COS
Computersystemer

Lektion #9
Algoritmer.

Morten Hansen
moh@sdu.dk

Computersystemer
Computer Science

Computer Science er læren om algoritmer.

- **Algorithm:** A set of steps that defines how a task is performed
- **Program:** A representation of an algorithm
- **Programming:** The process of developing a program
- **Software:** Programs and the algorithms they represent
- **Hardware:** The machinery

Morten Hansen
moh@sdu.dk

Figure 0.1
An algorithm for a magic trick

Effect: The performer places some cards from a normal deck of playing cards face down on a table and mixes them thoroughly while spreading them out on the table. Then, as the audience requests either red or black cards, the performer turns over cards of the requested color.

Secret and Patter:

Step 1. From a normal deck of cards, select ten red cards and ten black cards. Deal these cards face up in two piles on the table according to color.

Step 2. Announce that you have selected some red cards and some black cards.

Step 3. Pick up the red cards. Under the pretense of aligning them into a small deck, hold them face down in your left hand and, with the thumb and first finger of your right hand, pull back on each end of the deck so that each card is given a slightly *backward* curve. Then place the deck of red cards face down on the table as you say, "Here are the red cards in this stack."

Step 4. Pick up the black cards. In a manner similar to that in step 3, give these cards a slight *forward* curve. Then return these cards to the table in a face-down deck as you say, "And here are the black cards in this stack."

Step 5. Immediately after returning the black cards to the table, use both hands to mix the red and black cards (still face down) as you spread them out on the tabletop. Explain that you are thoroughly mixing the cards.

Step 6. As long as there are face-down cards on the table, repeatedly execute the following steps:

6.1. Ask the audience to request either a red or a black card.

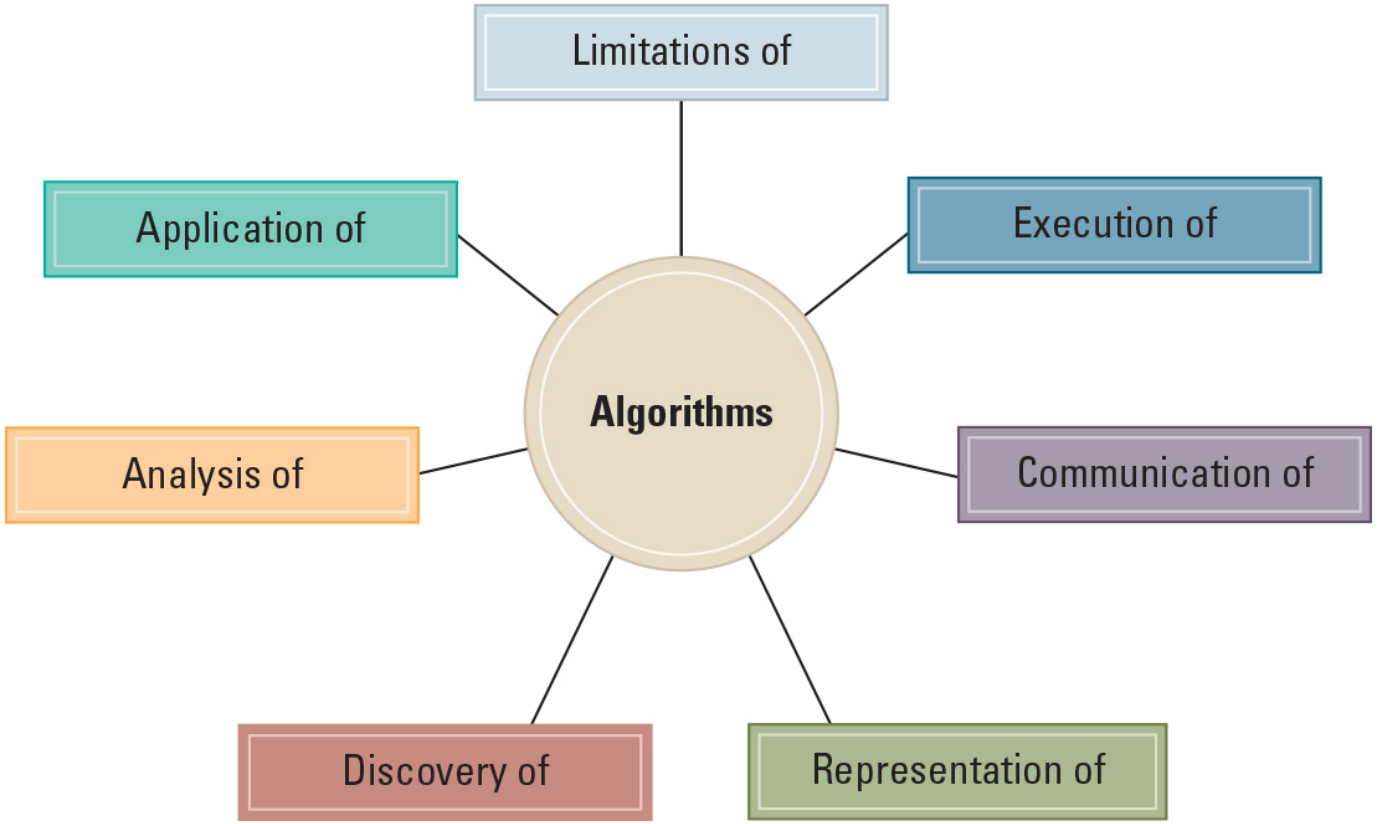
6.2. If the color requested is red and there is a face-down card with a concave appearance, turn over such a card while saying, "Here is a red card."

