University of Southern Denmark • Syddansk Universitet

DM503

Forelæsning 1

Indhold

- Introduktion til kurset
- Repetition af DM502
  - Rekursion
    - Fakultet $n!$
    - Fibonaccitallene
    - Hanois tårne

Introduktion

- Forelæser
  - Peter Schneider-Kamp (petersk@imada.sdu.dk)
- Instruktorer
  - M1/S1: Alessandro Maddaloni (maddaloni@imada.sdu.dk)
  - S7: Philipp Peters (phpeters@imada.sdu.dk)
- Hjemmeside
  - http://www.imada.sdu.dk/~petersk/DM503/
  - Kan også findes på BlackBoard

Introduktion

- 3 typer undervisning
  - Forelæsninger - ca. 2 gange om ugen
    - Slides vil være tilgængelige før forelæsningen
  - Labøvelser - 3 gange i løbet af kurset
    - Få hjælp til at programmere foran skærmen
    - Foregår i IMADAs terminalrum
  - Eksaminatorietimer - 1 gang om ugen
    - Blanding af teori og programmering
    - Klasseundervisning med tavle
    - Suppleret med praktiske øvelser
    - Ringe udbytte uden forberedelse

Introduktion

- For information om kuset, se hjemmesiden!
  - Information på hjemmesiden har præcedens
  - Fx er tid og sted for øvelsesholdene som angivet på hjemmesiden og IKKE som angivet på fakultetets hjemmeside
    - Kan i øvrigt ændre sig løbende
- Ugesedler udkommer hver uge
  - Indeholder information til ugen efter
  - Hvad der bliver gennemgået til forelæsningerne
  - Hvad der skal læses
  - Opgaverne til øvelserne og eksaminatorietimerne

Introduktion

- Litteratur
  - "Det samme som i DM502"
  - Lewis & Loftus: Java Software Solutions - Foundations of Program Design, 6. udgave
    - 5. udgave kan også bruges, men på eget ansvar
  - Niels Kjeldsen & Jacob Aae Mikkelsen: Noter og opgaver
- Derudover diverse noter
  - Allerede flere tilgængelig på hjemmesiden

Introduktion

- Evaluering
  - Ét projekt der er delt i to uafhænige dele
  - 1. delprojekt
    - Udleveres torsdag i næste uge
    - Afleveres torsdag den 2. december
  - 2. delprojekt
    - Udleveres torsdag den 9. december
    - Afleveres mandag den 10. januar
- Bedømmelse - B/IB
  - Begge delprojekter skal godkendes
  - Kun mulighed for genaflevering af 1. delprojekt

Introduktion

- Dit ansvar
  - At holde dig opdateret vha. hjemmesiden
  - Aflevere og afhente opgaver

Rekursion

Rekursion

- Prøv at definere "en liste af tal" uden at bruge ordet "liste"
- Svært, ik?
- "En liste af tal er en lis... øhh... sekvens af tal?"
  - "Sekvens" er bare et andet ord for "liste"
- Man behøver måske ikke undgå at bruge ordet "liste"

Rekursion

Rekursion

- Hvad med?

Rekursion

- Hvad med?
  - "En liste af tal er enten

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller
    - et tal efterfulgt af en liste af tal"

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller
    - et tal efterfulgt af en liste af tal"
- Lister af tal

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller
    - et tal efterfulgt af en liste af tal"
- Lister af tal
  - 42

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller
    - et tal efterfulgt af en liste af tal"
- Lister af tal
  - 42
  - Et tal

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller
    - et tal efterfulgt af en liste af tal"
- Lister af tal
  - 42
    - Et tal
  - 7, 9, 13

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller
    - et tal efterfulgt af en liste af tal"
- Lister af tal
  - 42
    - Et tal
  - 7, 9, 13
    - et tal (7) efterfulgt af en liste af tal (9, 13)

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller
    - et tal efterfulgt af en liste af tal"
- Lister af tal
  - 42
    - Et tal
  - 7, 9, 13
    - et tal (7) efterfulgt af en liste af tal (9, 13)
    - ...

Rekursion

- Hvad med?
  - "En liste af tal er enten
    - et tal, eller
    - et tal efterfulgt af en liste af tal"
- Lister af tal
  - 42
    - Et tal
  - 7, 9, 13
    - et tal (7) efterfulgt af en liste af tal (9, 13)
    - ...
    - Et tal (7) efterfulgt af et tal (9) efterfulgt af et tal (13)

Rekursion

- En liste af tal kan altså defineres i termer af sig selv
  - Definitionen kaldes rekursiv
  - En liste af tal er defineret rekursivt
- Rekursion
  - Når noget er defineret ved brug af sig selv

Rekursion

- Vi kender funktioner der kalder andre funktioner
  - Fx `main`-metoden der kalder `power(x, y)`
  - Osv...

Power eksempel

```
public class PowerExample {
    public static void main( String[] args ) {
        int result;
        int a, b;

        a = 2;
        b = 4;
        result = power( a, b );
        System.out.println( result );

        result = power( b, 0 );
        System.out.println( result );

        System.out.println( power( 1, 2 ) );
    }

    public static int power( int x, int y ) {
        int result = 1;
        int i;

        for( i = 1; i <= y; ++i ) {
            result = result * x;
        }

        return result;
    }
}
```

Rekursion

- Rekursion i programmering
  - Når en funktion kalder sig selv
  - Vi skal se tre eksempler på anvendelse af rekursion
- Fakultet
  - $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$
  - $n! = n \cdot (n-1)!$
- Fibonacci-tallene
  - f(5) = 5
  - f(n) = f(n-1) + f(n-2)
- Hanois tårne

University of Southern Denmark • Syddansk Universitet

Fakultet

Fakultet

- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$

Fakultet

- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$
- Skrives normalt som:
  $n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$

Fakultet

- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$
- Skrives normalt som:
  $n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$
- Fakultet kan også skrives som:
  $n! = n \cdot (n-1)!$

Fakultet

- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$
- Skrives normalt som:
  $n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$
- Fakultet kan også skrives som:
  $n! = n \cdot (n-1)!$
- Bemærk at definitionen er rekursiv

Fakultet

- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$
- Skrives normalt som:
  $n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$
- Fakultet kan også skrives som:
  $n! = n \cdot (n-1)!$
- Bemærk at definitionen er rekursiv
- Vi har defineret fakultet i termer af sig selv

Fakultet

- Et første forsøg

```
public class Factorial {
    public static void main( String[] args ) {
        System.out.println( "5! = " + fac(5) );
    }

    public static int fac( int n ) {
        int n1 = fac( n-1 );
        return n * n1;
    }
}
```

Fakultet

- Et første forsøg

```
public class Factorial {
    public static void main( String[] args ) {
        System.out.println( "5! = " + fac(5) );
    }

    public static int fac( int n ) {
        int n1 = fac( n-1 );
        return n * n1;
    }
}
```

```
Terminal — bash — 66×8
mac-sdu-00001:10 petersk$ javac Factorial.java
mac-sdu-00001:10 petersk$ java Factorial
Exception in thread "main" java.lang.StackOverflowError
        at Factorial.fac(Factorial.java:12)
        at Factorial.fac(Factorial.java:12)
        at Factorial.fac(Factorial.java:12)
        at Factorial.fac(Factorial.java:12)
        at Factorial.fac(Factorial.java:12)
        at Factorial.fac(Factorial.java:12)
```

University of Southern Denmark • Syddansk Universitet

Fakultet

Fakultet

- Hvad er der galt!?!

Fakultet

- Hvad er der galt!?!
- $fac(5) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0 \cdot -1 \cdot -2 \cdot -3 \cdot \ldots$

Fakultet

- Hvad er der galt!?!
- $fac(5) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0 \cdot -1 \cdot -2 \cdot -3 \cdot \ldots$
- Rekursionen stopper aldrig

Fakultet

- Hvad er der galt!?!
- $fac(5) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0 \cdot -1 \cdot -2 \cdot -3 \cdot \ldots$
- Rekursionen stopper aldrig
- Basistilfælde

Fakultet

- Hvad er der galt!?!
- $fac(5) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0 \cdot -1 \cdot -2 \cdot -3 \cdot \ldots$
- Rekursionen stopper aldrig
- Basistilfælde
  - $1! = 1$

Fakultet

- Et nyt forsøg

```
public class Factorial {
    public static void main( String[] args ) {
        System.out.println( "5! = " + fac(5) );
    }

    public static int fac( int n ) {
        int n1;
        if( n == 1 ) {
            return 1;
        } else {
            n1 = fac( n-1 );
            return n * n1;
        }
    }
}
```

Fakultet

- Et nyt forsøg

```
public class Factorial {
    public static void main( String[] args ) {
        System.out.println( "5! = " + fac(5) );
    }

    public static int fac( int n ) {
        int n1;
        if( n == 1 ) {
            return 1;
        } else {
            n1 = fac( n-1 );
            return n * n1;
        }
    }
}
```

```
Terminal — bash — 66×5
mac-sdu-00001:10 petersk$ javac Factorial.java
mac-sdu-00001:10 petersk$ java Factorial
5! = 120
mac-sdu-00001:10 petersk$
```

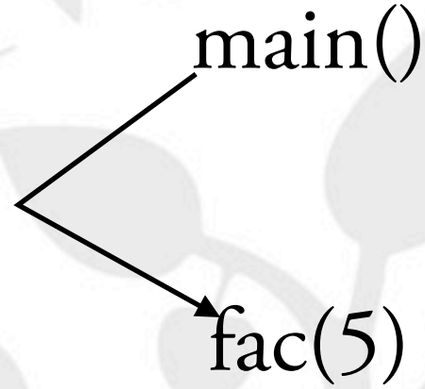
University of Southern Denmark • Syddansk Universitet

Fakultet

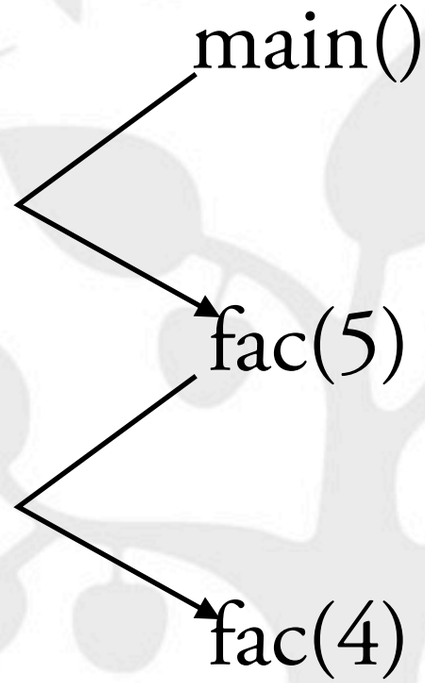
Fakultet

main()

