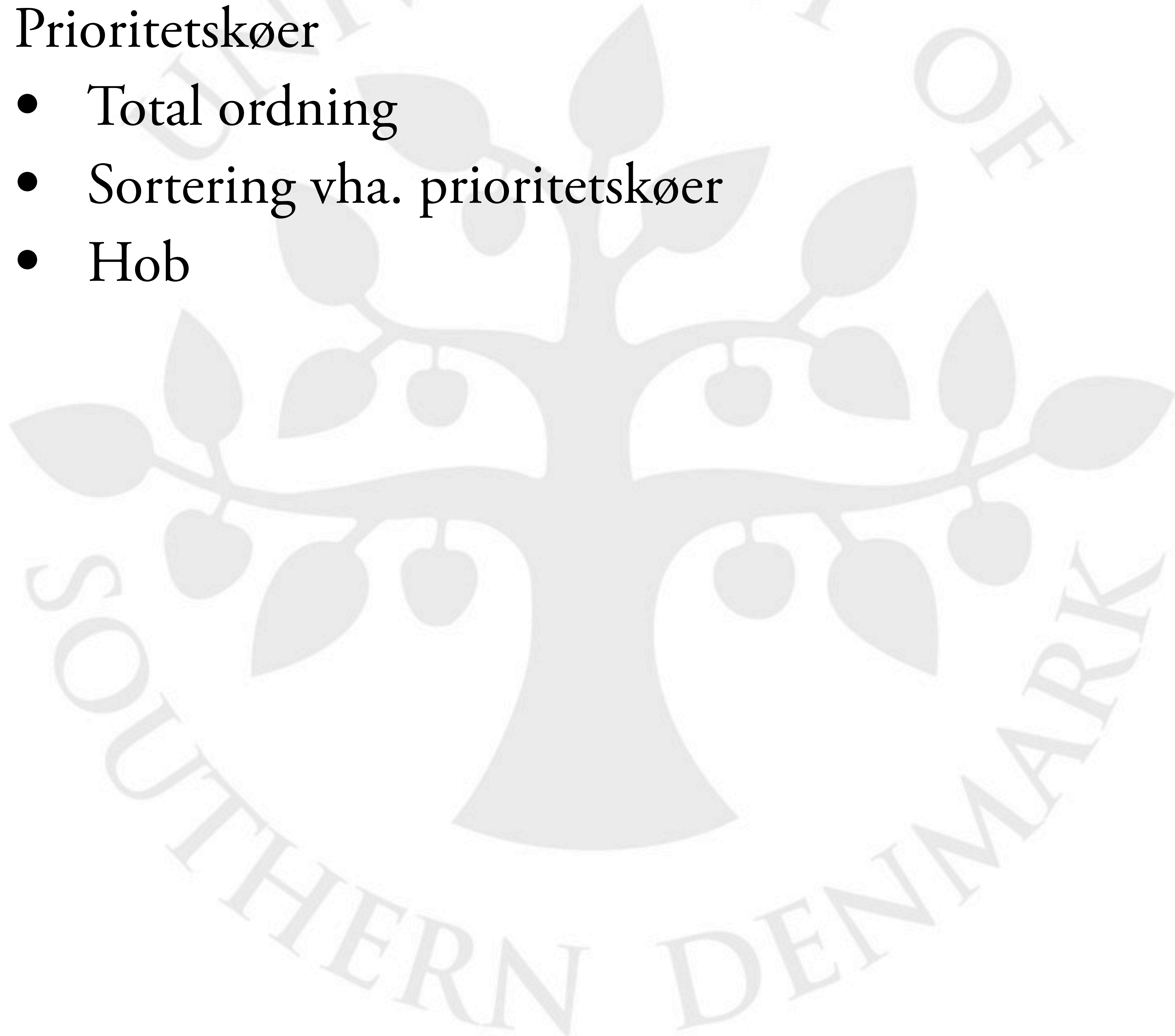
DM503

Forelæsning 6

Indhold

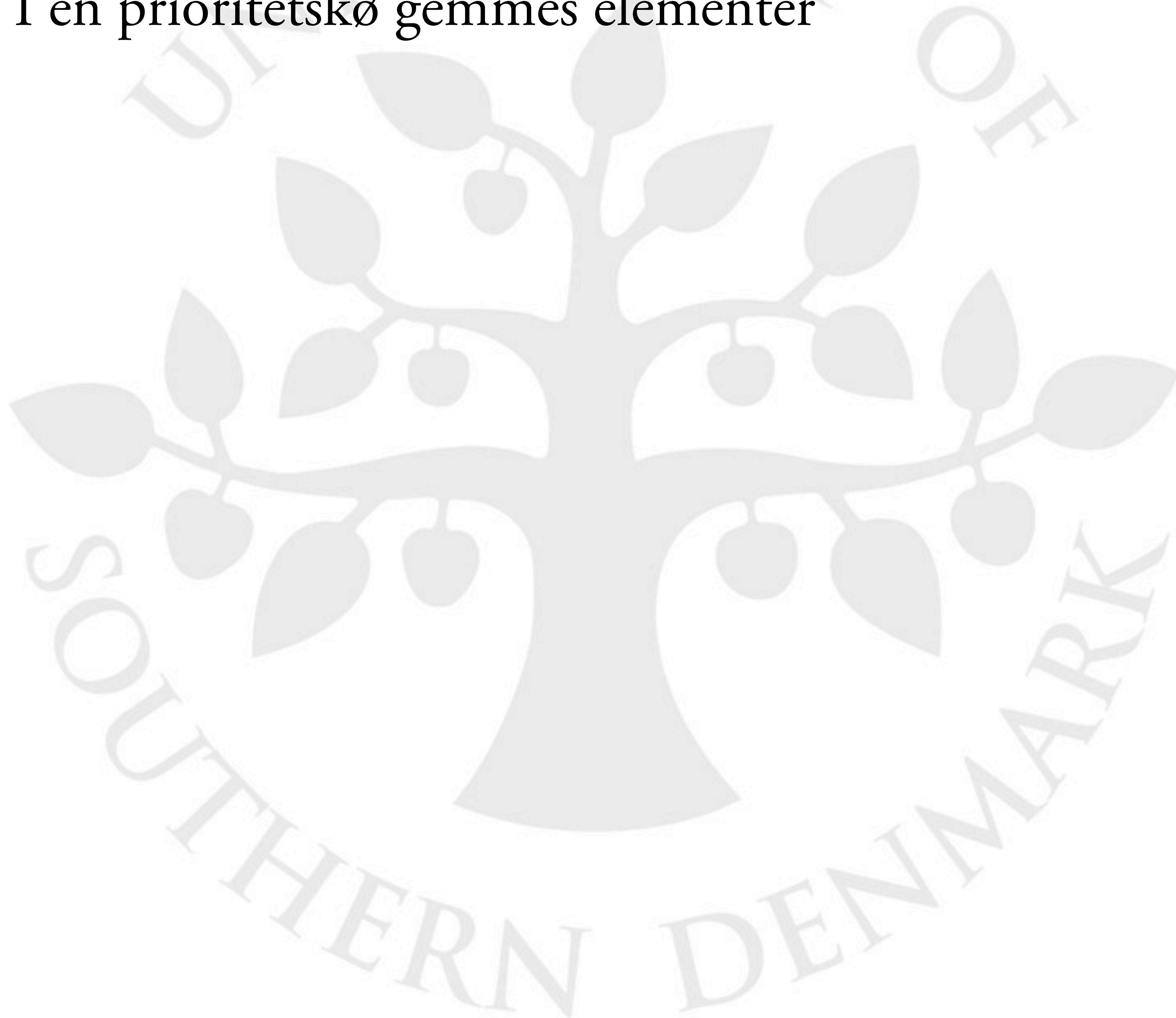


- Prioritetskøer
  - Total ordning
  - Sortering vha. prioritetskøer
  - Hob

Prioritetskø

University of Southern Denmark • Syddansk Universitet

Prioritetskø



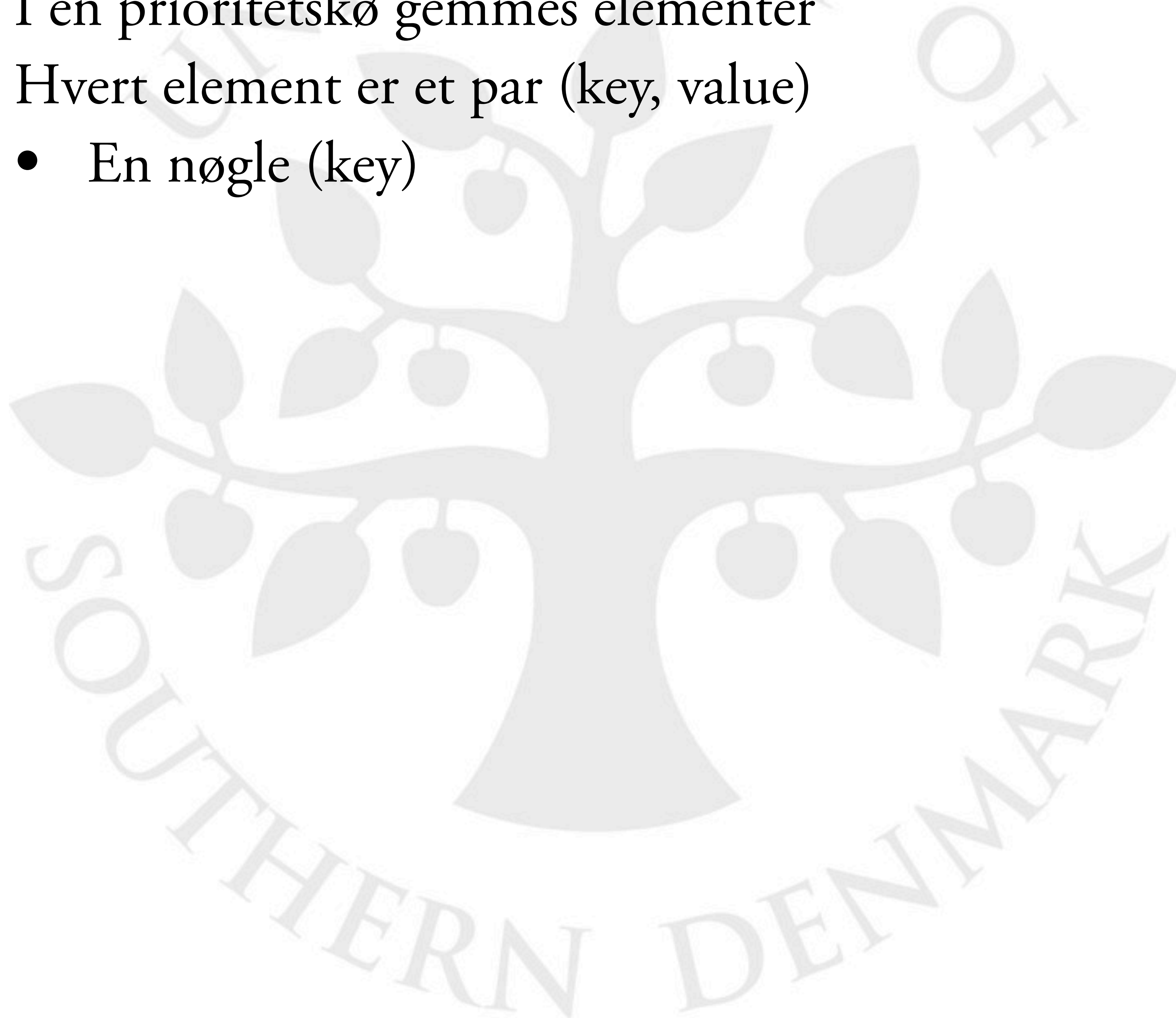
- I en prioritetskø gemmes elementer

Prioritetskø

- I en prioritetskø gemmes elementer
- Hvert element er et par (key, value)

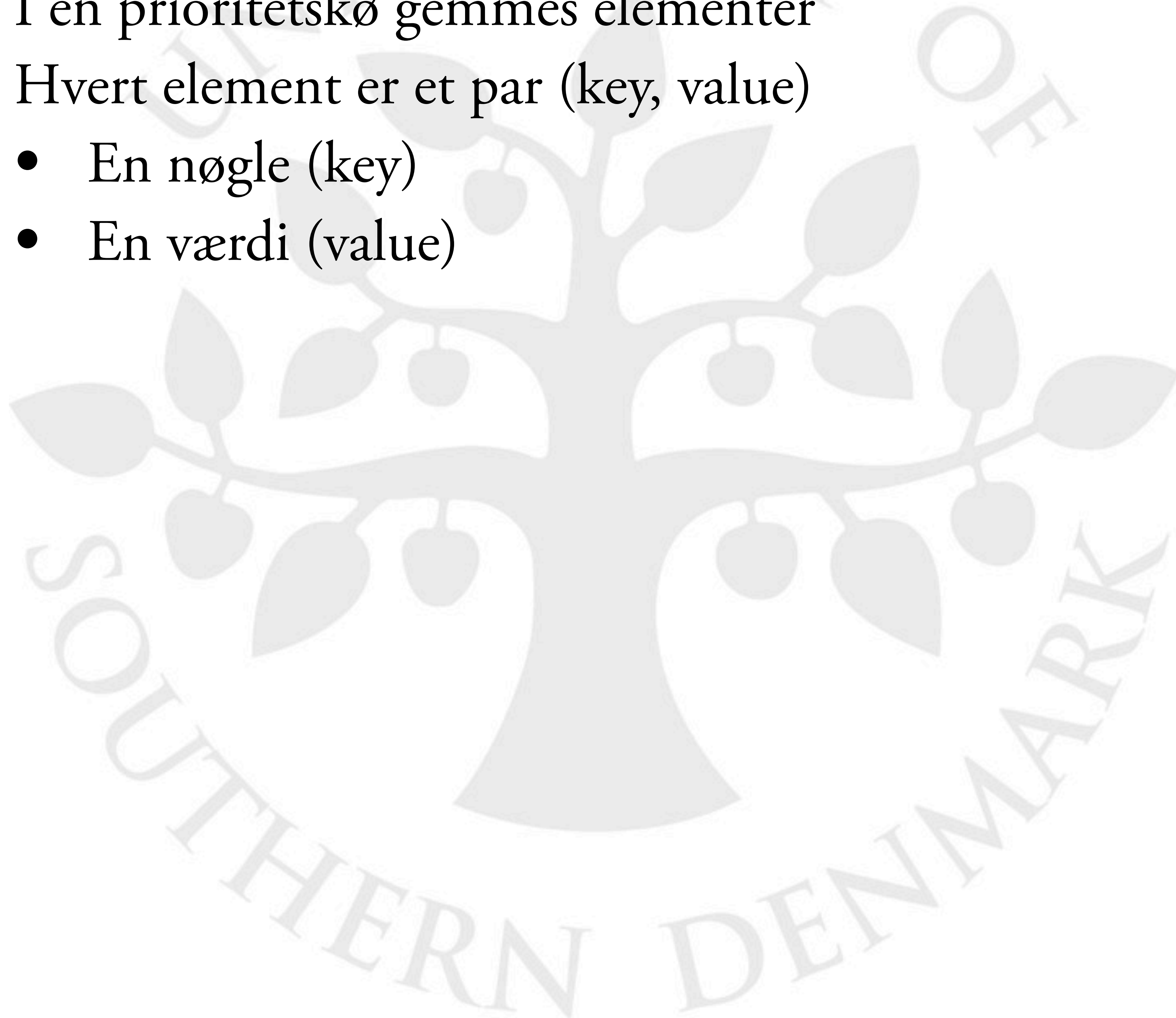
Prioritetskø

- I en prioritetskø gemmes elementer
- Hvert element er et par (key, value)
  - En nøgle (key)



Prioritetskø

- I en prioritetskø gemmes elementer
- Hvert element er et par (key, value)
  - En nøgle (key)
  - En værdi (value)



University of Southern Denmark • Syddansk Universitet

Prioritetskø

- I en prioritetskø gemmes elementer
- Hvert element er et par (key, value)
    - En nøgle (key)
    - En værdi (value)
- Værdien er typisk det data man gerne vil have gemt

Prioritetskø

- I en prioritetskø gemmes elementer
- Hvert element er et par (key, value)
  - En nøgle (key)
  - En værdi (value)
- Værdien er typisk det data man gerne vil have gemt
- Nøglen er den prioritet elementet har

Prioritetskø

- I en prioritetskø gemmes elementer
- Hvert element er et par (key, value)
    - En nøgle (key)
    - En værdi (value)
- Værdien er typisk det data man gerne vil have gemt
- Nøglen er den prioritet elementet har
    - Prioriteten af værdien

University of Southern Denmark • Syddansk Universitet

Prioritetskø

- I en prioritetskø gemmes elementer
- Hvert element er et par (key, value)
  - En nøgle (key)
  - En værdi (value)
- Værdien er typisk det data man gerne vil have gemt
- Nøglen er den prioritet elementet har
  - Prioriteten af værdien
  - Nøgle med lille værdi == Høj prioritet

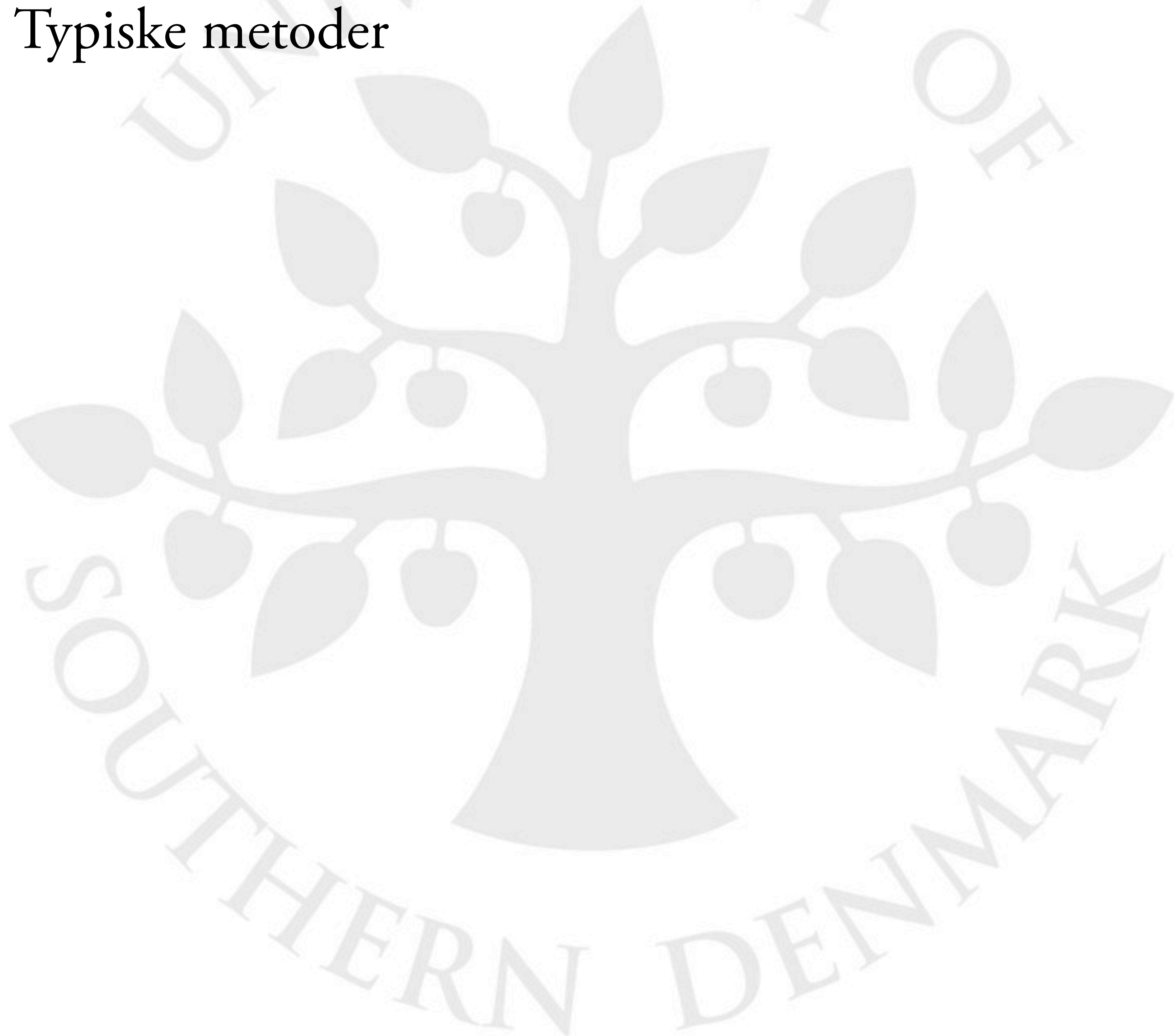
Prioritetskø

- I en prioritetskø gemmes elementer
- Hvert element er et par (key, value)
    - En nøgle (key)
    - En værdi (value)
- Værdien er typisk det data man gerne vil have gemt
- Nøglen er den prioritet elementet har
    - Prioriteten af værdien
    - Nøgle med lille værdi == Høj prioritet
        - (0, x) har højere prioritet end (20, y)

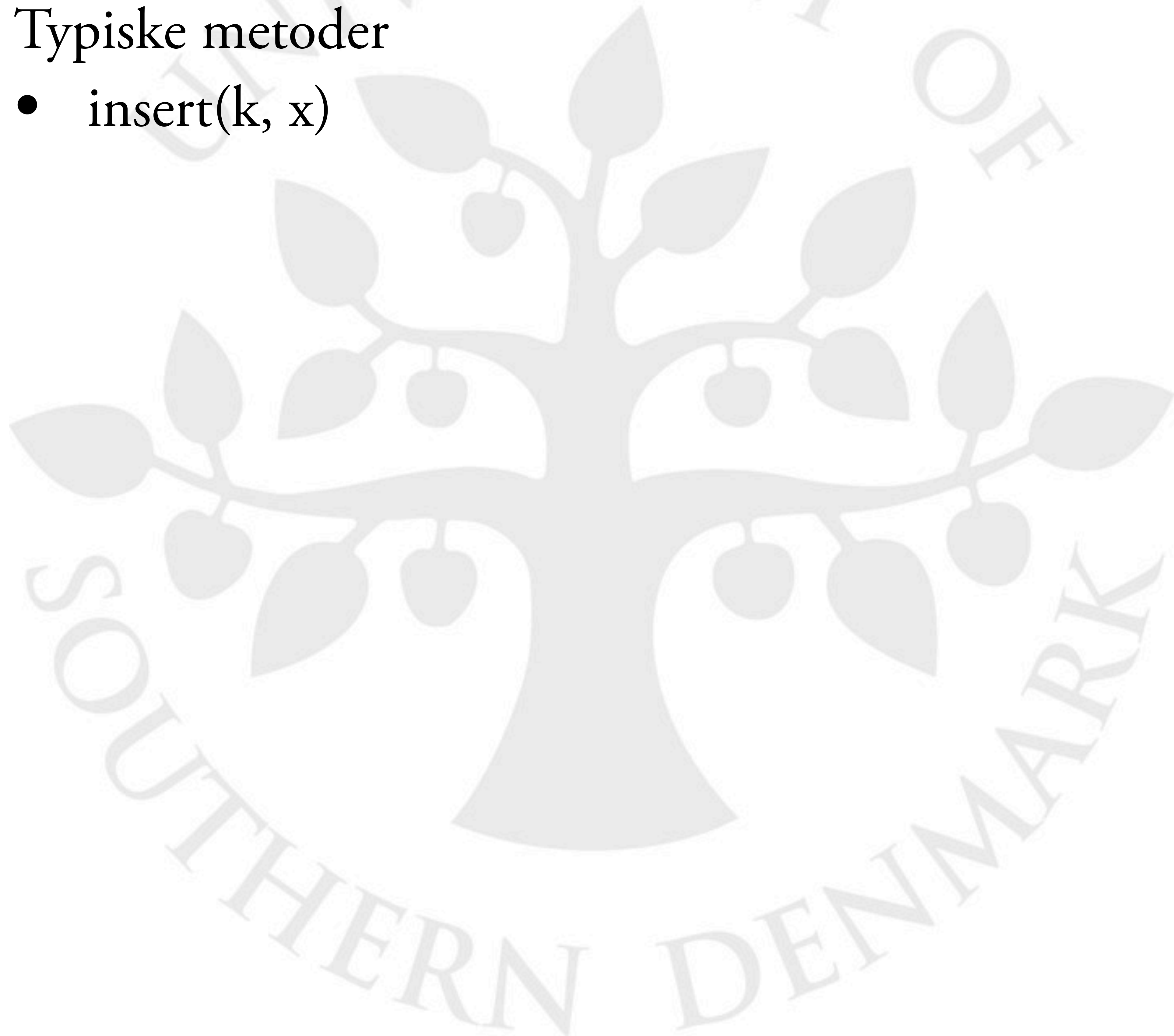
Prioritetskøer

Prioritetskøer

- Typiske metoder



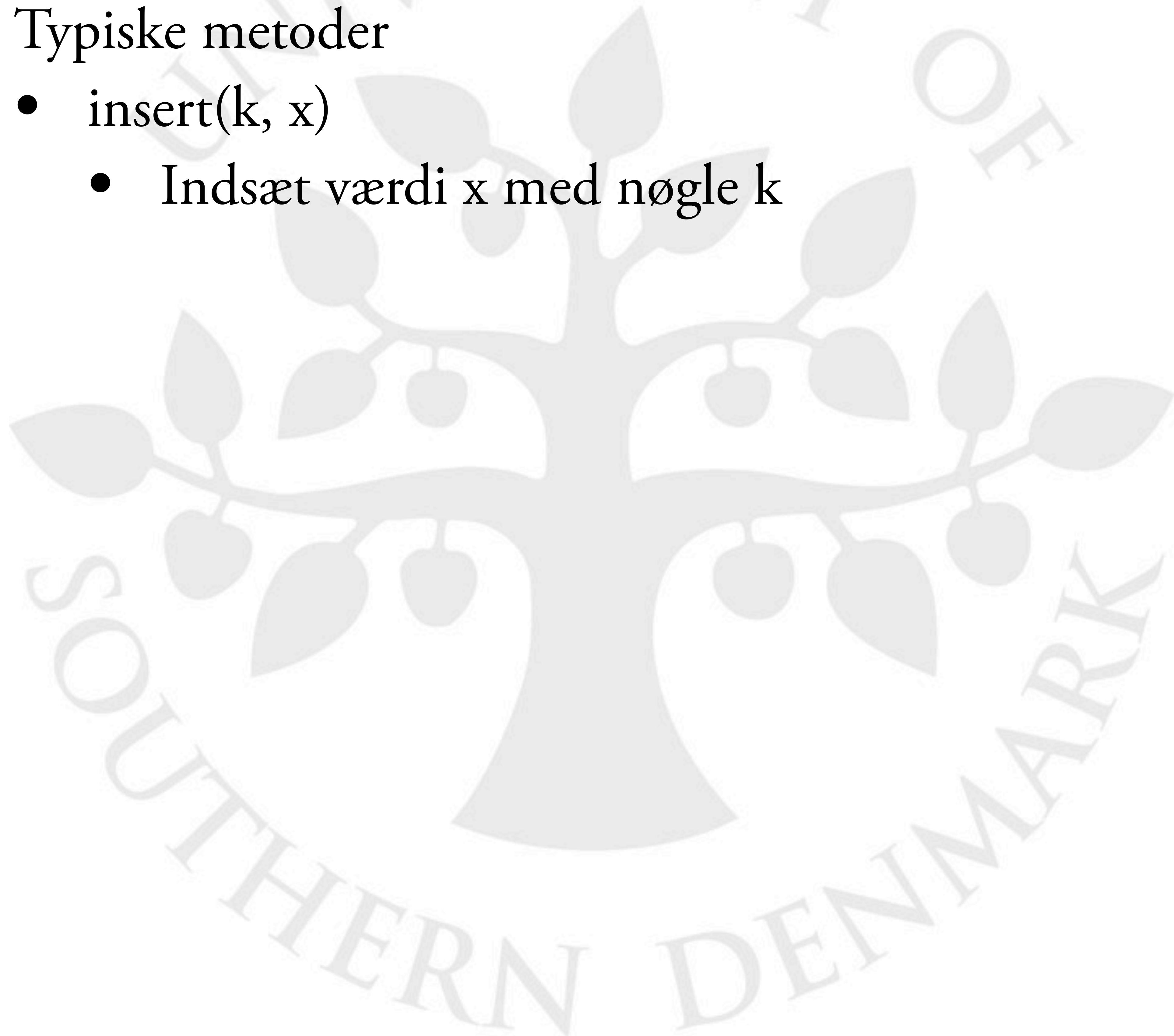
Prioritetskøer



- Typiske metoder
  - insert(k, x)

Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k



University of Southern Denmark • Syddansk Universitet

Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k
  - removeMin()

University of Southern Denmark • Syddansk Universitet

Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k
  - removeMin()
    - Udtager det element med den mindste nøgle (dvs. højest prioritet) i køen

Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k
  - removeMin()
    - Udtager det element med den mindste nøgle (dvs. højest prioritet) i køen
  - min()

Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k
  - removeMin()
    - Udtager det element med den mindste nøgle (dvs. højest prioritet) i køen
  - min()
    - Som removeMin() dog uden at fjerne elementet fra prioritetskøen

Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k
  - removeMin()
    - Udtager det element med den mindste nøgle (dvs. højest prioritet) i køen
  - min()
    - Som removeMin() dog uden at fjerne elementet fra prioritetskøen
  - size()

Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k
  - removeMin()
    - Udtager det element med den mindste nøgle (dvs. højest prioritet) i køen
  - min()
    - Som removeMin() dog uden at fjerne elementet fra prioritetskøen
  - size()
    - Antallet af elementer i køen

Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k
  - removeMin()
    - Udtager det element med den mindste nøgle (dvs. højest prioritet) i køen
  - min()
    - Som removeMin() dog uden at fjerne elementet fra prioritetskøen
  - size()
    - Antallet af elementer i køen
  - isEmpty()