6.3. If the color requested is black and there is a face-down card with a convex appearance, turn over such a card while saying, "Here is a black card."

6.4. Otherwise, state that there are no more cards of the requested color and turn over the remaining cards to prove your claim.

Morten Hansen
moh@sdu.dk

Figure 0.5
The central role of algorithms in computer science



Morten Hansen
moh@sdu.dk

Hvilke algoritmer har vi snakket om i løbet af kurset?

Morten Hansen
moh@sdu.dk

Hvilke algoritmer har vi snakket om i løbet af kurset?

- Inden for Netværk?
  - Routing
  - Swithcing
  - Transmission af pakker
- Indenfor processorer
  - Afvikling af program i CPUen
    - fetch, decode, execute cycle
- Indenfor operativsystemer
  - schedulering

Morten Hansen
moh@sdu.dk

Hvad er en algoritme?

An algorithm is an ordered set of unambiguous, executable steps that defines a terminating process

Morten Hansen
DET TEKNISKE FAKULTET
moh@sdu.dk

Hvad er en algoritme?

*An algorithm is an **ordered** set of unambiguous, executable steps that defines a terminating process*

The steps of an algorithm can be sequenced in different ways
Linear (1, 2, 3, …)
Parallel (multiple processors)
Cause and Effect (circuits)

Morten Hansen
DET TEKNISKE FAKULTET
moh@sdu.dk

Hvad er en algoritme?

*An algorithm is an ordered set of **unambiguous**, executable steps that defines a terminating process*

- Clear in meaning. Can only be understood in one way

- Entydig
- Ikke til at misforstå

Morten Hansen
DET TEKNISKE FAKULTET
moh@sdu.dk

Hvad er en algoritme?

An algorithm is an ordered set of unambiguous,
***executable** steps*
that defines a terminating process

- Effektiv
- Kan beregnes

Morten Hansen
DET TEKNISKE FAKULTET
moh@sdu.dk

Hvad er en algoritme?

*An algorithm is an ordered set of unambiguous, executable steps that defines a **terminating process***

- A Terminating Process
  - Culminates with a result
  - Can include systems that run continuously ?
    - ec. Hospital systems
- A Non-terminating Process
  - Does not produce an answer

Morten Hansen
DET TEKNISKE FAKULTET
moh@sdu.dk

The Abstract Nature of Algorithms

- There is a difference between an algorithm and its representation.
  - Analogy:
    - Difference between a story and a book.
    - Difference between a recipe and a dish.
  - How is this reflected computer science?
    - Which hardware must run the representation
- A Program is a representation of an algorithm.
- A Process is the activity of executing an algorithm.

Morten Hansen
moh@sdu.dk

Termer brugt om blandt andet algoritmer

- **Precondition**
  - Hvad skal der til for at vi kan bruge vores algoritme?
  - ”In computer programming, a precondition is a condition or predicate that must always be true just prior to the execution of some section of code or before an operation in a formal specification.”
- **Body**
  - Selve algoritmen
- **Postcondition**
  - Hvad er sandt efter en algoritme er kørt.
  - ”In computer programming, a postcondition is a condition or predicate that must always be true just after the execution of some section of code or after an operation in a formal specification.”

