Velkommen til DMA 2021!

Plan for forelæsningen i dag:

- Plan for ugen.
- Introduktion af MA.
- Motivation og baggrund for algoritmik.
- Pseudokode: hvad og hvorfor?

Velkommen til DMA 2021!

Plan for forelæsningen i dag:

- Plan for ugen.
- Introduktion af MA.
- Motivation og baggrund for algoritmik.
- Pseudokode: hvad og hvorfor?

Plan for resten af dagen:

- Kl. 10: Instruktorer følger jer over til øvelsestimerne.
- Kl. 10–12: Øvelsestimer
- Kl. 12: Svare på spørgeskema i Store UP 1 til inddeling i studiegrupper.
- Derefter: Selvstudium.

Plan for ugen

- Mandag: Generel introduktion, pseudokode.
- Tirsdag: Formelle rammer for DMA, RAM-modellen, arrays, køretidsanalyse.
- Fredag: Rekursive algoritmer, mere køretidsanalyse.

Kort introduktion

Mikkel Abrahamsen, adjunkt, DIKU

- BSc i mat.
- MSc og PhD på DIKU

Kort introduktion

Mikkel Abrahamsen, adjunkt, DIKU

- BSc i mat.
- MSc og PhD på DIKU

Kort introduktion

Mikkel Abrahamsen, adjunkt, DIKU

- BSc i mat.
- MSc og PhD på DIKU

Kort introduktion

Mikkel Abrahamsen, adjunkt, DIKU

- BSc i mat.
- MSc og PhD på DIKU

- Forskning: algoritmisk geometri
- Anvendelser inden for computergrafik, computer-aided design/manufacturing, geografiske informationssystemer, mv.

Kort introduktion

Mikkel Abrahamsen, adjunkt, DIKU

- BSc i mat.
- MSc og PhD på DIKU

- Forskning: algoritmisk geometri
- Anvendelser inden for computergrafik, computer-aided design/manufacturing, geografiske informationssystemer, mv.

Mest almindelige forsker- og underviserstillinger på universiteter:

PhD-studerende $\longrightarrow$ Postdoc $\longrightarrow$ Adjunkt $\longrightarrow$ Lektor $\longrightarrow$ Professor

Pakningsproblemer: Eksempel på algoritmisk geometri

Pakningsproblemer: Eksempel på algoritmisk geometri

Pakningsproblemer: Eksempel på algoritmisk geometri

Pakningsproblemer: Eksempel på algoritmisk geometri

Pakningsproblemer: Eksempel på algoritmisk geometri

Pakningsproblemer: Eksempel på algoritmisk geometri

Pakningsproblemer: Eksempel på algoritmisk geometri

Brug for algoritmikere

Danmark vil mangle 22.000 it-professionelle i 2030

Manglen på it-folk fortsætter så langt øjet rækker, og vil være næsten tre gange så høj om ti år

Ny FN-måling: Danmark er fortsat verdensmestre i offentlig digitalisering

12-07-2020 | Strategier

FN har netop offentliggjort E-Government Survey 2020, og Danmark ligger for anden gang i træk på en flot første plads. I 2018 rykkede Danmark fra en tidligere 9. plads til første pladsen.

DIKU i verdensklasse!

CSRankings: Computer Science Rankings

CSRankings is a metrics-based ranking of top computer science institutions around the world. **Click on a triangle** (▶) to expand areas or institutions. **Click on a name** to go to a faculty member's home page. **Click on a pie** (the after a name or institution) to see their publication profile as a pie chart. **Click on a Google Scholar icon** ( ) to see publications, and **click on the DBLP logo** ( ) to go to a DBLP entry. ***Consider sponsoring CSrankings.***

Applying to grad school? Read this first.

