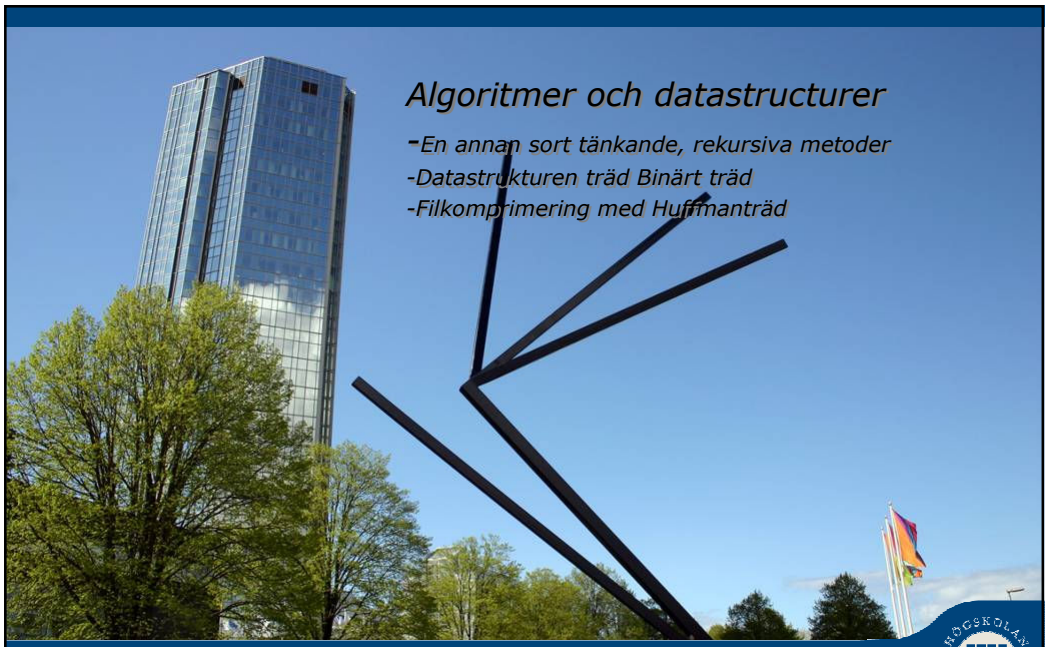


Rekursion ?

- En rekursiv metod är en metod som anropar sig själv.
- Liknar loopar / iterationer
- Mycket kraftfull men något förvirrande...
- Rekursion är ett sätt att lösa ett problem genom att dela problemmet i flera identiska men mindre subproblem, dvs. metoden anropar sig själv med ” ett mindre argument”.

1*2*3*4*……(n-1) * n fakultet(n)

fakultet(n -1) * n

Implementation

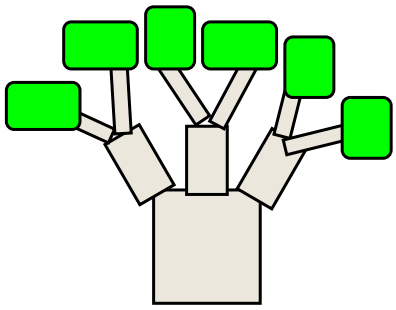
```
public static int fak(int n)
{
  if(n==0)
    return 1;

  else
    return  fak (n-1) * n;
}
```

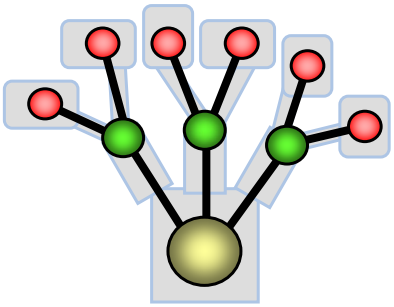
!! **basfall**

Se till att den rekursiva anropet alltid leder till **basfallet**

Vad är ett träd datastruktur? En abstaction som beskriver ...ja..just det...



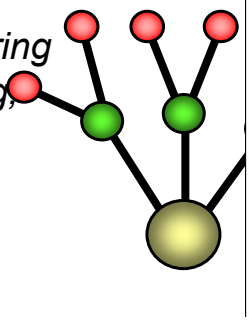
Datoranpassad träd...

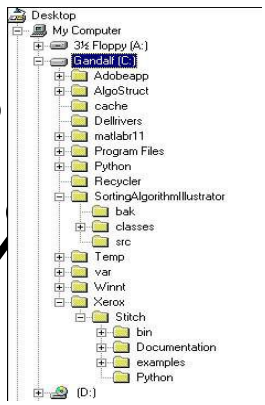


För utveckling av verksamhet, produkter och livskvalitet.

Var finns träd?

Fil / Katalog strukturer!
Sökning och sortering
Filkomprimenring,
Kryptering
Kompilatorer

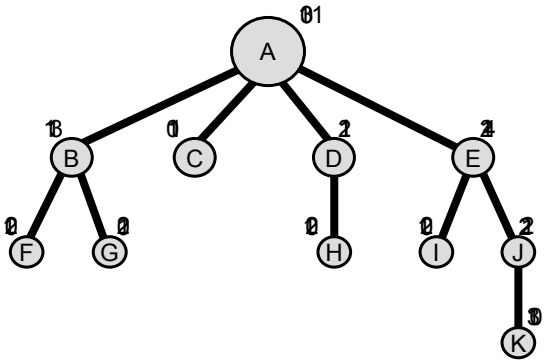




För utveckling av verksamhet, produkter och livskvalitet.