Morten Hansen
moh@sdu.dk

Algoritme repræsentation

Morten Hansen
moh@sdu.dk

Informel

As long as the halt instruction has not been executed, continue to execute the following steps:

- *Fetch an instruction*
- *Decode the instruction*
- *Execute the instruction*

Morten Hansen
moh@sdu.dk

Formel (f.eks.: Python)

```
instruction = UNDEFINED
while instruction != HALT:
   instruction = fetch( PC )
   commands    = decode( instruction )
   execute( commands )
```


Morten Hansen
moh@sdu.dk

Algorithm Representation

- Is done formally with well-defined Primitives
- A collection of primitives constitutes a programming language.
- Is done informally with Pseudocode
  - Pseudocode is between natural language and a programming language.

Morten Hansen
moh@sdu.dk

Pseudokode.

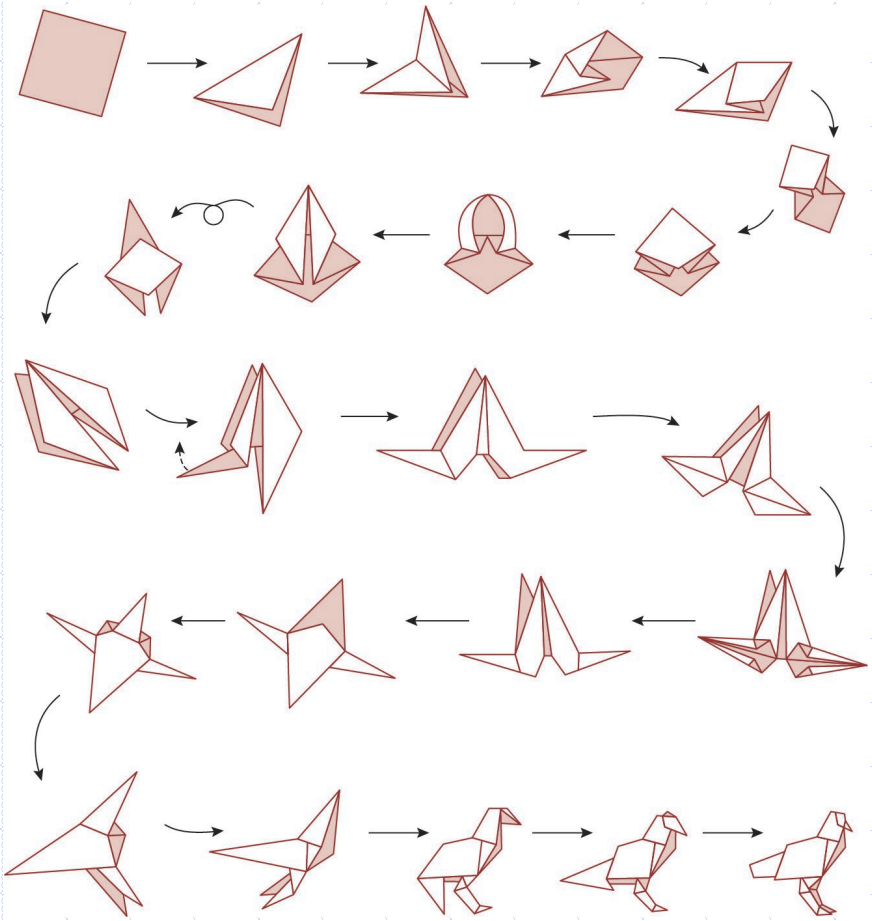
```
while not HALT:
   fetch
   decode
   execute
```