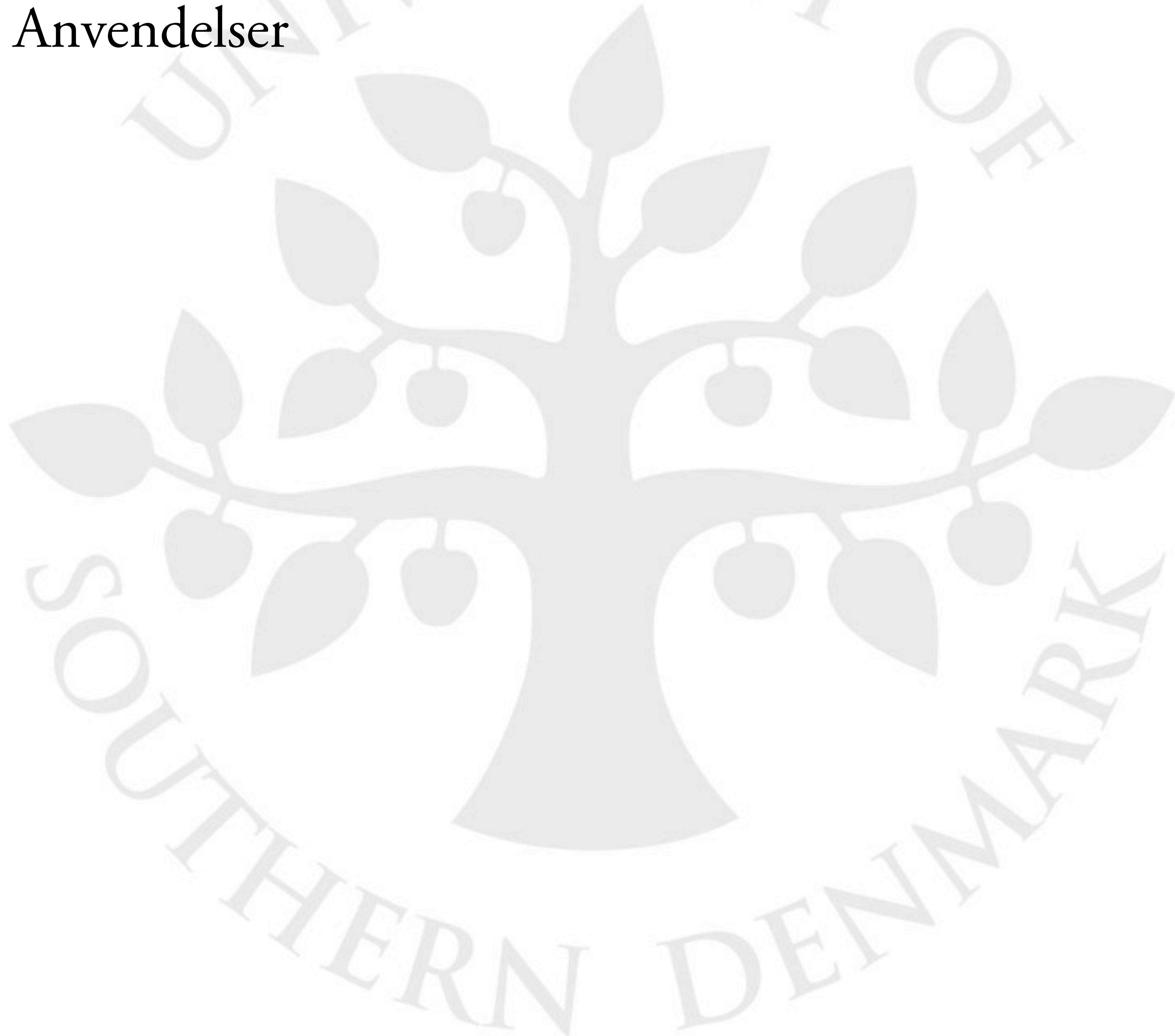
Prioritetskøer

- Typiske metoder
  - insert(k, x)
    - Indsæt værdi x med nøgle k
  - removeMin()
    - Udtager det element med den mindste nøgle (dvs. højest prioritet) i køen
  - min()
    - Som removeMin() dog uden at fjerne elementet fra prioritetskøen
  - size()
    - Antallet af elementer i køen
  - isEmpty()
    - Er køen tom

Prioritetskøer

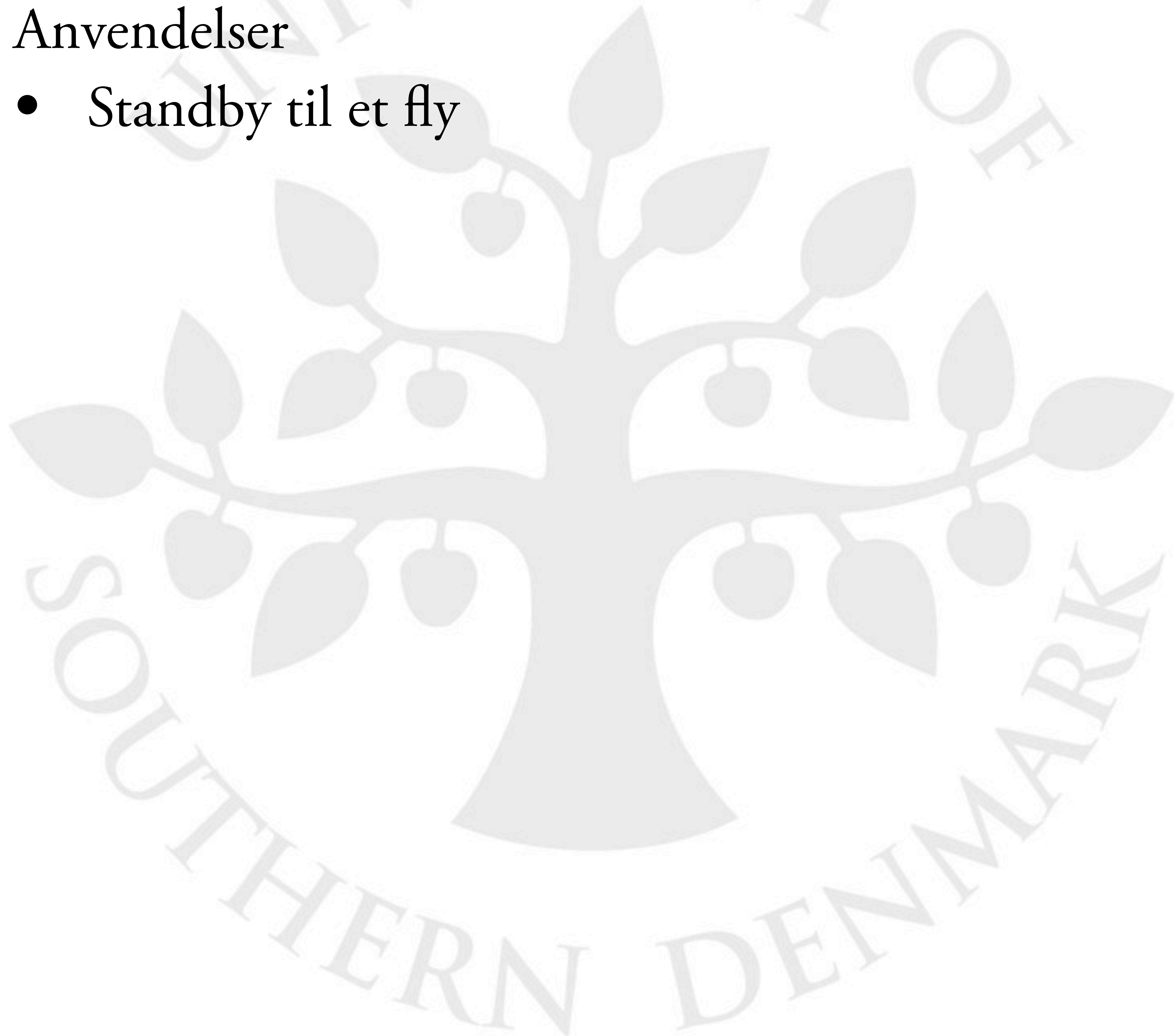
Prioritetskøer

- Anvendelser



Prioritetskøer

- Anvendelser
  - Standby til et fly



University of Southern Denmark • Syddansk Universitet

Prioritetskøer

- Anvendelser
  - Standby til et fly
  - Hvilket program skal næste gang få lov til at køre på CPU'en

Prioritetskøer

- Anvendelser
  - Standby til et fly
  - Hvilket program skal næste gang få lov til at køre på CPU'en
  - Auktioner

Prioritetskøer

- Anvendelser
  - Standby til et fly
  - Hvilket program skal næste gang få lov til at køre på CPU'en
  - Auktioner
  - Aktiemarkedet

Prioritetskøer

- Anvendelser
  - Standby til et fly
  - Hvilket program skal næste gang få lov til at køre på CPU'en
  - Auktioner
  - Aktiemarkedet
  - ...

University of Southern Denmark • Syddansk Universitet

Prioritetskøer

Prioritetskøer

- Hvad skal vi kræve af nøglerne?

Prioritetskøer

- Hvad skal vi kræve af nøglerne?
    - Kan de være hvad som helst?

University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

Prioritetskøer

- Hvad skal vi kræve af nøglerne?
  - Kan de være hvad som helst?
- Vi skal gentagne gange kunne finde det element med mindst nøgle

Prioritetskøer

- Hvad skal vi kræve af nøglerne?
    - Kan de være hvad som helst?
- Vi skal gentagne gange kunne finde det element med mindst nøgle
    - Hvad vil det sige at have mindst nøgle (k_{min})?

University of Southern Denmark • Syddansk Universitet

Prioritetskøer

- Hvad skal vi kræve af nøglerne?
  - Kan de være hvad som helst?
- Vi skal gentagne gange kunne finde det element med mindst nøgle
  - Hvad vil det sige at have mindst nøgle (k_{min})?
    - $\forall k \in K : k_{min} \leq k$

Prioritetskøer

- Hvad skal vi kræve af nøglerne?
  - Kan de være hvad som helst?
- Vi skal gentagne gange kunne finde det element med mindst nøgle
  - Hvad vil det sige at have mindst nøgle (k_{min})?
    - $\forall k \in K : k_{min} \leq k$
- Vi skal altså kunne sammenligne elementer

Total ordning

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem
  - To elementer kan godt have samme nøgle (prioritet)

University of Southern Denmark • Syddansk Universitet

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem
  - To elementer kan godt have samme nøgle (prioritet)
- Total ordning $\leq$

University of Southern Denmark • Syddansk Universitet

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem
  - To elementer kan godt have samme nøgle (prioritet)
- Total ordning $\leq$
  - Refleksiv

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem
  - To elementer kan godt have samme nøgle (prioritet)
- Total ordning $\leq$
  - Refleksiv
    - $x \leq x$

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem
  - To elementer kan godt have samme nøgle (prioritet)
- Total ordning $\leq$
  - Refleksiv
    - $x \leq x$
  - Anti-symmetri

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem
  - To elementer kan godt have samme nøgle (prioritet)
- Total ordning $\leq$
  - Refleksiv
    - $x \leq x$
  - Anti-symmetri
    - $x \leq y \wedge y \leq x \Rightarrow x = y$

University of Southern Denmark • Syddansk Universitet

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem
  - To elementer kan godt have samme nøgle (prioritet)
- Total ordning $\leq$
  - Refleksiv
    - $x \leq x$
  - Anti-symmetri
    - $x \leq y \wedge y \leq x \Rightarrow x = y$
  - Transitivitet

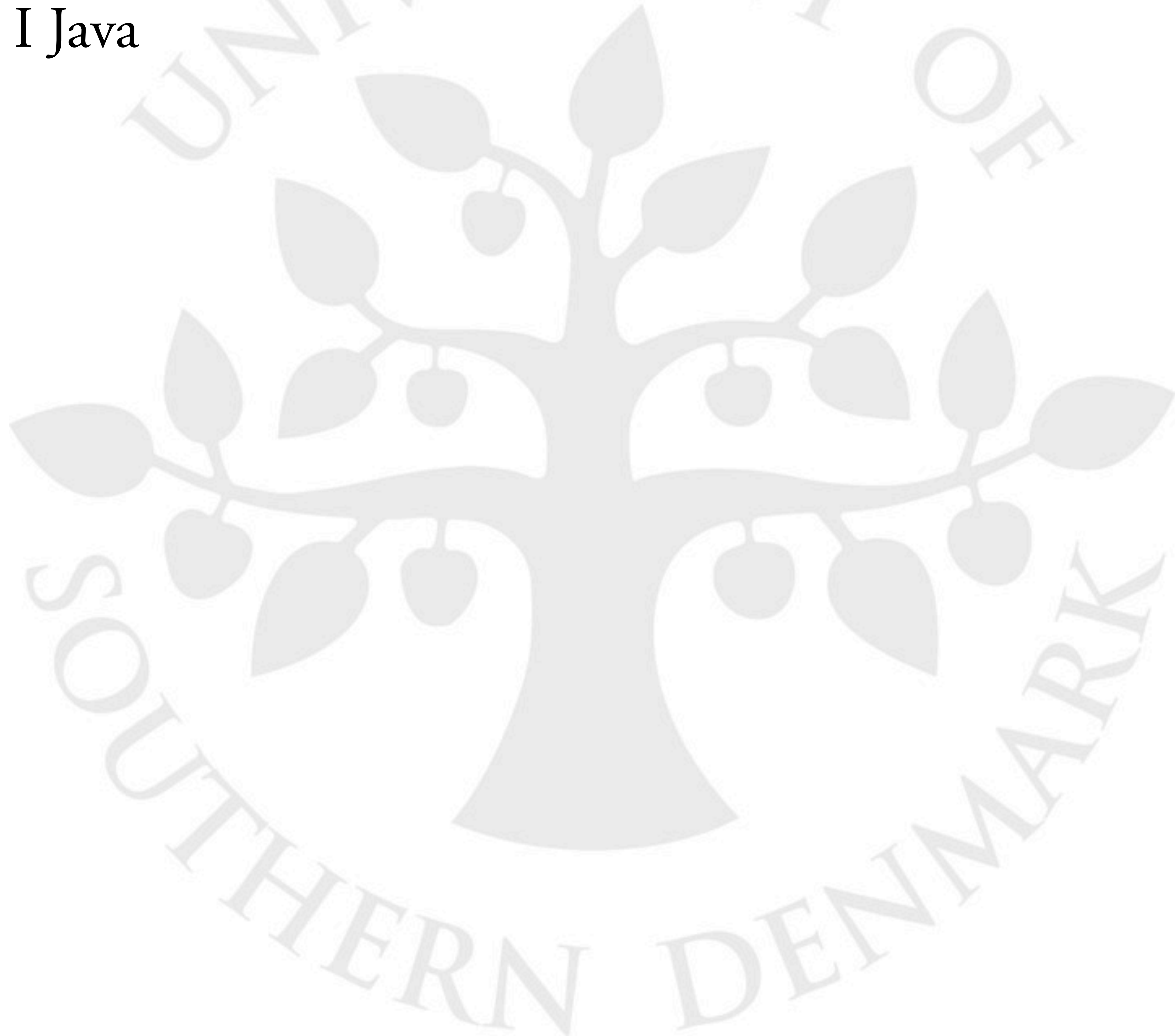
University of Southern Denmark • Syddansk Universitet

Total ordning

- Nøglerne kan altså være arbitrære objekter sålænge der er defineret en ordning på dem
  - To elementer kan godt have samme nøgle (prioritet)
- Total ordning $\leq$
  - Refleksiv
    - $x \leq x$
  - Anti-symmetri
    - $x \leq y \wedge y \leq x \Rightarrow x = y$
  - Transitivitet
    - $x \leq y \wedge y \leq z \Rightarrow x \leq z$

Prioritetskøer

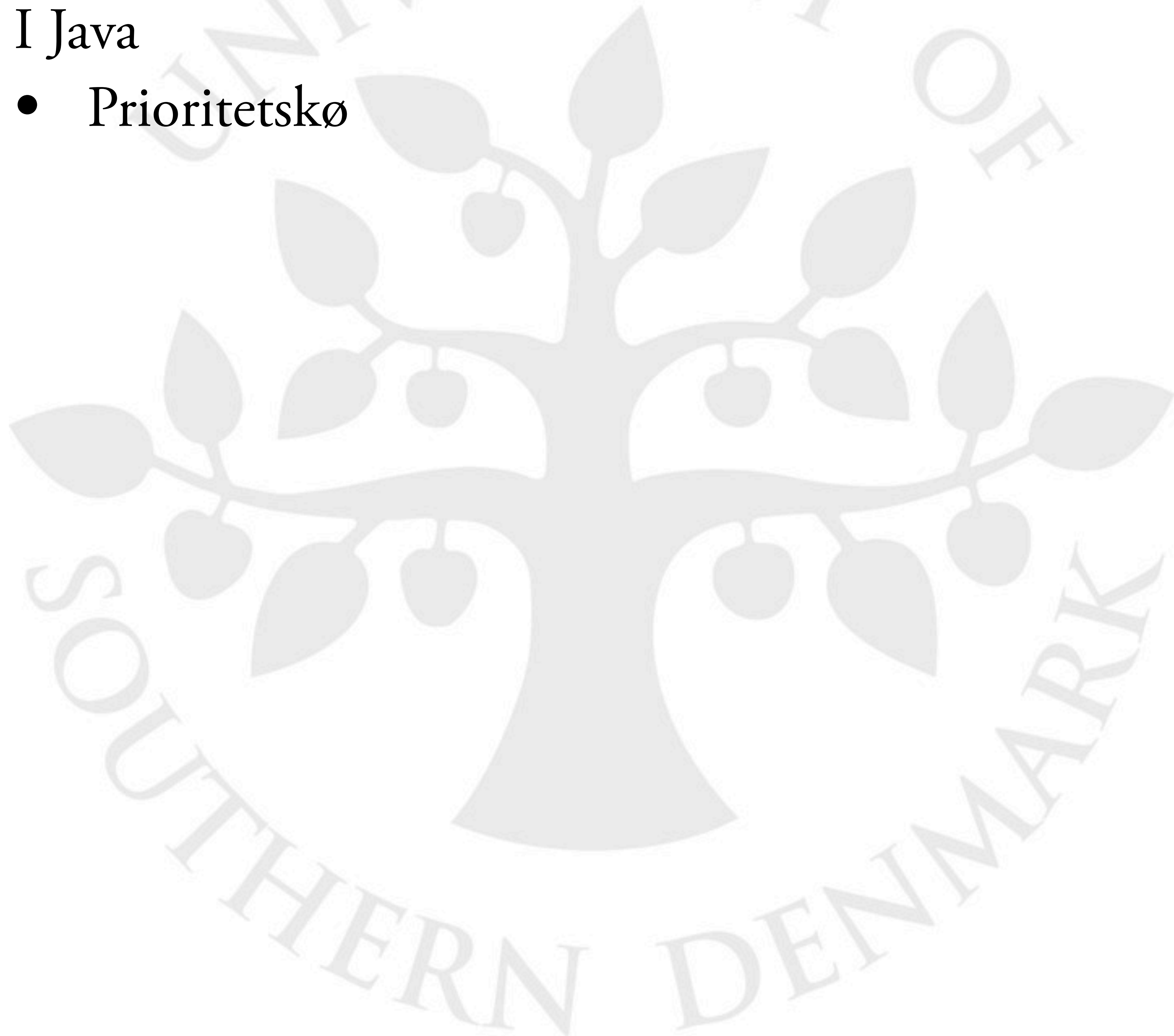
Prioritetskøer



- I Java

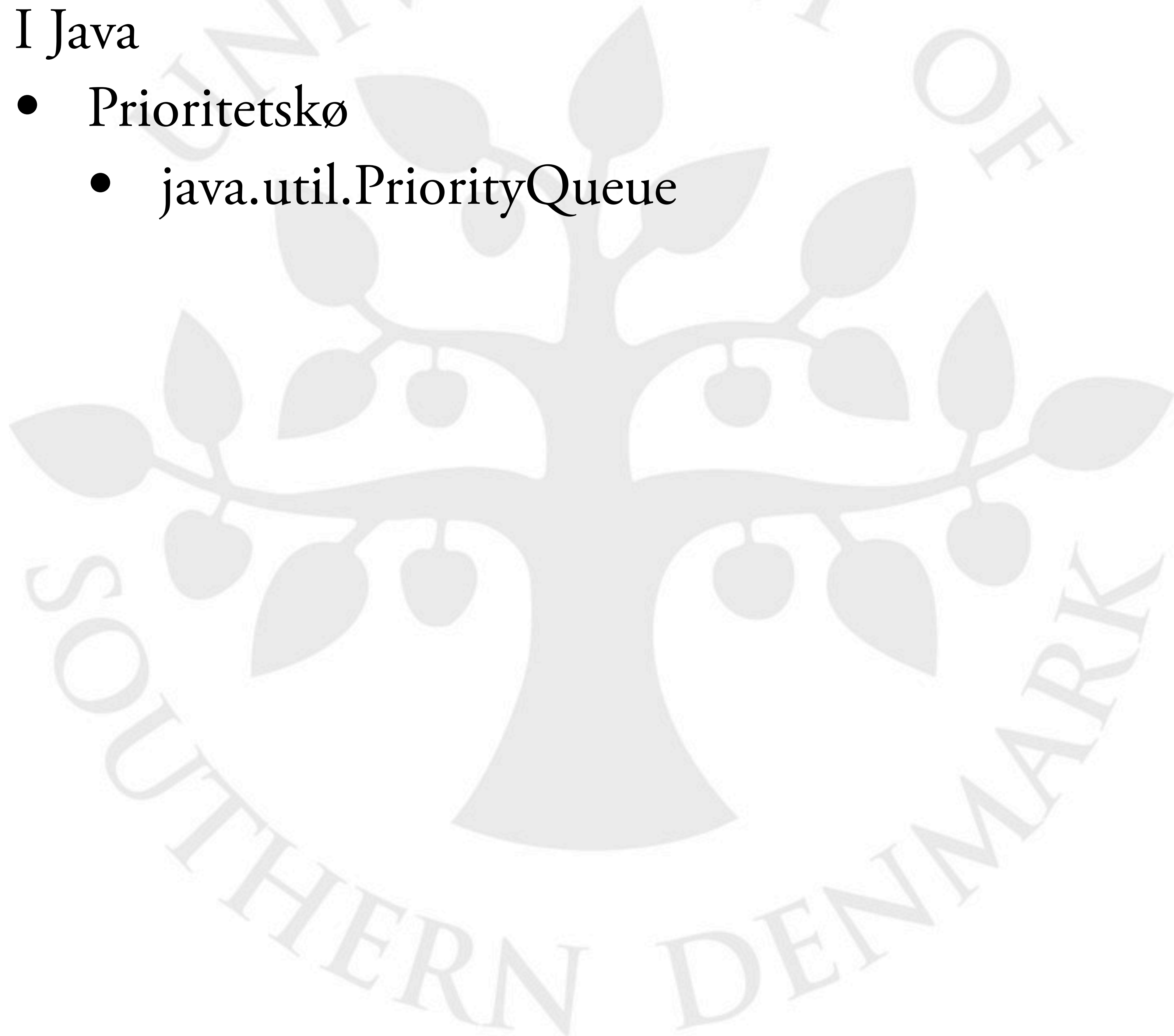
Prioritetskøer

- I Java
  - Prioritetskø



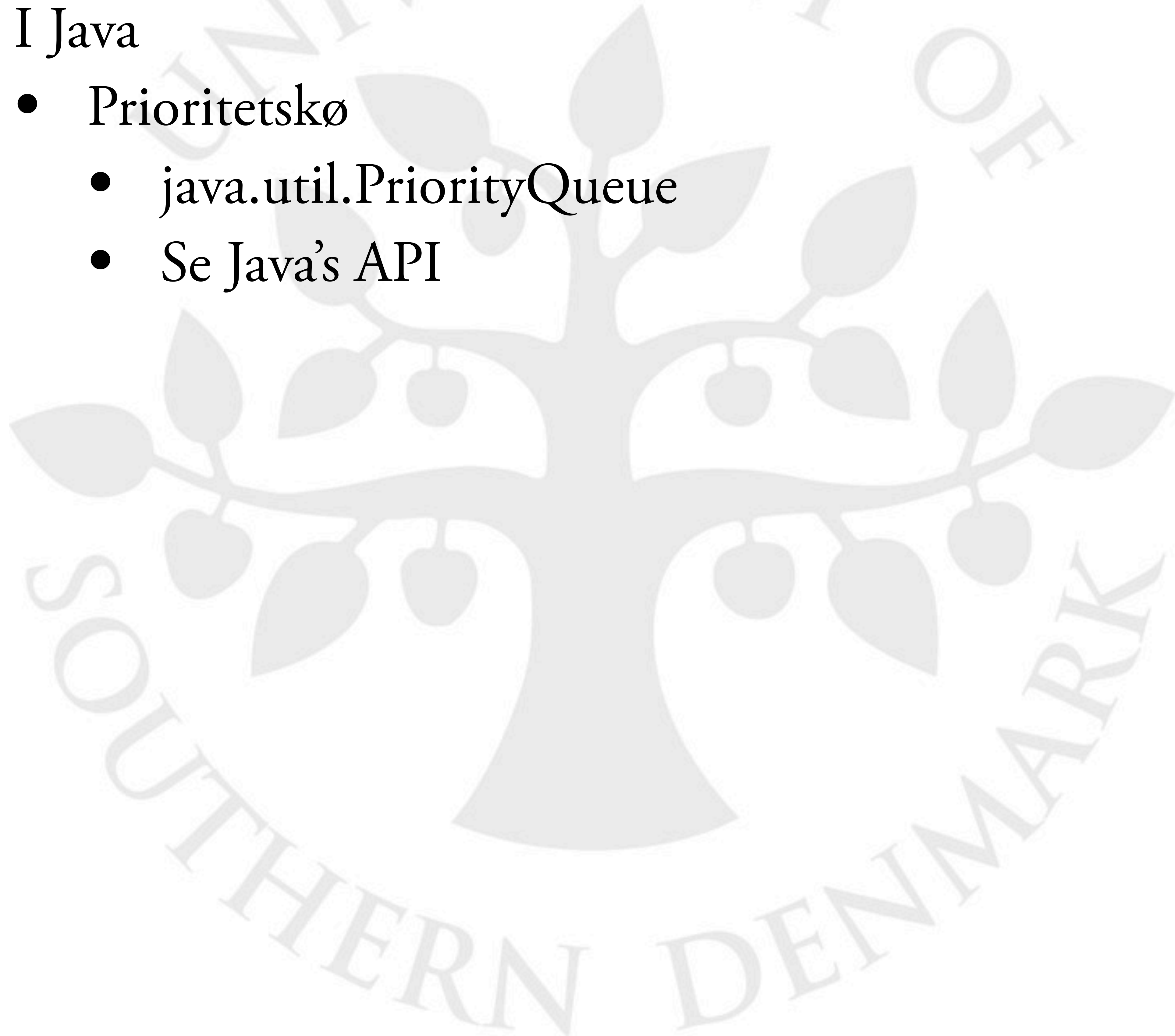
University of Southern Denmark • Syddansk Universitet

Prioritetskøer



- I Java
  - Prioritetskø
    - java.util.PriorityQueue

Prioritetskøer



- I Java
  - Prioritetskø
    - java.util.PriorityQueue
    - Se Java's API

Prioritetskøer

- I Java
  - Prioritetskø
    - java.util.PriorityQueue
    - Se Java's API
  - Elementerne

Prioritetskøer

- I Java
  - Prioritetskø
    - java.util.PriorityQueue
    - Se Java's API
  - Elementerne
    - Objekter der har en total ordning

University of Southern Denmark • Syddansk Universitet

Prioritetskøer

- I Java
  - Prioritetskø
    - java.util.PriorityQueue
    - Se Java's API
  - Elementerne
    - Objekter der har en total ordning
  - Total ordning i Java?!?

University of Southern Denmark • Syddansk Universitet

Prioritetskøer

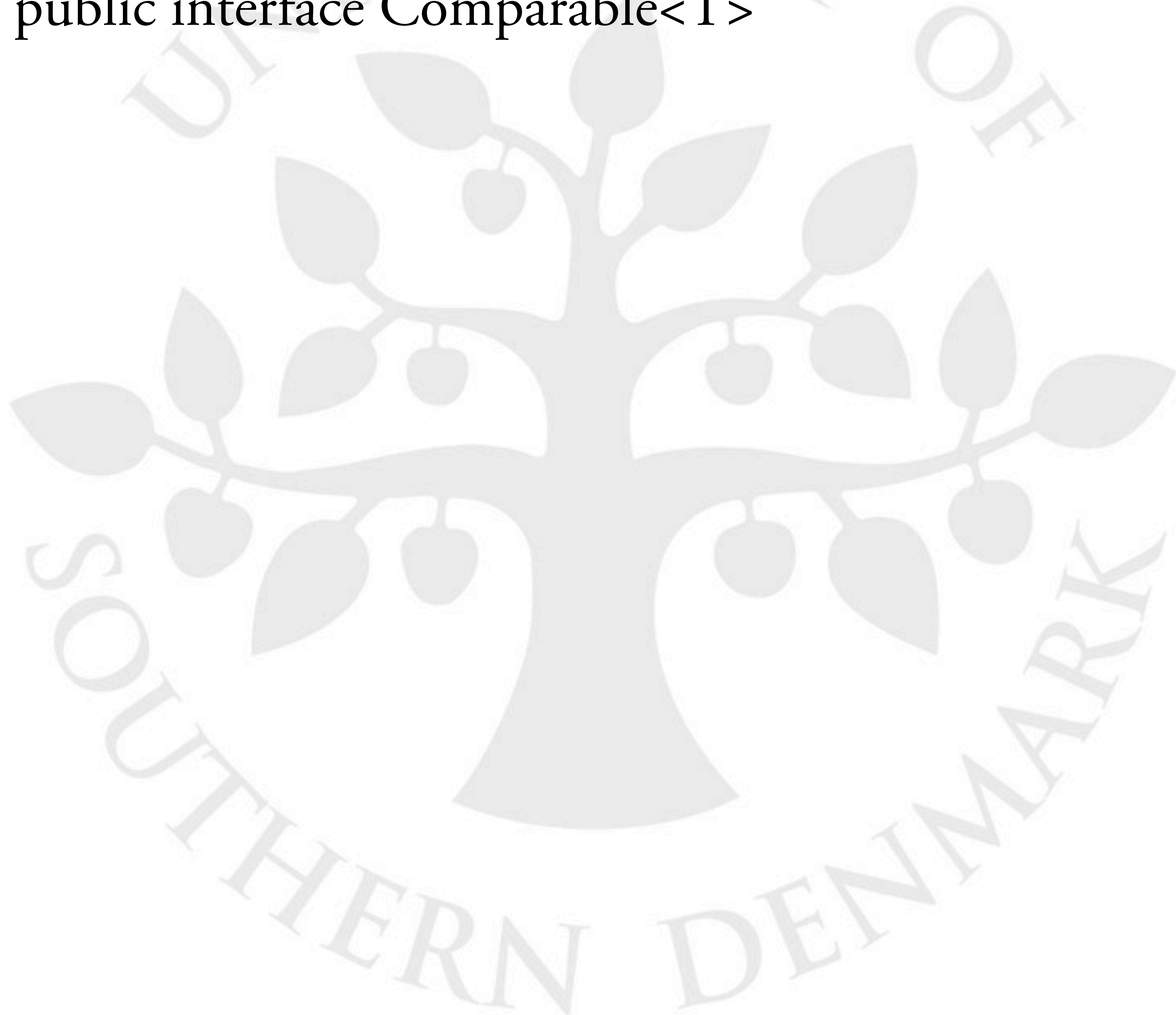
- I Java
  - Prioritetskø
    - java.util.PriorityQueue
    - Se Java's API
  - Elementerne
    - Objekter der har en total ordning
  - Total ordning i Java?!?
    - Interface java.lang.Comparable

Comparable

University of Southern Denmark • Syddansk Universitet

Comparable

- public interface Comparable<T>



Comparable

- public interface Comparable<T>
- Kun én metode

University of Southern Denmark • Syddansk Universitet

Comparable

- public interface Comparable<T>
- Kun én metode
  - int compareTo(T o)

Comparable

- public interface Comparable<T>
- Kun én metode
  - int compareTo(T o)
    - < 0

Comparable

- public interface Comparable<T>
- Kun én metode
    - int compareTo(T o)
        - < 0
            - Det aktuelle objekt er mindre end o

Comparable

- public interface Comparable<T>
- Kun én metode
  - int compareTo(T o)
    - < 0
      - Det aktuelle objekt er mindre end o
    - = 0

University of Southern Denmark • Syddansk Universitet

Comparable

- public interface Comparable<T>
- Kun én metode
  - int compareTo(T o)
    - < 0
      - Det aktuelle objekt er mindre end o
    - = 0
      - Det aktuelle objekt er lig med o

Comparable

- public interface Comparable<T>
- Kun én metode
  - int compareTo(T o)
    - < 0
      - Det aktuelle objekt er mindre end o
    - = 0
      - Det aktuelle objekt er lig med o
    - > 0