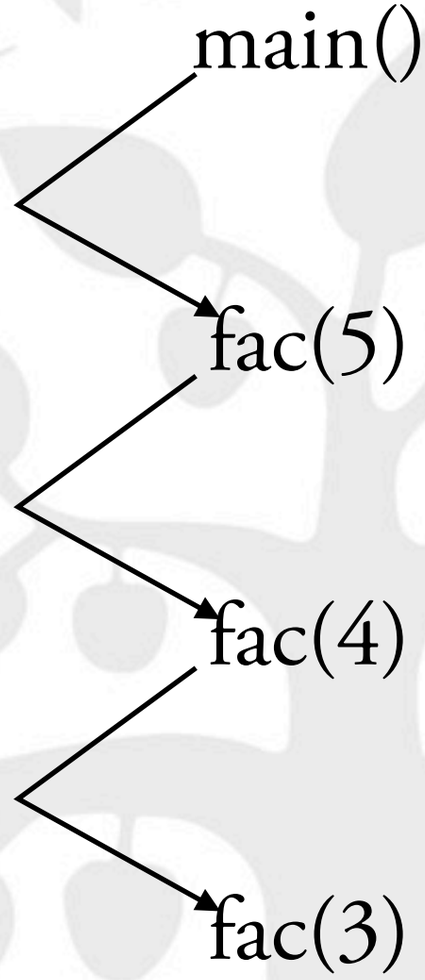
Fakultet



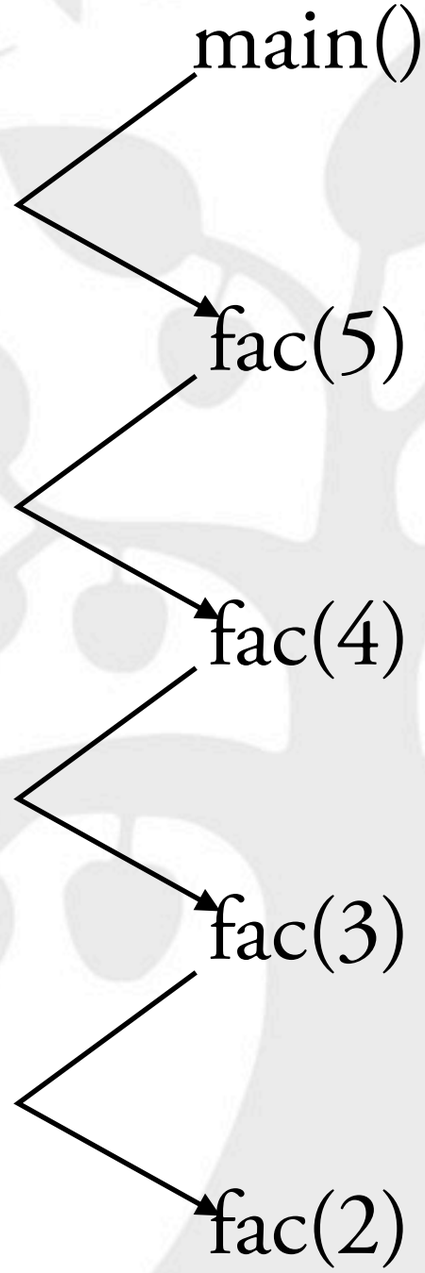
Fakultet



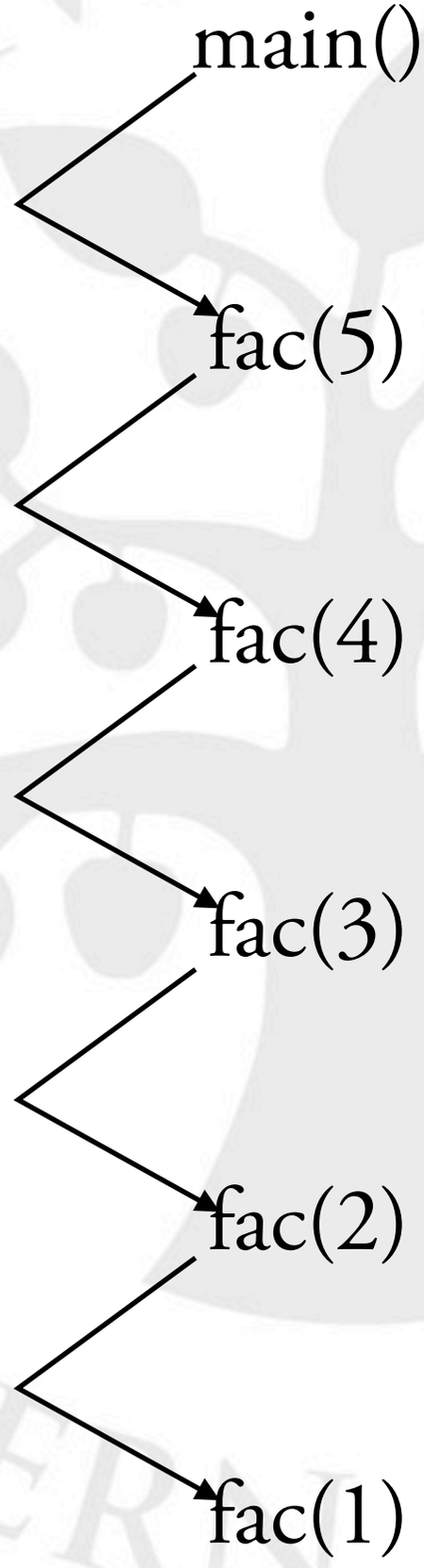
Fakultet



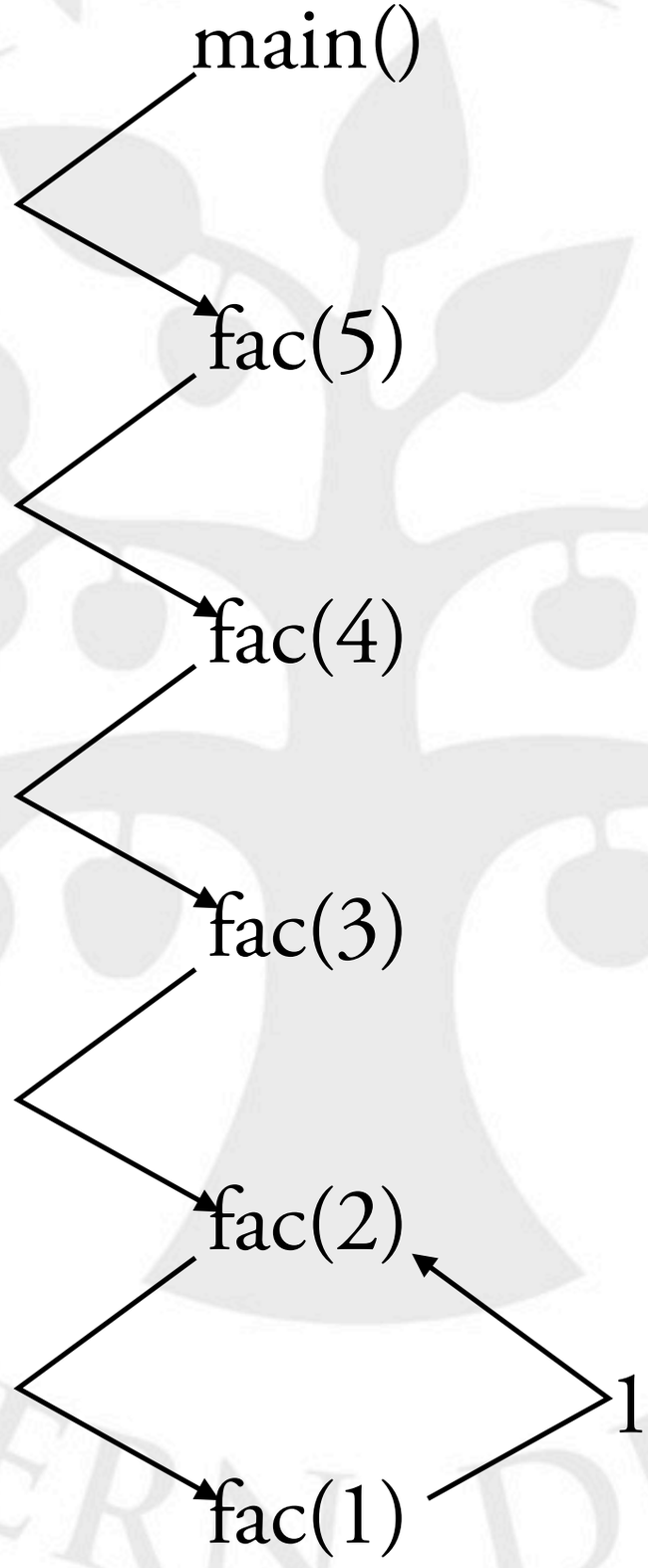
Fakultet



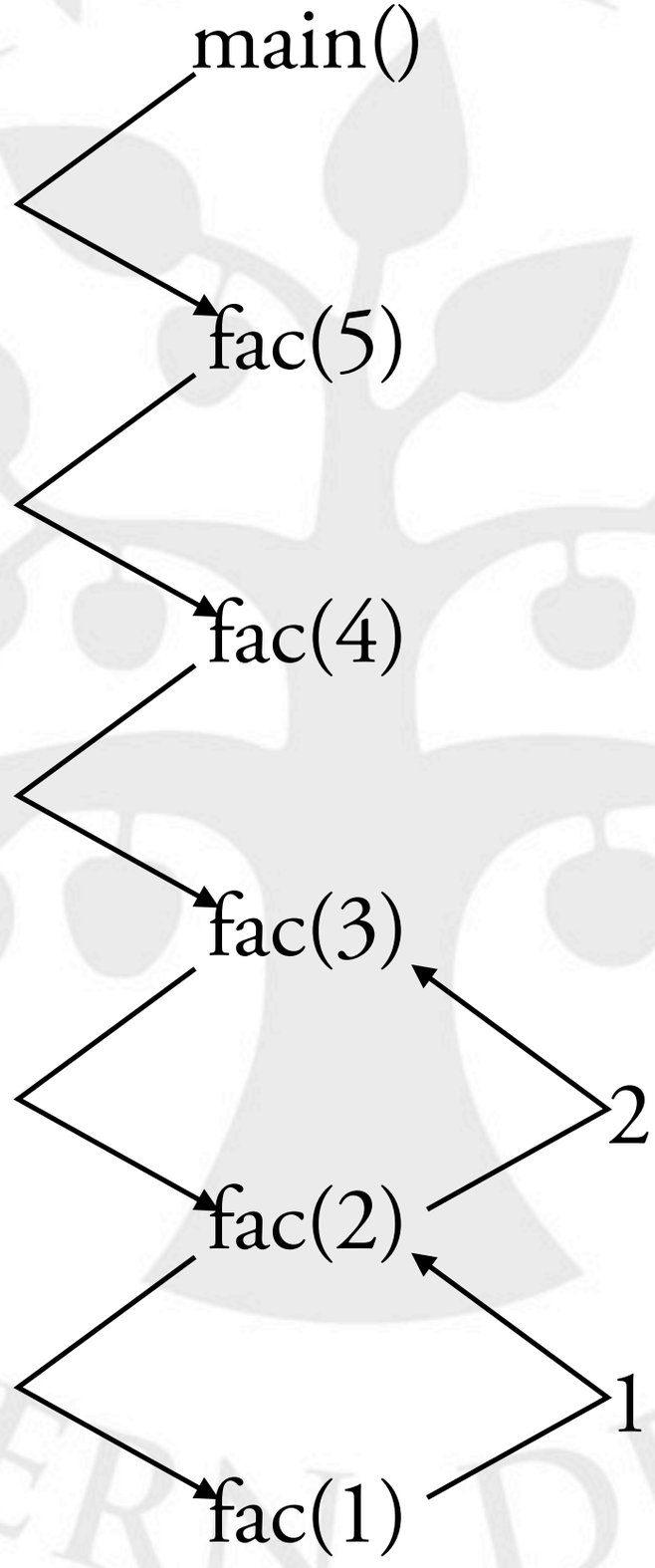
Fakultet



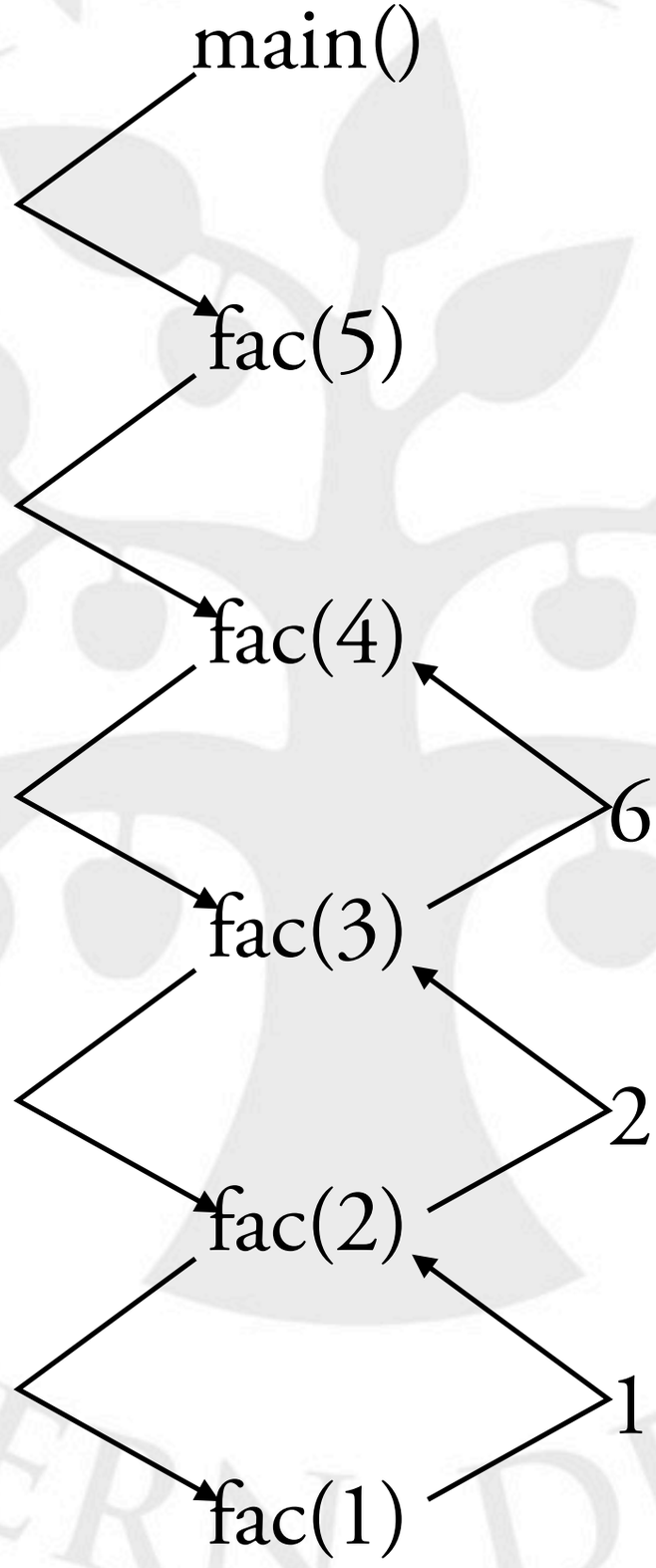
Fakultet



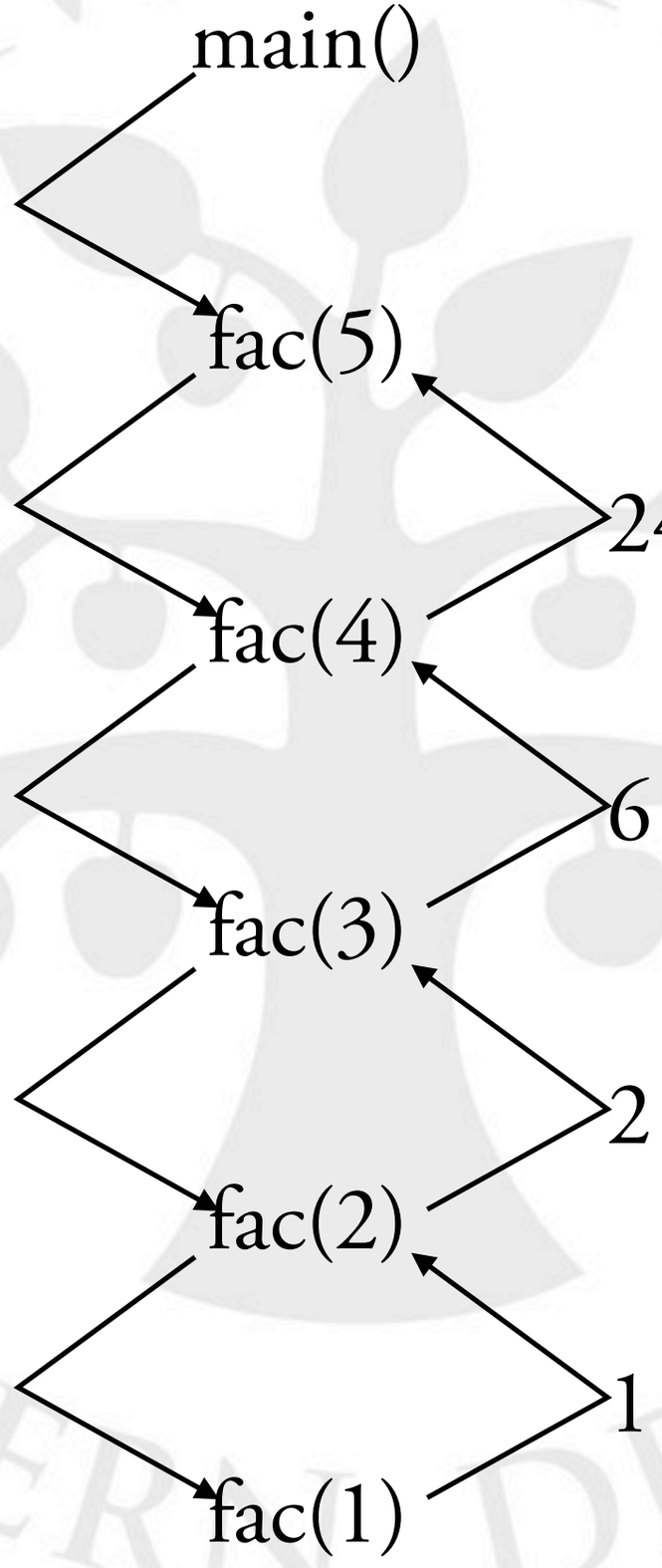
Fakultet



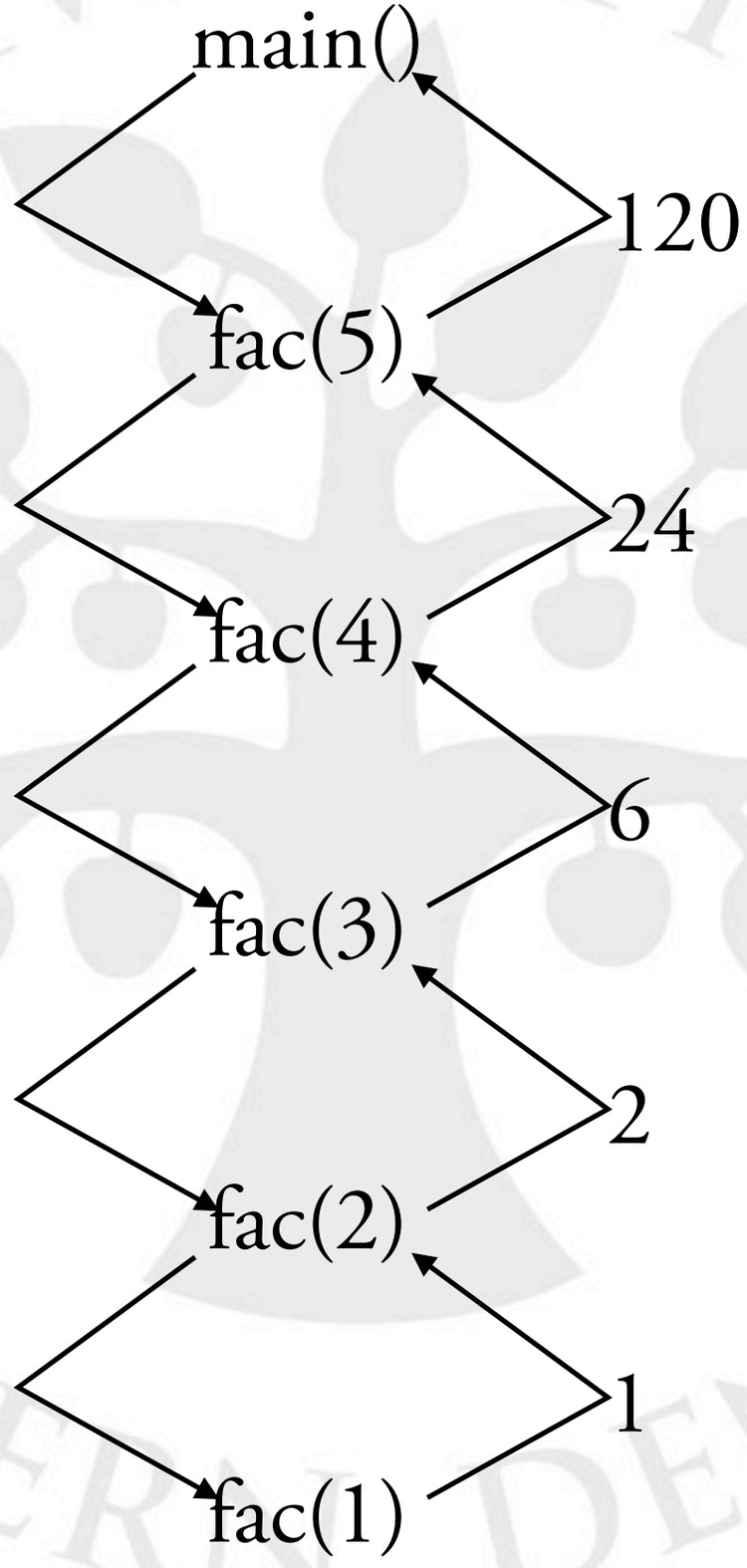
Fakultet



Fakultet



Fakultet



University of Southern Denmark • Syddansk Universitet

Fakultet

Fakultet

- Vigtig pointe

Fakultet

- Vigtig pointe
  - Rekursion er meningsløs uden et eller flere basistilfælde

Fakultet

- Vigtig pointe
  - Rekursion er meningsløs uden et eller flere basistilfælde
    - Det giver ikke mening at definere noget i termer af sige selv, medmindre man har en (eller flere) mindste "byggesten"

Fakultet

- Vigtig pointe
  - Rekursion er meningsløs uden et eller flere basistilfælde
    - Det giver ikke mening at definere noget i termer af sige selv, medmindre man har en (eller flere) mindste "byggesten"
    - For lister er basistilfældet "et tal"

Fakultet

- Vigtig pointe
  - Rekursion er meningsløs uden et eller flere basistilfælde
    - Det giver ikke mening at definere noget i termer af sige selv, medmindre man har en (eller flere) mindste "byggesten"
    - For lister er basistilfældet "et tal"
    - "En liste af tal er enten et tal, eller et tal efterfulgt af en liste af tal"

Fakultet

- Vigtig pointe
  - Rekursion er meningsløs uden et eller flere basistilfælde
    - Det giver ikke mening at definere noget i termer af sige selv, medmindre man har en (eller flere) mindste "byggesten"
    - For lister er basistilfældet "et tal"
    - "En liste af tal er enten et tal, eller et tal efterfulgt af en liste af tal"
    - Ellers ville definitionen bare sige:
      "En liste af tal er et tal efterfulgt af en liste af tal"

Fakultet

- Vigtig pointe
  - Rekursion er meningsløs uden et eller flere basistilfælde
    - Det giver ikke mening at definere noget i termer af sige selv, medmindre man har en (eller flere) mindste "byggesten"
    - For lister er basistilfældet "et tal"
    - "En liste af tal er enten et tal, eller et tal efterfulgt af en liste af tal"
    - Ellers ville definitionen bare sige: "En liste af tal er et tal efterfulgt af en liste af tal"
      - Uendeligt lange lister

Fakultet

- Vigtig pointe
  - Rekursion er meningsløs uden et eller flere basistilfælde
    - Det giver ikke mening at definere noget i termer af sige selv, medmindre man har en (eller flere) mindste "byggesten"
    - For lister er basistilfældet "et tal"
      - "En liste af tal er enten et tal, eller et tal efterfulgt af en liste af tal"
      - Ellers ville definitionen bare sige: "En liste af tal er et tal efterfulgt af en liste af tal"
        - Uendeligt lange lister
  - For fakultet er basistilfældet at 1! = 1

Fakultet

- Vigtig pointe
  - Rekursion er meningsløs uden et eller flere basistilfælde
    - Det giver ikke mening at definere noget i termer af sige selv, medmindre man har en (eller flere) mindste "byggesten"
    - For lister er basistilfældet "et tal"
      - "En liste af tal er enten et tal, eller et tal efterfulgt af en liste af tal"
      - Ellers ville definitionen bare sige: "En liste af tal er et tal efterfulgt af en liste af tal"
        - Uendeligt lange lister
  - For fakultet er basistilfældet at 1! = 1
    - Kunne lige så godt være 0! = 1

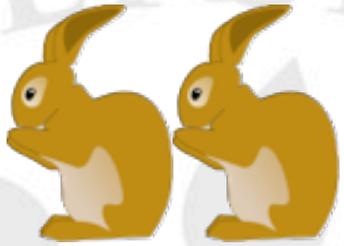
Fibonacci-tallene

- Introduceret i vesten af Leonardo af Pisa (Fibonacci) i en bog fra 1202
  - Kendt tidligere og kommer oprindeligt fra Indien
- Betragt en (urealistisk) kanin-population
  - Der starter med at være et kanin-par
  - Et nyt kanin-par kan ikke få unger, før de er 1 måned gammel
  - Et kanin-par kan få to unger på 1 måned

Fibonacci-tallene

Fibonacci-tallene

Fibonacci-tallene

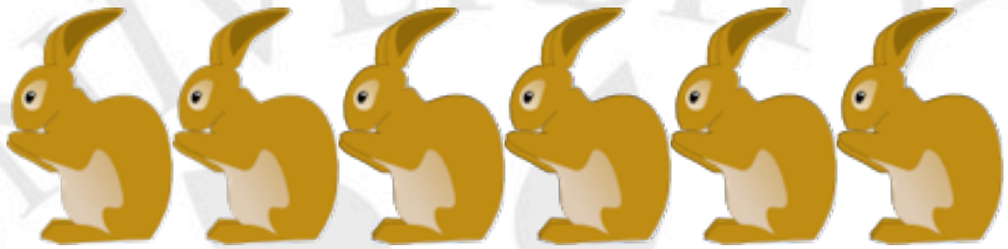


Fibonacci-tallene

Fibonacci-tallene

Fibonacci-tallene

Fibonacci-tallene



Fibonacci-tallene

Fibonacci-tallene

Fibonacci-tallene

Fibonacci-tallene

- 1. måned - 1 par kaniner
- 2. måned - 1 par kaniner
- 3. måned - 2 par kaniner
- 4. måned - 3 psr kaniner
- 5. måned - 5 par kaniner
- 6. måned - 8 par kaniner
- 7. måned - ?

Fibonacci-tallene

Fibonacci-tallene

- Måned n

Fibonacci-tallene

- Måned n
  - (#kaninpar i måned n-1) + (#kaninpar i måned n-2)

Fibonacci-tallene

- Måned n
  - (#kaninpar i måned n-1) + (#kaninpar i måned n-2)
  - Alle kaniner der var der i måned n-1 er der også i måned n

Fibonacci-tallene

- Måned n
  - (#kaninpar i måned n-1) + (#kaninpar i måned n-2)
  - Alle kaniner der var der i måned n-1 er der også i måned n
  - Alle kaniner der var der i måned n-2 får unger i måned n

Fibonacci-tallene

- Måned n
  - (#kaninpar i måned n-1) + (#kaninpar i måned n-2)
  - Alle kaniner der var der i måned n-1 er der også i måned n
  - Alle kaniner der var der i måned n-2 får unger i måned n
- Lad f(n) være antallet af kaninpar i måned n

Fibonacci-tallene

- Måned n
  - (#kaninpar i måned n-1) + (#kaninpar i måned n-2)
  - Alle kaniner der var der i måned n-1 er der også i måned n
  - Alle kaniner der var der i måned n-2 får unger i måned n
- Lad f(n) være antallet af kaninpar i måned n
  - f(n) = f(n-1) + f(n-2)

Fibonacci-tallene

- Måned n
  - (#kaninpar i måned n-1) + (#kaninpar i måned n-2)
  - Alle kaniner der var der i måned n-1 er der også i måned n
  - Alle kaniner der var der i måned n-2 får unger i måned n
- Lad f(n) være antallet af kaninpar i måned n
  - f(n) = f(n-1) + f(n-2)
  - f(n) - det n'te fibonacci-tal

Fibonacci-tallene

- Måned n
  - (#kaninpar i måned n-1) + (#kaninpar i måned n-2)
  - Alle kaniner der var der i måned n-1 er der også i måned n
  - Alle kaniner der var der i måned n-2 får unger i måned n
- Lad f(n) være antallet af kaninpar i måned n
  - f(n) = f(n-1) + f(n-2)
  - f(n) - det n'te fibonacci-tal
  - f(1) = 1, f(2) = 1, f(3) = 2, f(4) = 3, f(5) = 5, f(6) = 8, osv...

Fibonacci-tallene

- Måned n
  - (#kaninpar i måned n-1) + (#kaninpar i måned n-2)
  - Alle kaniner der var der i måned n-1 er der også i måned n
  - Alle kaniner der var der i måned n-2 får unger i måned n
- Lad f(n) være antallet af kaninpar i måned n
  - f(n) = f(n-1) + f(n-2)
  - f(n) - det n'te fibonacci-tal
  - f(1) = 1, f(2) = 1, f(3) = 2, f(4) = 3, f(5) = 5, f(6) = 8, osv...
  - Bemærk definitionen er rekursiv
    f(n) afhænger af f(n-1) og f(n-2)

Fibonacci-tallene

- f(n) = f(n-1) + f(n-2)
  = (f(n-2) + f(n-3)) + (f(n-3) + f(n-4))
  = (f(n-3) + f(n-4)) + (f(n-4) + f(n-5)) +
  (f(n-4) + f(n-5)) + (f(n-5) + f(n-6))
  = osv...
- F.eks.
  - f(5) = f(4) + f(3)
    = (f(3) + f(2)) + (f(2) + f(1))
    = ((f(2) + f(1)) + f(2)) + (f(2) + f(1))
    = ((1 + 1) + 1) + (1 + 1)
    = 5

Fibonacci-tallene

```
public class Fibonacci {
    public static void main( String[] args ) {
        System.out.println( f(6) );
    }

    public static int f( int n ) {
        return f(n-1) + f(n-2);
    }
}
```

Fibonacci-tallene

```
public class Fibonacci {
    public static void main( String[] args ) {
        System.out.println( f(6) );
    }

    public static int f( int n ) {
        return f(n-1) + f(n-2);
    }
}
```

```
Terminal — bash — 66×8
mac-sdu-00001:10 petersk$ javac Fibonacci.java
mac-sdu-00001:10 petersk$ java Fibonacci
Exception in thread "main" java.lang.StackOverflowError
        at Fibonacci.f(Fibonacci.java:7)
        at Fibonacci.f(Fibonacci.java:7)
        at Fibonacci.f(Fibonacci.java:7)
        at Fibonacci.f(Fibonacci.java:7)
        at Fibonacci.f(Fibonacci.java:7)
        at Fibonacci.f(Fibonacci.java:7)
```

Fibonacci-tallene

Fibonacci-tallene

- Samme fejl som sidst

Fibonacci-tallene

- Samme fejl som sidst
  - Mangler basis-tilfælde

Fibonacci-tallene

- Samme fejl som sidst
  - Mangler basis-tilfælde
  - Hmm....

