G
Panjang     : 8
Nama garis : e12
Masih ada edge dari vertex H ? (Y/T) T


Edge e7 terpilih ? (Y/T) Y
Edge e5 terpilih ? (Y/T) Y
Edge e3 terpilih ? (Y/T) Y
Edge e2 terpilih ? (Y/T) Y
Edge e11 terpilih ? (Y/T) Y
Edge e1 terpilih ? (Y/T) Y
Edge e9 terpilih ? (Y/T) T
Edge e6 terpilih ? (Y/T) Y
```

OUTPUT :

```
Spanning Tree Minimal dibentuk oleh edge - edge :
-------------------------------------------------
e7  -  e5  -  e3  -  e2  -  e11  -  e1  -  e6

Dengan panjang  : 32
```