Morten Hansen
moh@sdu.dk

Figure 5.2 Folding a bird from a square piece of paper

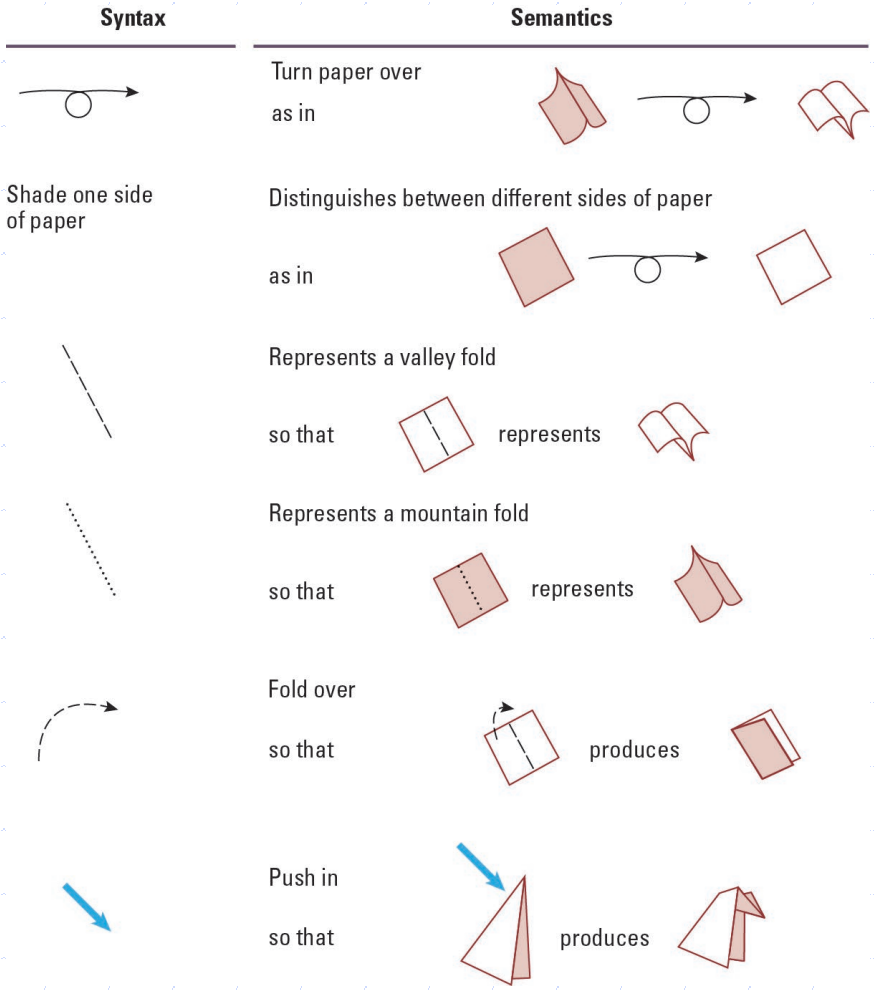
En opskrift over hvordan man laver en origami fugl.

Dog meget svær at udføre før vi har defineret primitivere.



Morten Hansen
moh@sdu.dk

Figure 5.3 Origami primitives



Morten Hansen
moh@sdu.dk

Pseudocode

- Findes mange notationer af Pseudocode
  - Typisk bundet op ad et programmerings sprog
- Løsere defineret end sproget
  - Tillader i nogen grad brug af naturligt sprog
- Bruger kortfattede notation
- Vi bruger Python baseret Pseudocode

Morten Hansen
moh@sdu.dk

Pseudocode Primitives

- Assignment

```
name = expression
```

- example

```
RemainingFunds = CheckingBalance +
                 SavingsBalance
```


Morten Hansen
moh@sdu.dk

Pseudocode Primitives (continued)

- Conditional selection

```
if (condition):
    activity
```

- example

```
if (sales have decreased):
    lower the price by 5%
```


Morten Hansen
moh@sdu.dk

Pseudocode Primitives (continued)

- Conditional selection

```
if (condition):
    activity
else:
    activity
```

- example

```
if (year is leap year):
    daily total = total / 366
else:
    daily total = total / 365
```


Morten Hansen
moh@sdu.dk

Pseudocode Primitives (continued)

- Repeated execution

```
while (condition):
    body
```

- example

```
while (tickets remain to be sold):
    sell a ticket
```


Morten Hansen
moh@sdu.dk

Pseudocode Primitives (continued)

- Indentation shows **nested** conditions

```
if (not raining):
    if (temperature == hot):
        go swimming
    else:
        play golf
else:
    watch television
```


Morten Hansen
moh@sdu.dk

Pseudocode Primitives (continued)

- Define a function

```
def name():
```

- example

```
def ProcessLoan():
```

- Executing a function

```
if (. . .):
    ProcessLoan()
else:
    RejectApplication()
```


Morten Hansen
moh@sdu.dk

Pseudocode Primitives (continued)

- Using parameters

```
def Sort(List):
    .
    .
```

- Executing Sort on different lists

```
Sort(the membership list)
Sort(the wedding guest list)
```


Morten Hansen
moh@sdu.dk

Figure 5.4 The procedure Greetings in pseudocode

```
def Greetings():
    Count = 3
    while (Count > 0):
        print('Hello')
        Count = Count - 1
```


Morten Hansen
moh@sdu.dk

Hvordan udvikles en algoritme?