Fibonacci-tallene

- Samme fejl som sidst
  - Mangler basis-tilfælde
  - Hmm....
  - Vi ved at f(1) = 1

Fibonacci-tallene

```
public static int f( int n ) {
     if( n == 1 )
          return 1;
     else {
          int n2 = f(n-2);
          int n1 = f(n-1);
          return n1 + n2;
     }
}
```

Fibonacci-tallene

```
public static int f( int n ) {
     if( n == 1 )
          return 1;
     else {
          int n2 = f(n-2);
          int n1 = f(n-1);
          return n1 + n2;
     }
}
```

```
Terminal — bash — 66×8
mac-sdu-00001:10 petersk$ javac Fibonacci.java
mac-sdu-00001:10 petersk$ java Fibonacci
Exception in thread "main" java.lang.StackOverflowError
        at Fibonacci.f(Fibonacci.java:7)
        at Fibonacci.f(Fibonacci.java:10)
        at Fibonacci.f(Fibonacci.java:10)
        at Fibonacci.f(Fibonacci.java:10)
        at Fibonacci.f(Fibonacci.java:10)
```

Fibonacci-tallene

- Stadig fejl...?!?
  - Lad os prøve at debugge det
  - Hmmm...
  - f(2) burde give 1

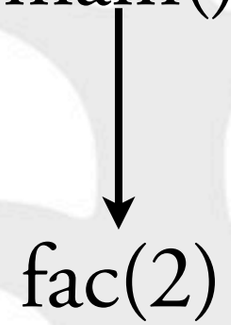
Fibonacci-tallene

- Stadig fejl...?!?
  - Lad os prøve at debugge det
  - Hmmm...
  - f(2) burde give 1

main()

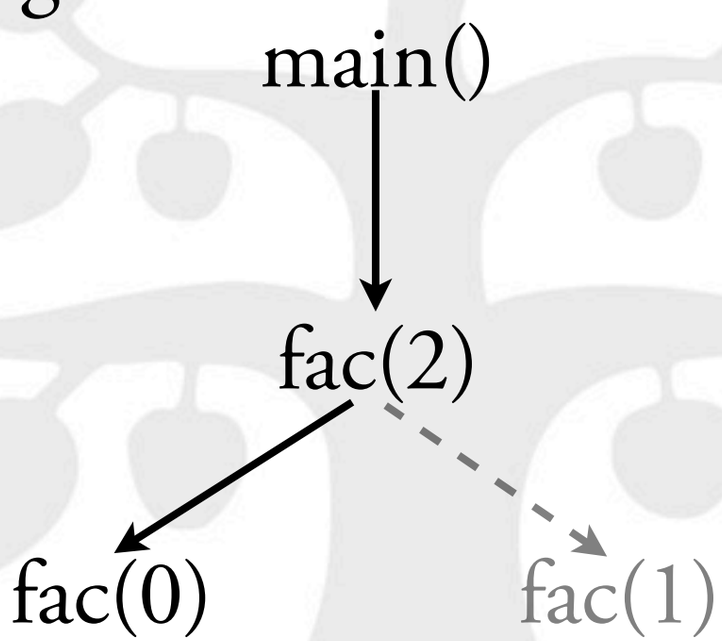
Fibonacci-tallene

- Stadig fejl...?!?
  - Lad os prøve at debugge det
  - Hmmm...
  - f(2) burde give 1



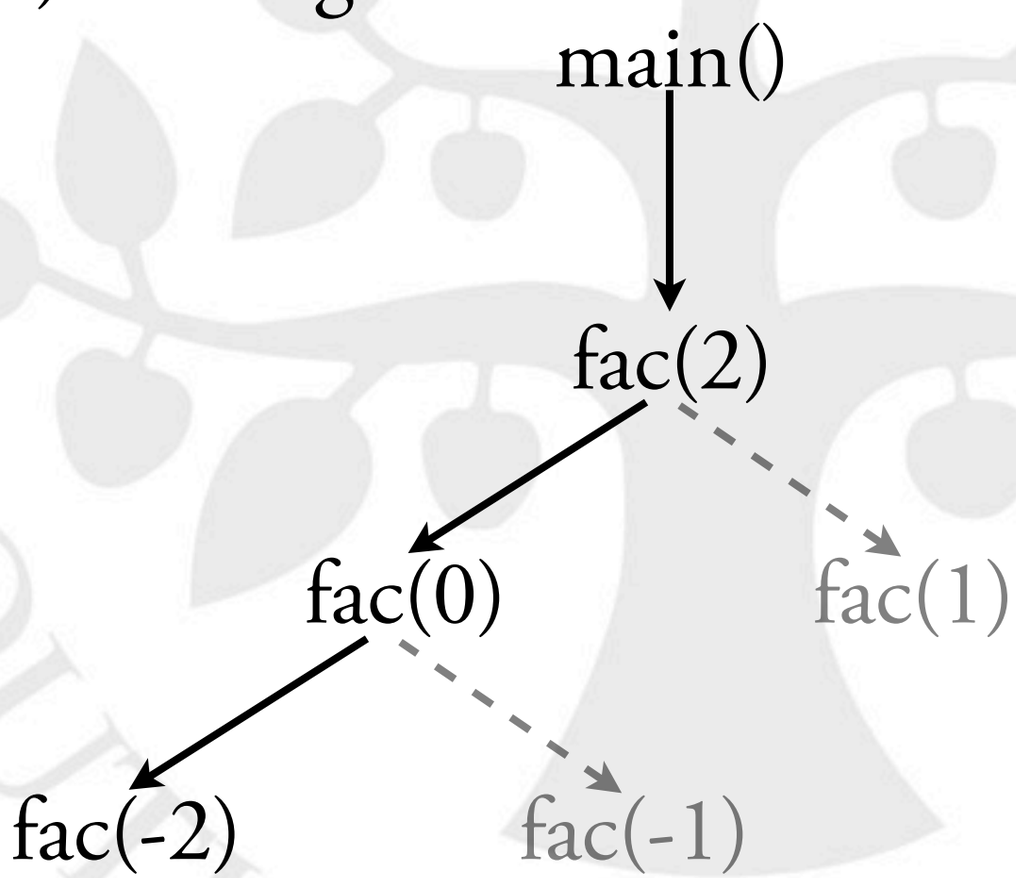
Fibonacci-tallene

- Stadig fejl...?!?
  - Lad os prøve at debugge det
  - Hmmm...
  - f(2) burde give 1



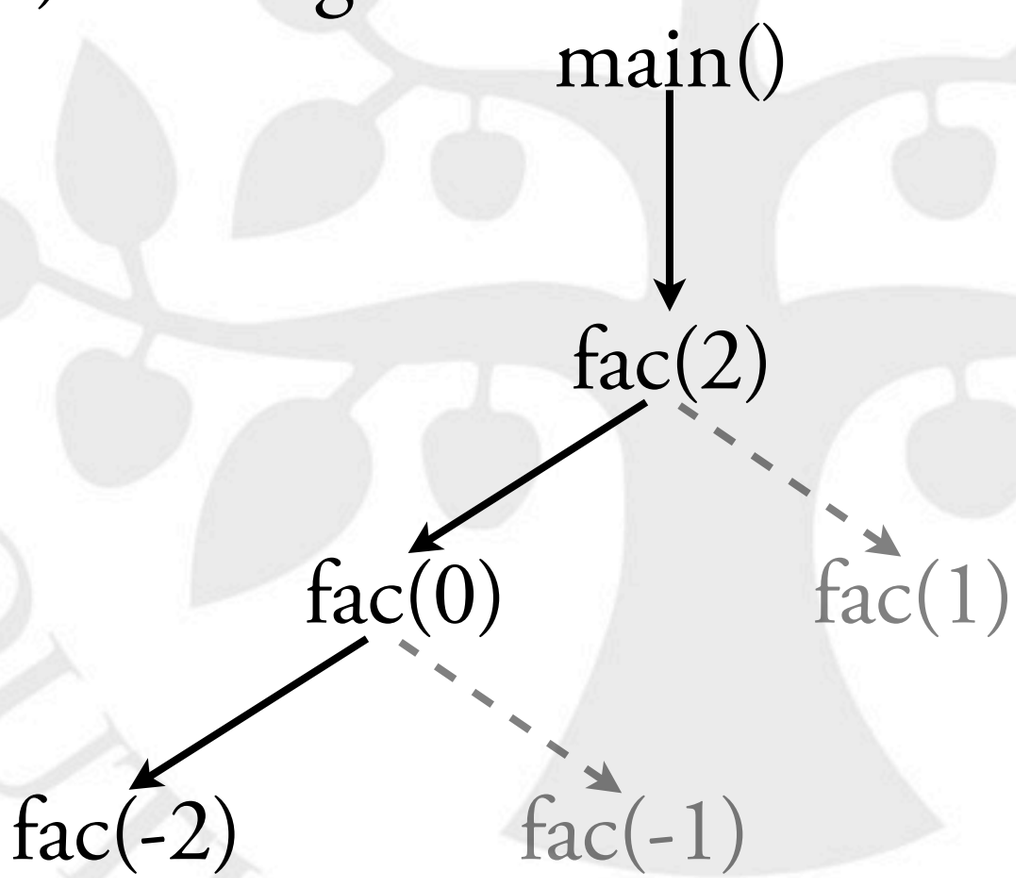
Fibonacci-tallene

- Stadig fejl...?!?
  - Lad os prøve at debugge det
  - Hmmm...
  - f(2) burde give 1



Fibonacci-tallene

- Stadig fejl...?!?
  - Lad os prøve at debugge det
  - Hmmm...
  - f(2) burde give 1



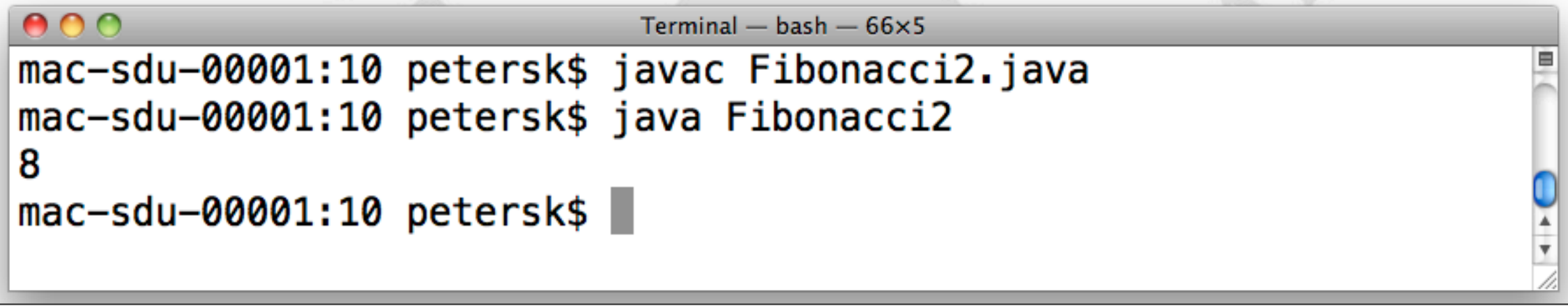
Ups....

Fibonacci-tallene

```
public static int f( int n ) {
     if( n == 1 )
          return 1;
     else if( n == 2 )
          return 1;
     else
          return f(n-1) + f(n-2);
}
```

Fibonacci-tallene

```
public static int f( int n ) {
     if( n == 1 )
          return 1;
     else if( n == 2 )
          return 1;
     else
          return f(n-1) + f(n-2);
}
```

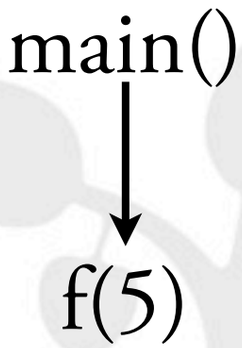


Fibonacci-tallene

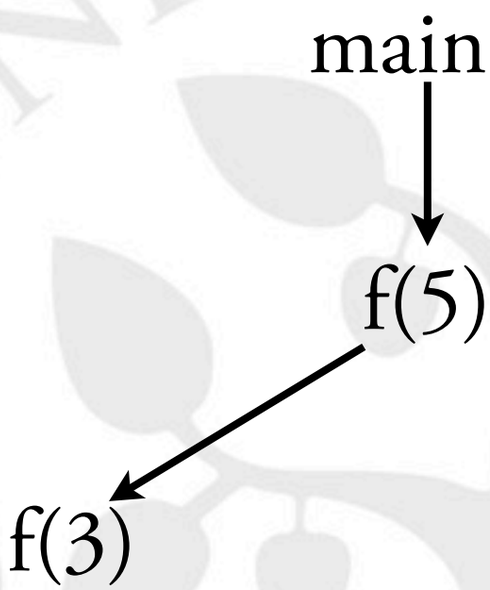
Fibonacci-tallene

main()

