

Föreläsning 4 i ADK20

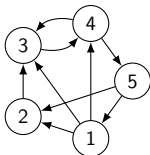
Grafer: djupetförstsökning, breddenförstsökning

Stefan Nilsson, Viggo Kann

KTH

Representation av graf

Riktad graf:



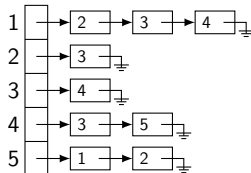
**Som kantmatrix
(incidence matrix):**

	1	2	3	4	5
1	-1	-1	-1	-1	1
2	1	-1			1
3		1	1	-1	1
4			1	-1	1
5				1	-1

**Som grannmatrix
(adjacency matrix):**

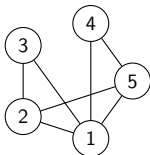
	1	2	3	4	5
1		1	1	1	
2			1		
3				1	
4				1	1
5	1	1			

**Som grannlista
(adjacency list):**



Representation av graf

Oriktad graf:



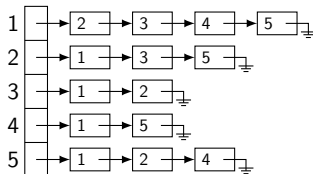
Som grannmatrix
(adjacency matrix):

	1	2	3	4	5
1		1	1	1	1
2			1		
3				1	
4					1
5					

Som kantmatrix
(incidence matrix):

1	1	1	1		1
2	1	1			1
3		1	1		
4				1	1
5				1	1

Som grannlista
(adjacency list):



Breddenförstsökning går igenom alla hörn som kan nås från ett speciellt starthörn s i följande ordning:

- Först alla grannar till s
- Sedan grannarna till grannarna till s
- Sedan alla hörn på avstånd 3 från s
- Sedan alla hörn på avstånd 4
- o.s.v.

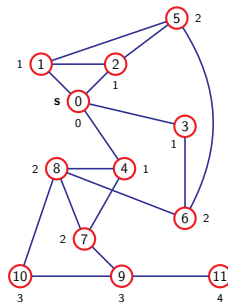
Breddenförstsökning i graf

```
function BFS( $V$ ,  $E$ ,  $s$ )  
  for varje hörn  $u \in V$  do  
     $d[u] \leftarrow \infty$   
   $d[s] \leftarrow 0$   
   $Q \leftarrow \{s\}$   
  while  $Q \neq \emptyset$  do  
     $u \leftarrow \text{DEQUEUE}(Q)$   
    for varje granne  $v$  till  $u$  do  
      if  $d[v] = \infty$  then  
         $d[v] \leftarrow d[u] + 1$   
         $\text{ENQUEUE}(Q, v)$ 
```

Om något ska göras med varje hörn i grafen kan det göras med u här i algoritmen.

Tidskomplexitet: $\mathcal{O}(|V| + |E|)$

Exempel:

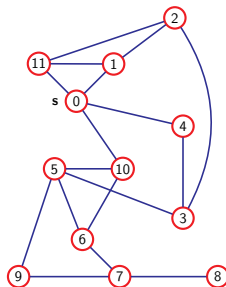


- Djupetförstsökning (DFS) börjar liksom BFS i starthörnet s , men går sedan så lång det går i grafen (utan att besöka något redan tidigare besökt hörn)
- När det inte går längre backar man tillbaka ett steg i taget tills det går att fortsätta framåt igen
- Detta implementeras enklast rekursivt

Djupetförstsökning i graf

```
function DFS(V,E,s)
|   for varje hörn  $u \in V$  do
|   |   color[u]  $\leftarrow$  white
|   DFS_VISIT(V,E,s)
function DFS_VISIT(V,E,u)
|   color[u]  $\leftarrow$  black
|   Gör något med u här
|   for varje granne v till u do
|   |   if color[v] = white then
|   |   |   DFS_VISIT(V,E,v)
```

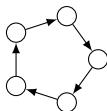
Exempel:



Tidskomplexitet: $\mathcal{O}(|V| + |E|)$

Avgör om en graf är en DAG

- Enkel tillämpning av sökning
- **DAG** = Riktad Acyklisk Graf
- **Cykel** = Stig av riktade kanter som börjar och slutar i samma hörn



Idé: Utvidga DFS så att man upptäcker kanter till förfäder (back edges)

Avgör om en graf är en DAG

```
function DFS(V,E,s)
|   for varje hörn  $u \in V$  do
|   |   color[u]  $\leftarrow$  white
|   for varje hörn  $u \in V$  do
|   |   if color[u] = white then
|   |   |   DFS_VISIT(u)
function DFS_VISIT(u)
|   color[u]  $\leftarrow$  gray
|   for varje granne v till u do
|   |   if color[v] = grey then
|   |   |   write "Cykel"
|   |   if color[v] = white then
|   |   |   DFS_VISIT(v)
|   color[u]  $\leftarrow$  black
```

Tidskomplexitet: $\mathcal{O}(|V| + |E|)$