Består af to aktiviteter:

- Opdage algoritmen
- Repræsenterer algoritmen

Det er en kreativ proces, og ikke en trivielt opgave.

- Test, test, test!!!

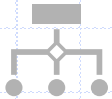
30

Morten Hansen
moh@sdu.dk

Polya's Problem Solving Steps – Programing development

Phase 1:
Understand the problem

Phase 2:
Get an idea of how an algorithmic function might solve the problem



Phase 3:
Formulate the algorithm and represent it as a program

Phase 4:
Evaluate the program for accuracy and for its potential as a tool for solving other problems

31

Morten Hansen
moh@sdu.dk

Ages of Children Problem

Person A is charged with the task of determining the ages of B's three children.

- B tells A that the **product** of the children's ages is **36**.
  - see (a) for triples.
- A replies that another clue is required.
- B tells A the **sum** of the children's ages.

- A replies that another clue is needed.
- B tells A that the **oldest** child plays the piano.
- A tells B the ages of the three children.

How old are the three children?

a. Triples whose product is 36

(1,1,36)	(1,6,6)
(1,2,18)	(2,2,9)
(1,3,12)	(2,3,6)
(1,4,9)	(3,3,4)

b. Sums of triples from part (a)

1 + 1 + 36 = 38	1 + 6 + 6 = 13
1 + 2 + 18 = 21	2 + 2 + 9 = 13
1 + 3 + 12 = 16	2 + 3 + 6 = 11
1 + 4 + 9 = 14	3 + 3 + 4 = 10

Morten Hansen
moh@sdu.dk

Tekniker til at løse et algoritme problem

Getting a foot in the Door

- Første del til at løse problemet.

Top-down methodology

- Fra det generelle problem til det specifikke.

Bottom-up methodology

- Fra det specifikke problem til det generelle.

33

Morten Hansen
moh@sdu.dk

Insertion Sort

Morten Hansen
moh@sdu.dk

Sortering

- Antag vi har en liste af ikke sorterede tal, kaldet L.
  - E.g. L = [6,5,1,3,5]
- Hvordan kan vi via en algoritme sorterer L?

Morten Hansen
moh@sdu.dk

Sortering (En generel algoritme)

```
6,5,1,3,5                  -> 1,3,5,5,6
1,2                        -> 1,2
3,0,1,8,7,2,5,4,9,6 -> 0,1,2,3,4,5,6,7,8,9
7                          -> 7
```

Diskuter med sidemanden i 5 min.
Hvordan kan vi få foden i døren?

Morten Hansen
moh@sdu.dk

Sortering

- Antag vi har en liste af ikke sorterede tal, kaldet L.
  - E.g. L = [6,5,1,3,5]
- Hvordan kan vi via en algoritme sorterer L?
- Getting a foot in the Door:
  - Sammenlign to tal af gangen og sorter.
    - L = [5,6,1,3,5]

Hvordan kommer vi videre?

Morten Hansen
moh@sdu.dk

Sortering

- Antag vi har en liste af ikke sorterede tal, kaldet L.
  - E.g. L = [6,5,1,3,5]
- Hvordan kan vi via en algoritme sorterer L?
- Getting a foot in the Door:
  - Sammenlign to tal af gangen og sorter.
    - L = [5,6,1,3,5]

Hvordan kommer vi videre?

- Vi kunne gøre brug af en løkke og et index tal til at holde styr på hvor langt vi er kommet.
- Bliv ved med at flytte det aktive tal tættere på starten indtil den næste plads har mindre værdi. Herved bliver tallene sorteret.

Morten Hansen
moh@sdu.dk

Insertion Sort

- Simpel sorterings algoritme at forstå og implementere.