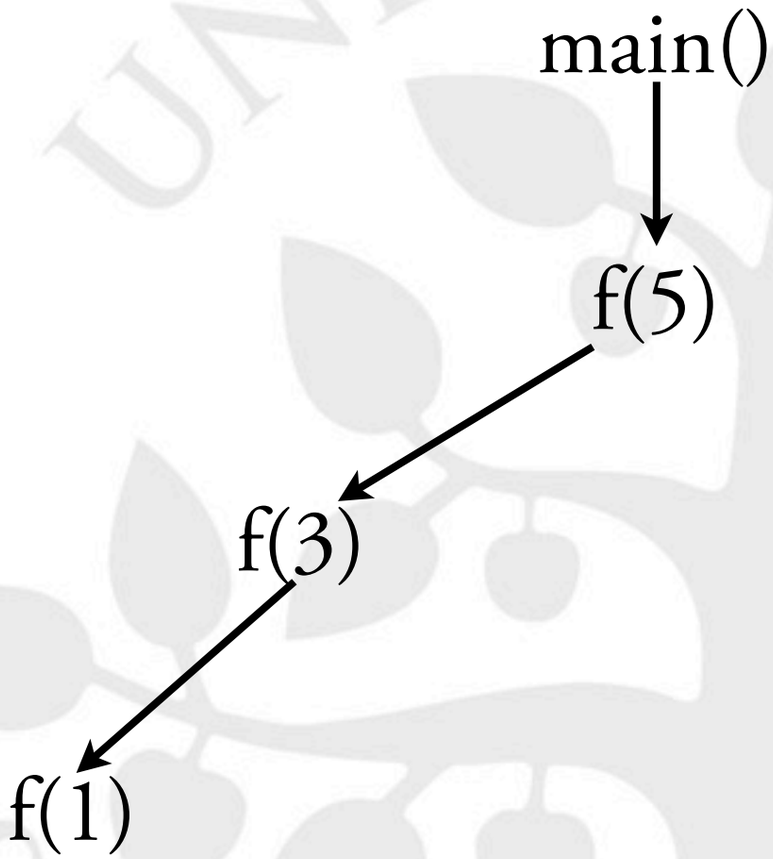
Fibonacci-tallene



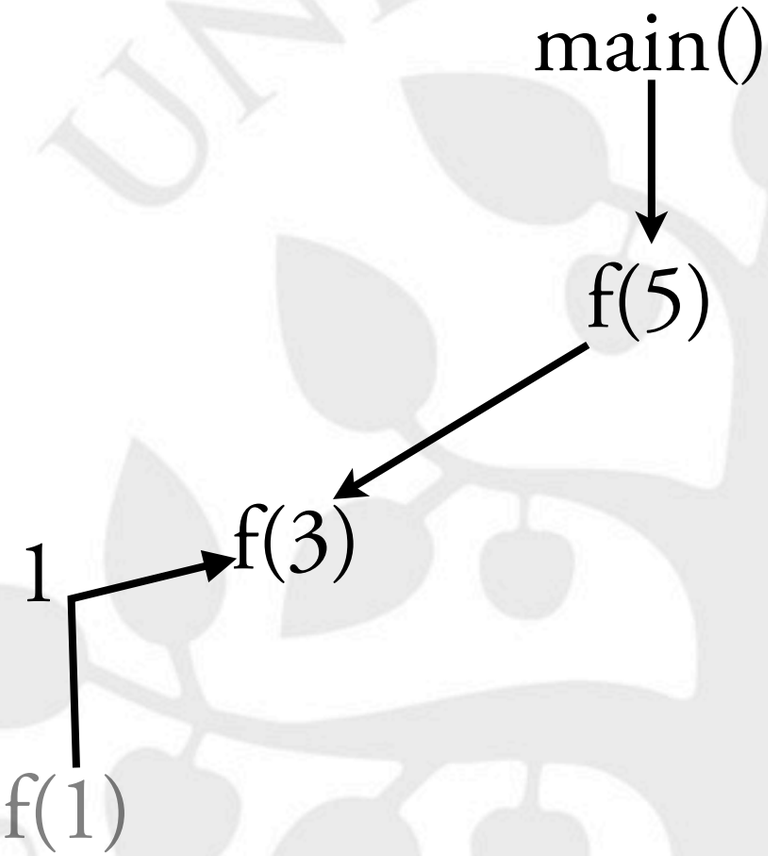
Fibonacci-tallene



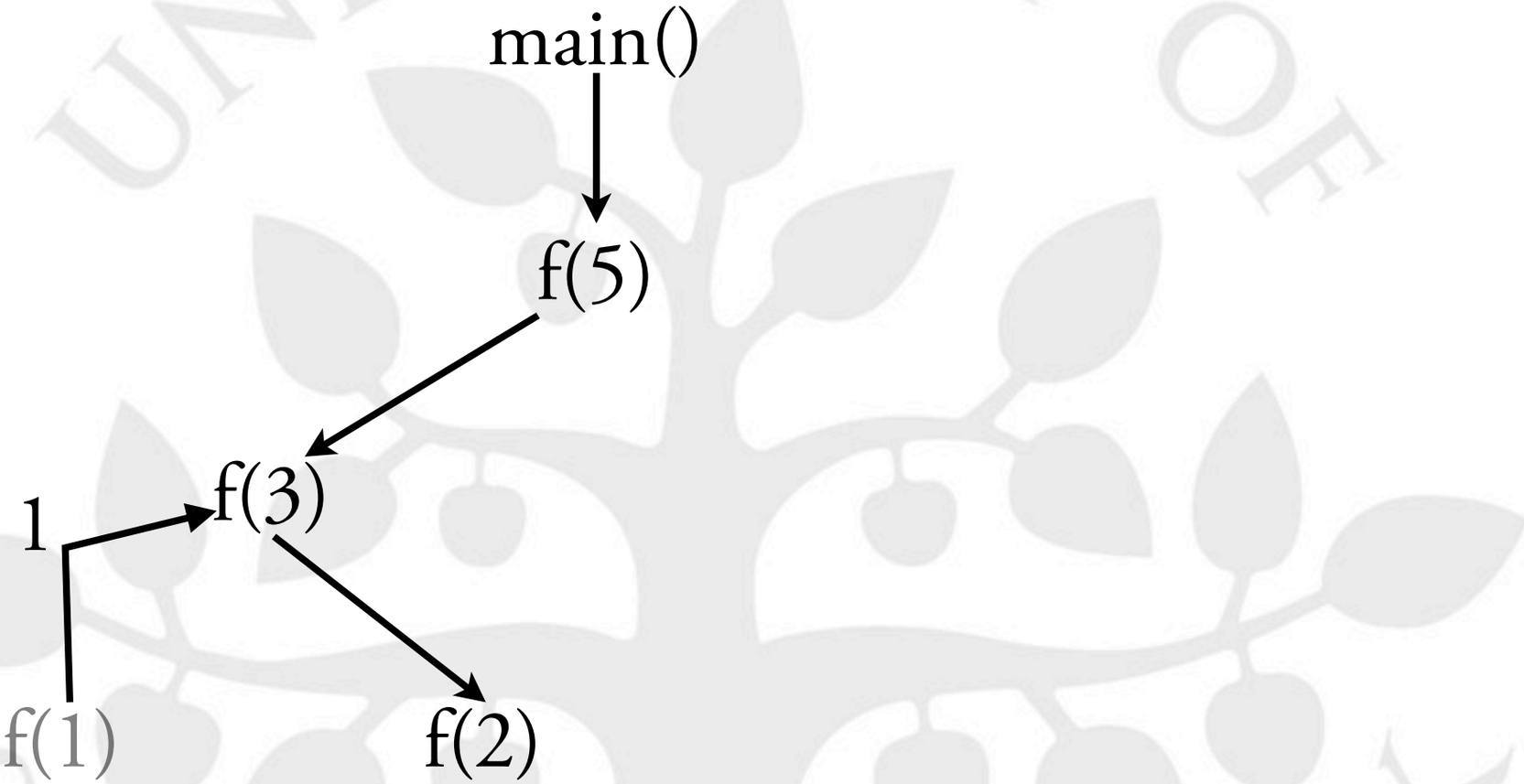
Fibonacci-tallene



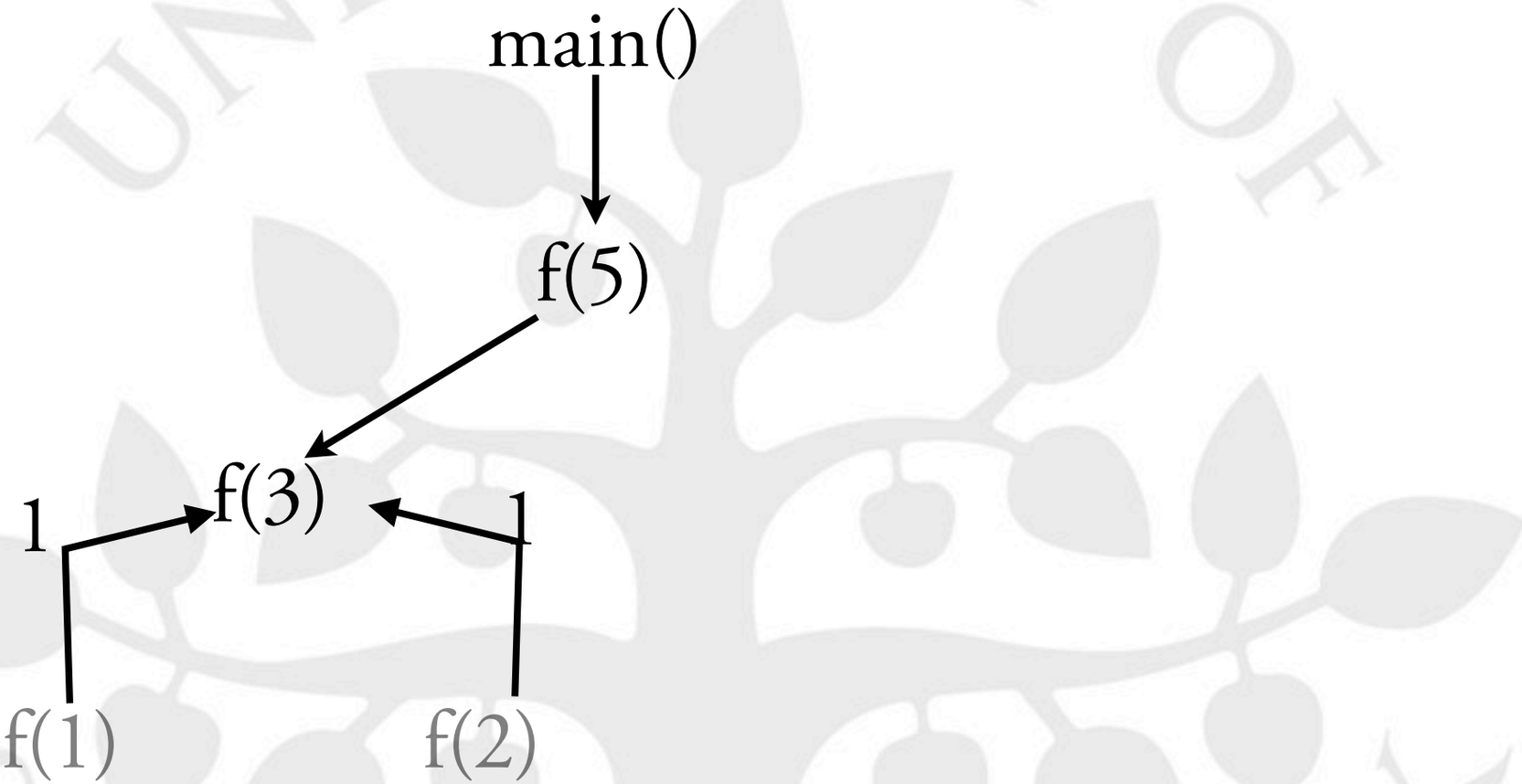
Fibonacci-tallene



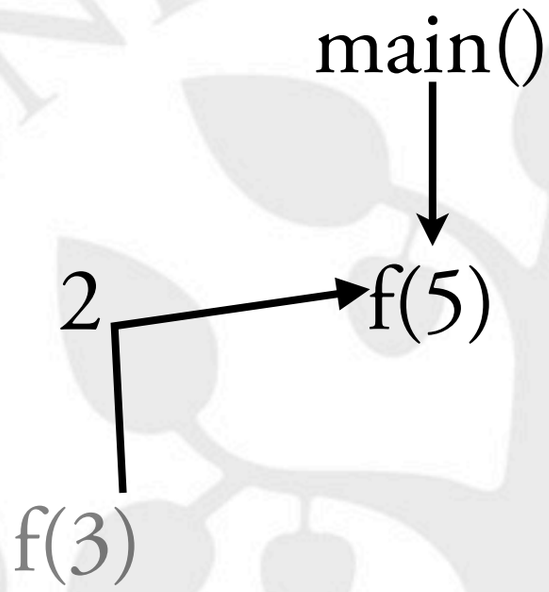
Fibonacci-tallene



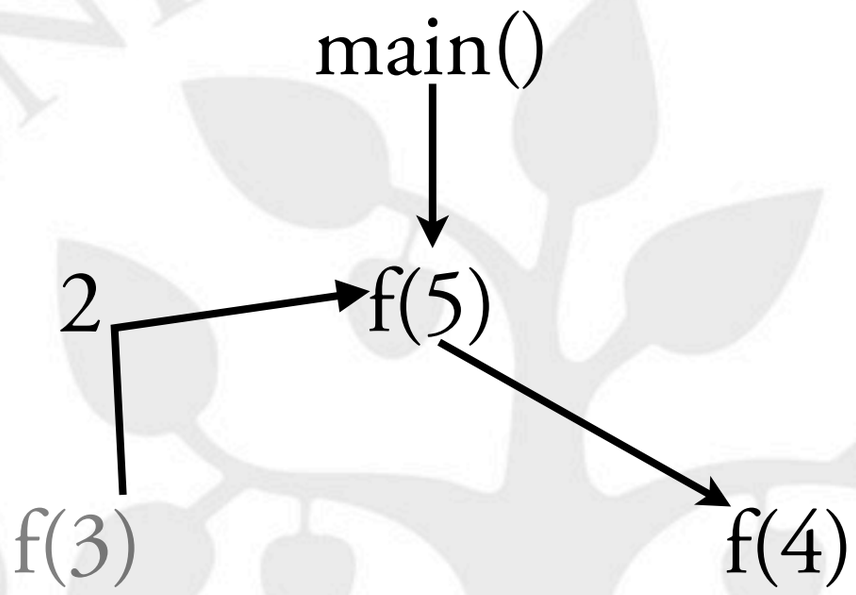
Fibonacci-tallene



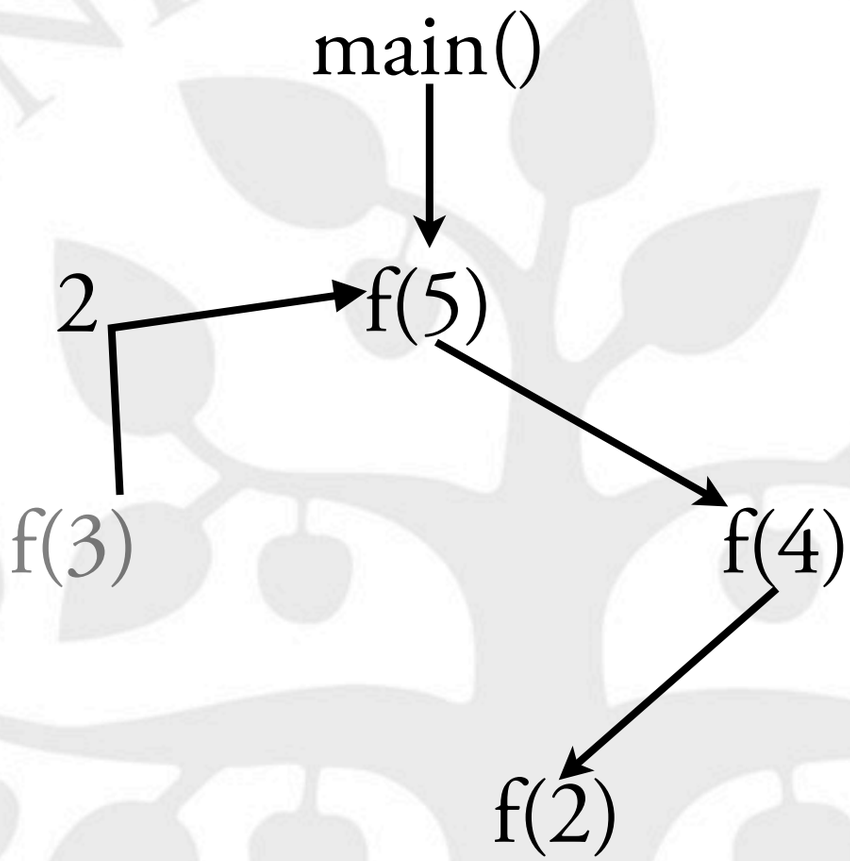
Fibonacci-tallene



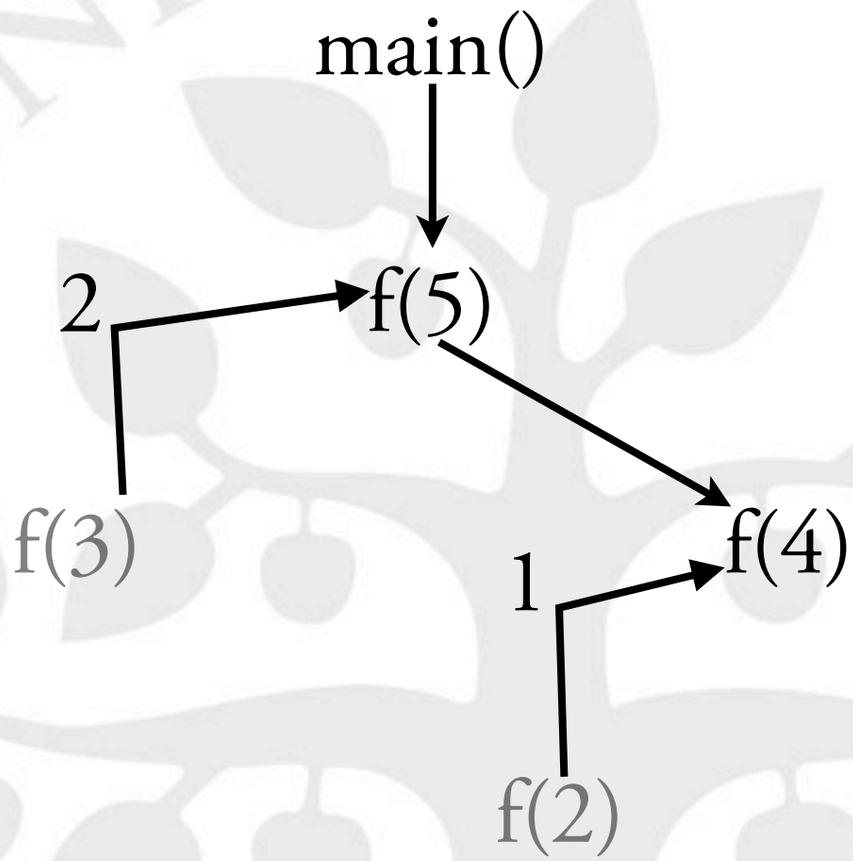
Fibonacci-tallene



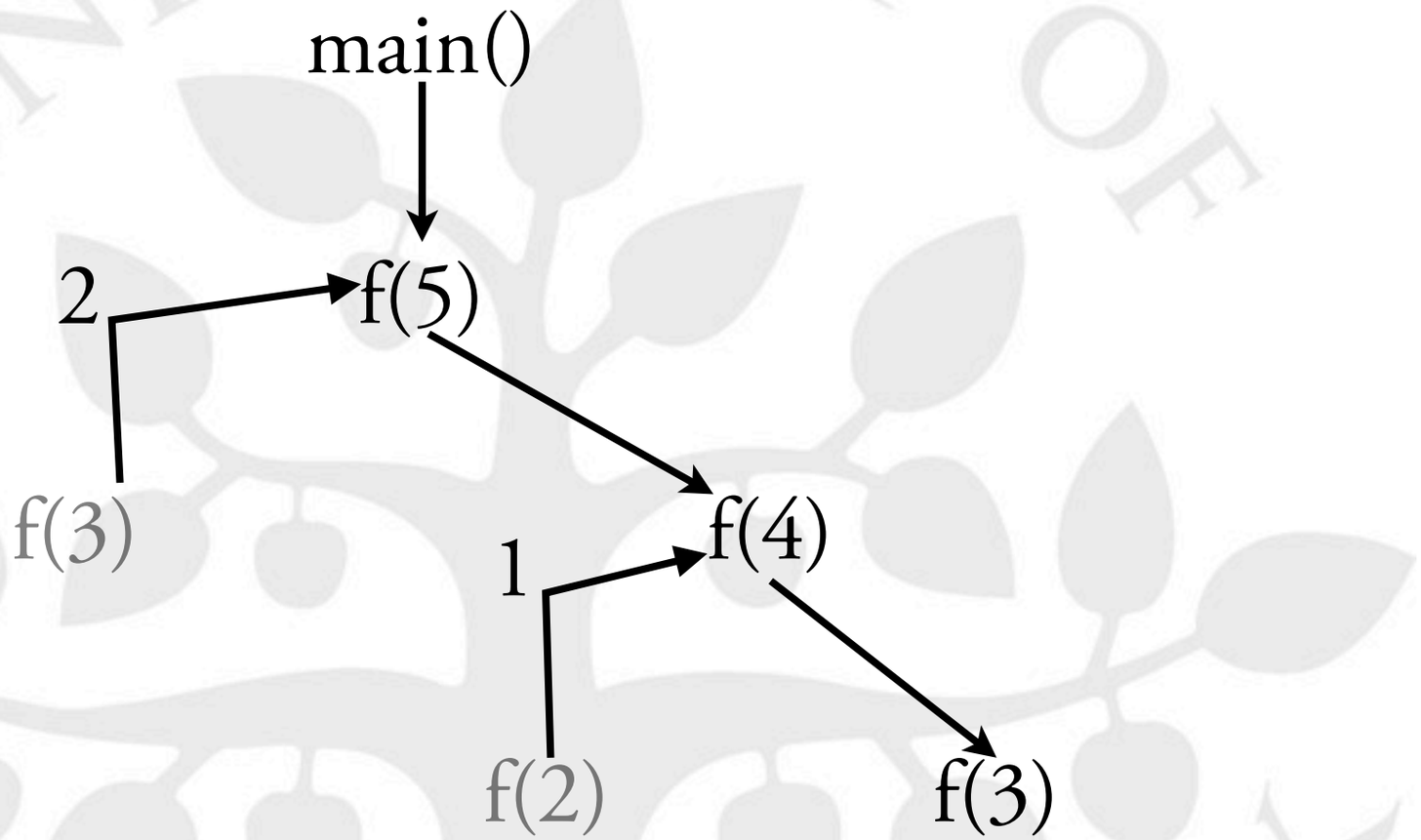
Fibonacci-tallene



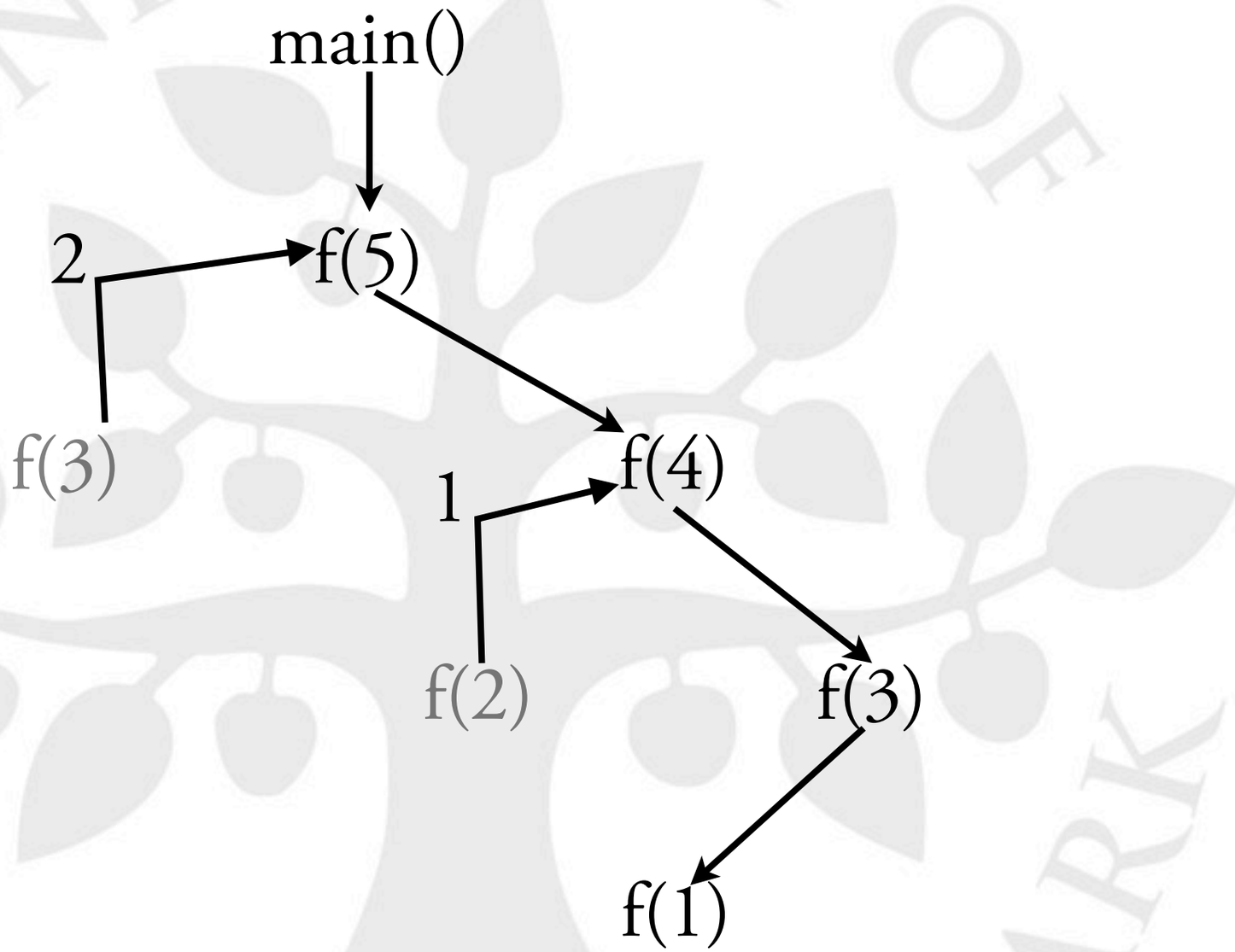
Fibonacci-tallene



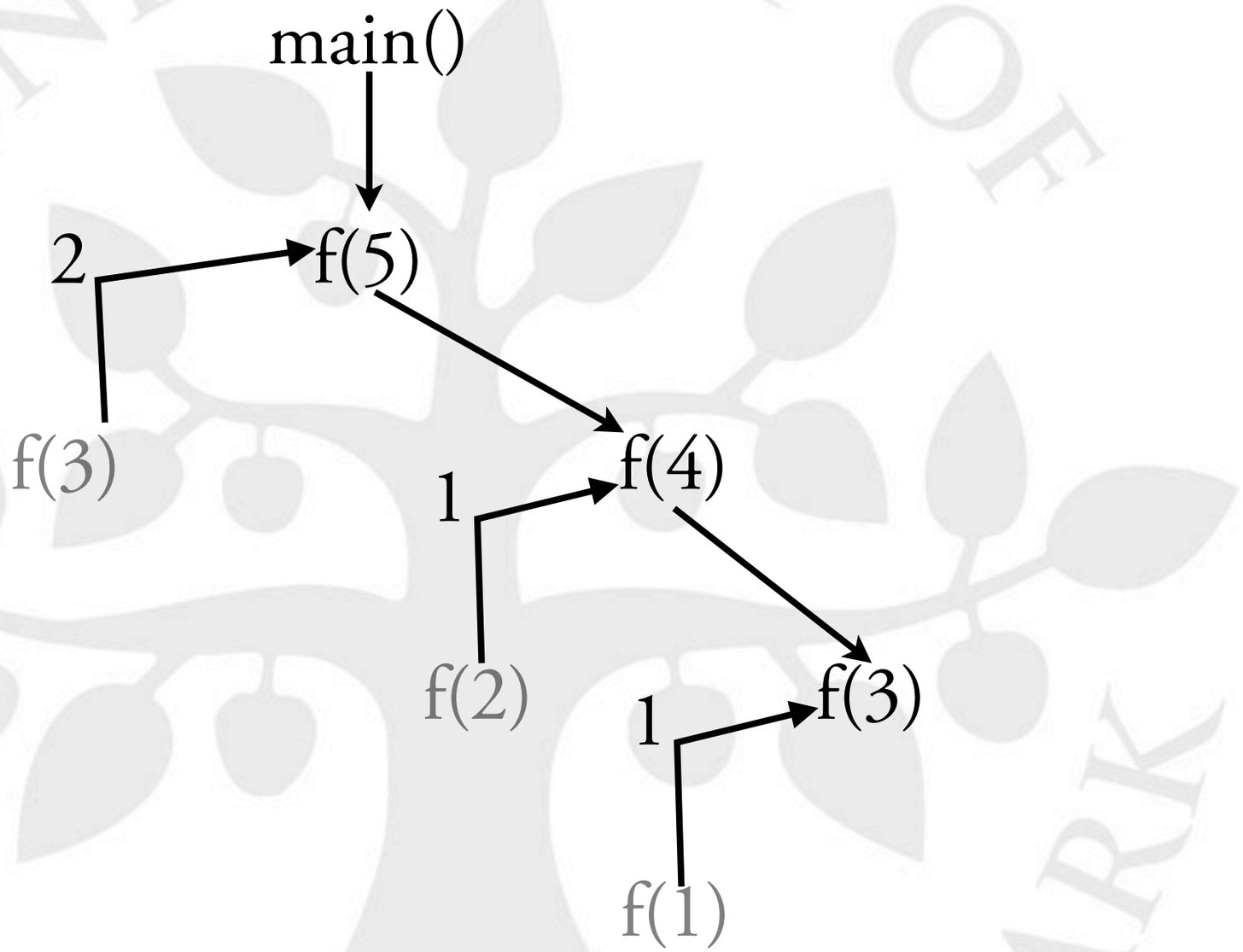
Fibonacci-tallene



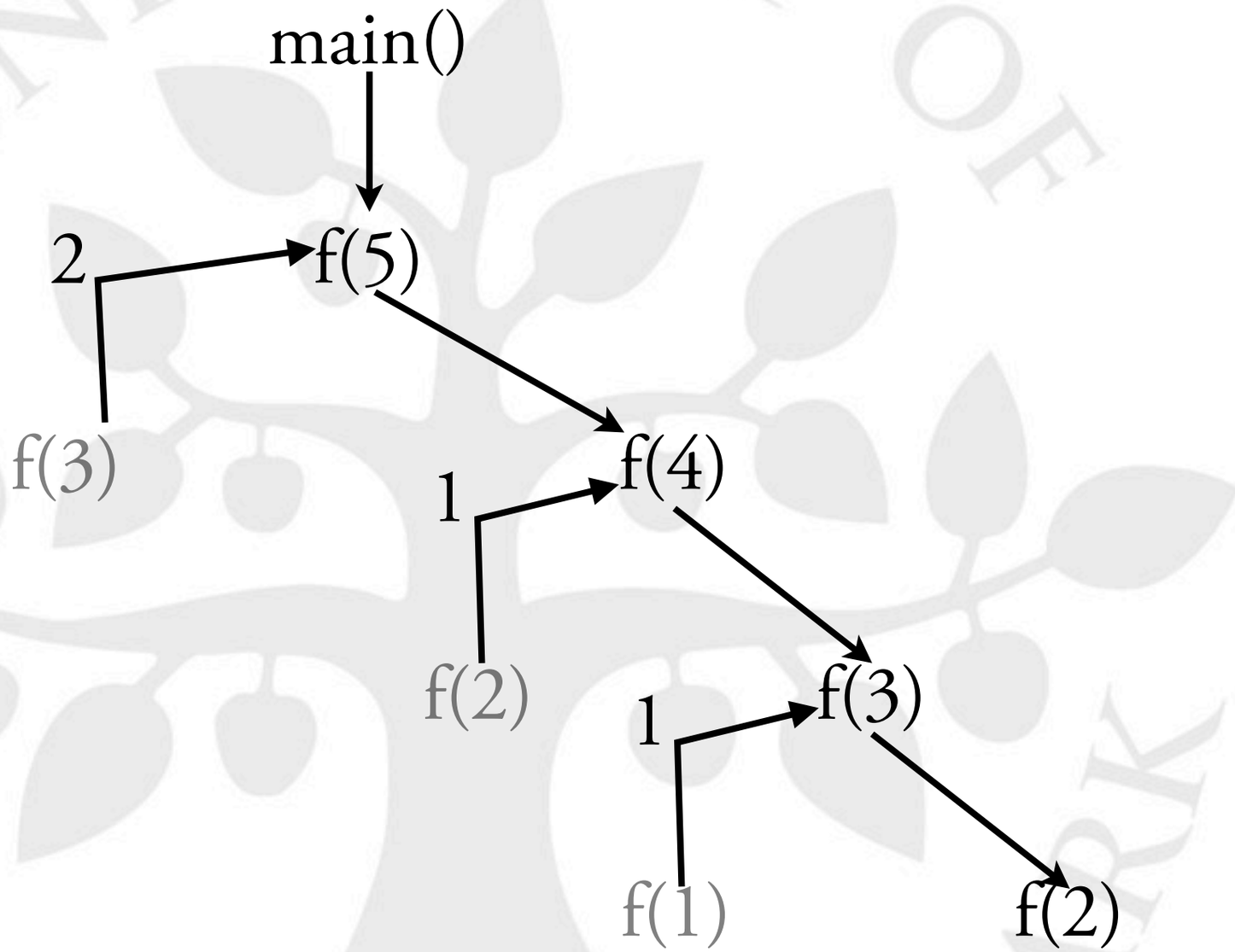
Fibonacci-tallene



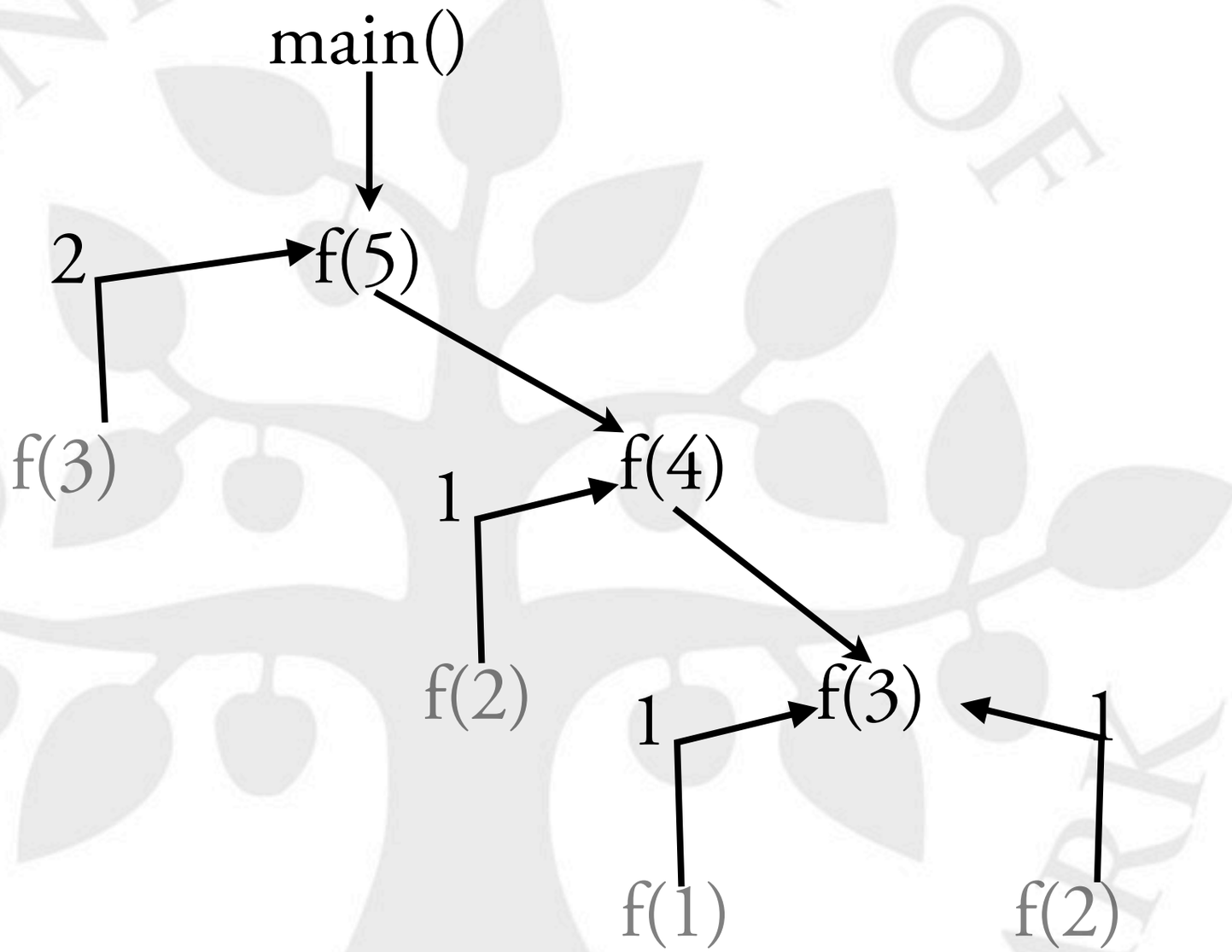
Fibonacci-tallene



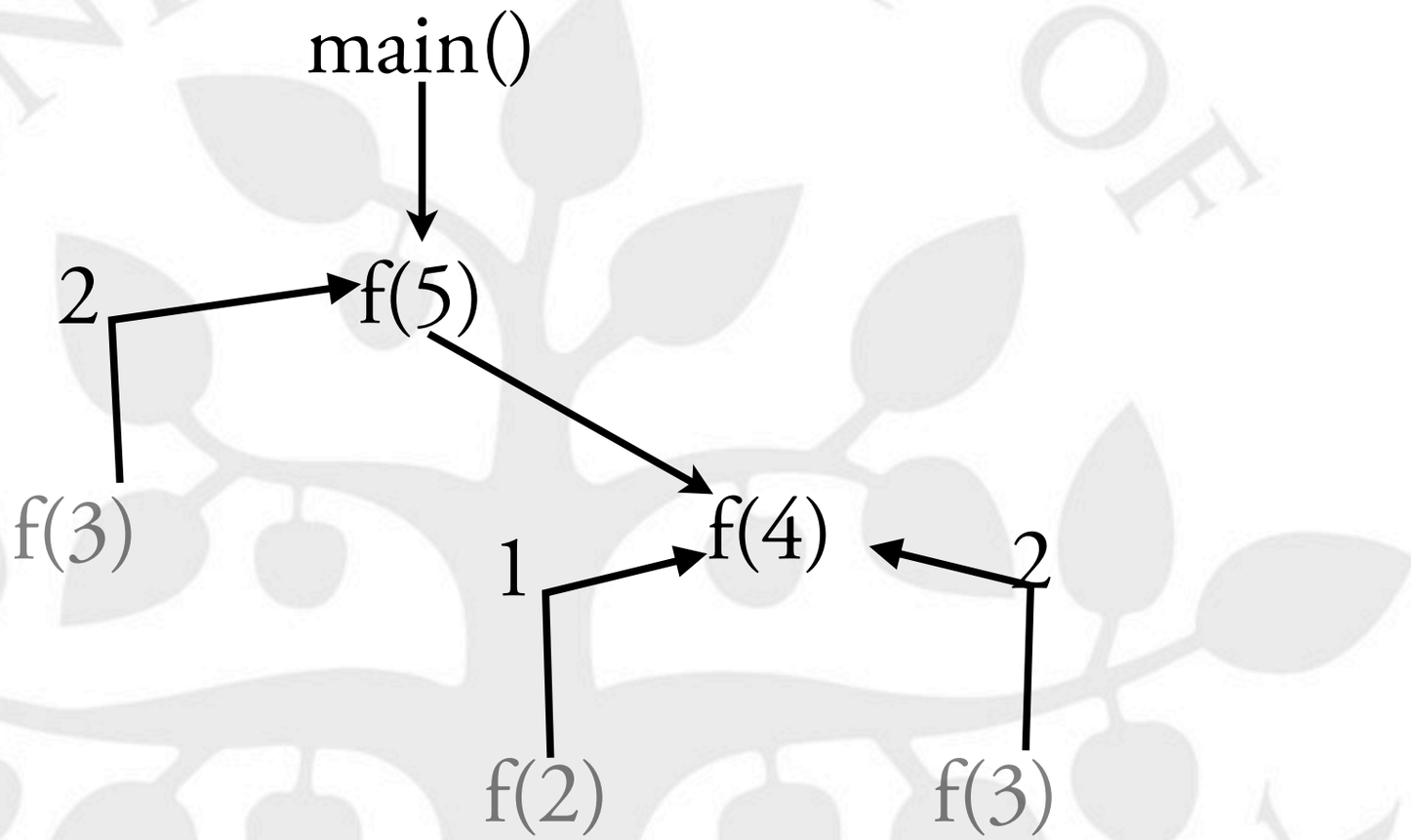