Comparable

- public interface Comparable<T>
- Kun én metode
  - int compareTo(T o)
    - < 0
      - Det aktuelle objekt er mindre end o
    - = 0
      - Det aktuelle objekt er lig med o
    - > 0
      - Det aktuelle objekt er større end o

Sortering

Sortering

- Sortering vha. en prioritetskø

Sortering

- Sortering vha. en prioritetskø
  - Lav en ny tom prioritetskø

University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

Sortering

- Sortering vha. en prioritetskø
    - Lav en ny tom prioritetskø
    - Indsæt de tal der ønskes sorteret i køen

Sortering

- Sortering vha. en prioritetskø
  - Lav en ny tom prioritetskø
  - Indsæt de tal der ønskes sorteret i køen
    - Nøglen (prioriteten) er tallet selv

Sortering

- Sortering vha. en prioritetskø
  - Lav en ny tom prioritetskø
  - Indsæt de tal der ønskes sorteret i køen
    - Nøglen (prioriteten) er tallet selv
  - Sålænge der stadig er tal i køen

Sortering

- Sortering vha. en prioritetskø
    - Lav en ny tom prioritetskø
    - Indsæt de tal der ønskes sorteret i køen
        - Nøglen (prioriteten) er tallet selv
    - Sålænge der stadig er tal i køen
        - Udtag det mindste element

Sortering

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme

University of Southern Denmark • Syddansk Universitet

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme
    - Prioritetskøen

University of Southern Denmark • Syddansk Universitet

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme
    - Prioritetskøen
        - En hægtet liste

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste
    - Elementerne ligger i den rækkefølge de blev indsat

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste
    - Elementerne ligger i den rækkefølge de blev indsat
- Indsættelsessortering kan også implementeres som en prioritetskø-sorterings-algoritme

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste
    - Elementerne ligger i den rækkefølge de blev indsat
- Indsættelsessortering kan også implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen

University of Southern Denmark • Syddansk Universitet

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste
    - Elementerne ligger i den rækkefølge de blev indsat

- Indsættelsessortering kan også implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste
    - Elementerne ligger i den rækkefølge de blev indsat

- Indsættelsessortering kan også implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste
    - Elementerne ligger i sorteret rækkefølge

University of Southern Denmark • Syddansk Universitet

Sortering

- Udvælgelsessortering kan implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste
    - Elementerne ligger i den rækkefølge de blev indsat

- Indsættelsessortering kan også implementeres som en prioritetskø-sorterings-algoritme
  - Prioritetskøen
    - En hægtet liste
    - Elementerne ligger i sorteret rækkefølge
      - Indsættes på deres rigtige plads

Hob

Hob

- Smart/hurtig implementering af en prioritetskø

Hob

- Smart/hurtig implementering af en prioritetskø
  - Det der bruges i java.util.PriorityQueue

Hob

- Smart/hurtig implementering af en prioritetskø
  - Det der bruges i java.util.PriorityQueue
  - Bemærk: Heap på engelsk

Hob

- Smart/hurtig implementering af en prioritetskø
    - Det der bruges i java.util.PriorityQueue
    - Bemærk: Heap på engelsk
- Målet

Hob

- Smart/hurtig implementering af en prioritetskø
  - Det der bruges i java.util.PriorityQueue
  - Bemærk: Heap på engelsk
- Målet
  - At lave en prioritetskø så vi kan sortere hurtigere end

Hob

- Smart/hurtig implementering af en prioritetskø
  - Det der bruges i java.util.PriorityQueue
  - Bemærk: Heap på engelsk
- Målet
  - At lave en prioritetskø så vi kan sortere hurtigere end $\frac{1}{2}n^2 - \frac{1}{2}n$

Hob

- Smart/hurtig implementering af en prioritetskø
  - Det der bruges i java.util.PriorityQueue
  - Bemærk: Heap på engelsk
- Målet
  - At lave en prioritetskø så vi kan sortere hurtigere end $\frac{1}{2}n^2 - \frac{1}{2}n$
- Íde

Hob

- Smart/hurtig implementering af en prioritetskø
  - Det der bruges i java.util.PriorityQueue
  - Bemærk: Heap på engelsk
- Målet
  - At lave en prioritetskø så vi kan sortere hurtigere end $\frac{1}{2}n^2 - \frac{1}{2}n$
- Íde
  - Brug et binært træ

University of Southern Denmark • Syddansk Universitet

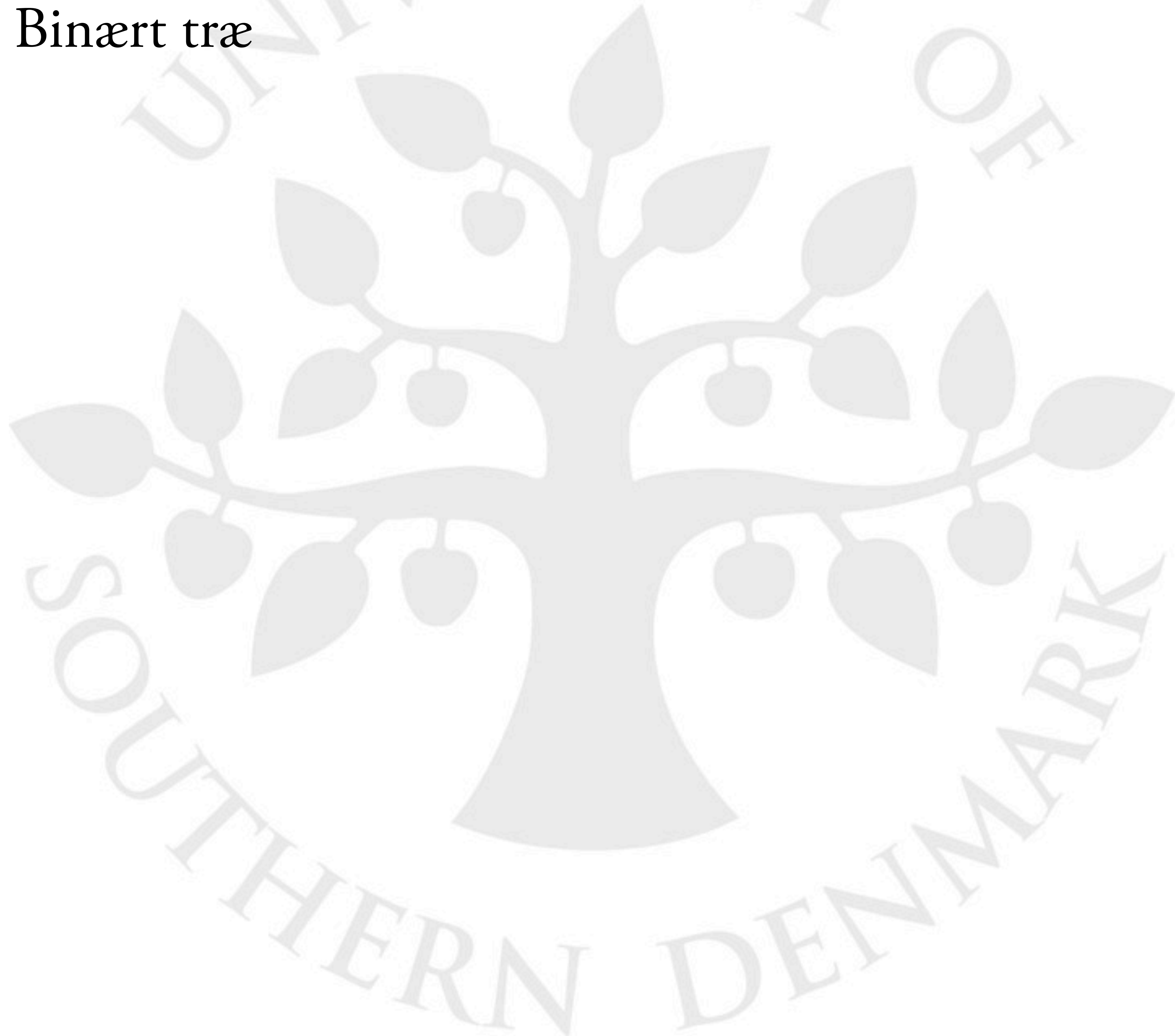
Hob

- Smart/hurtig implementering af en prioritetskø
  - Det der bruges i java.util.PriorityQueue
  - Bemærk: Heap på engelsk
- Målet
  - At lave en prioritetskø så vi kan sortere hurtigere end $\frac{1}{2}n^2 - \frac{1}{2}n$
- Íde
  - Brug et binært træ
    - IKKE et søgetræ

Hob

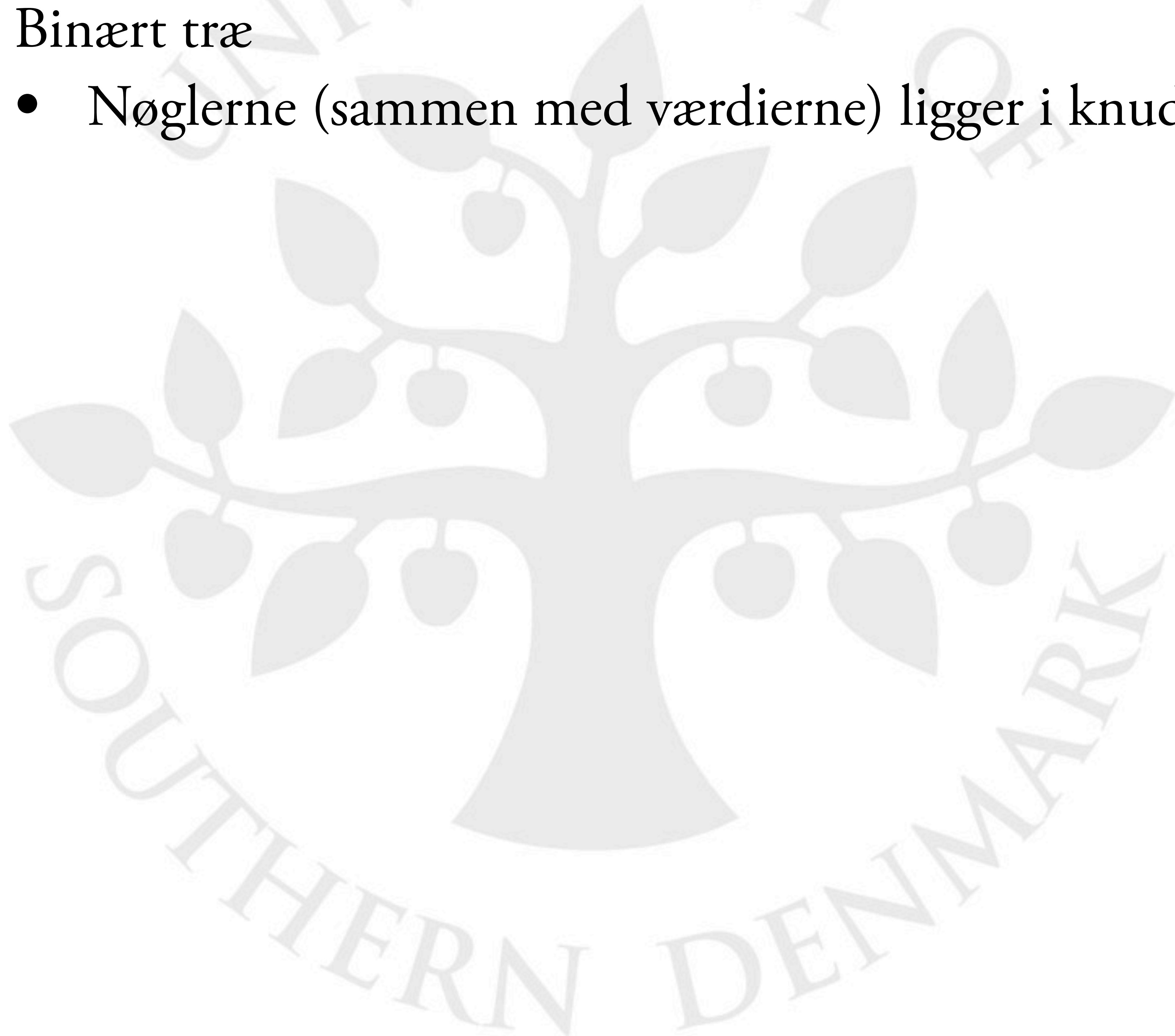
University of Southern Denmark • Syddansk Universitet

Hob



- Binært træ

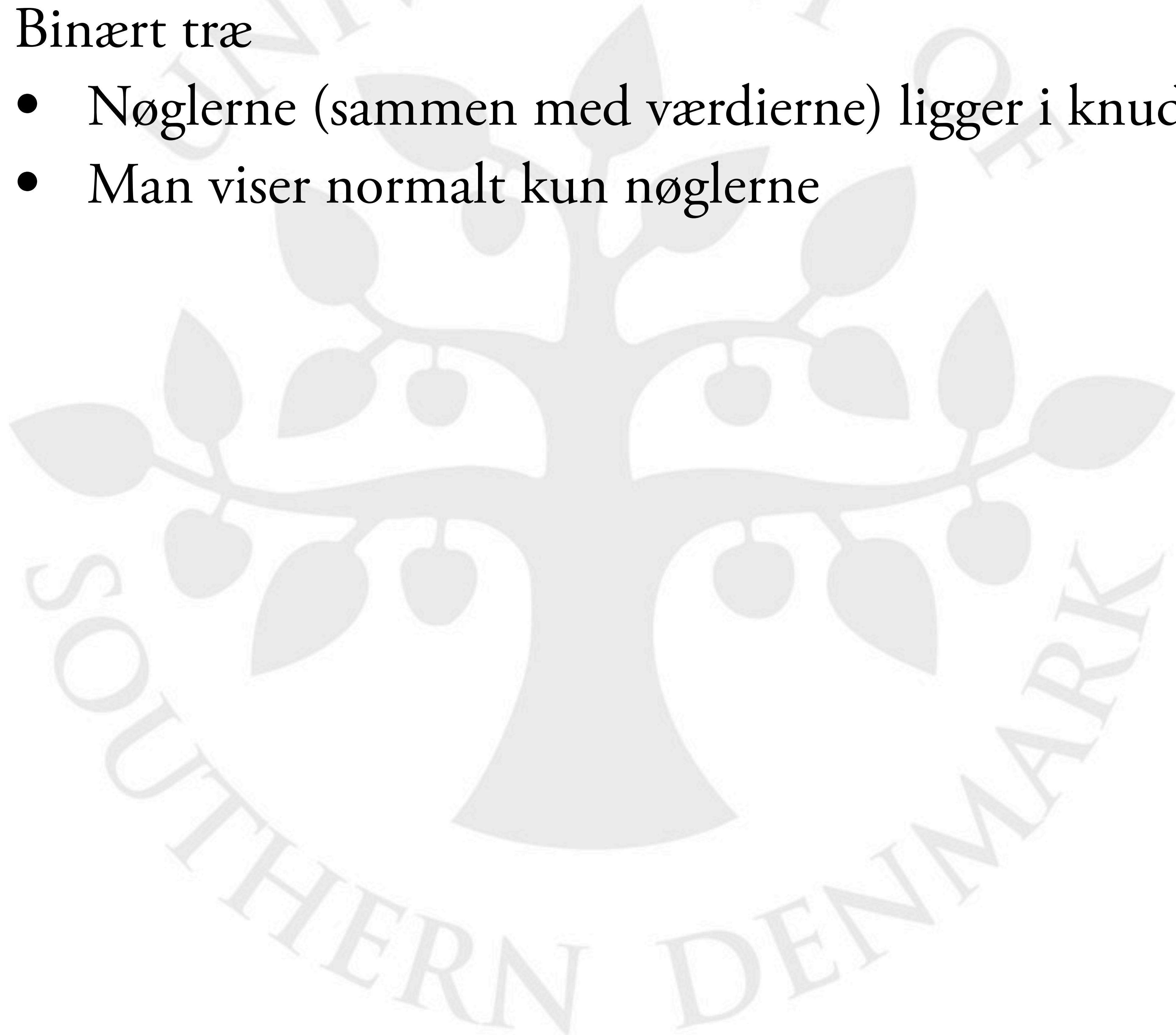
Hob



- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne

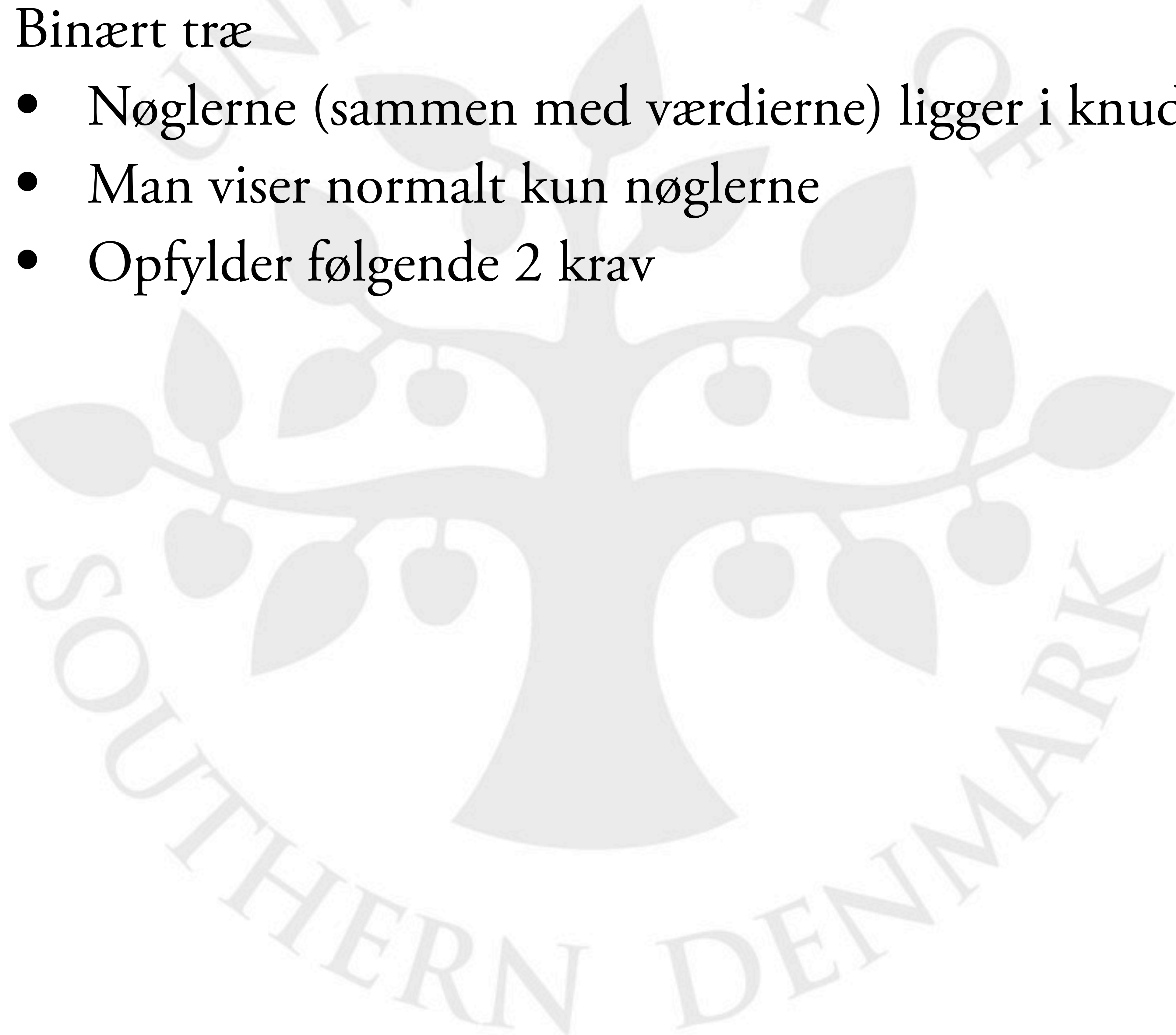
Hob

- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne



University of Southern Denmark • Syddansk Universitet

Hob



- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav

Hob

- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav
    - Hob-invariant

University of Southern Denmark • Syddansk Universitet

Hob

- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav
    - Hob-invariant
      - For alle knuder v pånær roden gælder: $key(v) \geq key(parent(v))$

Hob

- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav
    - Hob-invariant
      - For alle knuder v pånær roden gælder: $key(v) \geq key(parent(v))$
    - Fuldstændigt binært træ

Hob

- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav
    - Hob-invariant
      - For alle knuder v pånær roden gælder: $key(v) \geq key(parent(v))$
    - Fuldstændigt binært træ
      - “Halvtags-formet”

University of Southern Denmark • Syddansk Universitet

Hob

- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav
    - Hob-invariant
      - For alle knuder v pånær roden gælder: $key(v) \geq key(parent(v))$
    - Fuldstændigt binært træ
      - "Halvtags-formet"
      - Alle pånær det nederste lag er fyldt

Hob

- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav
    - Hob-invariant
      - For alle knuder v pånær roden gælder: $key(v) \geq key(parent(v))$
    - Fuldstændigt binært træ
      - "Halvtags-formet"
      - Alle pånær det nederste lag er fyldt
      - Knuderne i det nederste lag ligger længst til venstre

University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

Hob

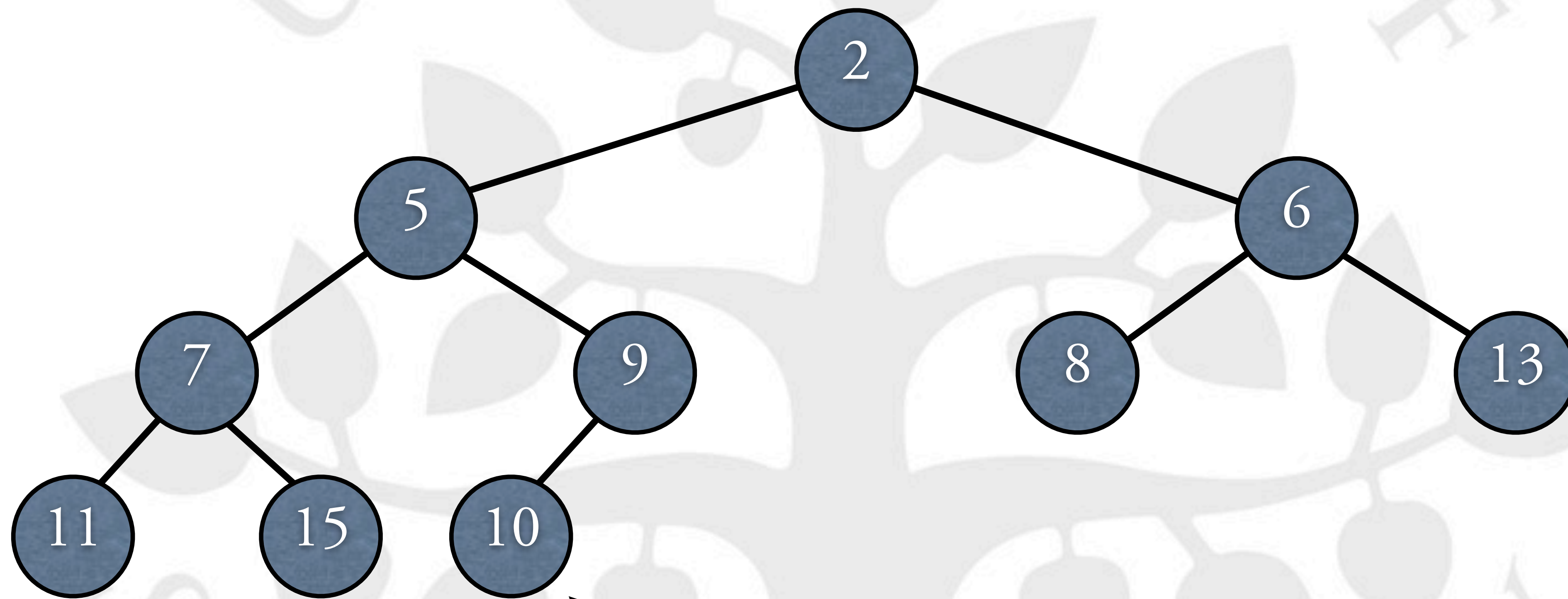
- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav
    - Hob-invariant
      - For alle knuder v pånær roden gælder: $key(v) \geq key(parent(v))$
    - Fuldstændigt binært træ
      - "Halvtags-formet"
      - Alle pånær det nederste lag er fyldt
      - Knuderne i det nederste lag ligger længst til venstre
  - Husker "den sidste knude"

Hob

- Binært træ
  - Nøglerne (sammen med værdierne) ligger i knuderne
  - Man viser normalt kun nøglerne
  - Opfylder følgende 2 krav
    - Hob-invariant
      - For alle knuder v pånær roden gælder: $key(v) \geq key(parent(v))$
    - Fuldstændigt binært træ
      - "Halvtags-formet"
      - Alle pånær det nederste lag er fyldt
      - Knuderne i det nederste lag ligger længst til venstre
  - Husker "den sidste knude"
    - Knuden længst til højre i det nederste lag

University of Southern Denmark • Syddansk Universitet

Hob

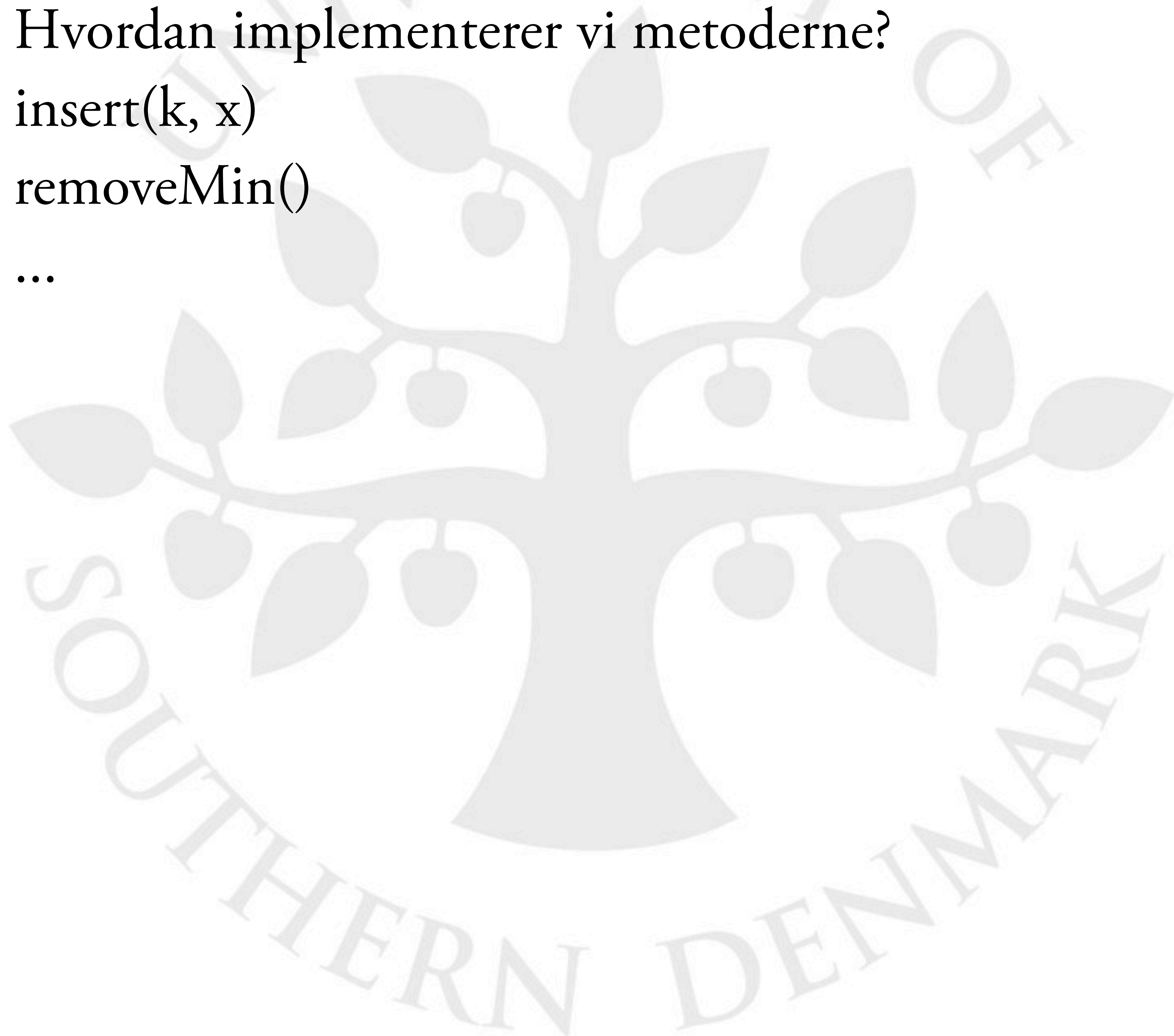


den sidste knude

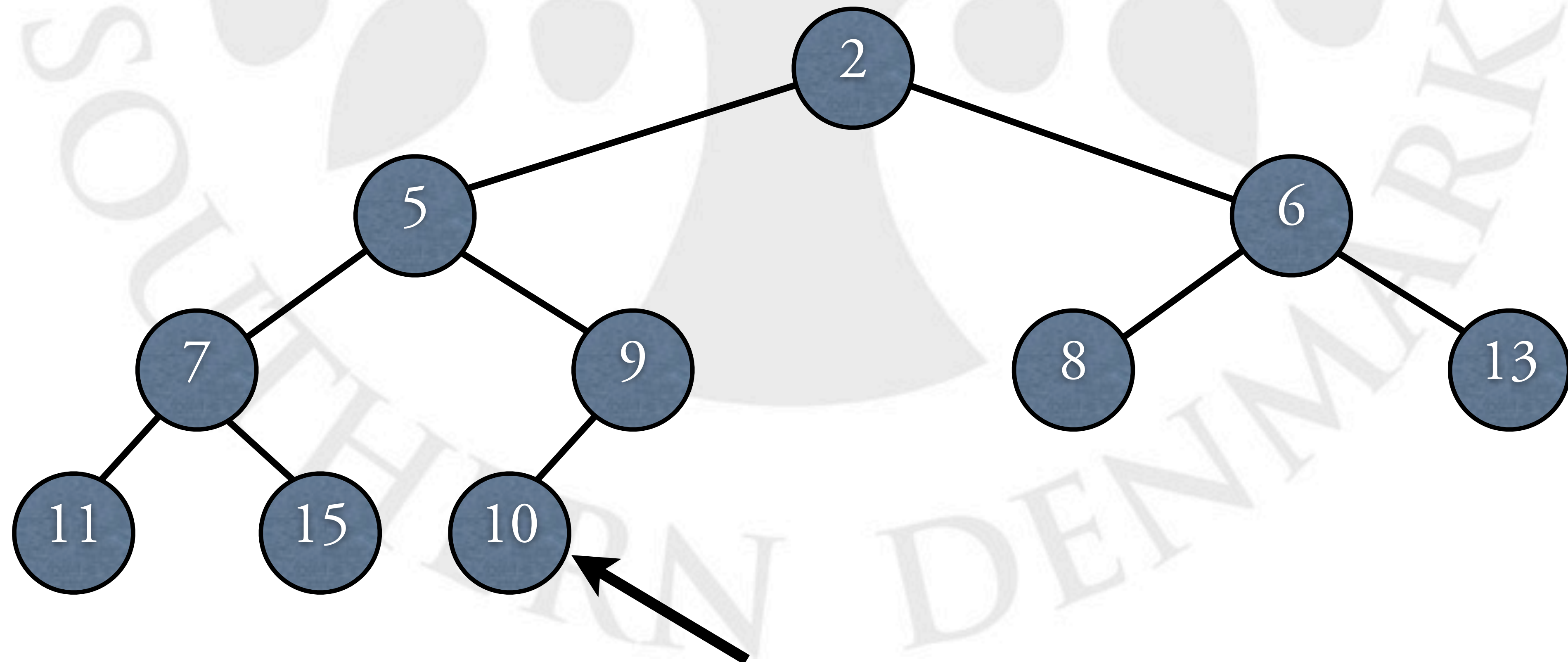
University of Southern Denmark • Syddansk Universitet

Hob

- Hvordan implementerer vi metoderne?
- insert(k, x)
- removeMin()
- ...