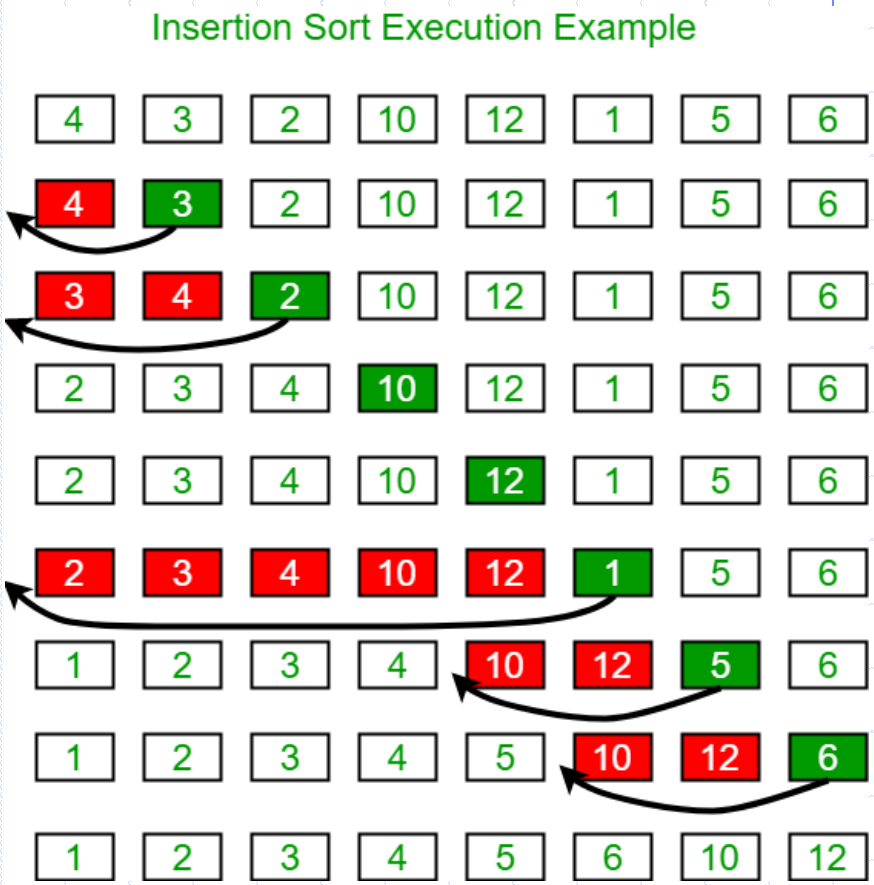
- Metoden itererer igennem samtlige elementer.
- Index: **i** sørger vi for at alle elementer op til det aktuelle er sorterede:
- Husk værdien af det **i**'te element
- Flyt alle elementer imellem den korrekte position for det **i**'te element og position **i** én position frem.
- Gem den oprindelige værdi af det **i**'teelement i den korrekte position

Morten Hansen
moh@sdu.dk

Insertion Sort

Index: i sørger vi for at alle elementer op til det aktuelle er sorterede:

1. Husk værdien af det i'teelement
2. Flyt alle elementer imellem den korrekte position for det i'te element og position i én position frem.
3. Gem den oprindelige værdi af det i'teelement i den korrekte position



40

Morten Hansen
moh@sdu.dk

Insertion Sort pseudocode

- **Precondition**
  - A skal være en liste med en orden.
    - E.g. linked list eller array
- **Body**
  - Selve methoden.
- **Postcondition**
  - Talrækken er sorteret.

41

https://varunsweblog.appspot.com/posts/insertion-sort-in-python

Morten Hansen
DET TEKNISKE FAKULTET
moh@sdu.dk

Array

Index	0	1	2	3	4	5	6	7	8
Value	x	x	x	x	x	x	x	x	x

f.eks.:

Index	0	1	2	3	4	5	6	7	8
Value	4	1	13	55	14	201	44	9	9

Morten Hansen
DET TEKNISKE FAKULTET
moh@sdu.dk

Insertion Sort pseudocode

```
def InsertionSort(myList):

while( index < length of list ):
  # sort indexed value into already sorted list.
  while( NOT the first value AND indexed value GREATER THAN Value to the left )
    Exchange indexed value with the value to the left
```


Morten Hansen
moh@sdu.dk

Insertion Sort implemenation

```
def InsertionSort(myList):
  n = len(myList) #find length af array
  i = 0
  while( i < n ):
    #store the value which need to be sorted
    tempValue = myList[i]
    j = i #used as temp index
    while( j > 0 and ( myList[j-1] > tempValue )):
      myList[j] = myList[j-1] #overwrite value
      j = j - 1 #update tempindex
      myList[j] = tempValue #overwrite value
    i = i + 1
```


Morten Hansen
moh@sdu.dk

Insertion Sort
Ungarnsk dans

https://www.youtube.com/watch?v=ROalU379l3U

Morten Hansen
moh@sdu.dk

Spørgsmål?

Morten Hansen
moh@sdu.dk