Fibonacci-tallene



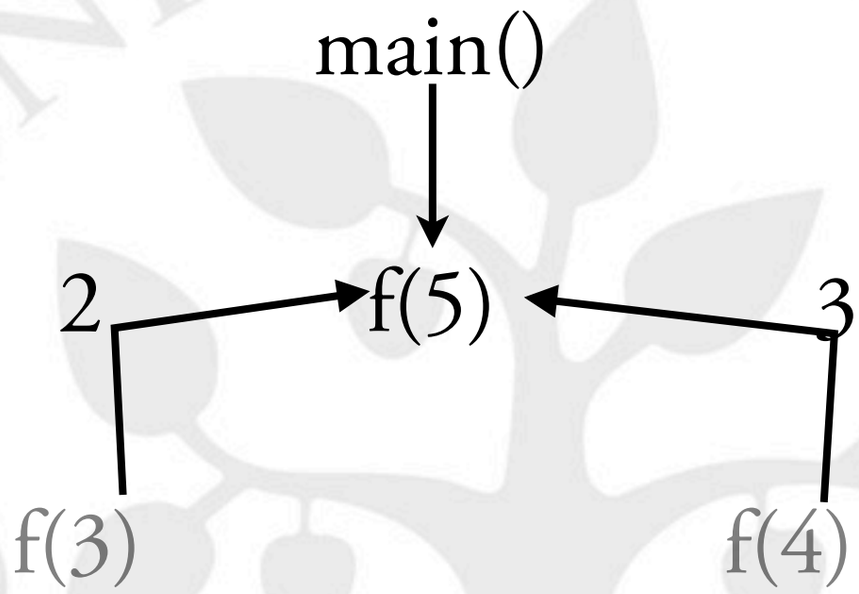
Fibonacci-tallene



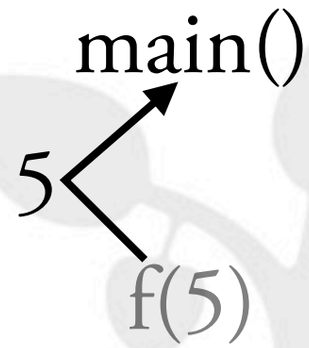
Fibonacci-tallene



Fibonacci-tallene



Fibonacci-tallene



Fibonacci-tallene

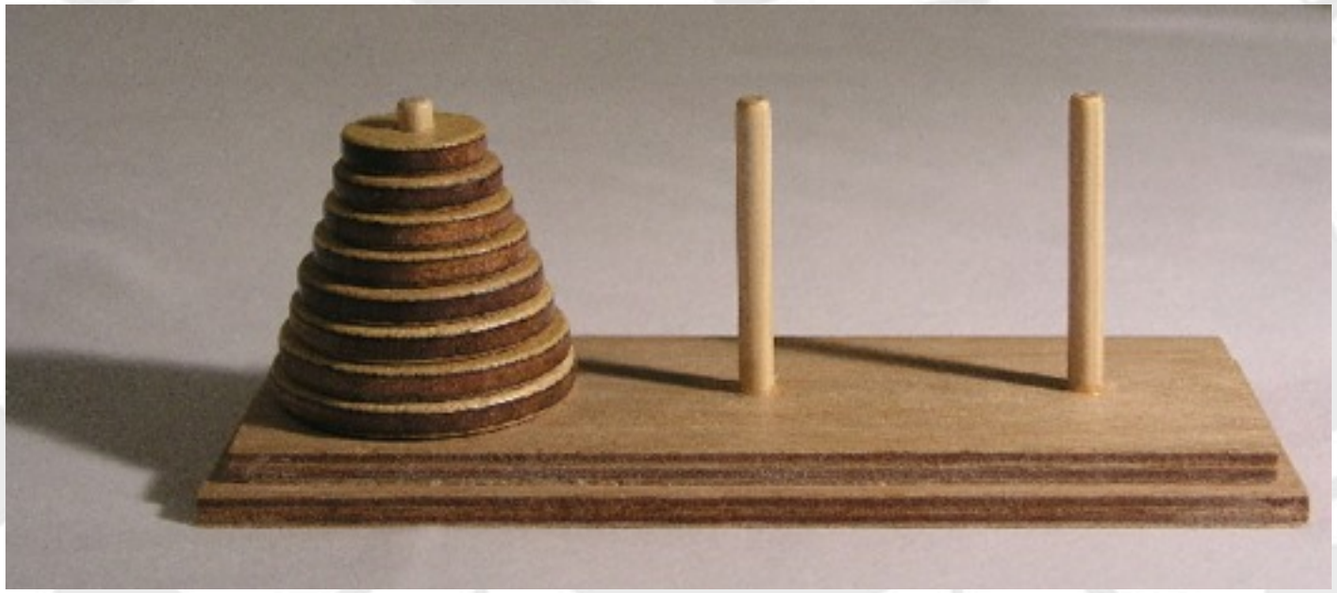
- Fibonacci-tallene kan ses mange steder i naturen
  - Solsikker

  - Grankogler

  - osv...
- Tæt forbundet med det gyldne snit
  - f(n+1) / f(n) -> det gyldne snit for n -> uendelig

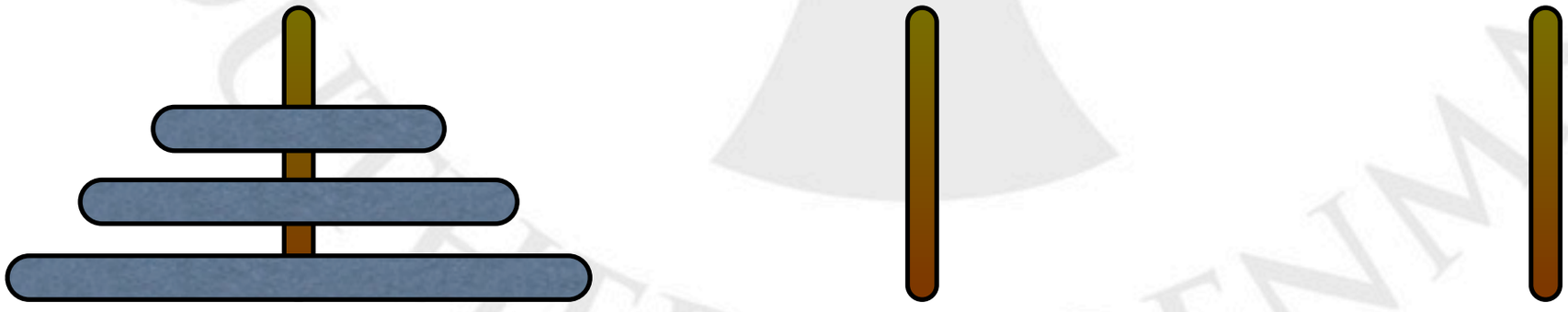
Hanois tårne

- Et spil opfundet af Edouard Lucas i 1883
- Tre pinde med skiver af varierende diameter



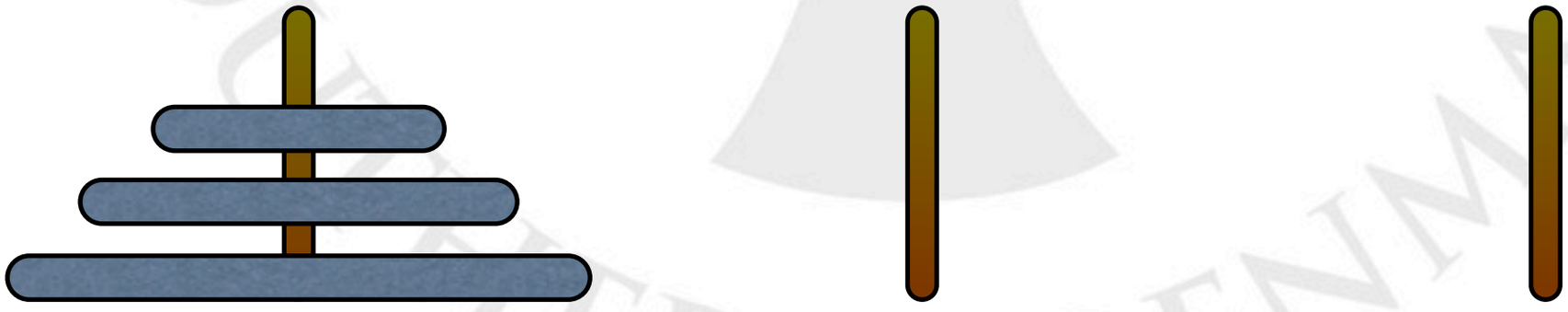
- Flyt alle skiver fra den venstre pind til den højre
  - Må kun flytte en skive af gangen
  - Må ikke sætte en større skive ovenpå en mindre
  - Alle skiverne, undtagen den der flyttes, skal være på en pind