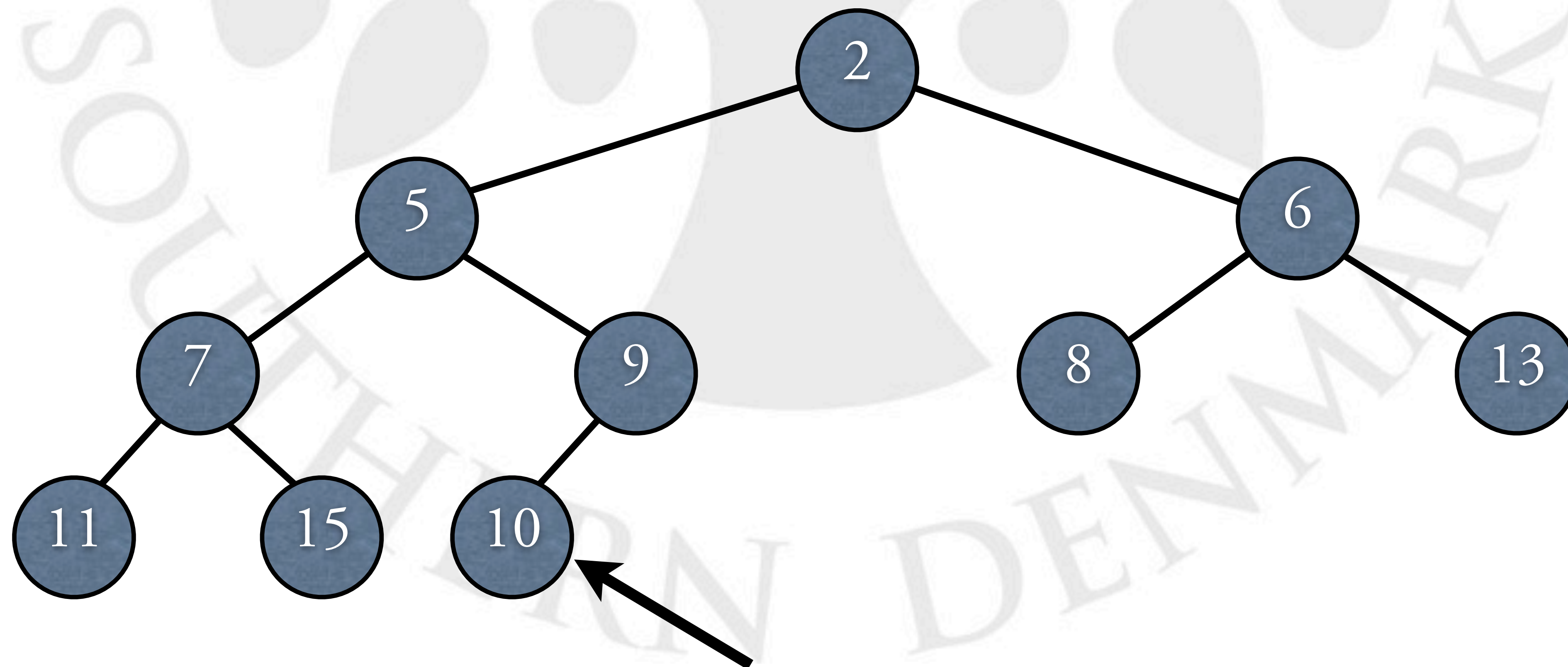


Hob



Hob

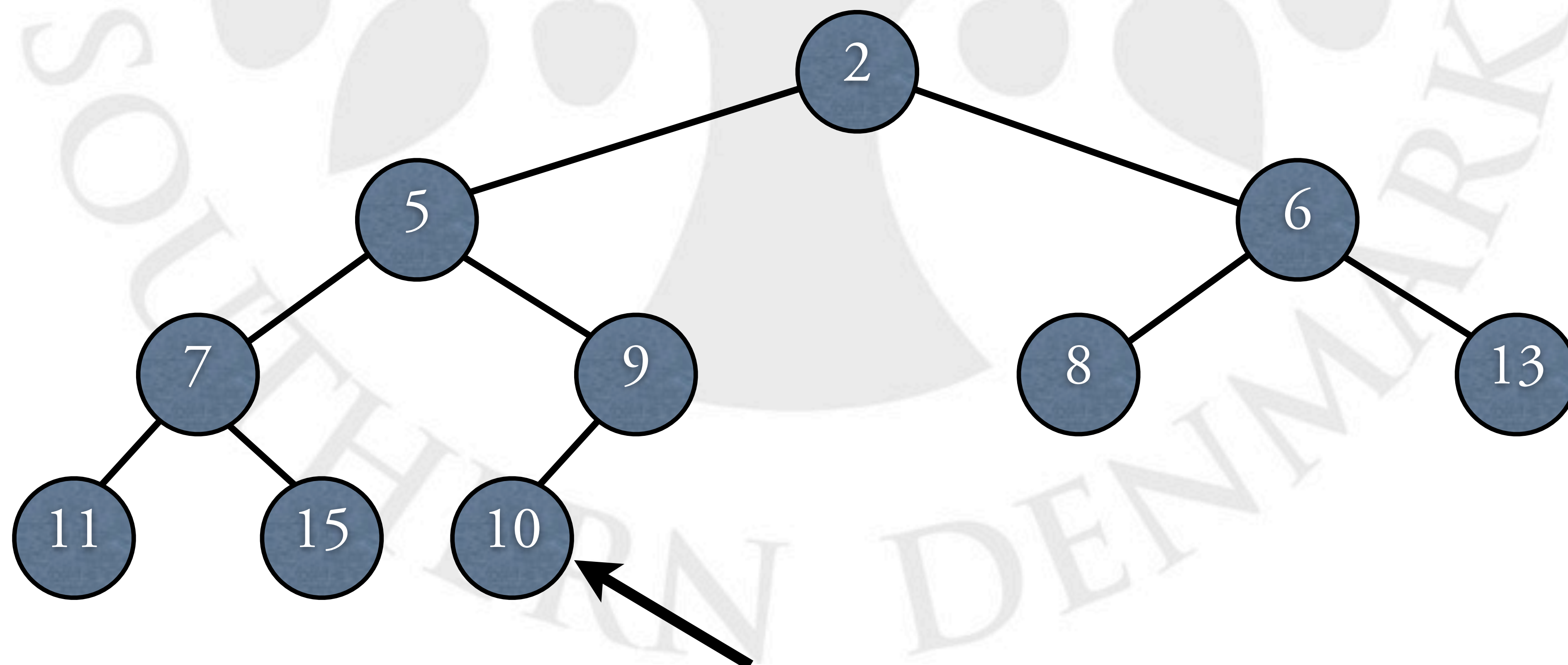
- insert(k, x)



University of Southern Denmark • Syddansk Universitet

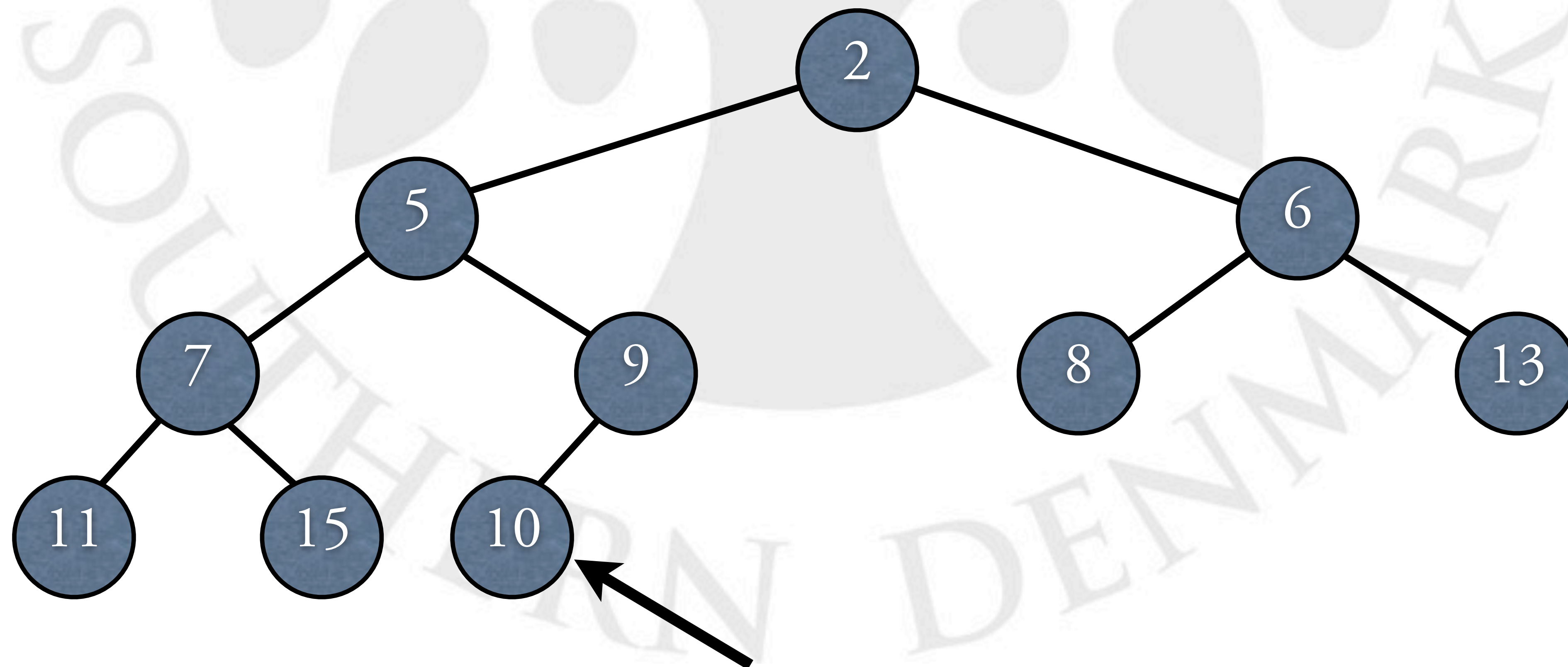
Hob

- insert(k, x)
- Tre skridt:



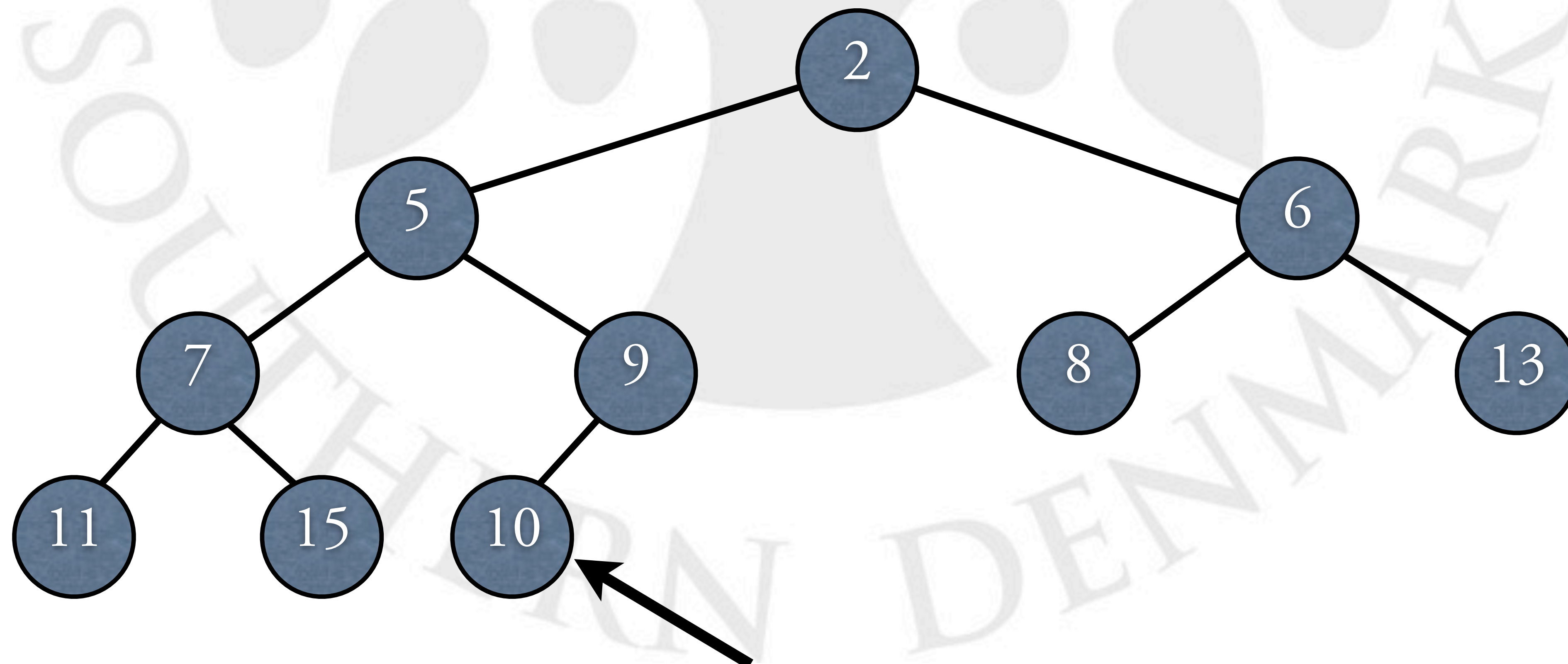
Hob

- insert(k, x)
- Tre skridt:
  - 1. Find den rette plads i træet, kaldet z (så træet stadig er halvtags-formet)



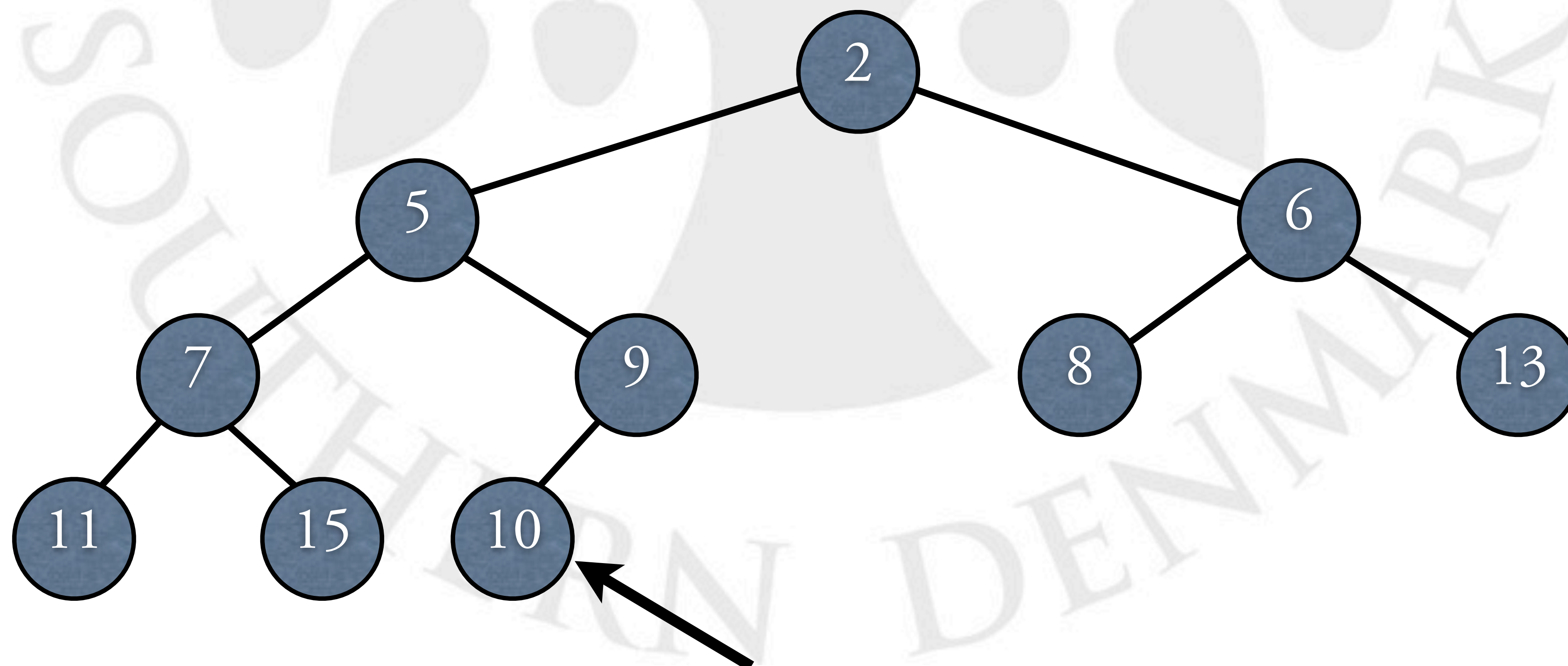
Hob

- insert(k, x)
- Tre skridt:
  - 1. Find den rette plads i træet, kaldet z (så træet stadig er halvtags-formet)
  - 2. Gem x i z



Hob

- insert(k, x)
- Tre skridt:
  - 1. Find den rette plads i træet, kaldet z (så træet stadig er halvtags-formet)
  - 2. Gem x i z
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten

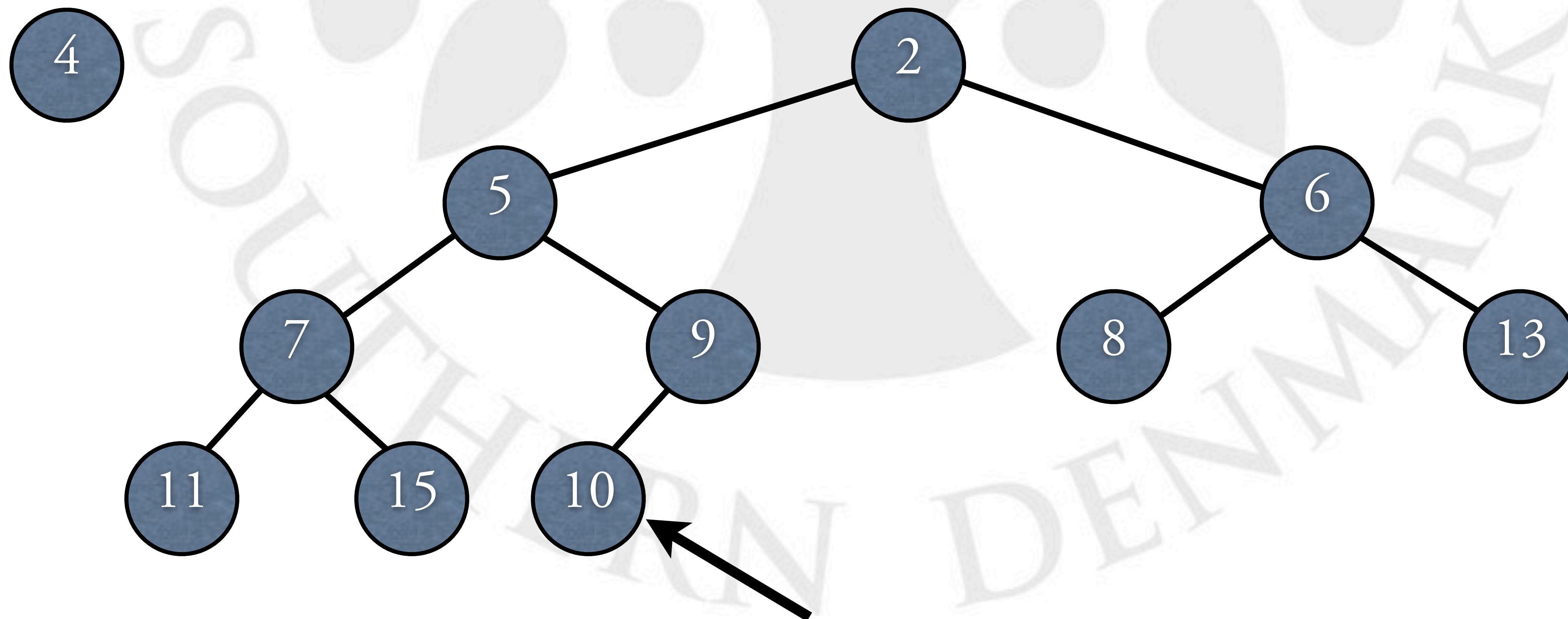


University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

Hob

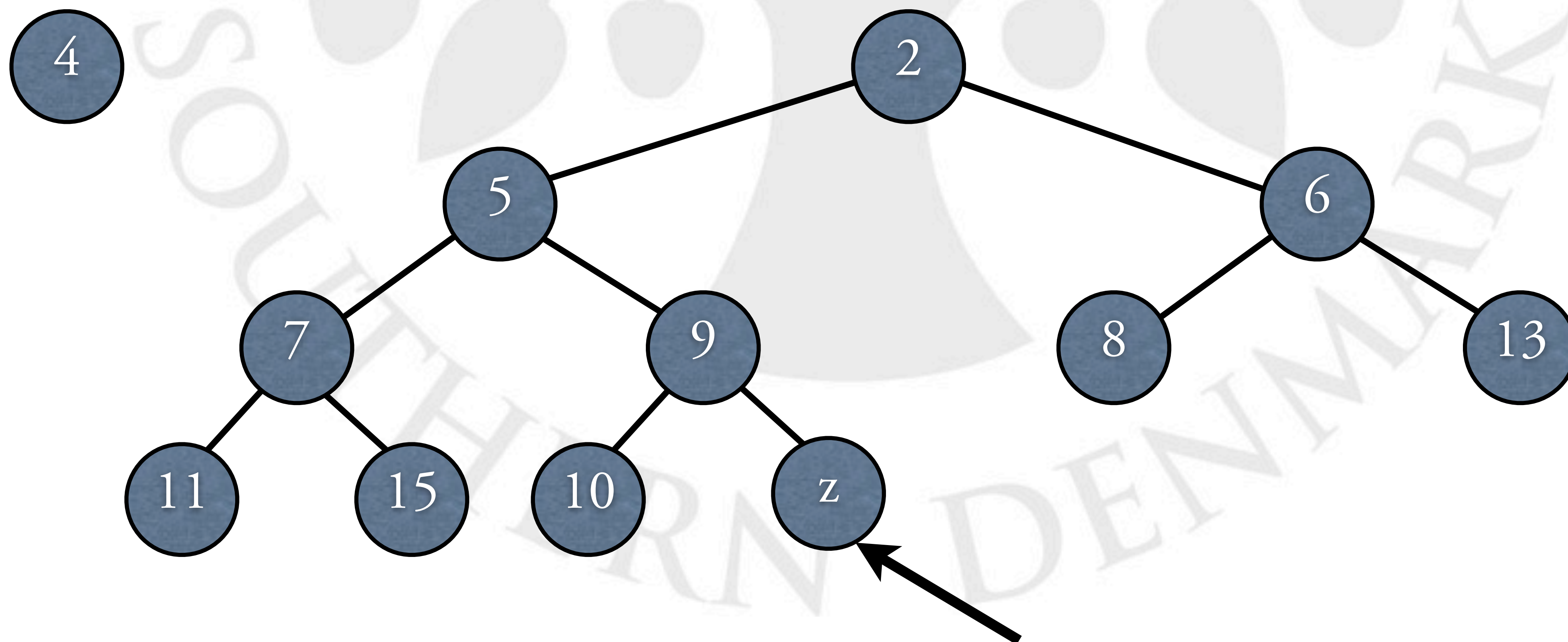
- insert(k, x)
- Tre skridt:
  - 1. Find den rette plads i træet, kaldet z
    (så træet stadig er halvtags-formet)
  - 2. Gem x i z
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten



University of Southern Denmark • Syddansk Universitet

Hob

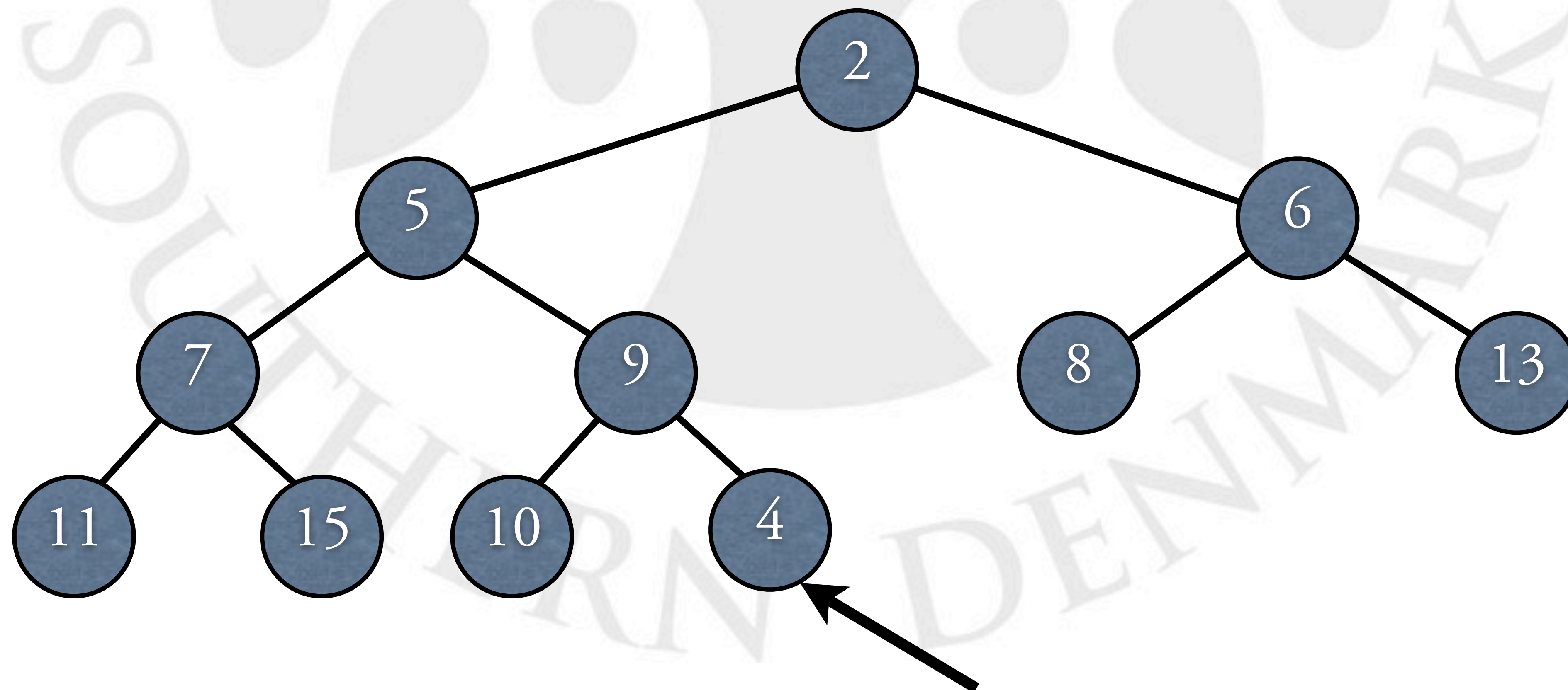
- insert(k, x)
- Tre skridt:
  - 1. Find den rette plads i træet, kaldet z (så træet stadig er halvtags-formet)
  - 2. Gem x i z
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten



University of Southern Denmark • Syddansk Universitet

Hob

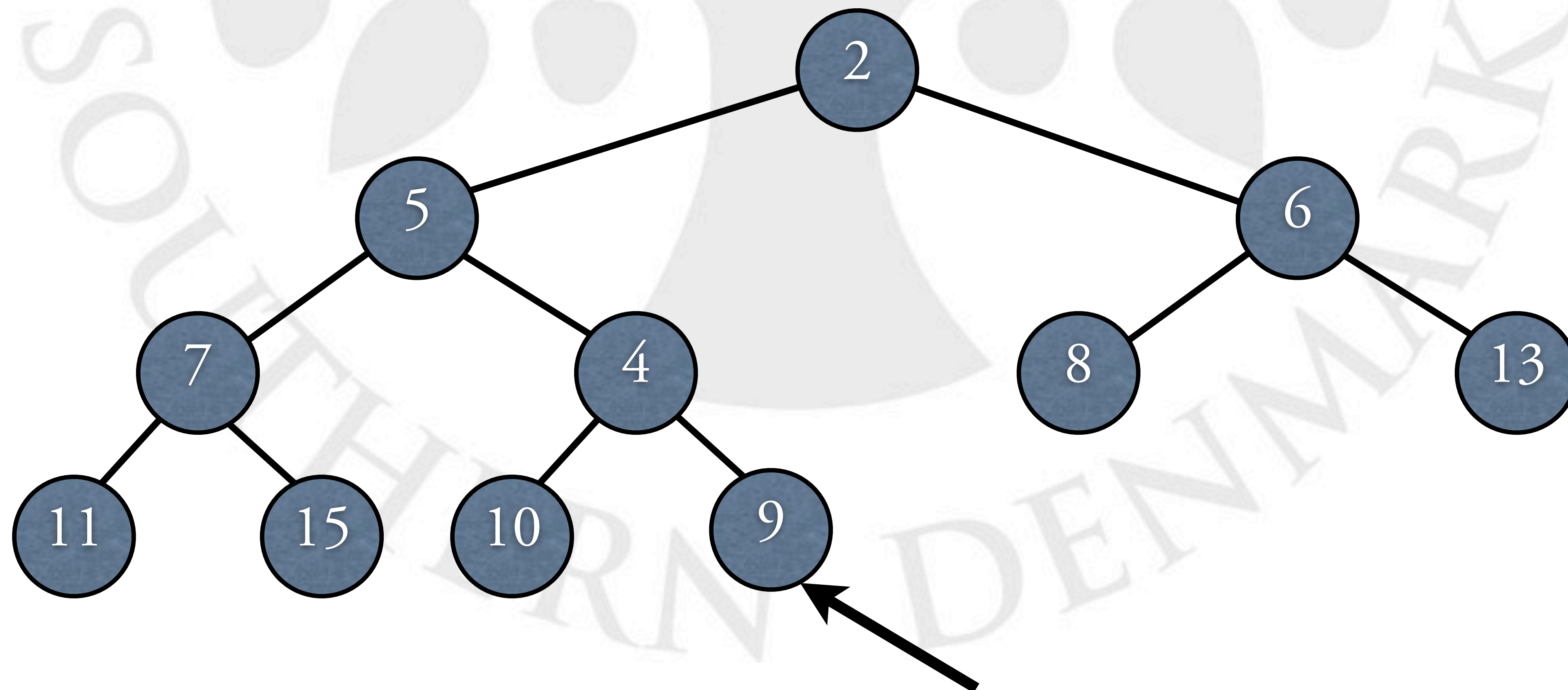
- insert(k, x)
- Tre skridt:
  - 1. Find den rette plads i træet, kaldet z (så træet stadig er halvtags-formet)
  - 2. Gem x i z
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten



University of Southern Denmark • Syddansk Universitet

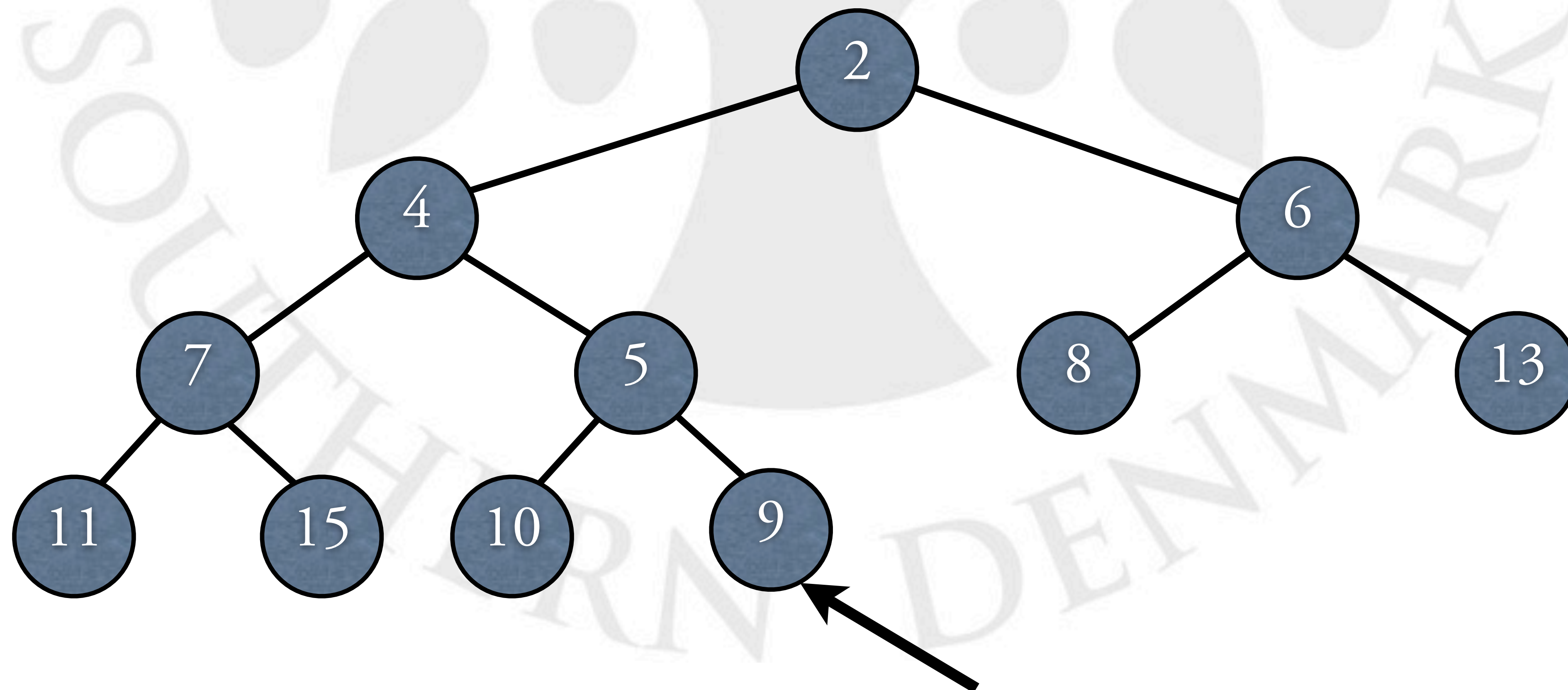
Hob

- insert(k, x)
- Tre skridt:
  - 1. Find den rette plads i træet, kaldet z (så træet stadig er halvtags-formet)
  - 2. Gem x i z
  - 3. “Bobbel” x opad indtil træet igen opfylder hob-invarianten



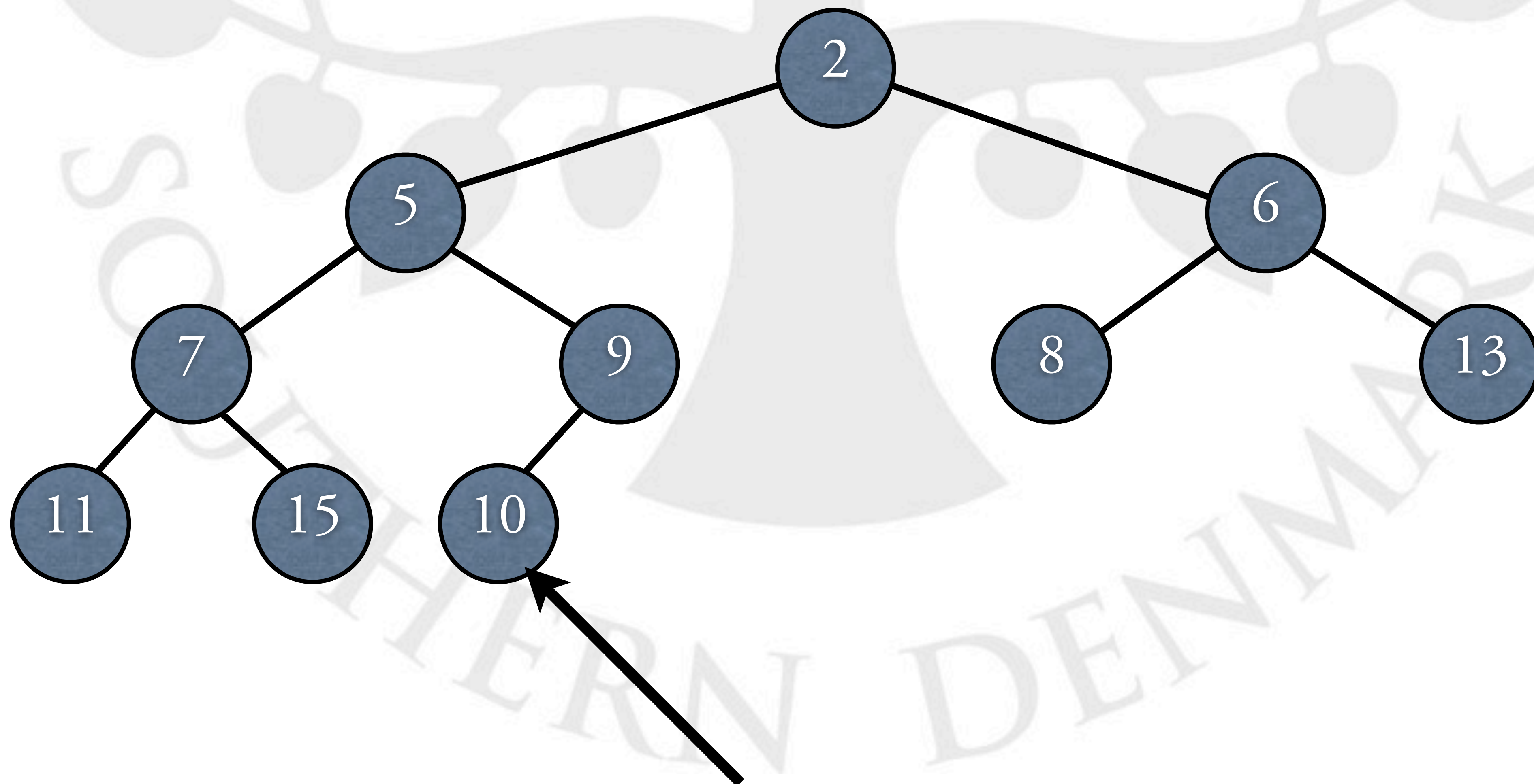
Hob

- insert(k, x)
- Tre skridt:
  - 1. Find den rette plads i træet, kaldet z (så træet stadig er halvtags-formet)
  - 2. Gem x i z
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten



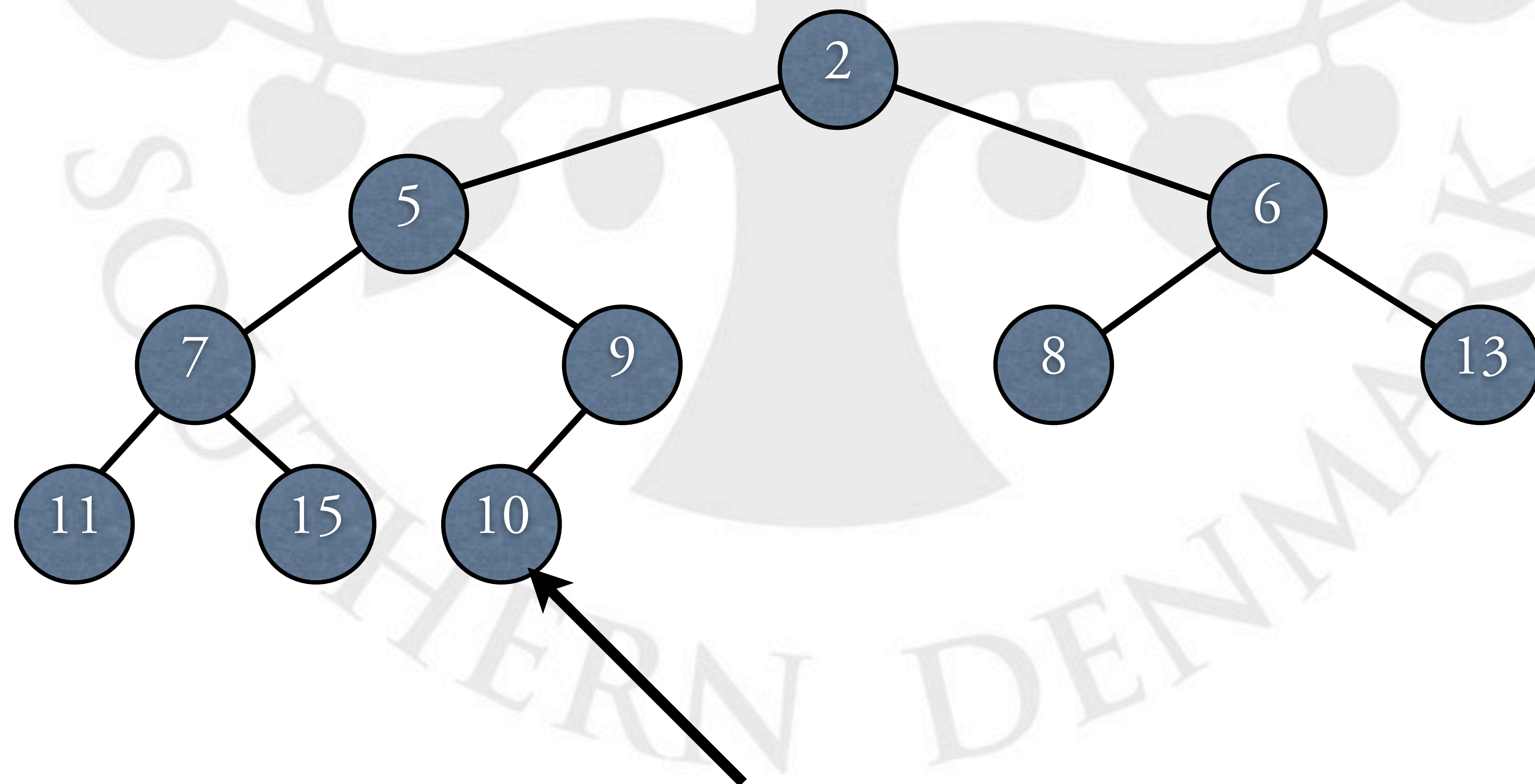
University of Southern Denmark • Syddansk Universitet

Hob



Hob

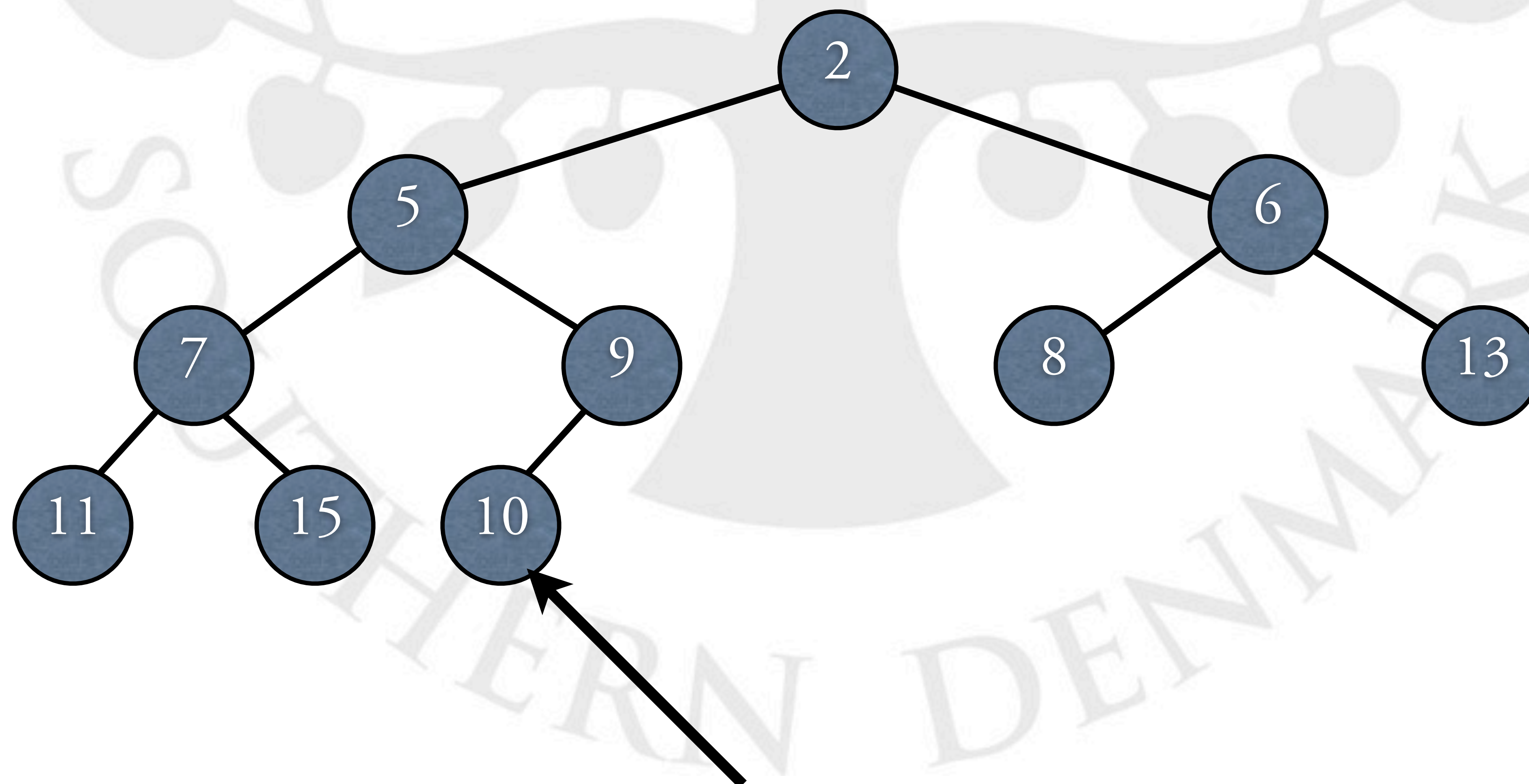
- deleteMin()



University of Southern Denmark • Syddansk Universitet

Hob

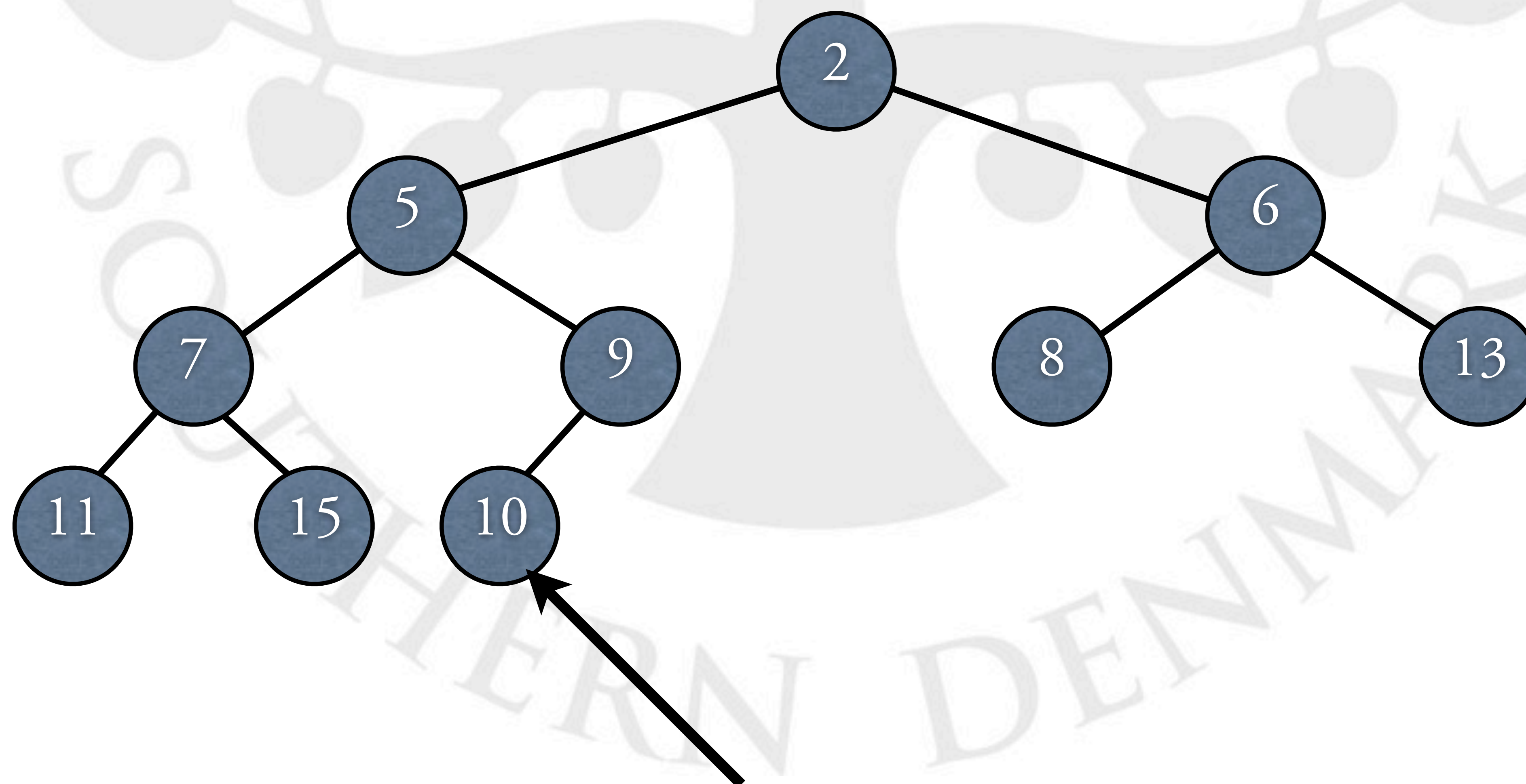
- deleteMin()
- 1. Udtag roden af træet (til senere returnering)



University of Southern Denmark • Syddansk Universitet

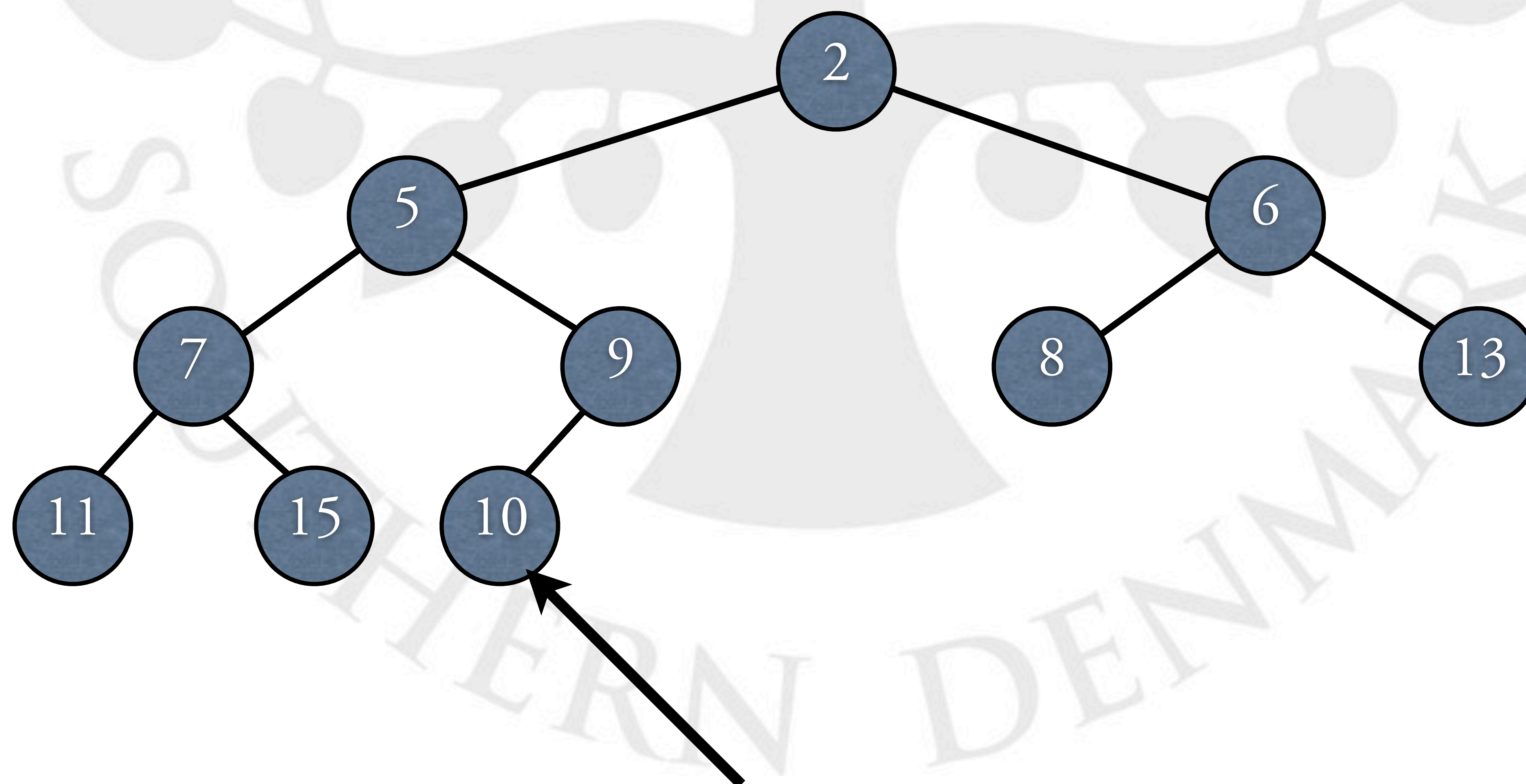
Hob

- deleteMin()
- 1. Udtag roden af træet (til senere returnering)
- 2. Flyt den sidste knude i træet op i roden
  (så træet igen hænger sammen og er halvtags-formet)



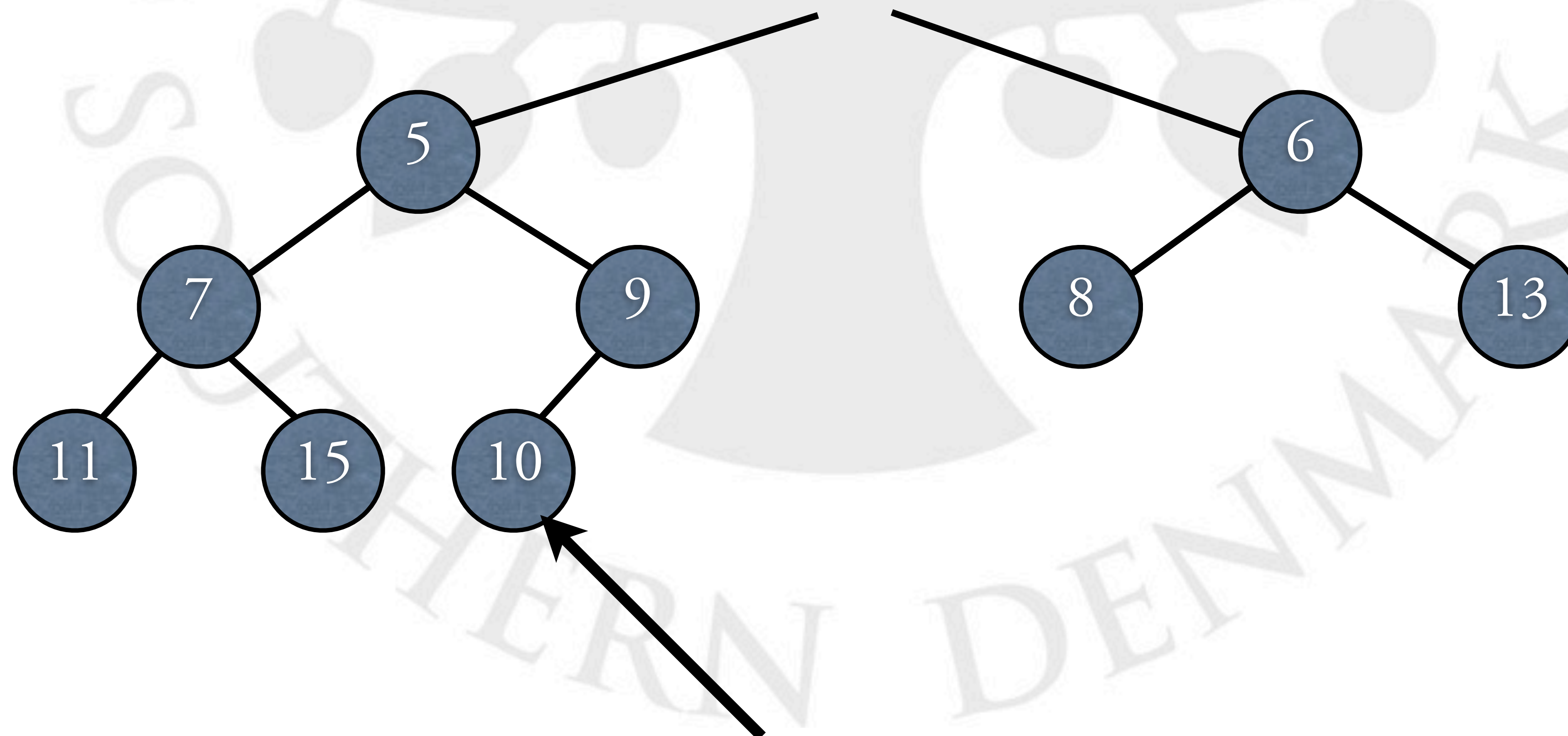
Hob

- deleteMin()
- 1. Udtag roden af træet (til senere returnering)
- 2. Flyt den sidste knude i træet op i roden
  (så træet igen hænger sammen og er halvtags-formet)
- 3. Bobbel roden ned indtil træet igen opfyder
  hob-invarianten



Hob

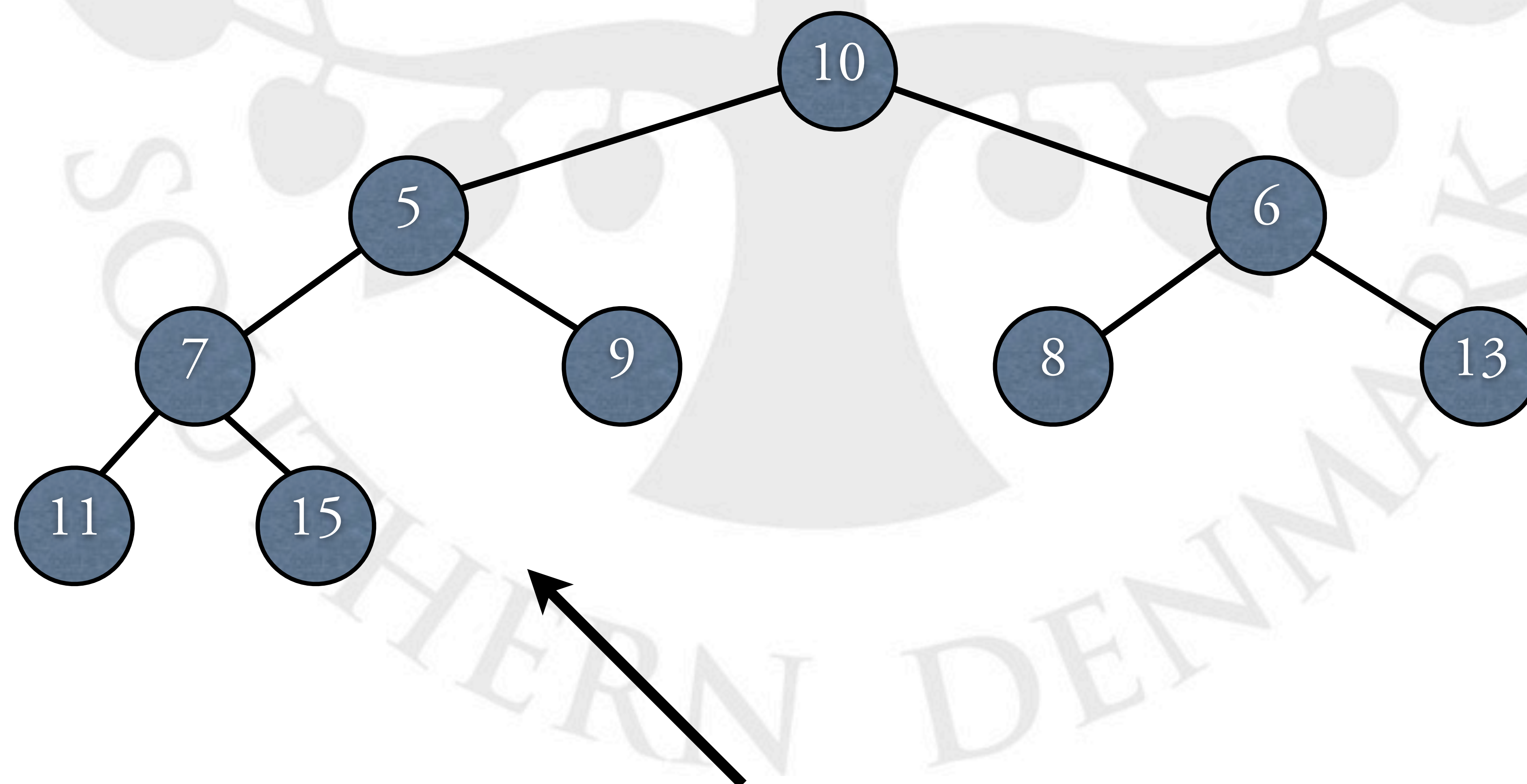
- deleteMin()
- 1. Udtag roden af træet (til senere returnering)
- 2. Flyt den sidste knude i træet op i roden
(så træet igen hænger sammen og er halvtags-formet)
- 3. Bobbel roden ned indtil træet igen opfyder
hob-invarianten



