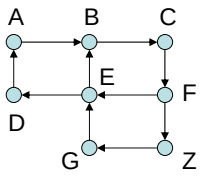
Übungen: Lösungen

1)



Betrachte jeweils ungerichteten/gerichteten Graphen:

- wieviele Kreise gibt es 6/4
- wieviele Wege von E nach Z gibt es 4/2
- wieviele Pfade von E nach Z gibt es 12/2
- Knotenreihenfolge bei Tiefensuche (Start in F) (nur für gerichteten Graphen; nimm an, dass alphabetische Ordnung existiert (z.B. A vor B)) F,E,B,C,D,A,Z,G
- Knotenreihenfolge bei Breitensuche (Start in F) (nur für gerichteten Graphen; nimm an, dass alphabetische Ordnung existiert (z.B. A vor B)) F,E,Z,B,D,G,C,A

Übungen: Lösungen

2)

```
function ist_zsh = test_zsh(adja)          % testet auf stark zusammenhängend

nodes=size(adja,1);
ist_zsh=1;

for i=1:nodes                              % teste für jeden Knoten ob alle Knoten erreichbar sind
   enthalten=zeros(1,nodes);
   folge=i;

   while(~isempty(folge))
      a=folge(1);
      folge(1)=[];                         %erstes Element loeschen
      neu=find(adja(a,:));                 %alle Knoten auf die a zeigt
      zuloeschen=[ ];
      for i=1:numel(neu)                   %numel: Anzahl der Elemenete in einem/r Vektor/Matrix
            if(enthalten(neu(i)))
                  zuloeschen=[zuloeschen i];   % schon dagewesen
            else
                  enthalten(neu(i))=1;
            end
      end
      neu(zuloeschen)=[];
      folge=[folge neu];                   %dranhaengen und weitersuchen
   end

   if(sum(enthalten)<nodes)
      ist_zsh=0;                           % Graph kann nicht zusammenhängend sein; verlasse …
      break;                               % … deshalb die Schleife
   end
end

if(ist_zsh)
            disp('Graph ist stark zusammenhaengend');
else
            disp('Graph ist nicht stark zusammenhaengend');
end
```

Übungen: Lösungen

3)

```
function reihenfolge = breitensuche(adja,s)

knoten=size(adja,1);
markierung=zeros(1,knoten);
markierung(s)=1;

reihenfolge=s;
naechster=1;

while(naechster<=length(reihenfolge))
   aktknoten=reihenfolge(naechster);
   nachfolger=find(adja(aktknoten,:));
   for i=1:length(nachfolger)
      nachfolgeridx=nachfolger(i);
      if(markierung(nachfolgeridx)==0)
         reihenfolge=[reihenfolge,nachfolgeridx];
         markierung(nachfolgeridx)=1;
      end
   end
   naechster=naechster+1;
end
```

Übungen: Lösungen

4)

```
function weglaenge = kuerzesterweg(adja,s)

knoten=size(adja,1);
weglaenge=ones(1,knoten)*Inf;
weglaenge(s)=0;

reihenfolge=s;
naechster=1;

while(naechster<=length(reihenfolge))
   aktknoten=reihenfolge(naechster);
   nachfolger=find(adja(aktknoten,:));
   for i=1:length(nachfolger)
      nachfolgeridx=nachfolger(i);
      if(weglaenge(nachfolgeridx)==Inf)
         reihenfolge=[reihenfolge,nachfolgeridx];
         weglaenge(nachfolgeridx)=weglaenge(aktknoten)+1;
      end
   end
   naechster=naechster+1;
end
```

Übungen: Lösungen

5)

```
function weglaengen = kuerzestewege(adja)

knoten=size(adja,1);
weglaengen=ones(knoten,knoten)*Inf;

for k=1:knoten
   reihenfolge=k;
   naechster=1;
   weglaengen(k,k)=0;

   while(naechster<=length(reihenfolge))
      aktknoten=reihenfolge(naechster);
      nachfolger=find(adja(aktknoten,:));
      for i=1:length(nachfolger)
         nachfolgeridx=nachfolger(i);
         if(weglaengen(k,nachfolgeridx)==Inf)
            reihenfolge=[reihenfolge,nachfolgeridx];
            weglaengen(k,nachfolgeridx)=weglaengen(k,aktknoten)+1;
         end
      end
      naechster=naechster+1;
   end
end
```

Übungen: Lösungen

6)

```
function reihenfolge = tiefensuche(adja,reihenfolge)

neighbors=find(adja(reihenfolge(end),:));
for i=1:length(neighbors)
        if(~any(reihenfolge==neighbors(i)))
              reihenfolge(end+1)=neighbors(i);
              reihenfolge=tiefensuche(adja,reihenfolge);%rekursiv!
        end
end
```