Hanois tårne



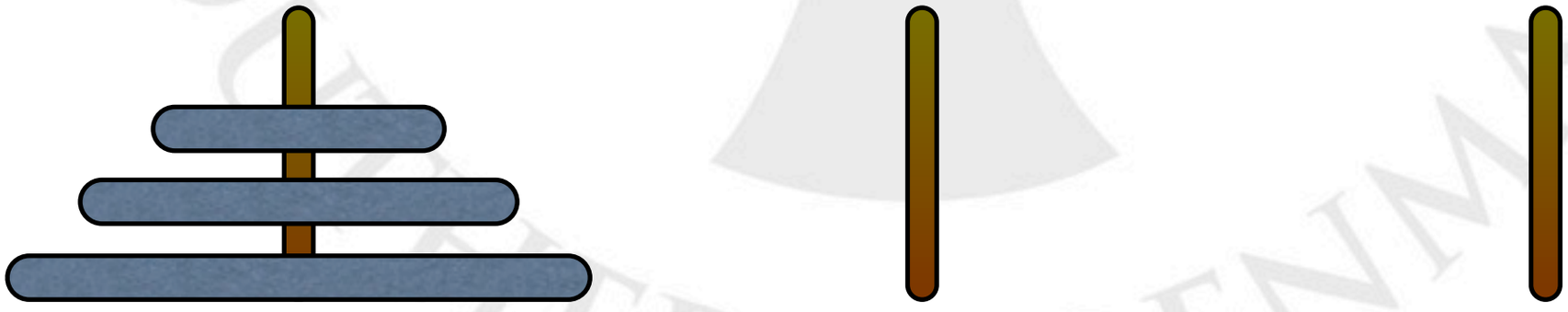
Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste



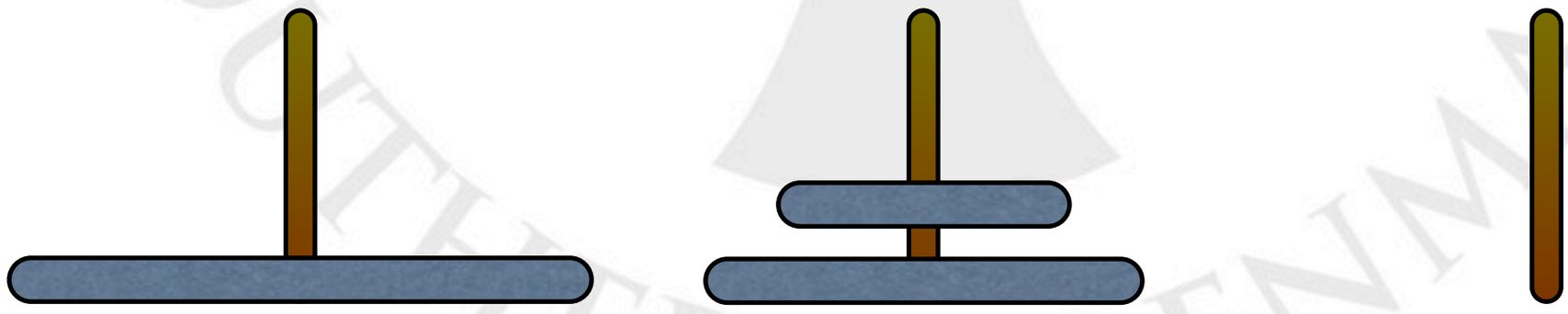
Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind



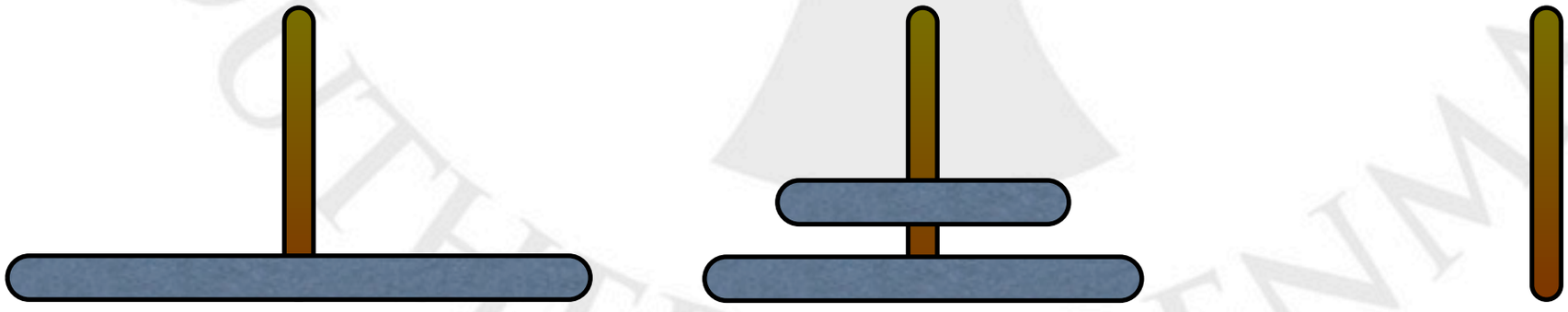
Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind



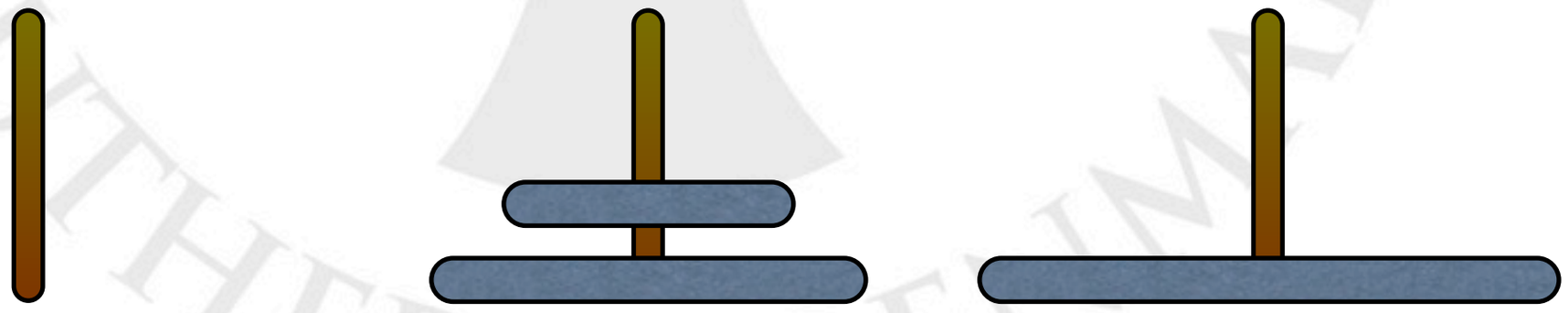
Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind
  - Så flyttes den største skive til den højre pind



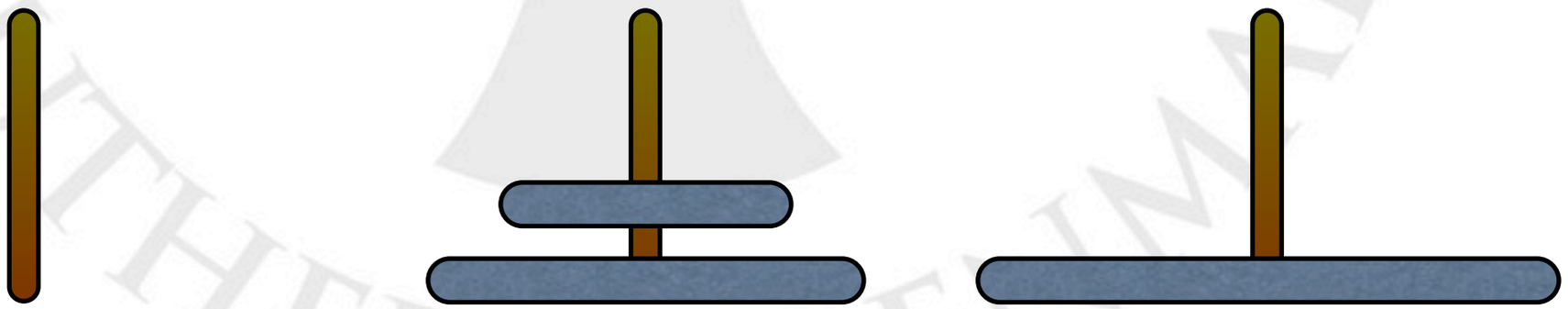
Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind
  - Så flyttes den største skive til den højre pind



Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind
  - Så flyttes den største skive til den højre pind
  - Til sidst skal alle skiverne på den midterste pind flyttes til den højre pind, ved brug af den venstre pind



Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind
  - Så flyttes den største skive til den højre pind
  - Til sidst skal alle skiverne på den midterste pind flyttes til den højre pind, ved brug af den venstre pind

Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind
  - Så flyttes den største skive til den højre pind
  - Til sidst skal alle skiverne på den midterste pind flyttes til den højre pind, ved brug af den venstre pind
- Flyt N skiver fra venstre til højre ved brug af midterste

Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind
  - Så flyttes den største skive til den højre pind
  - Til sidst skal alle skiverne på den midterste pind flyttes til den højre pind, ved brug af den venstre pind
- Flyt N skiver fra venstre til højre ved brug af midterste
  - Flyt N-1 skiver fra den venstre pind til den midterste ved brug af den højre

Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind
  - Så flyttes den største skive til den højre pind
  - Til sidst skal alle skiverne på den midterste pind flyttes til den højre pind, ved brug af den venstre pind
- Flyt N skiver fra venstre til højre ved brug af midterste
  - Flyt N-1 skiver fra den venstre pind til den midterste ved brug af den højre
  - Flyt sidste skive fra den venstre pind til den højre

Hanois tårne

- Flyt alle skiver fra den venstre pind til den højre, ved brug af den midterste
  - På et tidspunkt skal alle skiver, pånær den største, være på den midterste pind
  - Så flyttes den største skive til den højre pind
  - Til sidst skal alle skiverne på den midterste pind flyttes til den højre pind, ved brug af den venstre pind
- Flyt N skiver fra venstre til højre ved brug af midterste
  - Flyt N-1 skiver fra den venstre pind til den midterste ved brug af den højre
  - Flyt sidste skive fra den venstre pind til den højre
  - Flyt N-1 skiver fra den midterste pind til den højre ved brug af den venstre

Hanois tårne

Hanois tårne

- Bemærk at denne løsning er rekursiv

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`
    - `flyt( antal-1, start, temp, slut );`
      `flyt( 1, start, slut, temp );`
      `flyt( antal-1, temp, slut, start );`

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`
    - `flyt( antal-1, start, temp, slut );`
      `flyt( 1, start, slut, temp );`
      `flyt( antal-1, temp, slut, start );`
  - `}`

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`
    - `flyt( antal-1, start, temp, slut );`
      `flyt( 1, start, slut, temp );`
      `flyt( antal-1, temp, slut, start );`
  - `}`
- Men vi mangler stadig en ting

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`
    - `flyt( antal-1, start, temp, slut );`
      `flyt( 1, start, slut, temp );`
      `flyt( antal-1, temp, slut, start );`
  - `}`
- Men vi mangler stadig en ting
  - Hvad?

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`
    - `flyt( antal-1, start, temp, slut );`
      `flyt( 1, start, slut, temp );`
      `flyt( antal-1, temp, slut, start );`
  - `}`
- Men vi mangler stadig en ting
  - Hvad?
  - Et basistilfælde

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`
    - `flyt( antal-1, start, temp, slut );`
      `flyt( 1, start, slut, temp );`
      `flyt( antal-1, temp, slut, start );`
  - `}`
- Men vi mangler stadig en ting
  - Hvad?
- Et basistilfælde
  - Hvad er det så?

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`
    - `flyt( antal-1, start, temp, slut );`
      `flyt( 1, start, slut, temp );`
      `flyt( antal-1, temp, slut, start );`
  - `}`
- Men vi mangler stadig en ting
  - Hvad?
- Et basistilfælde
  - Hvad er det så?
  - At flytte én skive fra start til slut

Hanois tårne

- Bemærk at denne løsning er rekursiv
  - `flyt( antal, start, slut, temp ) {`
    - `flyt( antal-1, start, temp, slut );`
      `flyt( 1, start, slut, temp );`
      `flyt( antal-1, temp, slut, start );`
  - `}`
- Men vi mangler stadig en ting
  - Hvad?
- Et basistilfælde
  - Hvad er det så?
  - At flytte én skive fra start til slut
    - Vi flytter den bare direkte

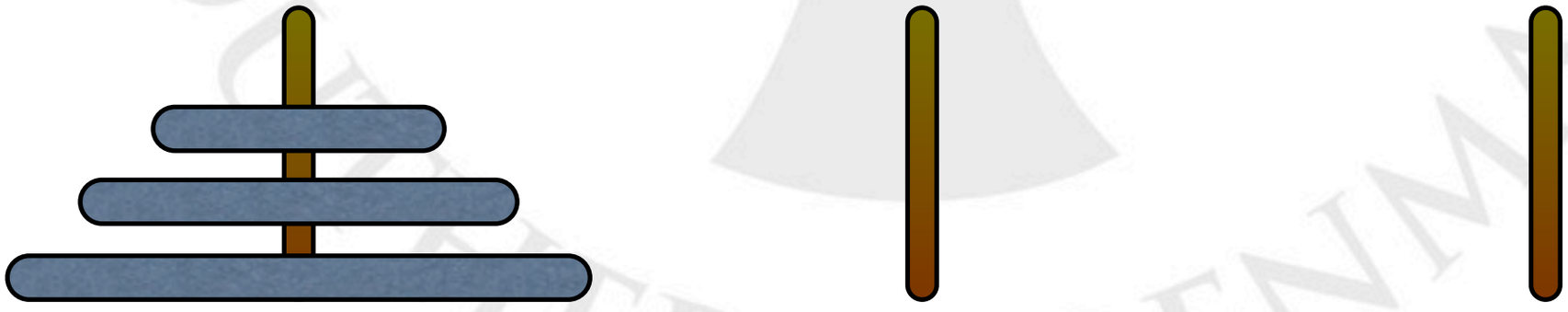
Hanois tårne

- Vi nummererer pindene fra venstre mod højre: 1, 2 og 3

```
public void move( int n, int start, int end, int temp ) {
    if( n == 1 )
        moveOneDisk( start, end );
    else {
        move( n-1, start, temp, end );
        move( 1, start, end, temp );
        move( n-1, temp, end, start );
    }
}
```

Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```



Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

move(3,1,3,2)

Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

move(3,1,3,2)

move(2,1,2,3)

Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

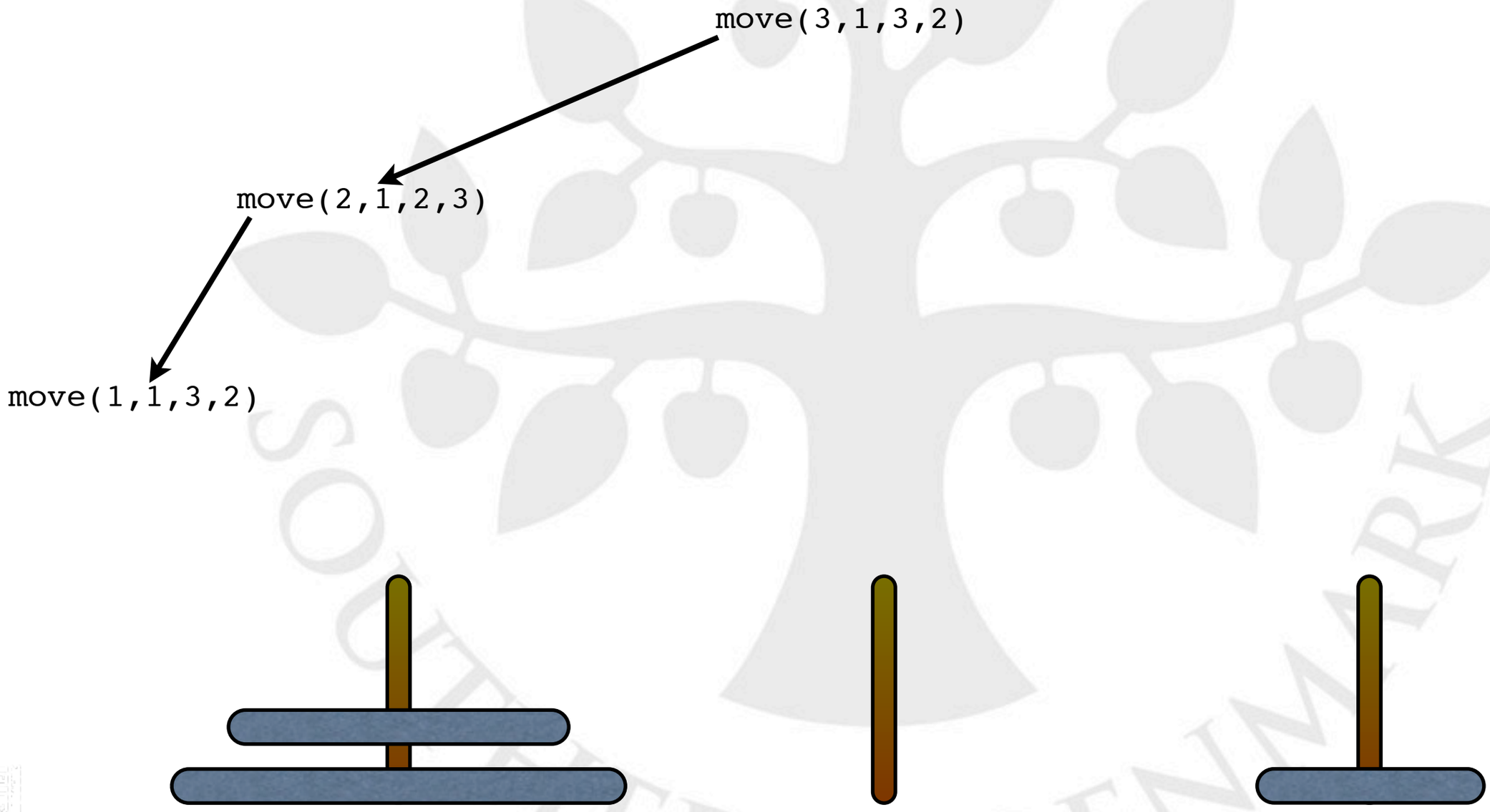
move(3,1,3,2)

move(2,1,2,3)

move(1,1,3,2)

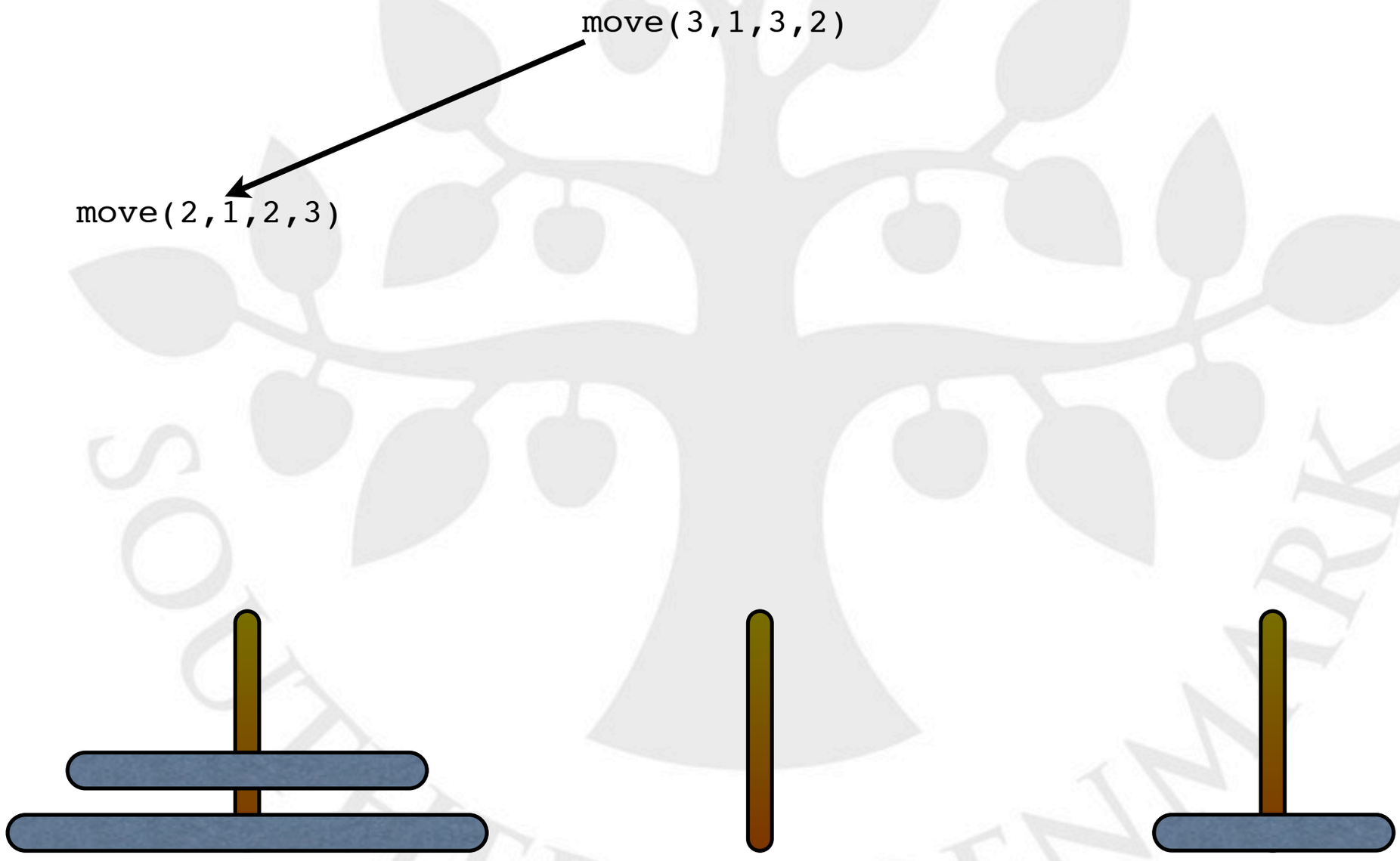
Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```



Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

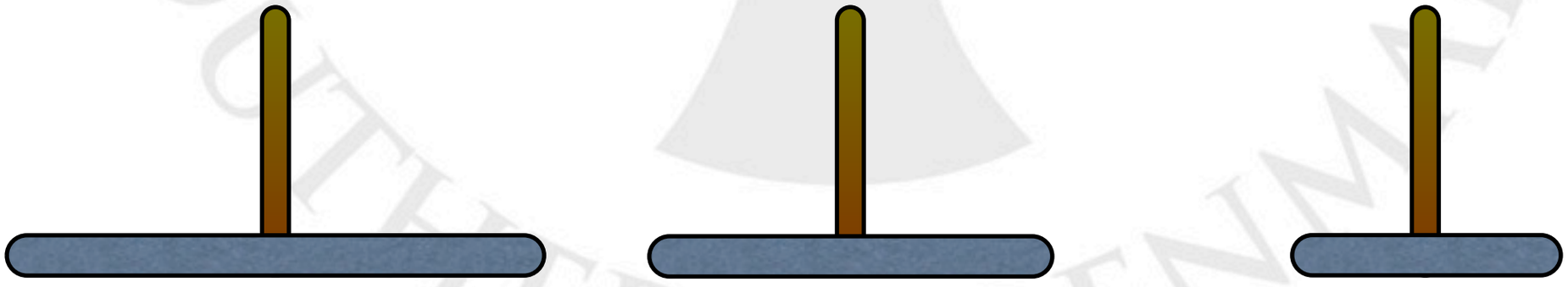


Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

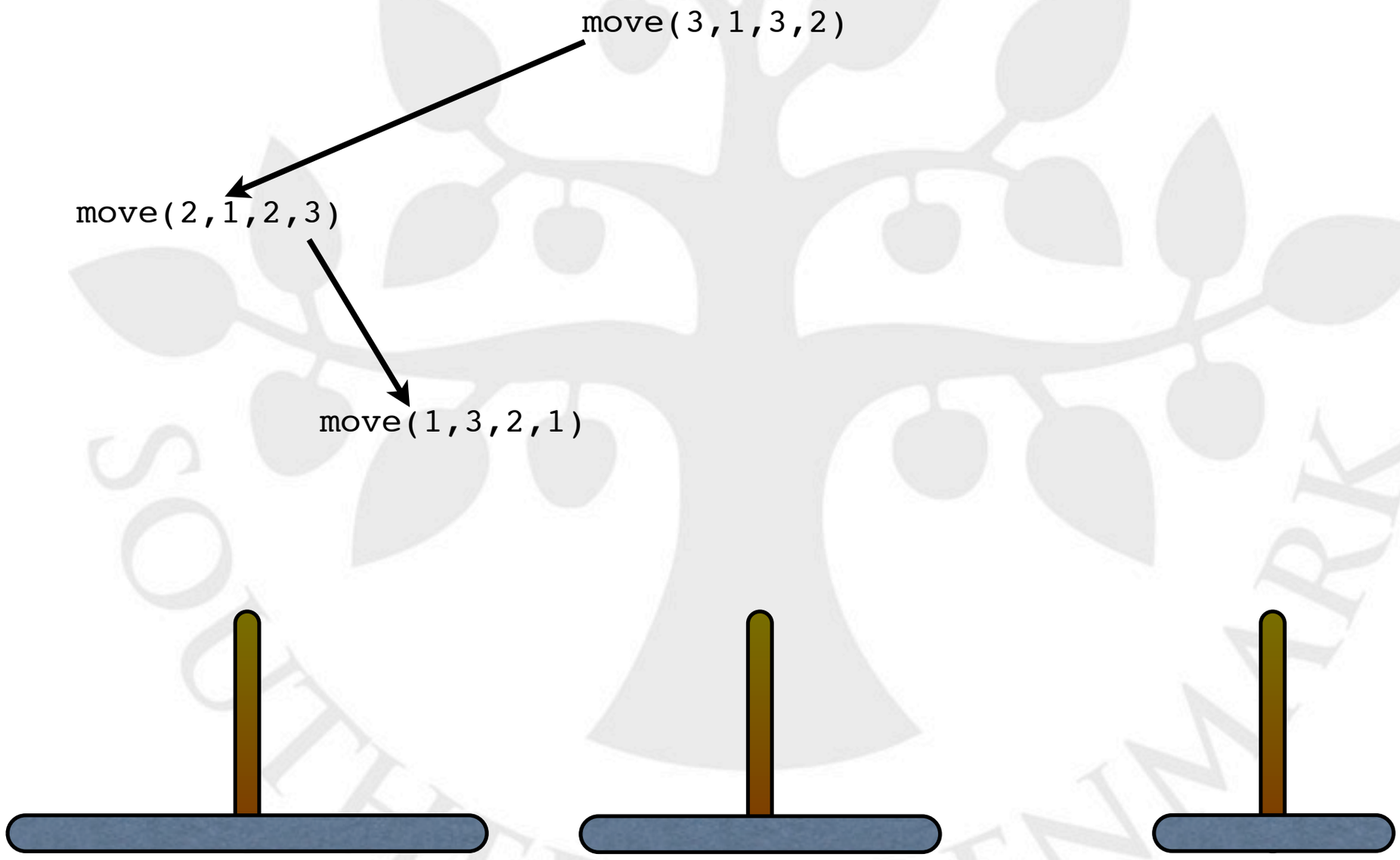
move(3,1,3,2)

move(2,1,2,3)



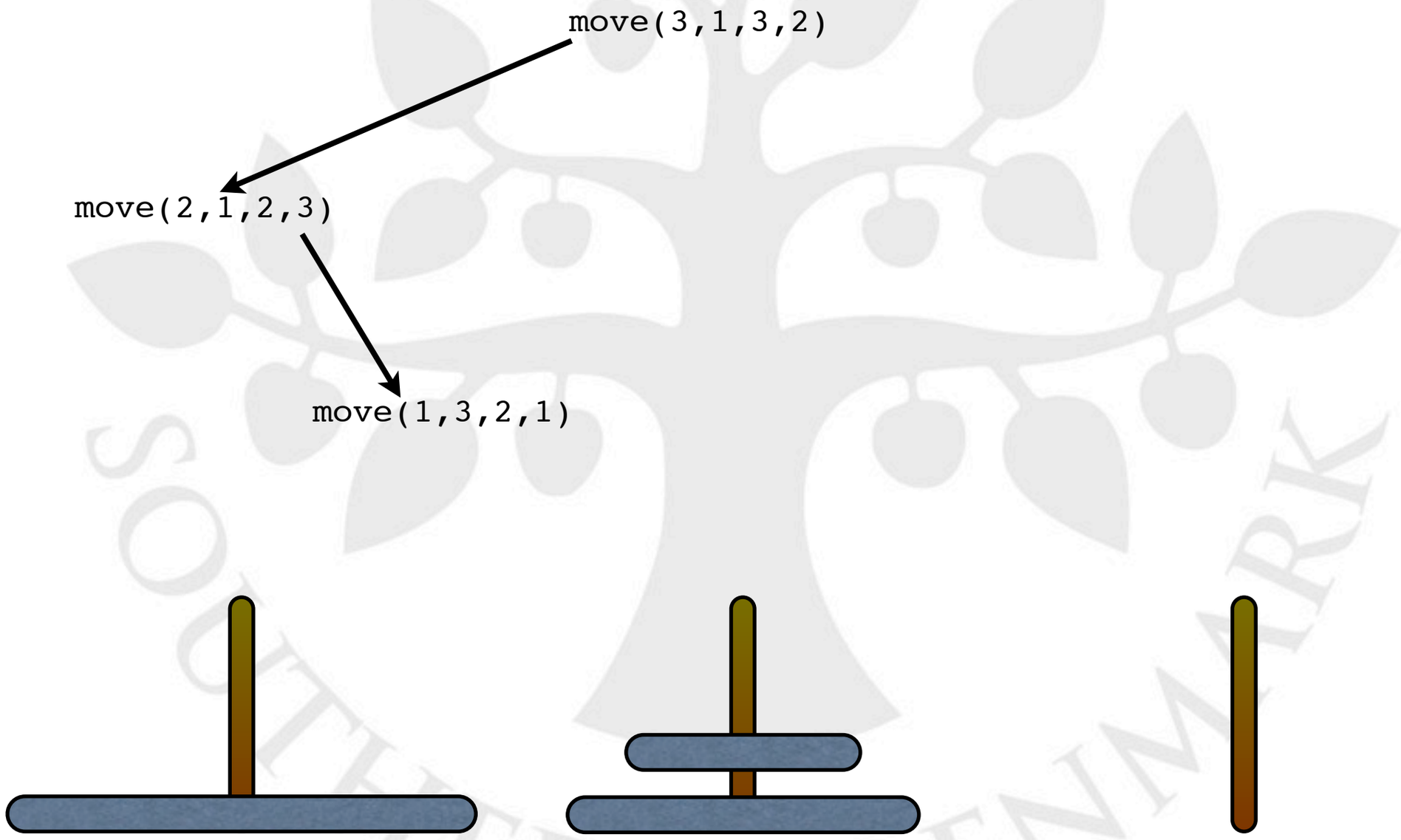
Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```



Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

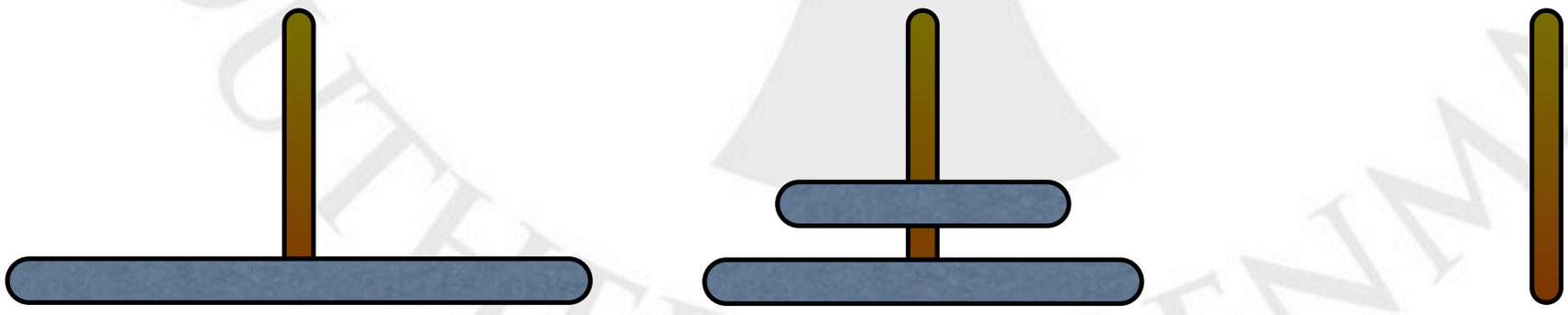


Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

move(3,1,3,2)

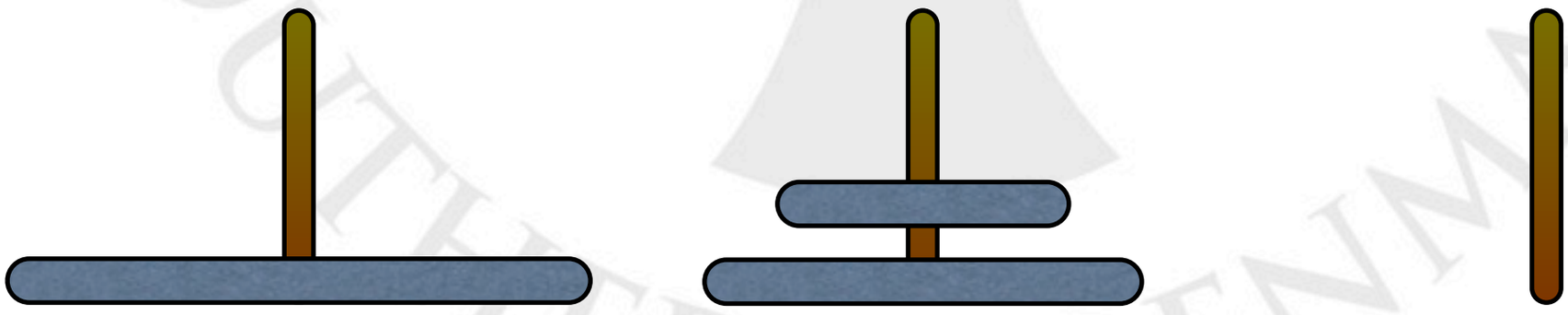
move(2,1,2,3)



Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

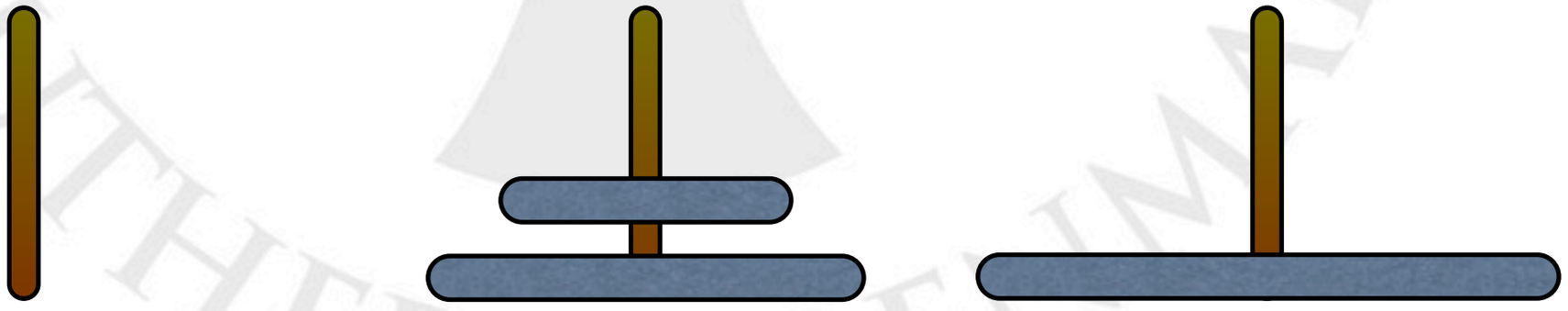
move(3,1,3,2)



Hanois tårne

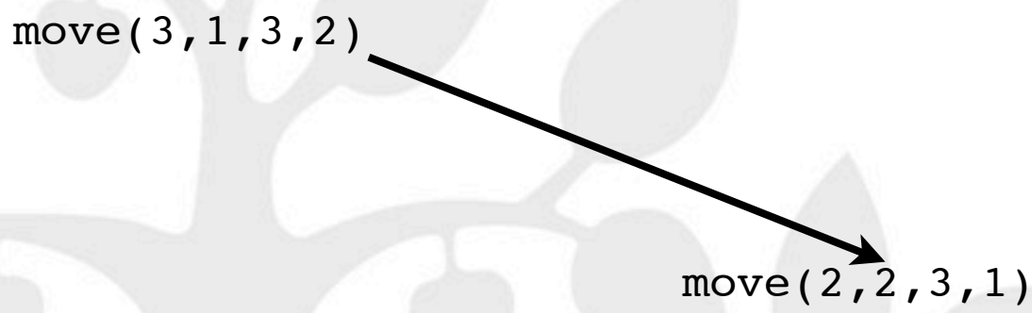
```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

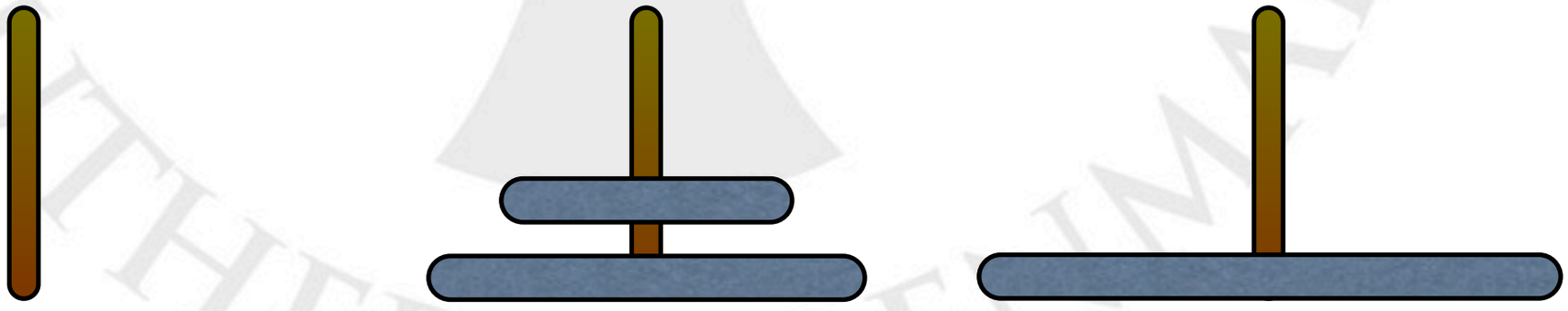
move(3,1,3,2)



Hanois tårne

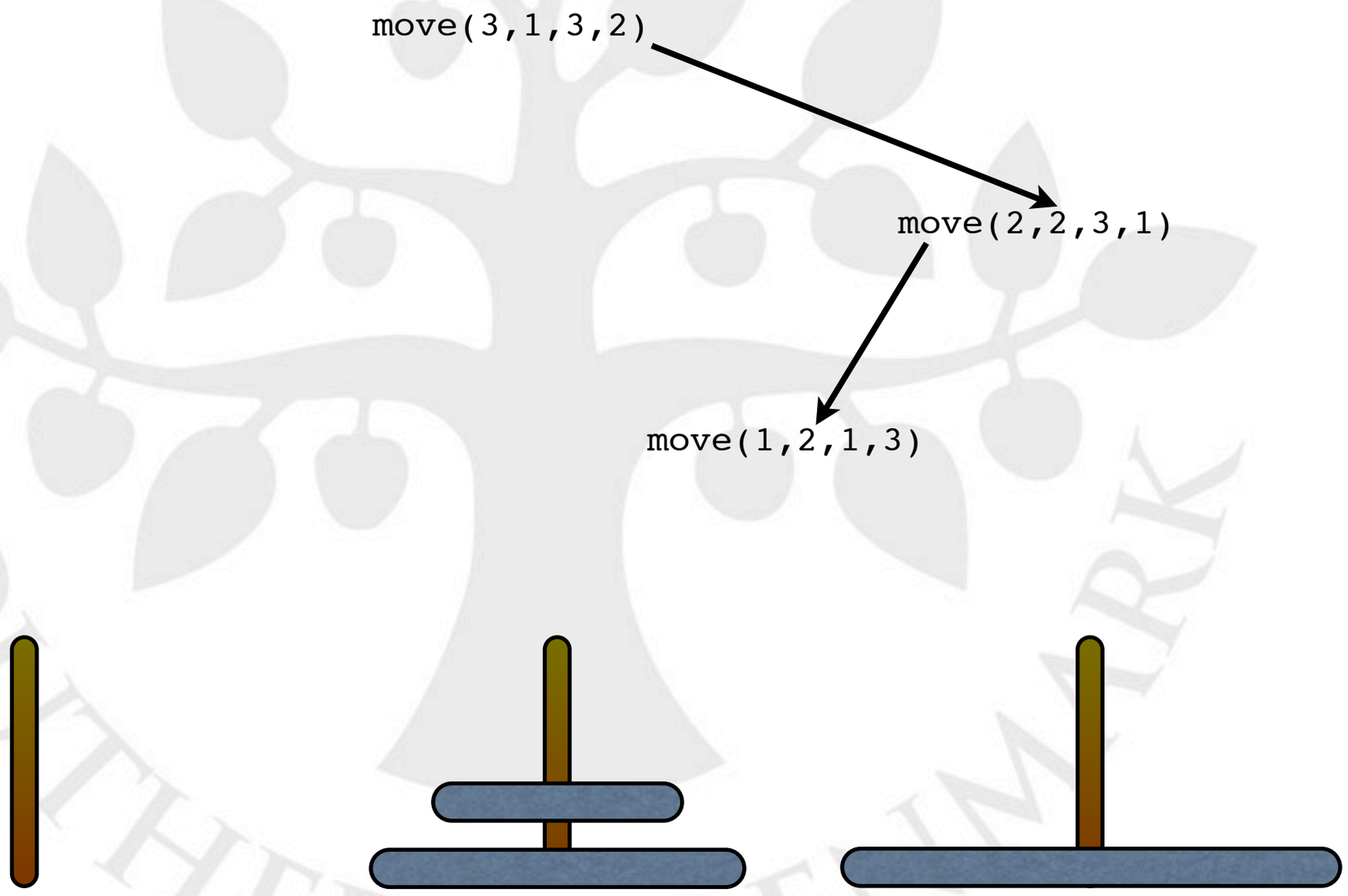
```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```