University of Southern Denmark • Syddansk Universitet

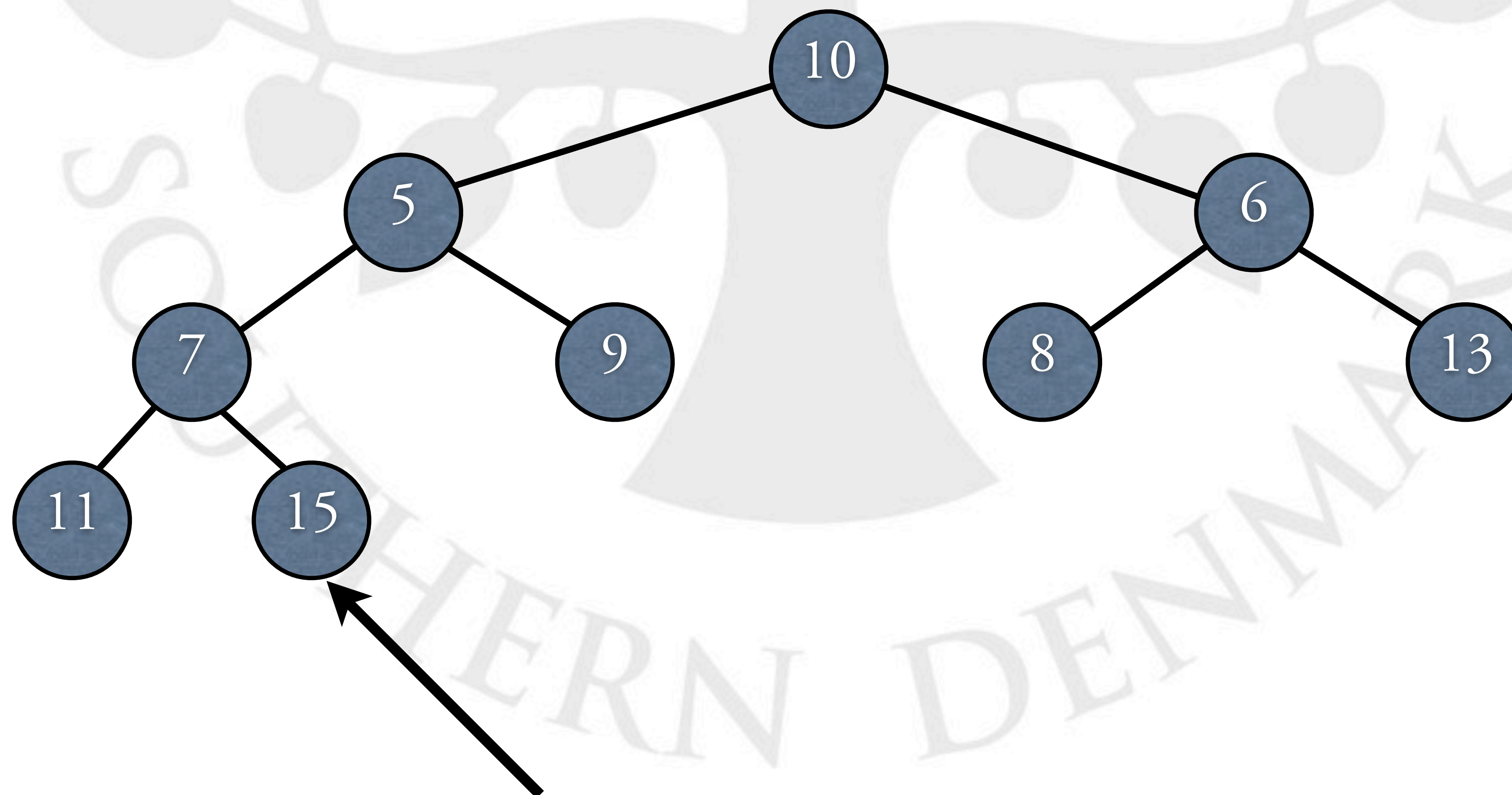
Hob

- deleteMin()
- 1. Udtag roden af træet (til senere returnering)
- 2. Flyt den sidste knude i træet op i roden
  (så træet igen hænger sammen og er halvtags-formet)
- 3. Bobbel roden ned indtil træet igen opfyder
  hob-invarianten



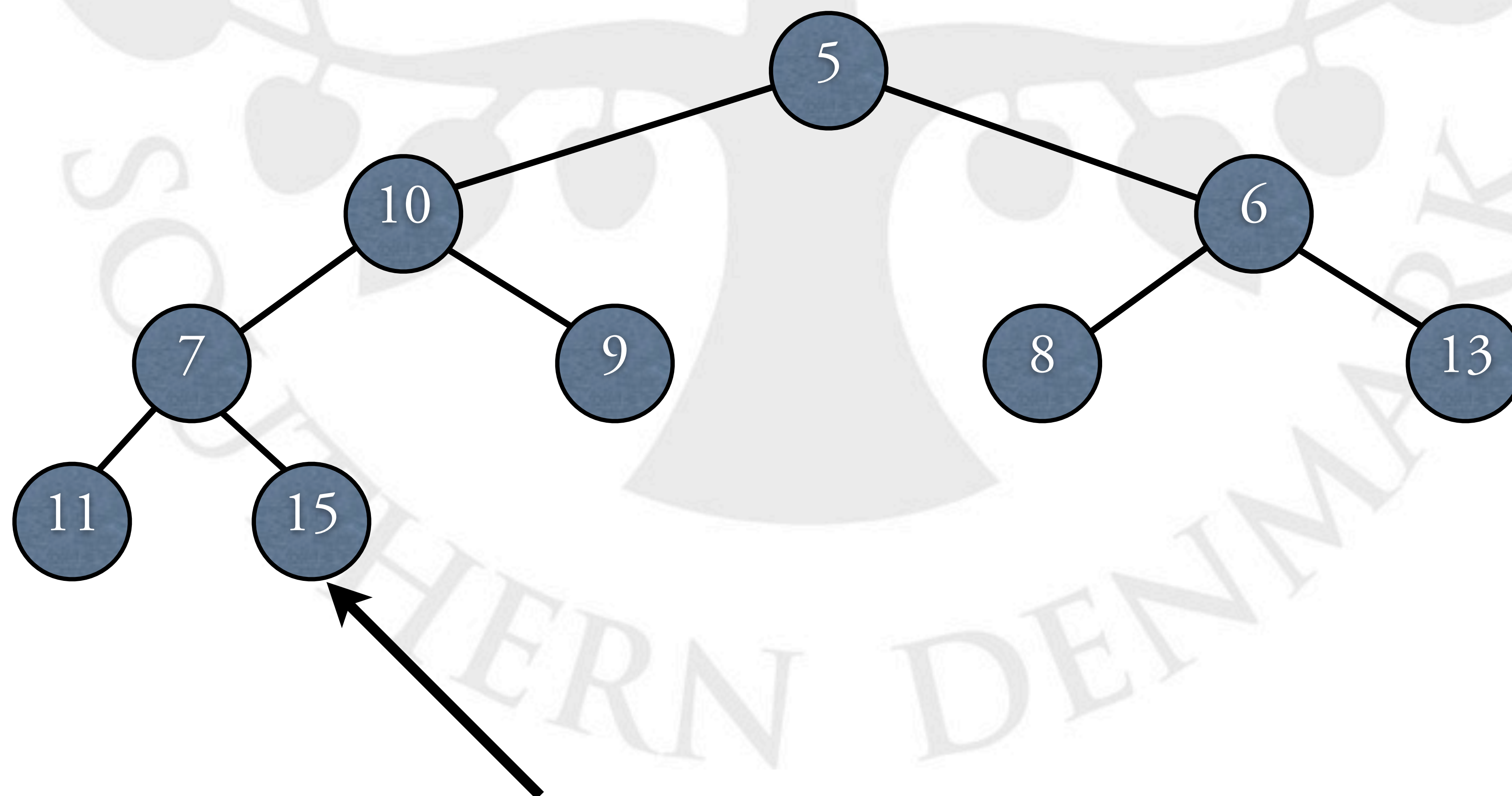
Hob

- deleteMin()
- 1. Udtag roden af træet (til senere returnering)
- 2. Flyt den sidste knude i træet op i roden
  (så træet igen hænger sammen og er halvtags-formet)
- 3. Bobbel roden ned indtil træet igen opfyder
  hob-invarianten



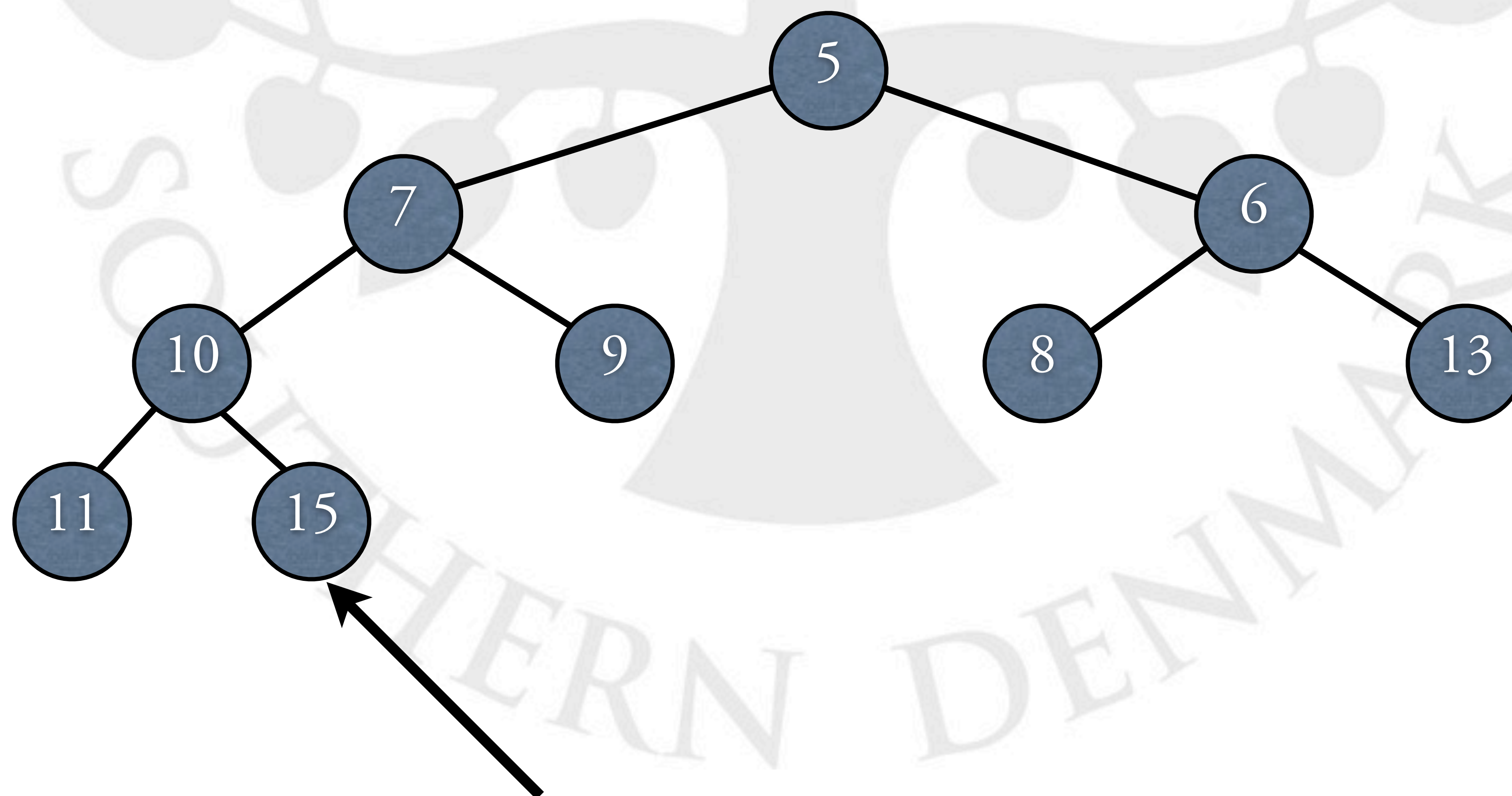
Hob

- deleteMin()
- 1. Udtag roden af træet (til senere returnering)
- 2. Flyt den sidste knude i træet op i roden
  (så træet igen hænger sammen og er halvtags-formet)
- 3. Bobbel roden ned indtil træet igen opfyder hob-invarianten

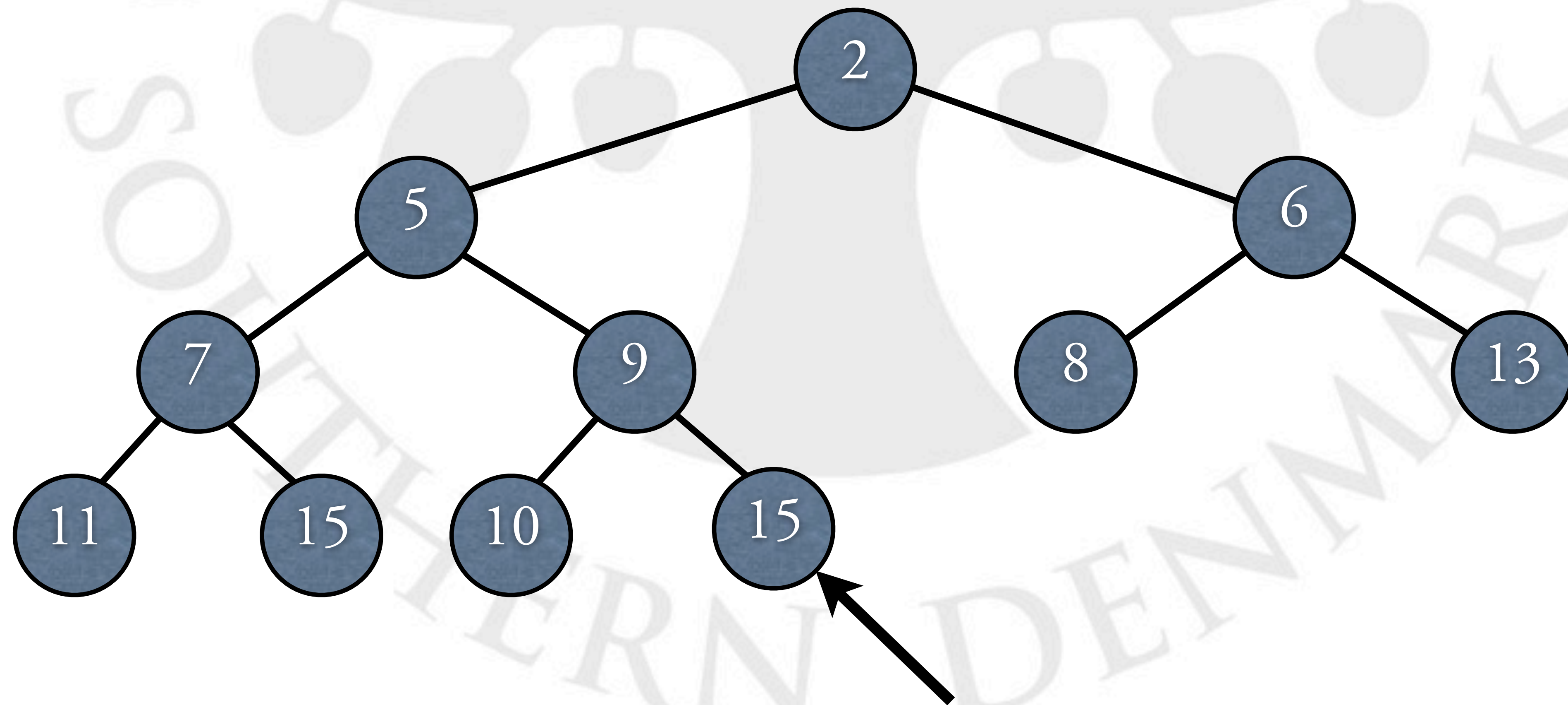


Hob

- deleteMin()
- 1. Udtag roden af træet (til senere returnering)
- 2. Flyt den sidste knude i træet op i roden
  (så træet igen hænger sammen og er halvtags-formet)
- 3. Bobbel roden ned indtil træet igen opfyder
  hob-invarianten



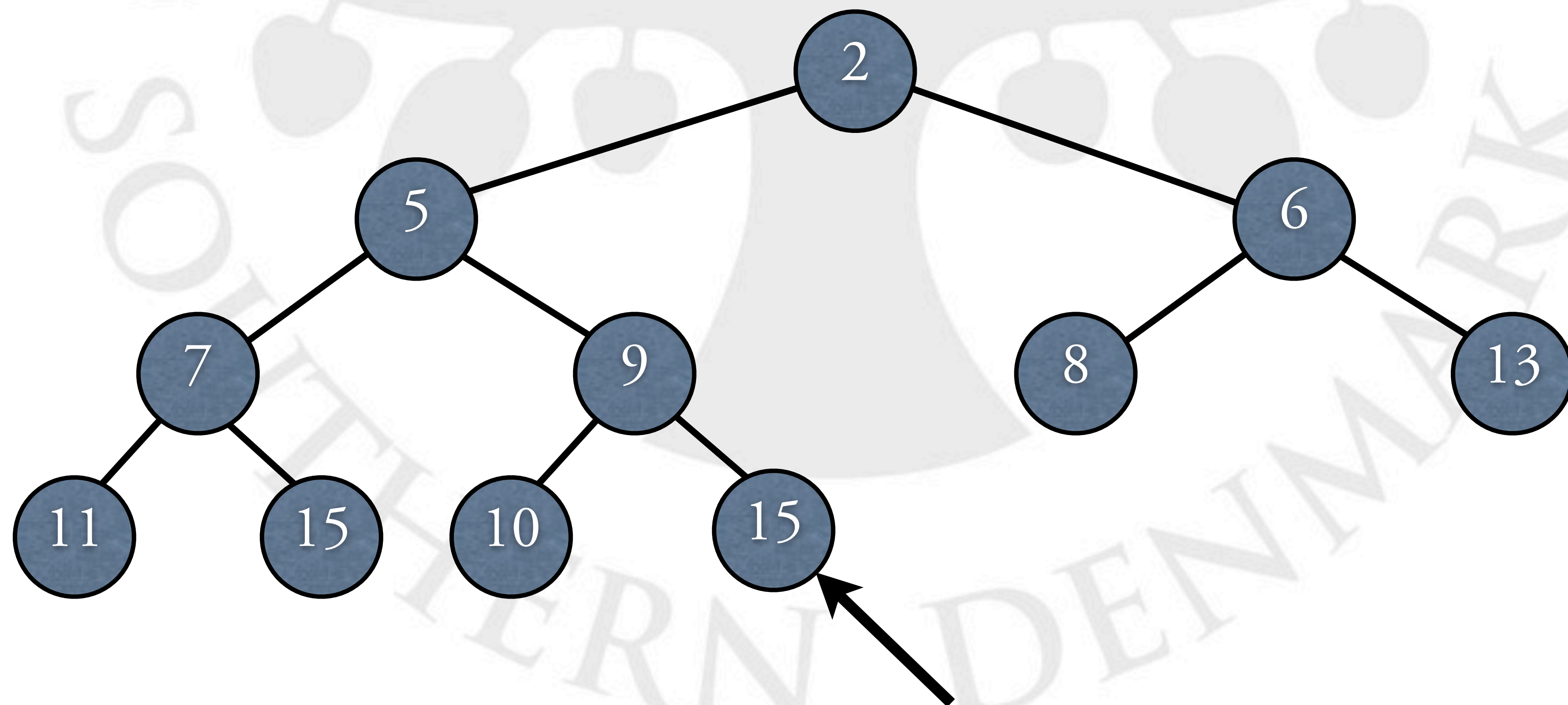
Hob



University of Southern Denmark • Syddansk Universitet

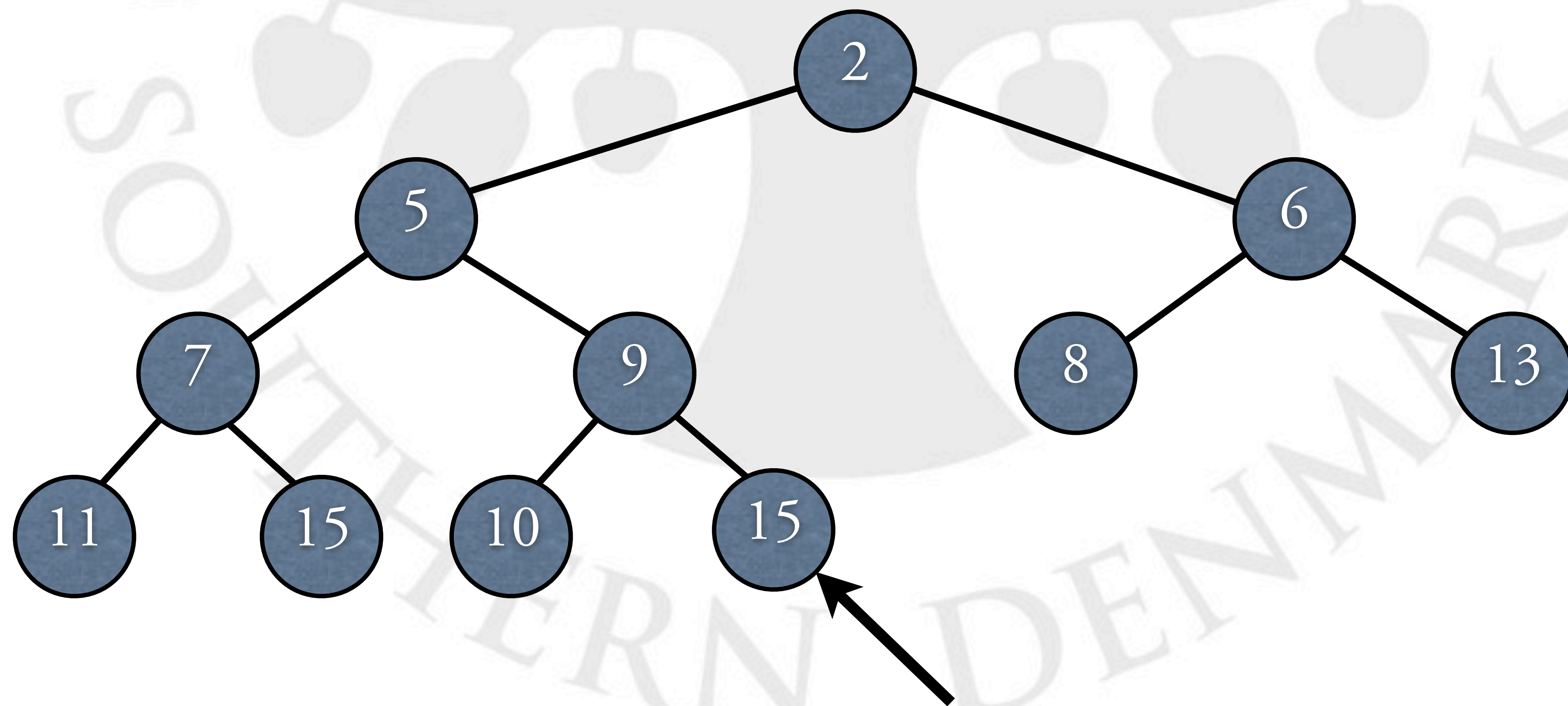
Hob

- Find den nye "sidste knude" for både insert og deleteMin



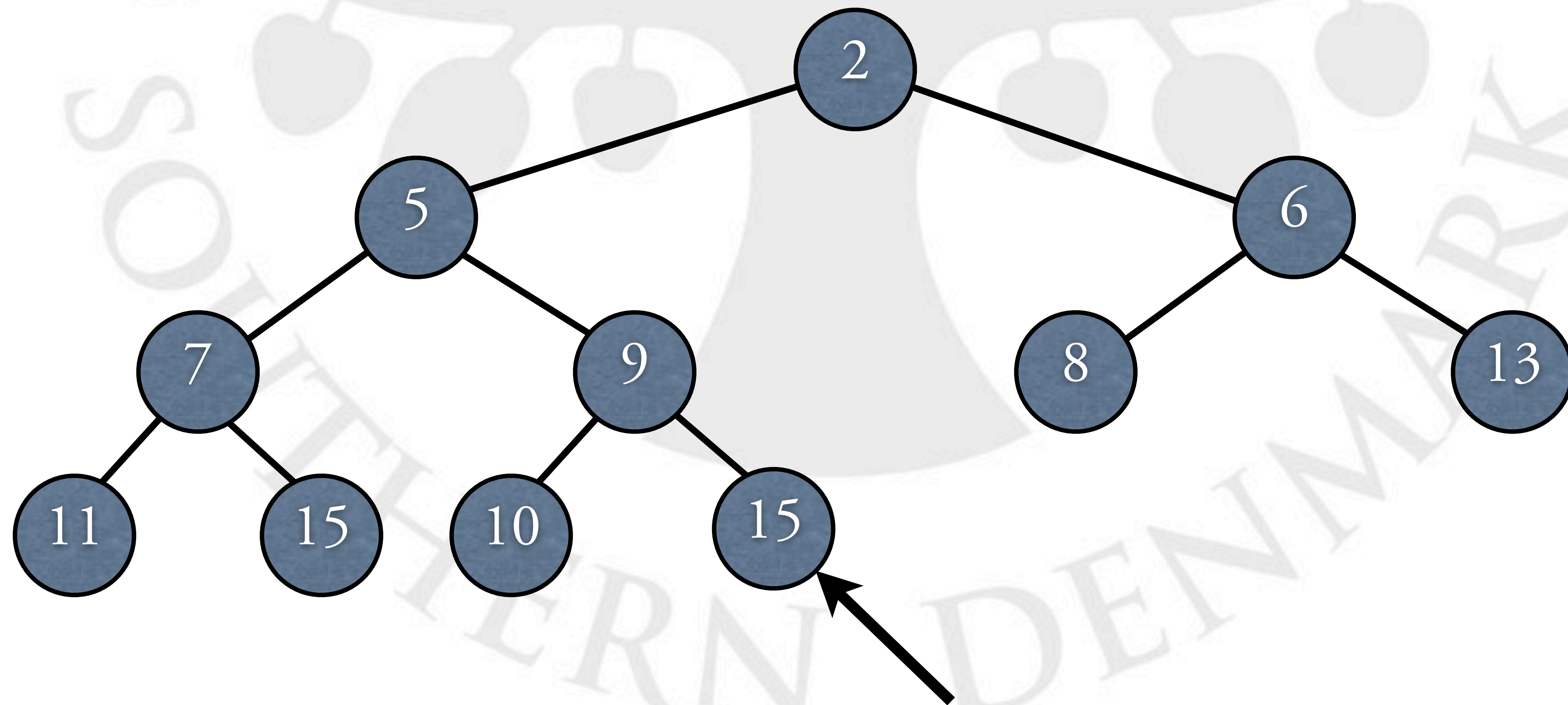
Hob

- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)



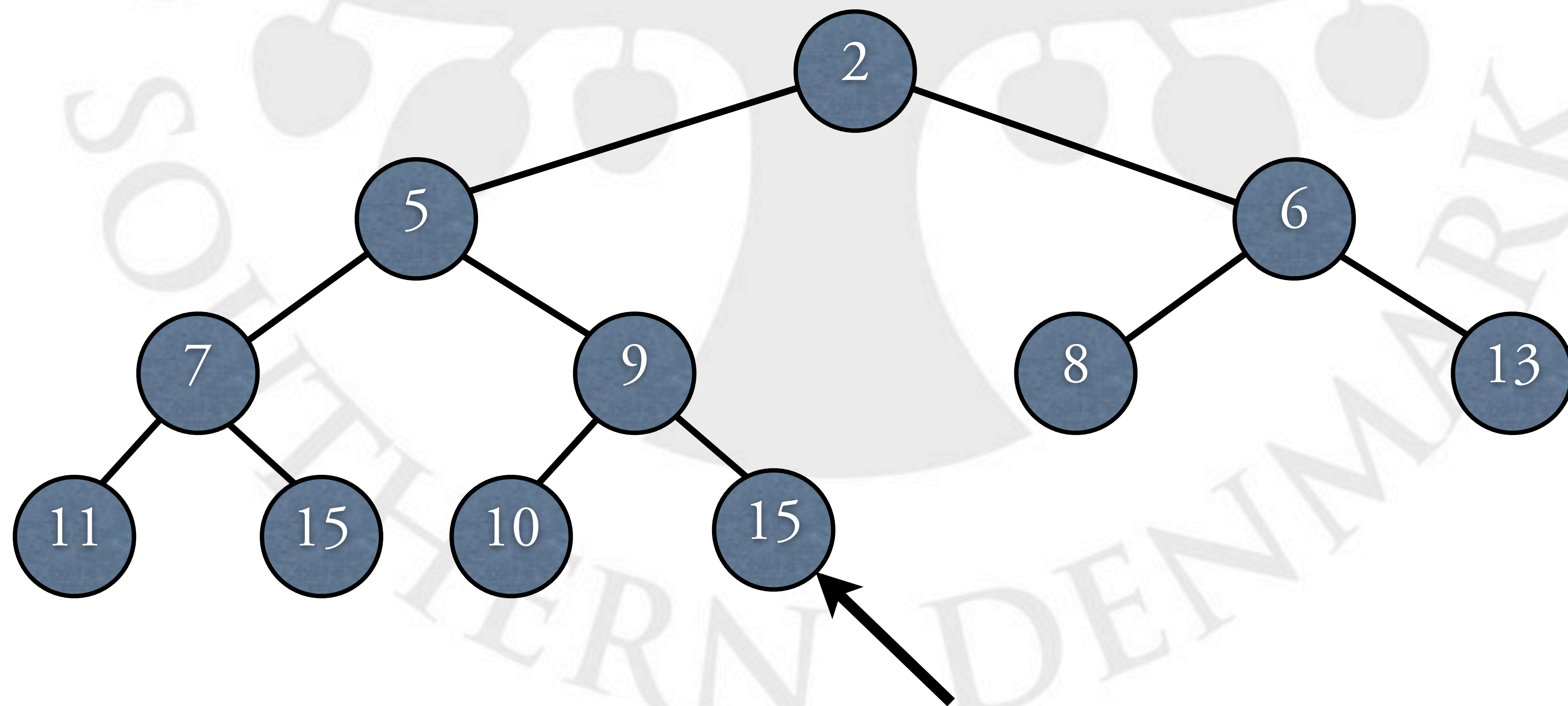
Hob

- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)
  - 1. Gå opad indtil et venstre-barn (eller roden) mødes



Hob

- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)
  - 1. Gå opad indtil et venstre-barn (eller roden) mødes
  - 2. Hvis det er et venstre-barn, gå til højre-barnet