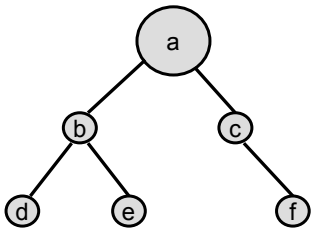
Några begrepp

- Rot
- Löv
- Djup
- Höjd
- Storlek

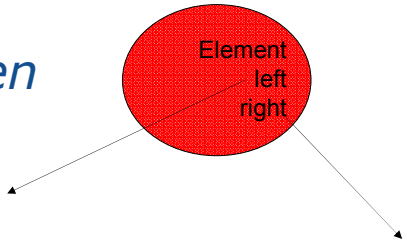


Binära träd

- Maximalt två barn / nod
  - Vänster / höger barn



Implementation- noden




```
class BinaryNode{
  private Object element;
  private BinaryNode left;
  private BinaryNode right;

public BinaryNode( Object theElement, BinaryNode lt, BinaryNode rt){
  element=theElement;
  left=lt;
  right=rt;
}
  public BinaryNode(){
  this( null,null,null); }
}
```


Implementation - metoder

- public Object getElement()
- public BinaryNode getLeft()
- public void printPreorder()
- public void printPostOrder()
- public void printInOrder();
- public static int size(BinaryNode n)
- public static int height(BinaryNode n)
- ……

Metoden –size()

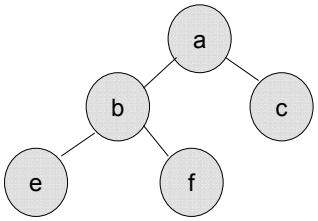
```
public static int size(BinaryNode n)
{
  if (n==null)

    return 0;
  else
    return 1+ size(n.left)+size(n.right);

}
```


Metoden printPreorder()

```
public void printPreorder(){
 System.out.println(element);
if(left!=null)
  left.printPreorder();
if(rigth!=null)
  right.printPreorder();
}
```

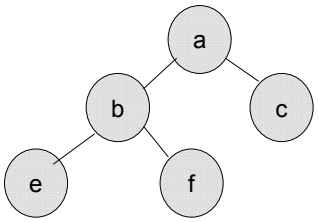


a b e f c

Metoden printPostOrder()

```
public void printPostOrder(){
if(left!=null)
  left.printPostOrder();
if(rigth!=null)
  right.printPostOrder();
System.out.println(element);

}
```



e f b c a

BinaryTree klassen

```
class BinaryTree{
    private BinaryNod root;

    public BinaryTree( Object item){
  root = new BinaryNode(item,null,null);
    }

    public BinaryTree(){
        root = null;
    }
}
```

BinaryTree-metoden merge()

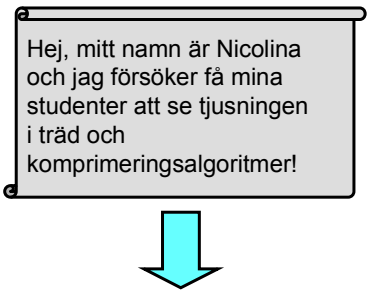
```
public void merge( Object item, BinaryTree t1, BinaryTree t2)
{
// Testa om inte t1 och t2 är samma träd
 if( t1.root== t2.root && t1.root!=null)
//  rapportera fel
 root = new BinaryNode( item, t1.root,t2.root);

if(this!=t1)
  t1.root=null;
 if(this!=t2)
  t2.root=null;

}
```

För utveckling av verksamhet, produkter och livskvalitet.

Datakomprimering - Huffman



01001000 01100101 01101010 00101100 00100000 01101101 01101001
01110100 01110100 00100000 01101110 01100001 01101101 01101111
00100000 10000100 ..

För utveckling av verksamhet, produkter och livskvalitet.

Datakomprimering - Huffman

01001000 01100101 01101010 00101100 00100000 01101101 01101001
01110100 01110100 00100000 01101110 01100001 01101101 01101111
00100000 10000100 ..

101 01 011 11 1101 10 001
1110 111 01 100 101 011 1100
010 10 ...

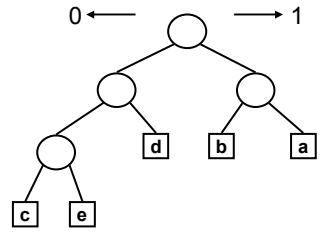
101 01 011 11 1101 10 001 1110 111 01 100 101 011 1100 010 10 ....

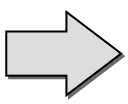
För utveckling av verksamhet, produkter och livskvalitet.

Datakomprimering – Huffman

(bokstäver för förekommer ofta hamnar högt upp i trädet och de som förekommer sällan hamnar längst ner i trädet)

a	01001101	25 ggr
b	01101101	31 ggr
c	00110101	5 ggr
d	10101001	19 ggr
e	01010111	14 ggr





a	=	11
b	=	10
c	=	000
e	=	001
d	=	01

För utveckling av verksamhet, produkter och livskvalitet.