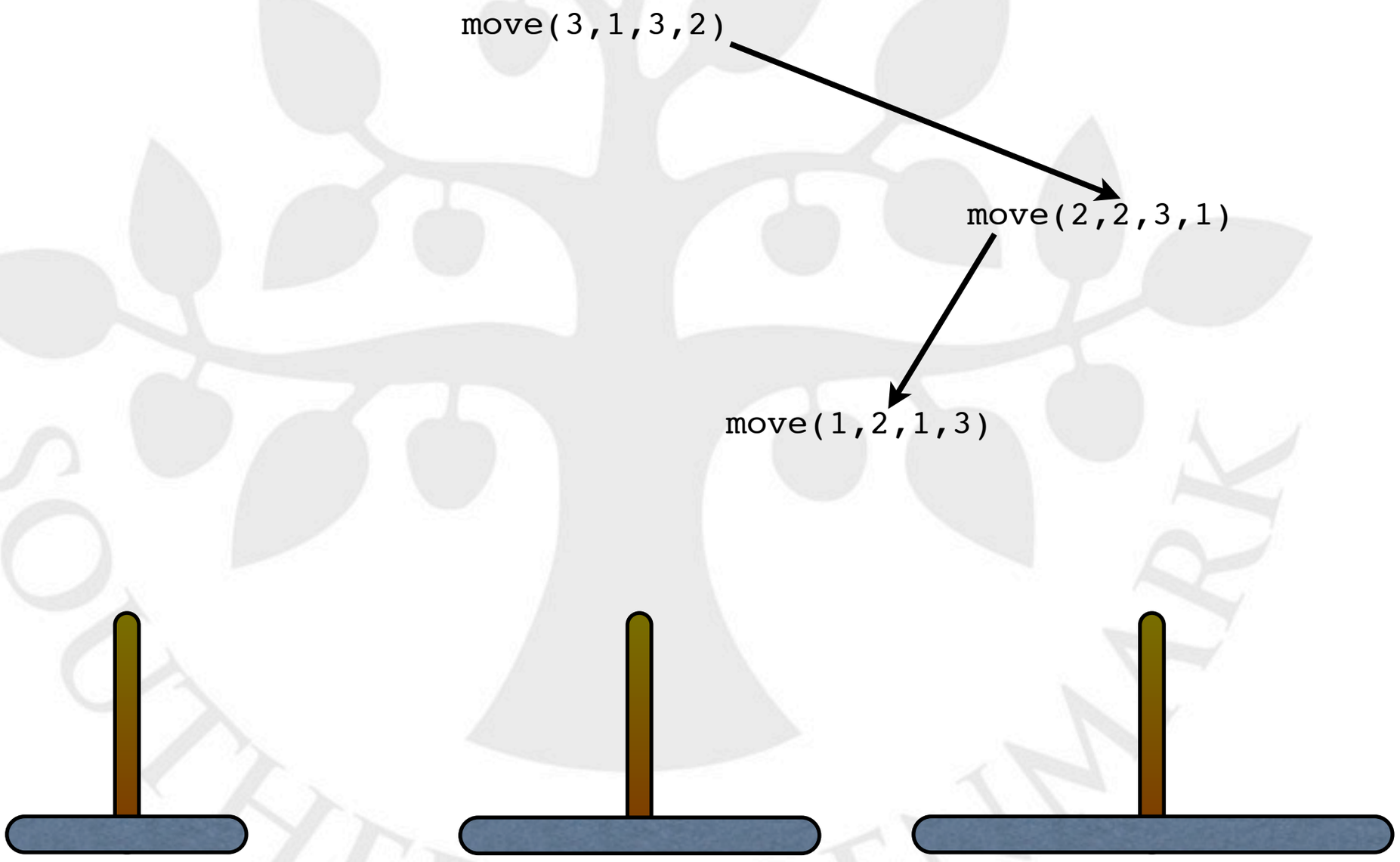
Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```



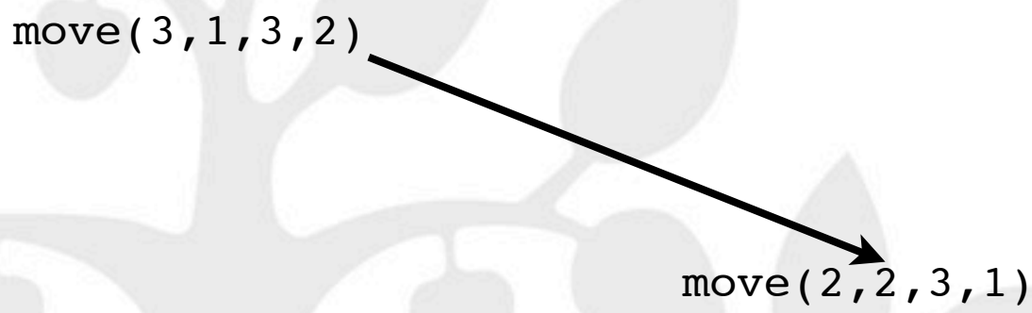
Hanois tårne

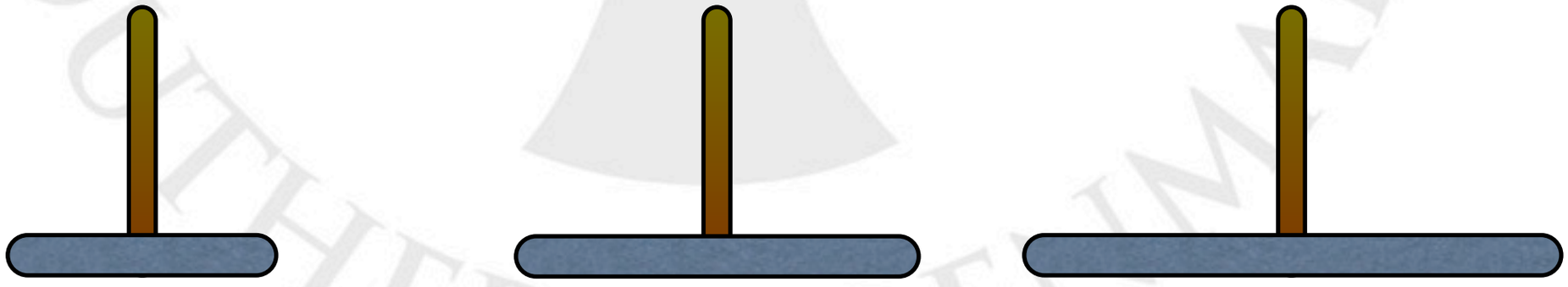
```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```



Hanois tårne

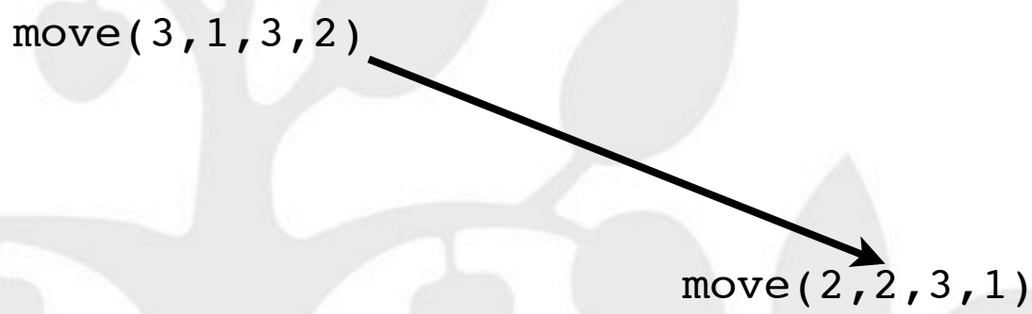
```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

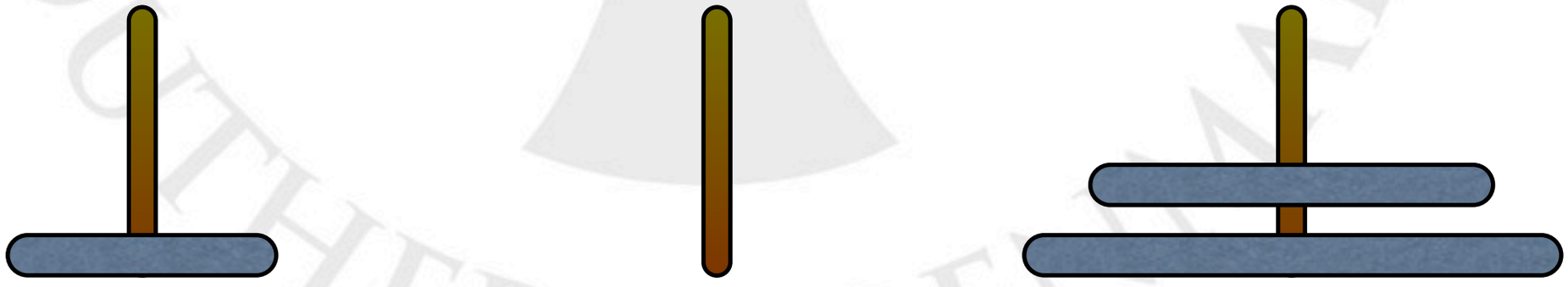




Hanois tårne

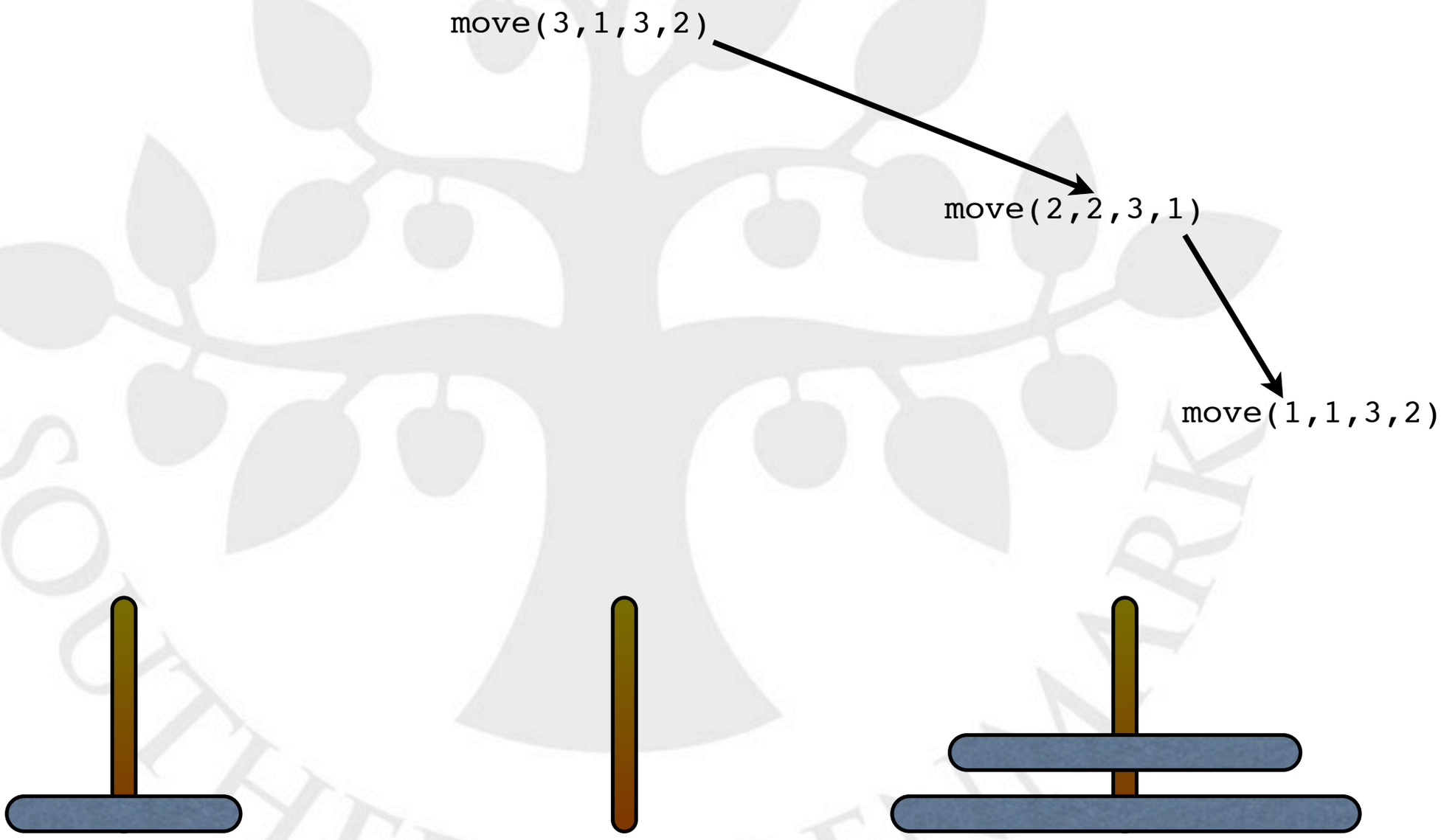
```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```





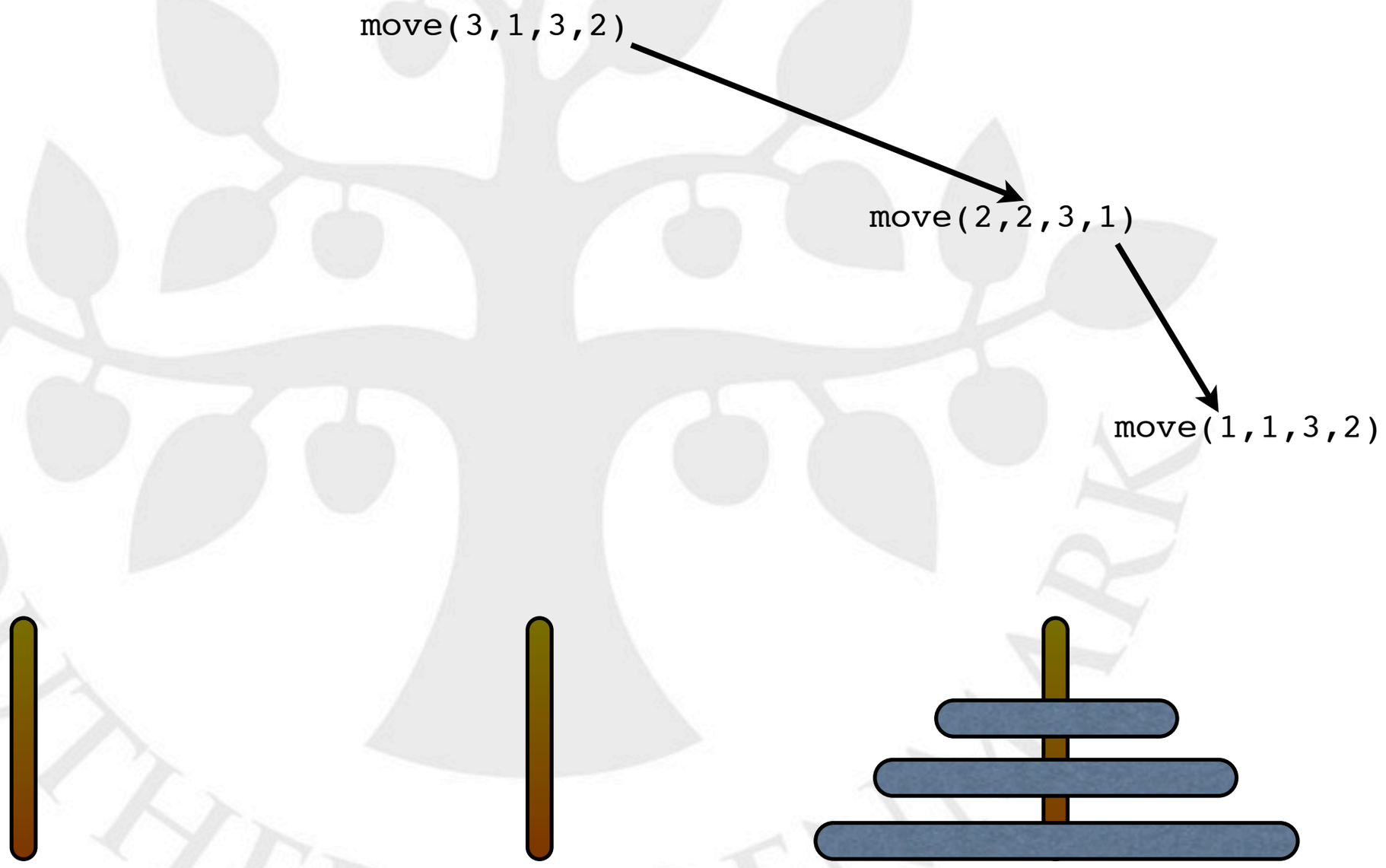
Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```



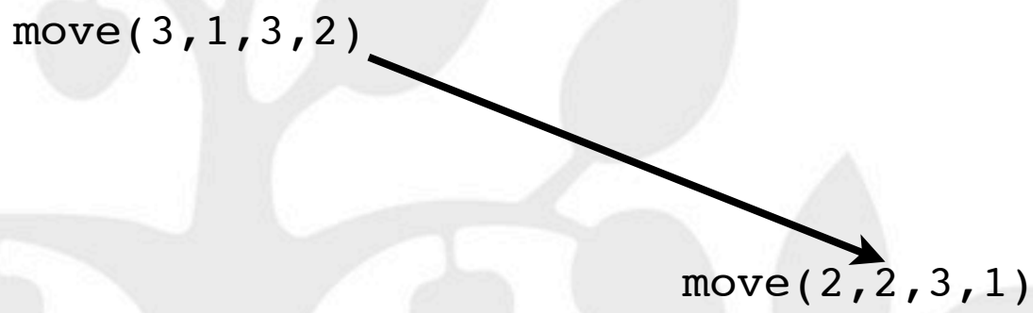
Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```



Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```



Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

move(3,1,3,2)

Hanois tårne

```
move(n, start, end, temp) {
  if(n == 1)
    moveOneDisk(start, end);
  else {
    move(n-1, start, temp, end);
    move(1, start, end, temp);
    move(n-1, temp, end, start);
  }
}
```

Hanois tårne

Hanois tårne

- Rekursiv løsning

Hanois tårne

- Rekursiv løsning
  - Elegant

Hanois tårne

- Rekursiv løsning
  - Elegant
  - Ineffektiv

Hanois tårne

- Rekursiv løsning
  - Elegant
  - Ineffektiv
    - For n skiver skal der bruges $2^n - 1$ flytninger

Hanois tårne

- Rekursiv løsning
  - Elegant
  - Ineffektiv
    - For n skiver skal der bruges $2^n - 1$ flytninger
    - Iflg. sagnet om de Brahmanske præster, så findes der et tempel hvor præster flytter 64 skiver lavet af guld, og når de er færdige går Verden under

Hanois tårne

- Rekursiv løsning
  - Elegant
  - Ineffektiv
    - For n skiver skal der bruges $2^n - 1$ flytninger
    - Iflg. sagnet om de Brahmanske præster, så findes der et tempel hvor præster flytter 64 skiver lavet af guld, og når de er færdige går Verden under
  - Bare rolig, hvis de flytter en skive hvert sekund, så tager det stadig ca. 600 millarder år

Hanois tårne

- Rekursiv løsning
  - Elegant
  - Ineffektiv
    - For n skiver skal der bruges $2^n - 1$ flytninger
    - Iflg. sagnet om de Brahmanske præster, så findes der et tempel hvor præster flytter 64 skiver lavet af guld, og når de er færdige går Verden under
      - Bare rolig, hvis de flytter en skive hvert sekund, så tager det stadig ca. 600 millarder år
      - Universets alder er kun omkring 14 millarder år

Rekursion

- Rekursion
  - "Når noget er defineret vha. sig selv"
    - Fx datastrukturer, såsom lister
    - Fx funktioner, såsom fakultet, fibonacci, ...
  - Skal indeholde mindst et basistilfælde
- Recursion: If you still don't get it, See: "Recursion".