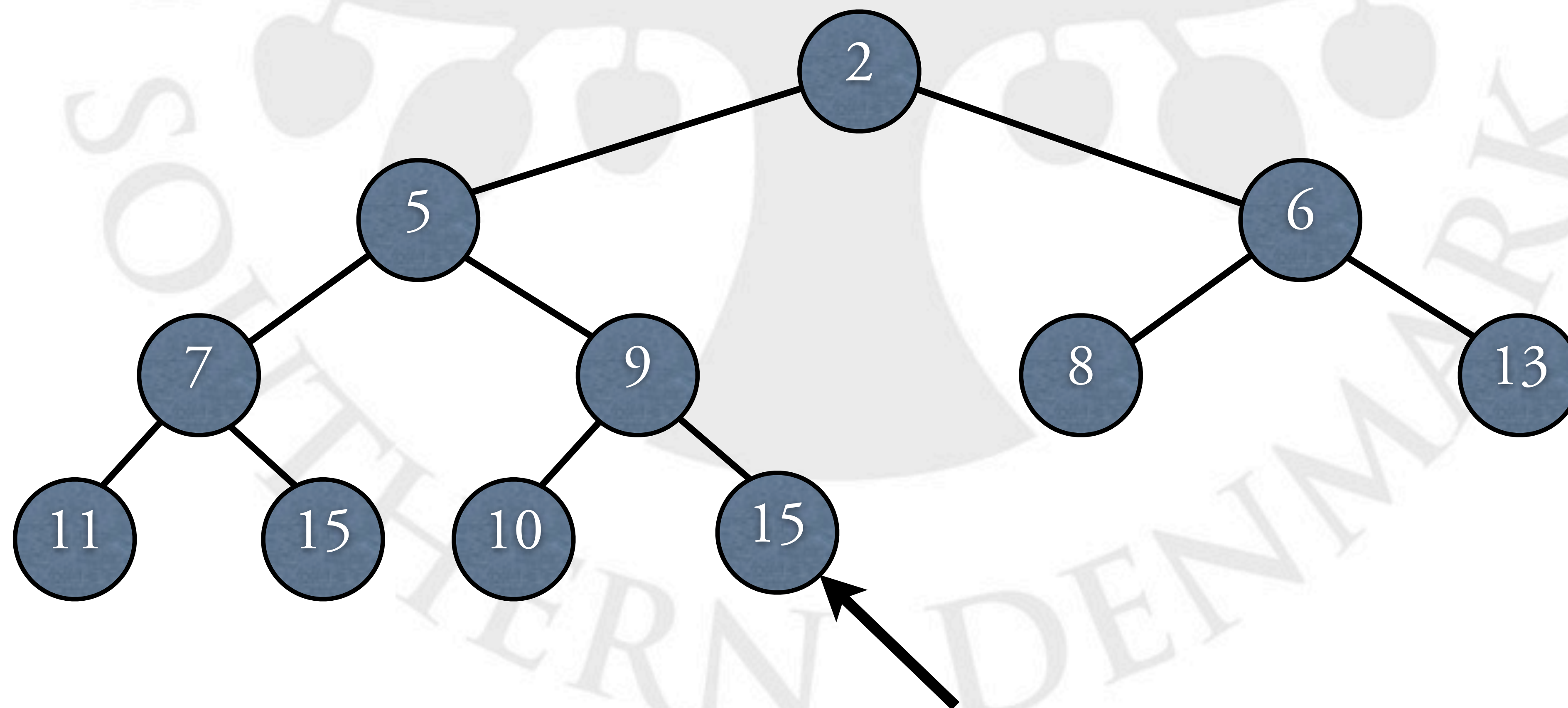


University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

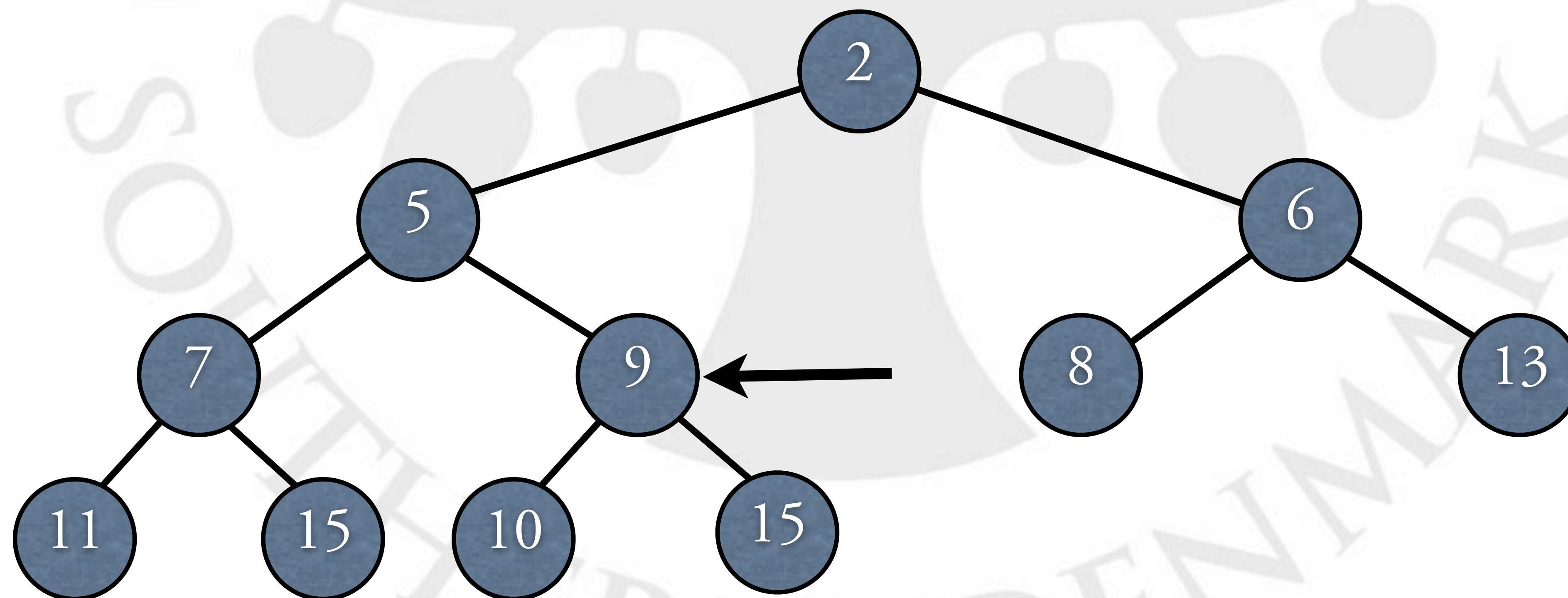
Hob

- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)
    - 1. Gå opad indtil et venstre-barn (eller roden) mødes
    - 2. Hvis det er et venstre-barn, gå til højre-barnet
    - 3. Gå nedad til venstre indtil et blad mødes



Hob

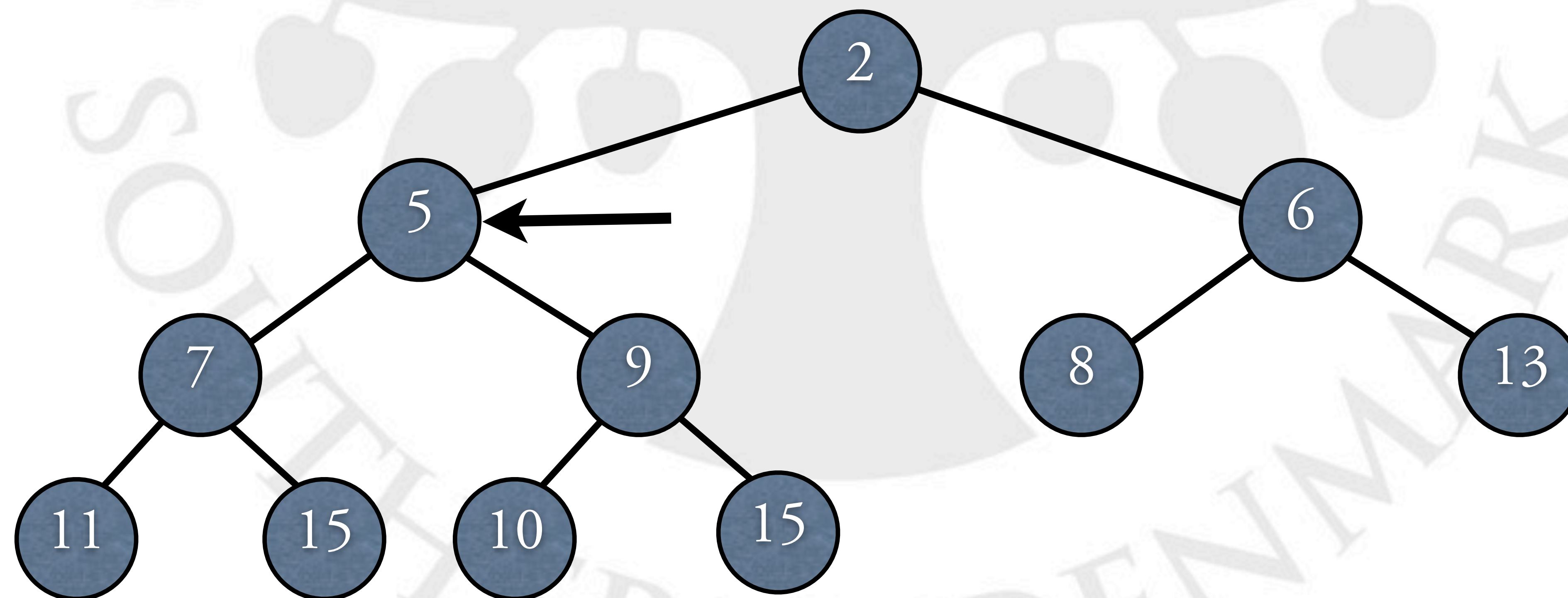
- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)
  - 1. Gå opad indtil et venstre-barn (eller roden) mødes
  - 2. Hvis det er et venstre-barn, gå til højre-barnet
  - 3. Gå nedad til venstre indtil et blad mødes



University of Southern Denmark • Syddansk Universitet

Hob

- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)
  - 1. Gå opad indtil et venstre-barn (eller roden) mødes
  - 2. Hvis det er et venstre-barn, gå til højre-barnet
  - 3. Gå nedad til venstre indtil et blad mødes

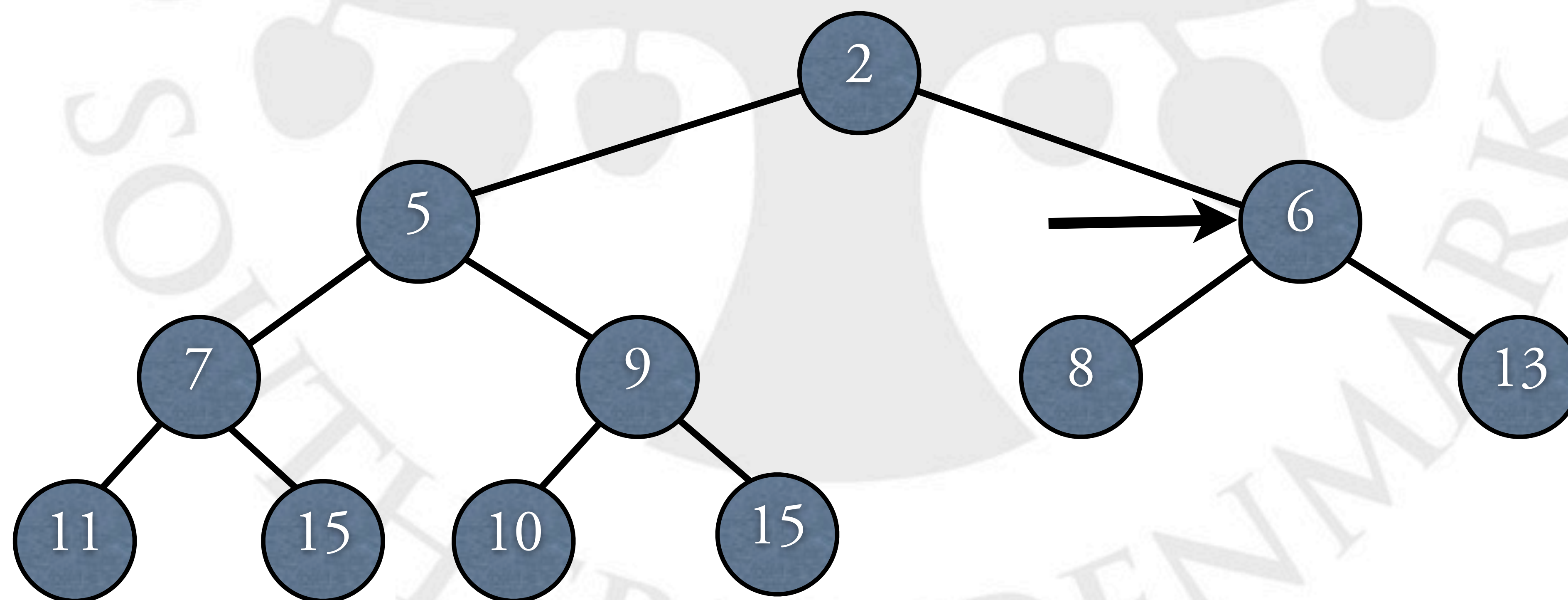


University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

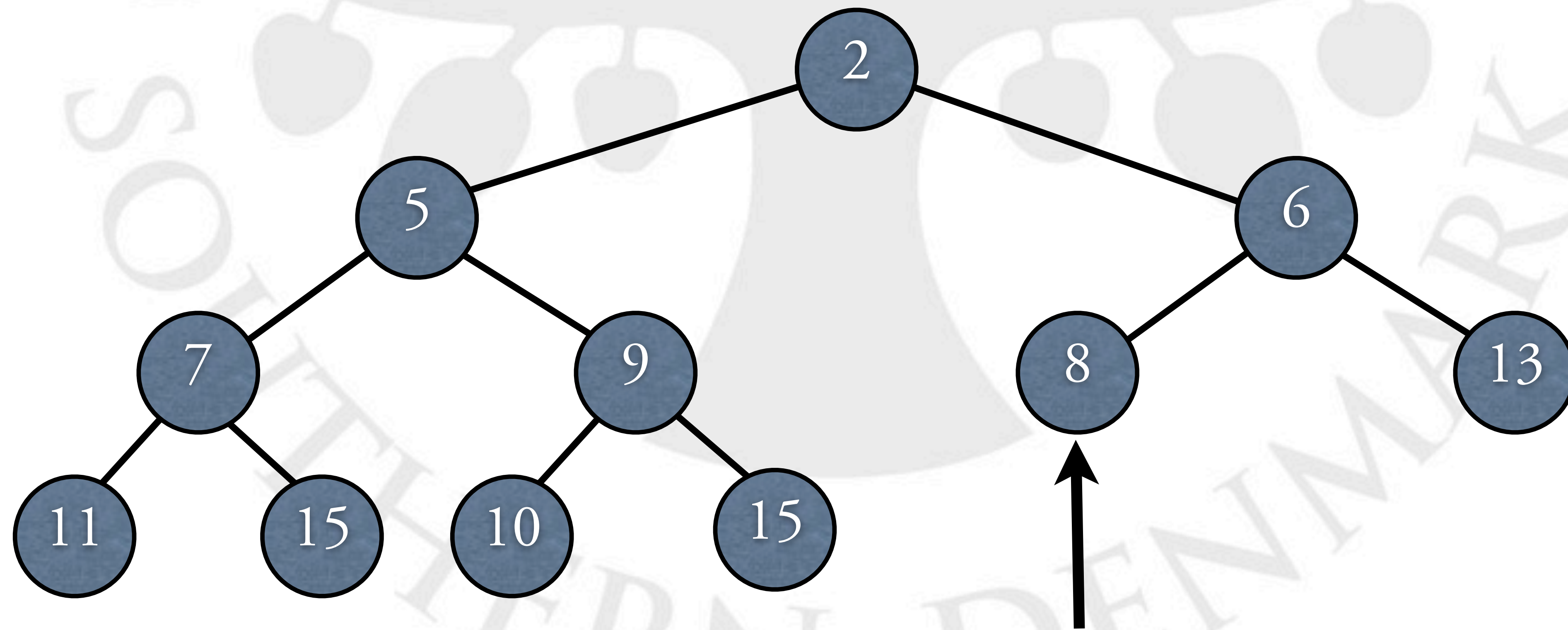
Hob

- Find den nye “sidste knude” for både insert og deleteMin
- insert(k, x)
    - 1. Gå opad indtil et venstre-barn (eller roden) mødes
    - 2. Hvis det er et venstre-barn, gå til højre-barnet
    - 3. Gå nedad til venstre indtil et blad mødes



Hob

- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)
    - 1. Gå opad indtil et venstre-barn (eller roden) mødes
    - 2. Hvis det er et venstre-barn, gå til højre-barnet
    - 3. Gå nedad til venstre indtil et blad mødes

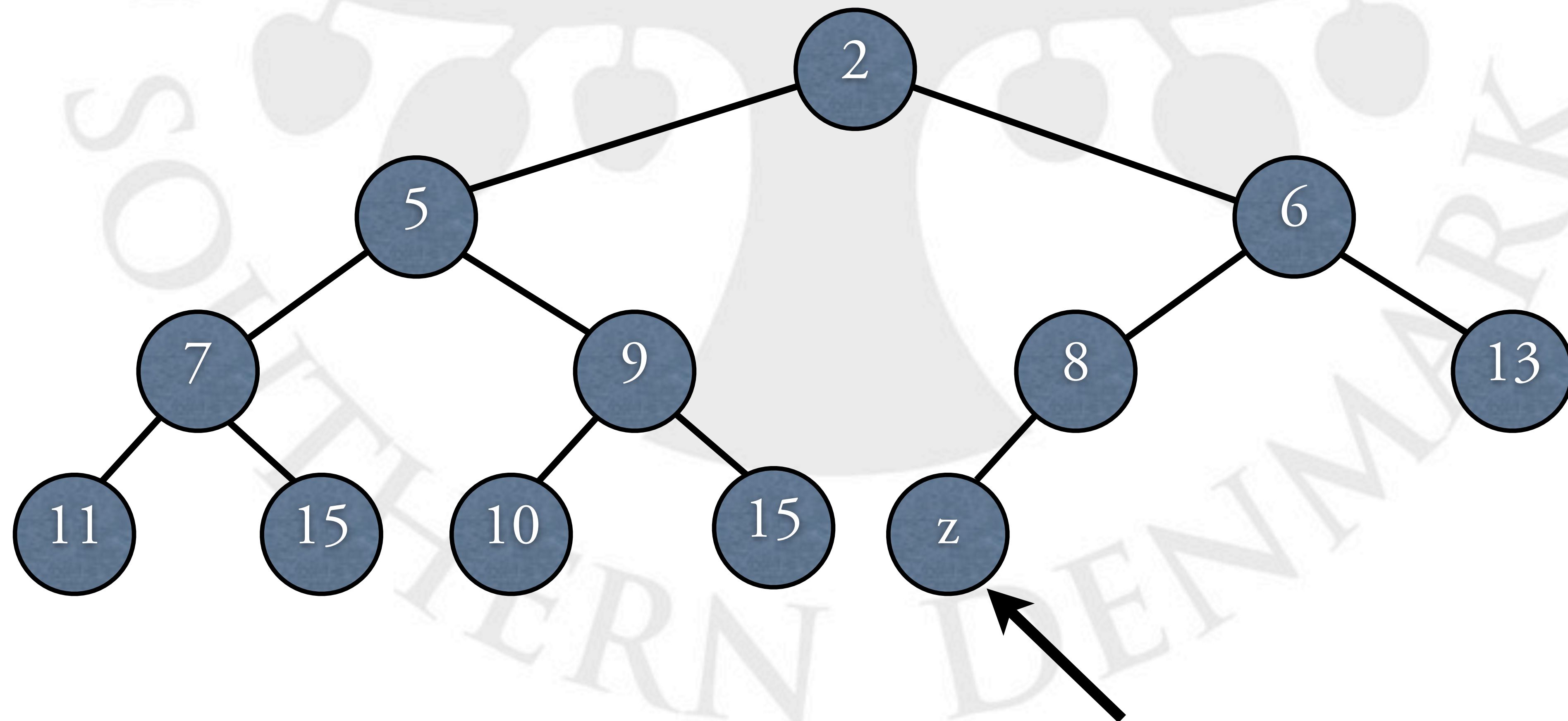


University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

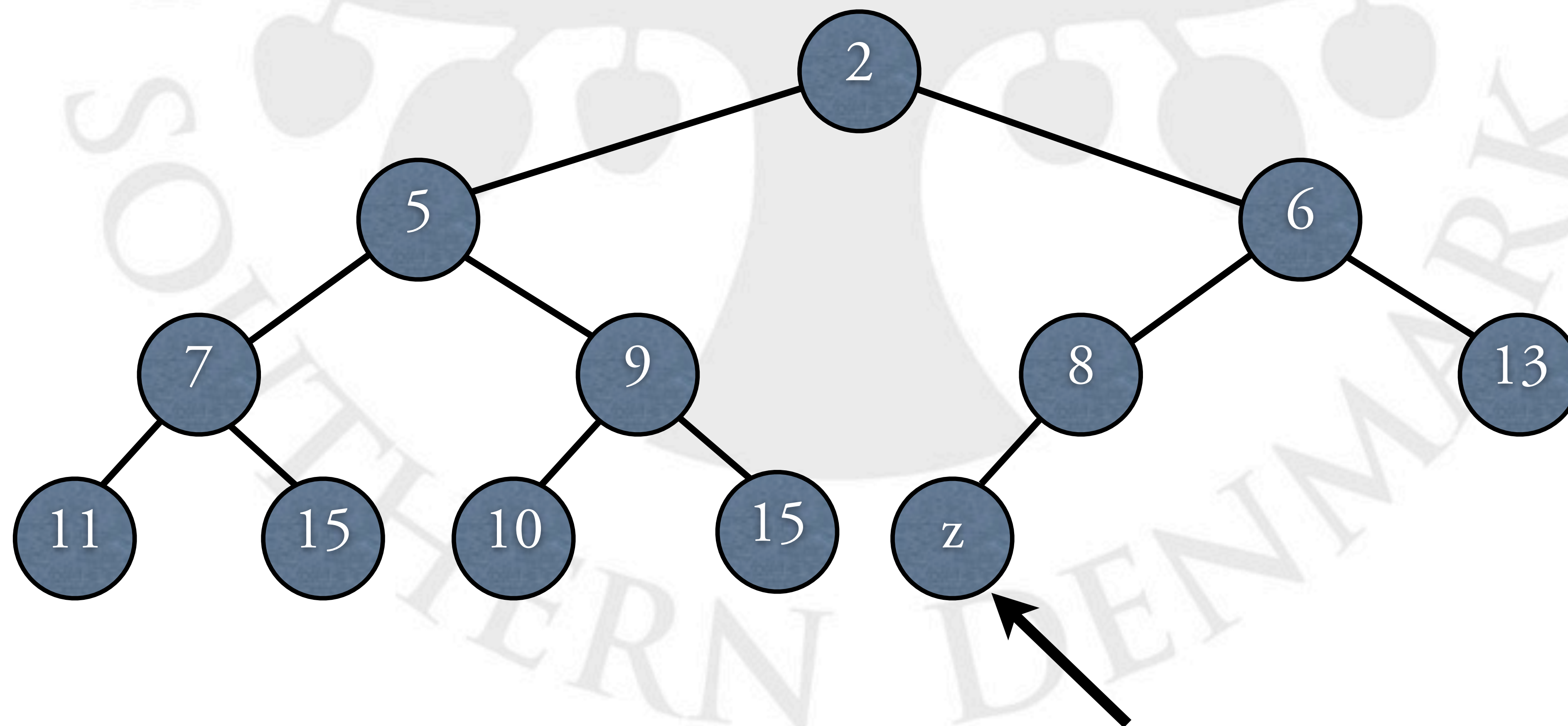
Hob

- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)
  - 1. Gå opad indtil et venstre-barn (eller roden) mødes
  - 2. Hvis det er et venstre-barn, gå til højre-barnet
  - 3. Gå nedad til venstre indtil et blad mødes



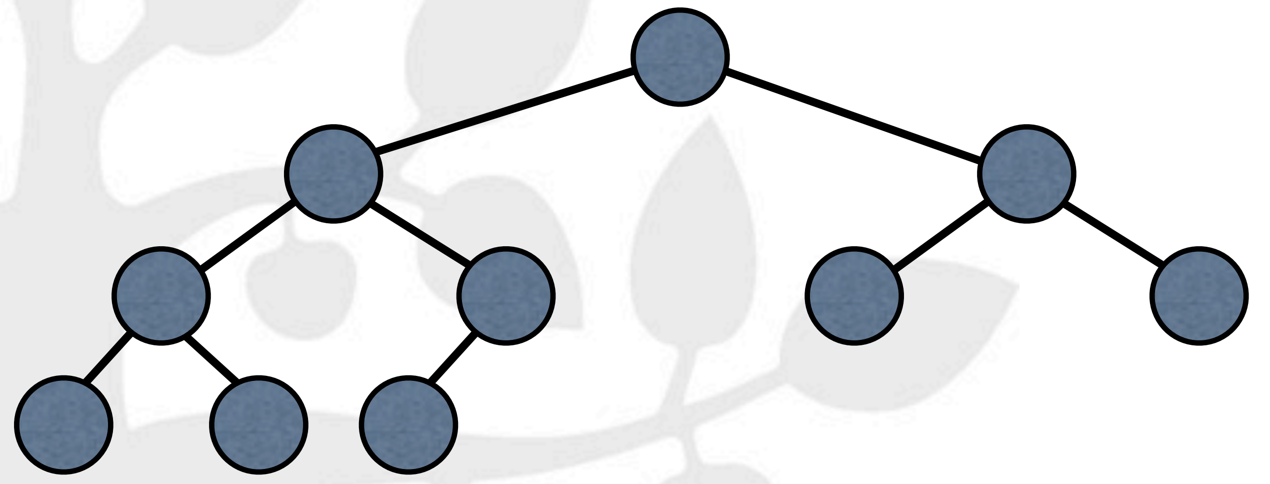
Hob

- Find den nye "sidste knude" for både insert og deleteMin
- insert(k, x)
  - 1. Gå opad indtil et venstre-barn (eller roden) mødes
  - 2. Hvis det er et venstre-barn, gå til højre-barnet
  - 3. Gå nedad til venstre indtil et blad mødes
- Tilsvarende for deleteMin()



University of Southern Denmark • Syddansk Universitet

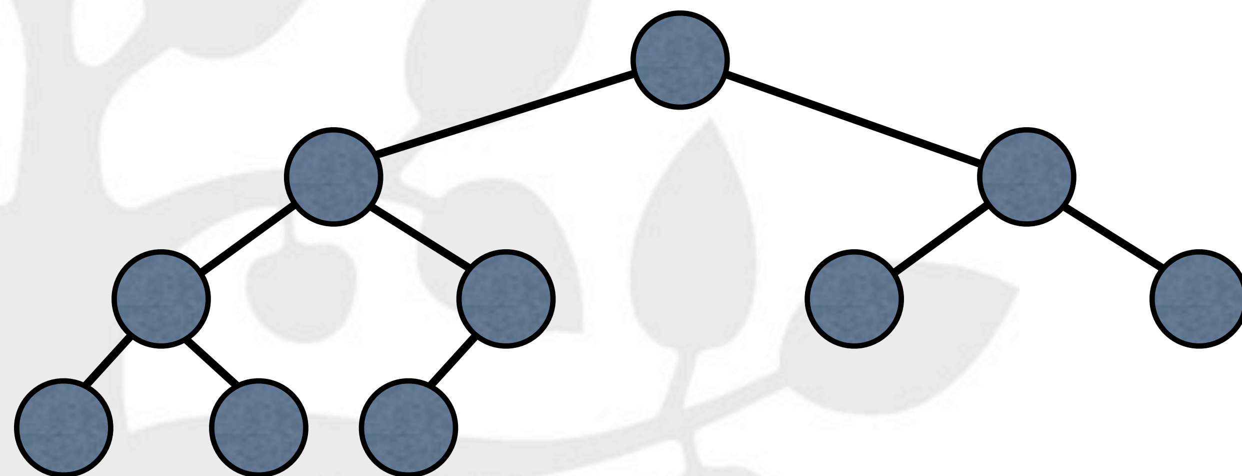
Hob



University of Southern Denmark • Syddansk Universitet

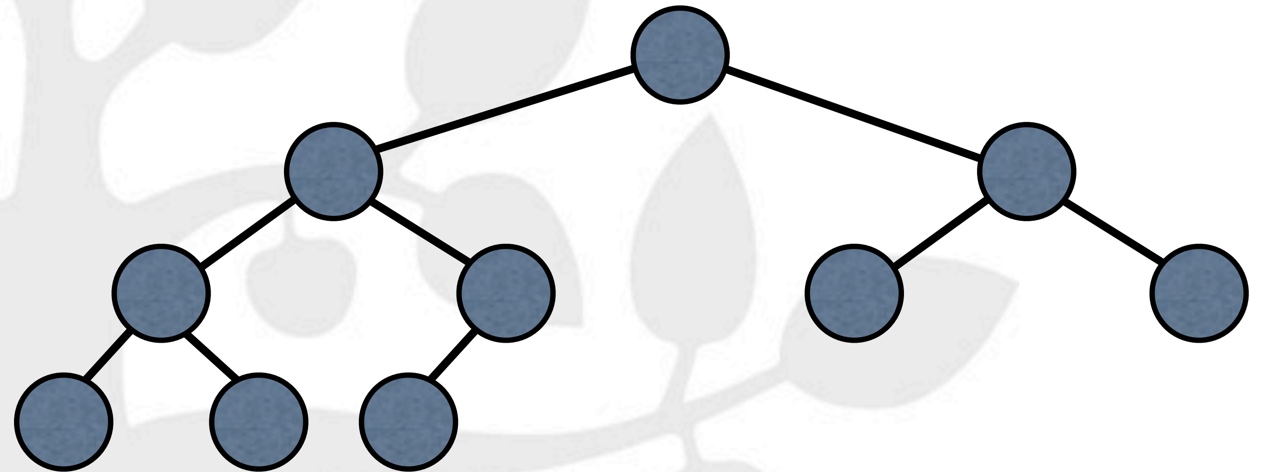
Hob

- En hob med n elementer har højst højde log(n)



Hob

- En hob med n elementer har højst højde log(n)
- Vi bruger at en hob er et fuldstændigt binært træ

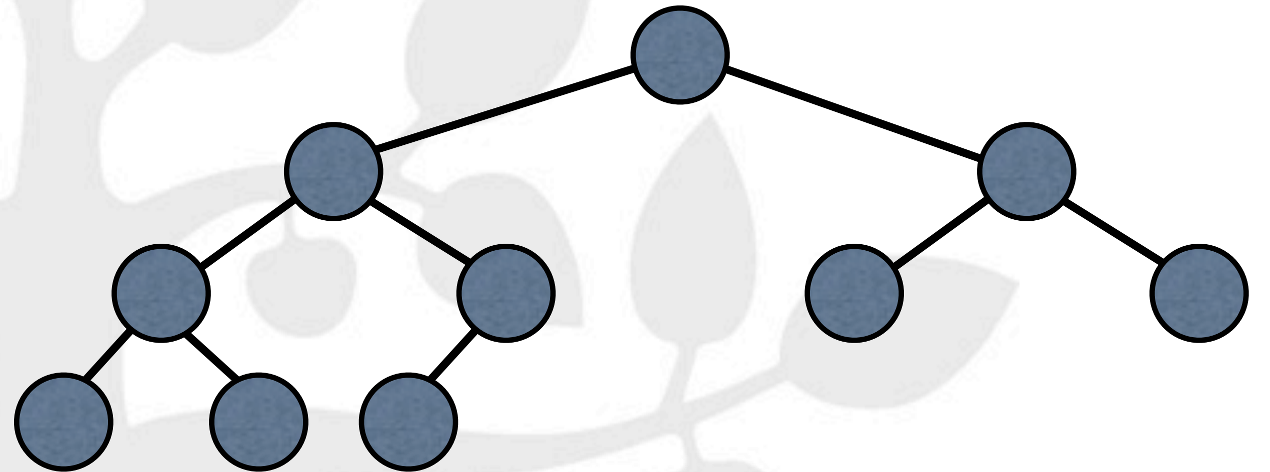


University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

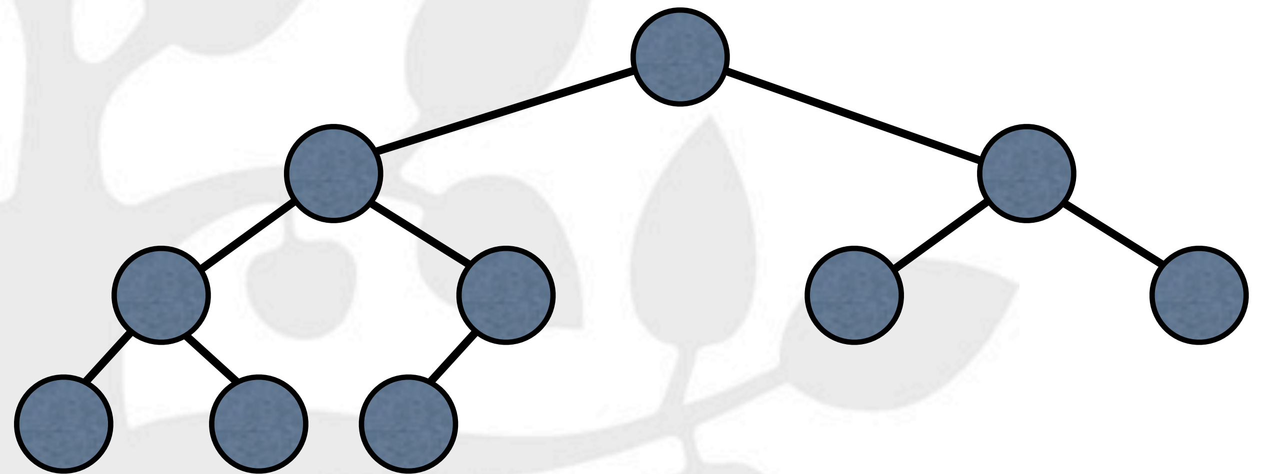
Hob

- En hob med n elementer har højst højde log(n)
- Vi bruger at en hob er et fuldstændigt binært træ
- Dybde 0: 1 knude



Hob

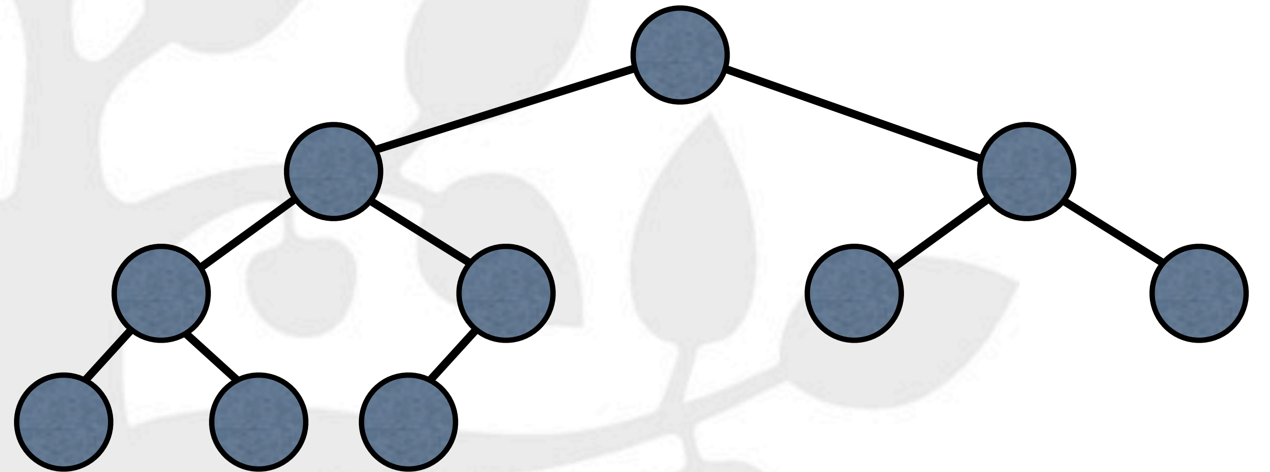
- En hob med n elementer har højst højde log(n)
- Vi bruger at en hob er et fuldstændigt binært træ
- Dybde 0: 1 knude
- Dybde 1: 2 knuder



University of Southern Denmark • Syddansk Universitet

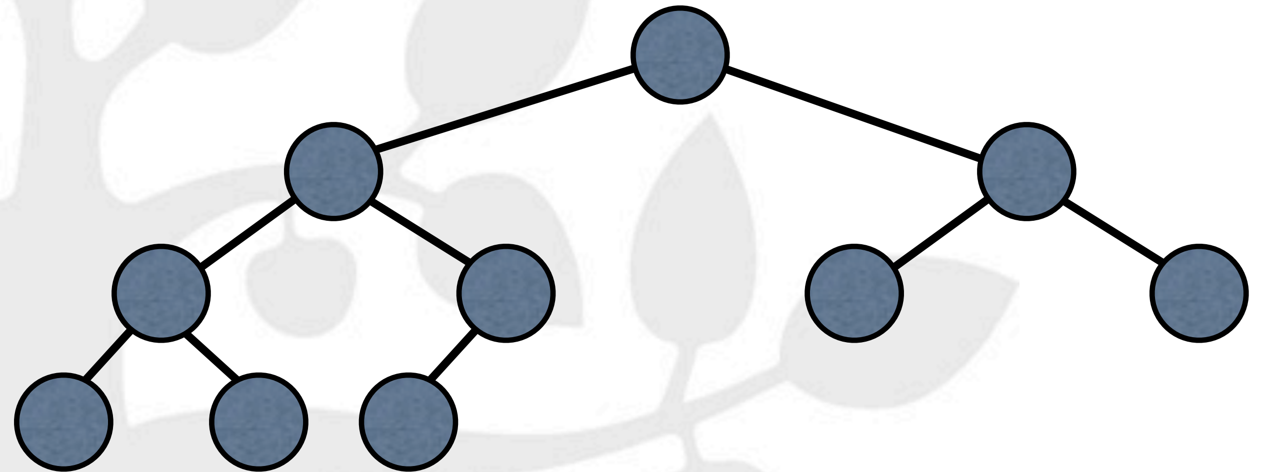
Hob

- En hob med n elementer har højst højde log(n)
- Vi bruger at en hob er et fuldstændigt binært træ
- Dybde 0: 1 knude
- Dybde 1: 2 knuder
- ...



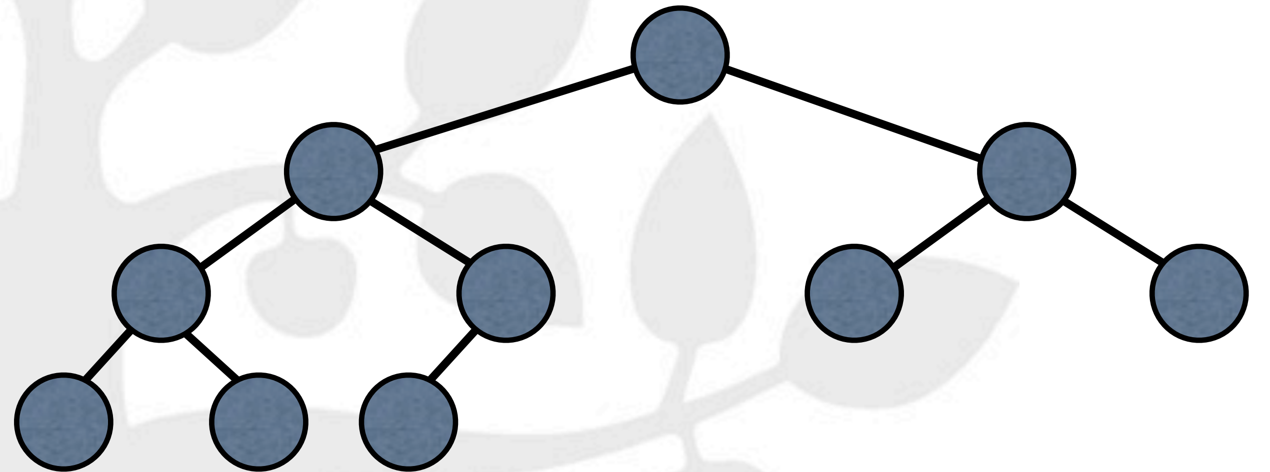
Hob

- En hob med n elementer har højst højde log(n)
- Vi bruger at en hob er et fuldstændigt binært træ
- Dybde 0: 1 knude
- Dybde 1: 2 knuder
- ...
- Dybde h-1: 2^{h-1}



Hob

- En hob med n elementer har højst højde log(n)
- Vi bruger at en hob er et fuldstændigt binært træ
- Dybde 0: 1 knude
- Dybde 1: 2 knuder
- ...
- Dybde h-1: 2^{h-1}
- Dybe h: Mindst 1
  Højst 2^h



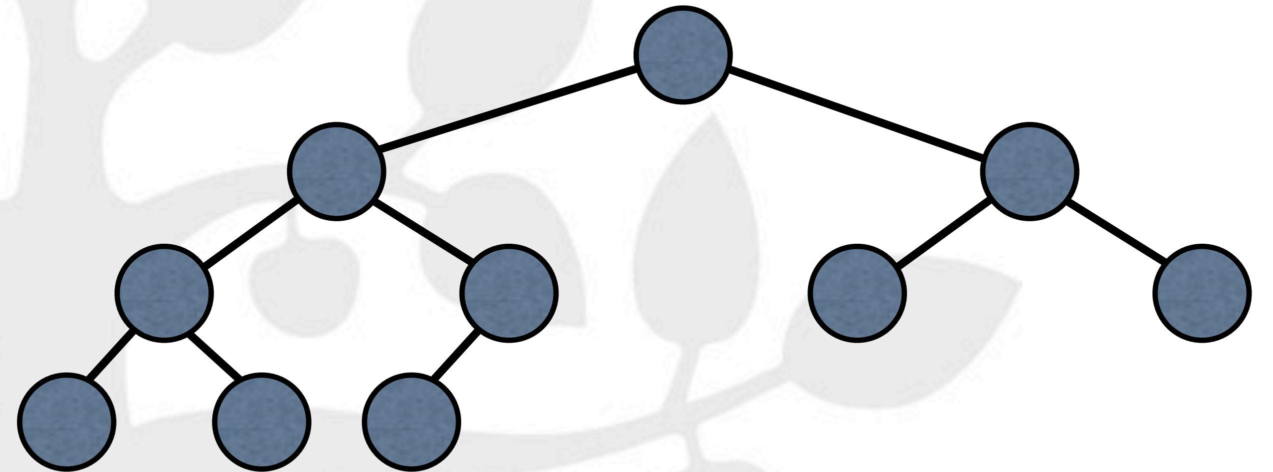
Hob

- En hob med n elementer har højst højde log(n)
- Vi bruger at en hob er et fuldstændigt binært træ
- Dybde 0: 1 knude
- Dybde 1: 2 knuder
- ...
- Dybde h-1: 2^{h-1}
- Dybe h: Mindst 1

  Højst 2^h
- $n \geq 1 + 2 + 4 + \cdots + 2^{h-1} + 1$

  $= (2^h - 1) + 1$

  $= 2^h$

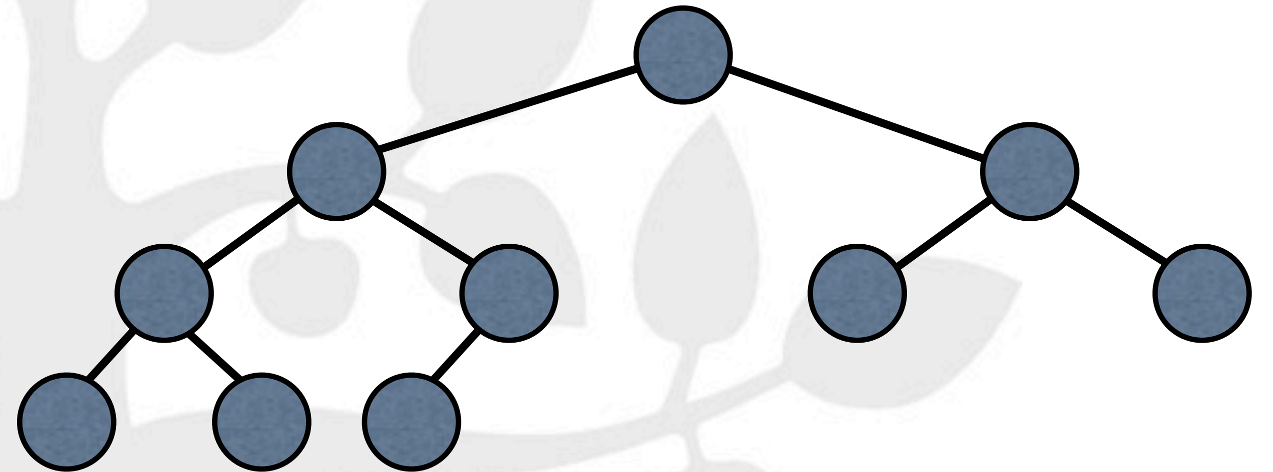


University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

Hob

- En hob med n elementer har højst højde log(n)
- Vi bruger at en hob er et fuldstændigt binært træ
- Dybde 0: 1 knude
- Dybde 1: 2 knuder
- ...
- Dybde h-1: 2^{h-1}
- Dybe h: Mindst 1
  Højst 2^h
- $n \geq 1 + 2 + 4 + \cdots + 2^{h-1} + 1$
  $= (2^h - 1) + 1$
  $= 2^h$



- Dvs. $h \leq \log n$

Hob

Hob

- Tid == antal sammenligninger

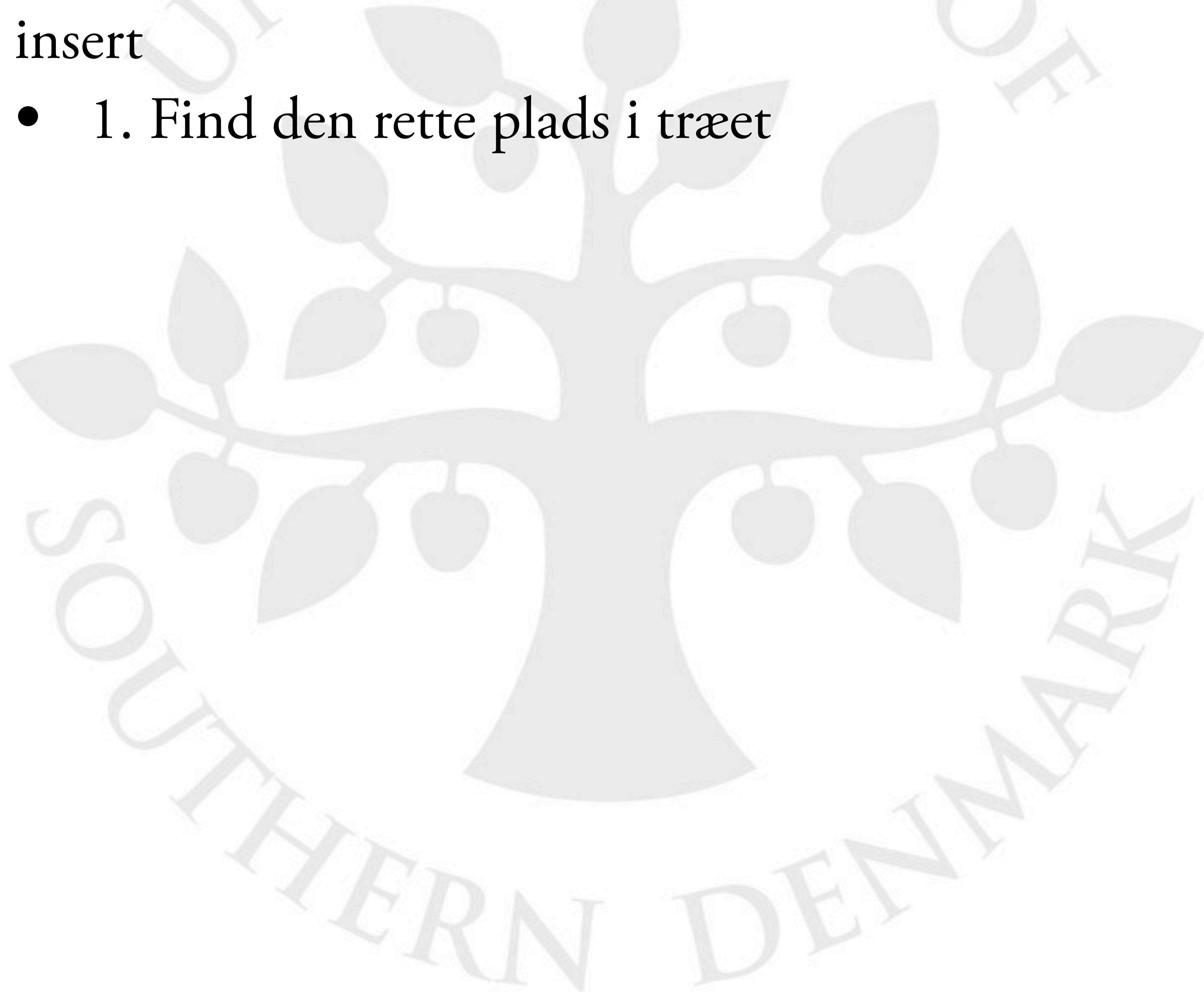
University of Southern Denmark • Syddansk Universitet

Hob

- Tid == antal sammenligninger
- insert

University of Southern Denmark • Syddansk Universitet

Hob



- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet

University of Southern Denmark • Syddansk Universitet

Hob

- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet
    - I værste fald fra det nederste lag, hele vejen til roden og tilbage til det nederste lag

Hob

- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet
    - I værste fald fra det nederste lag, hele vejen til roden og tilbage til det nederste lag
    - $\leq 2 \log n$

University of Southern Denmark • Syddansk Universitet

Hob

- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet
    - I værste fald fra det nederste lag, hele vejen til roden og tilbage til det nederste lag
    - $\leq 2 \log n$
  - 2. Gem x i z

Hob

- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet
    - I værste fald fra det nederste lag, hele vejen til roden og tilbage til det nederste lag
    - $\leq 2 \log n$
  - 2. Gem x i z
    - Ingen sammenligninger

Hob

- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet
    - I værste fald fra det nederste lag, hele vejen til roden og tilbage til det nederste lag
    - $\leq 2 \log n$
  - 2. Gem x i z
    - Ingen sammenligninger
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten

University of Southern Denmark • Syddansk Universitet

Hob

- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet
    - I værste fald fra det nederste lag, hele vejen til roden og tilbage til det nederste lag
    - $\leq 2 \log n$
  - 2. Gem x i z
    - Ingen sammenligninger
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten
    - I værste fald bobles x hele vejen til roden

University of Southern Denmark • Syddansk Universitet

University of Southern Denmark • Syddansk Universitet

Hob

- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet
    - I værste fald fra det nederste lag, hele vejen til roden og tilbage til det nederste lag
    - $\leq 2 \log n$
  - 2. Gem x i z
    - Ingen sammenligninger
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten
    - I værste fald bobles x hele vejen til roden
    - $\leq \log n$

Hob

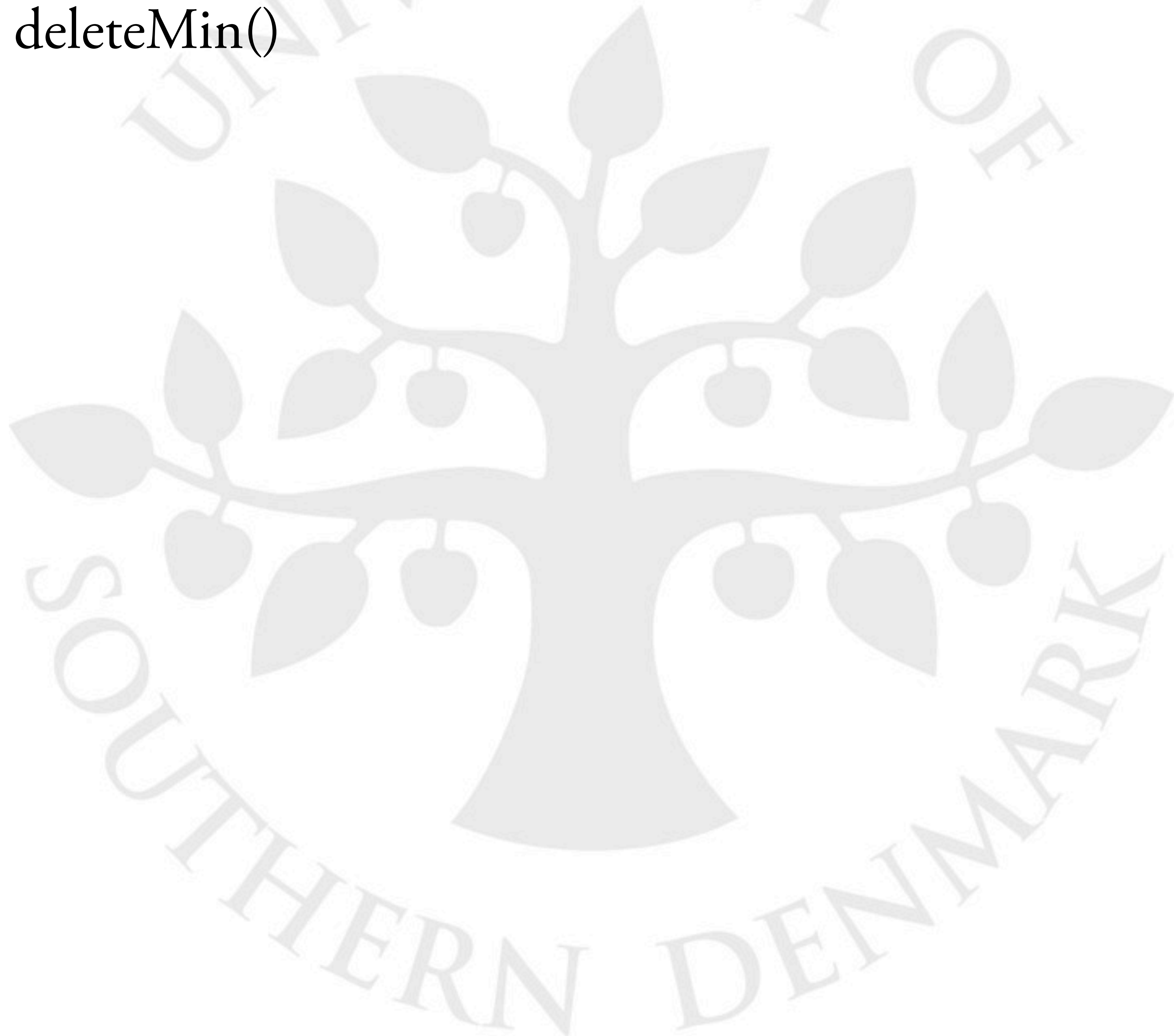
- Tid == antal sammenligninger
- insert
  - 1. Find den rette plads i træet
    - I værste fald fra det nederste lag, hele vejen til roden og tilbage til det nederste lag
    - $\leq 2 \log n$
  - 2. Gem x i z
    - Ingen sammenligninger
  - 3. "Bobbel" x opad indtil træet igen opfylder hob-invarianten
    - I værste fald bobles x hele vejen til roden
    - $\leq \log n$
  - Total $\leq 3 \log n$

University of Southern Denmark • Syddansk Universitet

Hob

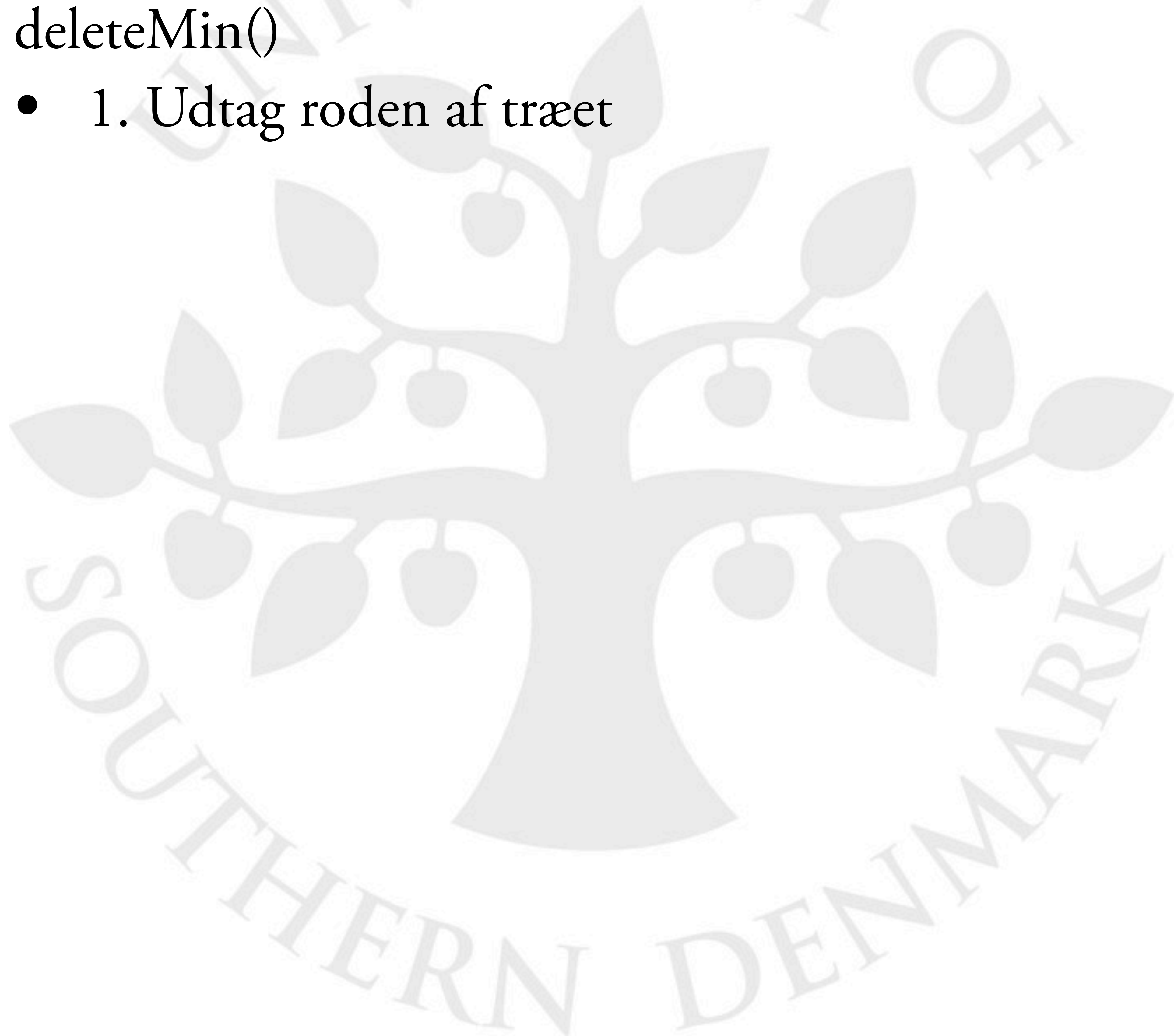
Hob

- deleteMin()



University of Southern Denmark • Syddansk Universitet

Hob



- deleteMin()
  - 1. Udtag roden af træet

University of Southern Denmark • Syddansk Universitet

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligninger

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligniner
  - 2. Flyt den sidste knude i træet z op i roden

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligniner
  - 2. Flyt den sidste knude i træet z op i roden
    - Umiddelbart ingen sammenligninger, dog skal vi genberegne hvor den sidste knude er

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligniner
  - 2. Flyt den sidste knude i træet z op i roden
    - Umiddelbart ingen sammenligninger, dog skal vi genberegne hvor den sidste knude er
    - $\leq 2 \log n$

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligniner
  - 2. Flyt den sidste knude i træet z op i roden
    - Umiddelbart ingen sammenligninger, dog skal vi genberegne hvor den sidste knude er
    - $\leq 2\log n$
  - 3. Bobbel roden ned indtil træet igen opfylder hob-invarianten

University of Southern Denmark • Syddansk Universitet

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligniner
  - 2. Flyt den sidste knude i træet z op i roden
    - Umiddelbart ingen sammenligninger, dog skal vi genberegne hvor den sidste knude er
    - $\leq 2 \log n$
  - 3. Bobbel roden ned indtil træet igen opfylder hob-invarianten
    - I hvert lag skal z sammenlignes med sine to børn

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligniner
  - 2. Flyt den sidste knude i træet z op i roden
    - Umiddelbart ingen sammenligninger, dog skal vi genberegne hvor den sidste knude er
    - $\leq 2 \log n$
  - 3. Bobbel roden ned indtil træet igen opfylder hob-invarianten
    - I hvert lag skal z sammenlignes med sine to børn
    - Dvs. to sammenligninger i hvert lag

University of Southern Denmark • Syddansk Universitet

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligniner
  - 2. Flyt den sidste knude i træet z op i roden
    - Umiddelbart ingen sammenligninger, dog skal vi genberegne hvor den sidste knude er
    - $\leq 2 \log n$
  - 3. Bobbel roden ned indtil træet igen opfylder hob-invarianten
    - I hvert lag skal z sammenlignes med sine to børn
    - Dvs. to sammenligninger i hvert lag
    - $\leq 2 \log n$

Hob

- deleteMin()
  - 1. Udtag roden af træet
    - Ingen sammenligniner
  - 2. Flyt den sidste knude i træet z op i roden
    - Umiddelbart ingen sammenligninger, dog skal vi genberegne hvor den sidste knude er
    - $\leq 2 \log n$
  - 3. Bobbel roden ned indtil træet igen opfylder hob-invarianten
    - I hvert lag skal z sammenlignes med sine to børn
    - Dvs. to sammenligninger i hvert lag
    - $\leq 2 \log n$
  - Total $\leq 4 \log n$

Sortering

Sortering

- Sortering vha. en prioritetskø implementeret som en hob

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret

University of Southern Denmark • Syddansk Universitet

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert
    - 1. element: $\leq 3 \log 1$

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert
  - 1. element: $\leq 3 \log 1$
  - 2. element: $\leq 3 \log 2$

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert
  - 1. element: $\leq 3 \log 1$
  - 2. element: $\leq 3 \log 2$
  - ...

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert
  - 1. element: $\leq 3 \log 1$
  - 2. element: $\leq 3 \log 2$
  - ...
  - n'te element: $\leq 3 \log n$

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert
  - 1. element: $\leq 3 \log 1$
  - 2. element: $\leq 3 \log 2$
  - ...
  - n'te element: $\leq 3 \log n$
  - Total $\leq \sum_{i=1}^{n} 3 \log i \leq 3n \log n$

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert
  - 1. element: $\leq 3 \log 1$
  - 2. element: $\leq 3 \log 2$
  - ...
  - n'te element: $\leq 3 \log n$
  - Total $\leq \sum_{i=1}^{n} 3 \log i \leq 3n \log n$
- 2. n gange deleteMin()

University of Southern Denmark • Syddansk Universitet

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert
  - 1. element: $\leq 3 \log 1$
  - 2. element: $\leq 3 \log 2$
  - ...
  - n'te element: $\leq 3 \log n$
  - Total $\leq \sum_{i=1}^{n} 3 \log i \leq 3n \log n$
- 2. n gange deleteMin()
  - Som for pkt. 1. dog med 4n log(n) i stedet for 3n log(n)

Sortering

- Sortering vha. en prioritetskø implementeret som en hob
- n elementer vi vil have sorteret
- 1. n gange insert
  - 1. element: $\leq 3 \log 1$
  - 2. element: $\leq 3 \log 2$
  - ...
  - n'te element: $\leq 3 \log n$
  - Total $\leq \sum_{i=1}^{n} 3 \log i \leq 3n \log n$
- 2. n gange deleteMin()
  - Som for pkt. 1. dog med 4n log(n) i stedet for 3n log(n)
- Total $\leq 3n \log n + 4n \log n = 7n \log n$

University of Southern Denmark • Syddansk Universitet

Hob

- Hvad med metoderne?
  - min()
  - size()
  - isEmpty()