Rank institutions in the world by publications from 2011 to 2021

All Areas [off | on]

AI [off | on]

- ▶ Artificial intelligence
- ▶ Computer vision
- ▶ Machine learning & data mining
- ▶ Natural language processing
- ▶ The Web & information retrieval

Systems [off | on]

- ▶ Computer architecture
- ▶ Computer networks
- ▶ Computer security
- ▶ Databases
- ▶ Design automation
- ▶ Embedded & real-time systems
- ▶ High-performance computing
- ▶ Mobile computing
- ▶ Measurement & perf. analysis
- ▶ Operating systems
- ▶ Programming languages
- ▶ Software engineering

Theory [off | on]

- ▶ Algorithms & complexity ☑
- ▶ Cryptography
- ▶ Logic & verification

Interdisciplinary Areas [off | on]

- ▶ Comp. bio & bioinformatics
- ▶ Computer graphics
- ▶ Economics & computation
- ▶ Human-computer interaction
- ▶ Robotics
- ▶ Visualization

#	Institution	Count	Faculty
1	▶ Carnegie Mellon University	96.6	19
2	▶ Massachusetts Institute of Technology	77.5	20
3	▶ Tel Aviv University	59.5	19
4	▶ Stanford University	47.3	11
5	▶ University of Copenhagen	46.5	8
6	▶ Univ. of Illinois at Urbana-Champaign	43.3	10
7	▶ Weizmann Institute of Science	42.5	11
8	▶ University of Washington	42.4	11
9	▶ Columbia University	40.0	10
10	▶ Univ. of California - Berkeley	38.4	17
11	▶ University of Michigan	33.8	10
12	▶ Princeton University	32.2	10
13	▶ Univ. of California - San Diego	32.1	10
14	▶ EPFL	30.1	8
15	▶ Rutgers University	29.9	9
16	▶ University of Warsaw	28.1	17
17	▶ Cornell University	27.7	14
18	▶ University of Texas at Austin	27.3	9
19	▶ Technion	26.6	21
20	▶ Georgia Institute of Technology	25.8	6
21	▶ Hebrew University of Jerusalem	25.7	15
22	▶ University of Pennsylvania	24.6	9
23	▶ Univ. of California - Los Angeles	24.4	4

Algoritmer før (moderne) computere

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

Peptide Synthesis

A Biological Solution to a Fundamental Distributed Computing Problem

Yehuda Afek,[1*] Noga Alon,[1,2*] Omer Barad,[3*] Eran Hornstein,[3] Naama Barkai,[3†] Ziv Bar-Joseph[4†]

Physarum can compute shortest paths ☆

Vincenzo Bonifaci [a,*,1], Kurt Mehlhorn [b], Girish Varma [c,1]

[a] *Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" – CNR, Rome, Italy*
[b] *MPI for Informatics, Saarbrücken, Germany*
[c] *Tata Institute of Fundamental Research, Mumbai, India*

Algoritmer før (moderne) computere

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

By Ezrdr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21954723

By Canticle at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10532728

Charles Babbage: The Analytical Engine

Algoritmer før (moderne) computere

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

By Ezrdr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21954723

By Canticle at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10532728

Charles Babbage: The Analytical Engine

$$\begin{array}{r} 19734 \\ +54283 \\ \hline \end{array}$$

Algoritmer før (moderne) computere

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

By Ezrdr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21954723

By Canticle at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10532728

Charles Babbage: The Analytical Engine

$$\begin{array}{r} 19734 \\ +54283 \\ \hline 7 \end{array}$$

Algoritmer før (moderne) computere

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

By Ezrdr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21954723

By Canticle at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10532728

Charles Babbage: The Analytical Engine

$$\begin{array}{r} {\scriptstyle 1}\phantom{{\scriptstyle 7}{\scriptstyle 3}{\scriptstyle 4}} \\ 19734 \\ +54283 \\ \hline 17 \end{array}$$

Algoritmer før (moderne) computere

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

By Ezrdr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21954723

By Canticle at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10532728

Charles Babbage: The Analytical Engine

$$\begin{array}{r} {\scriptstyle 1\ 1} \\ 19734 \\ +54283 \\ \hline 017 \end{array}$$

Algoritmer før (moderne) computere

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

By Ezrdr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21954723

By Canticle at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10532728

Charles Babbage: The Analytical Engine

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ \ \ \ \ } \\ 19734 \\ +54283 \\ \hline 4017 \end{array}$$

Algoritmer før (moderne) computere

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

By Ezrdr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21954723

By Canticle at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10532728

Charles Babbage: The Analytical Engine

$$
\begin{array}{r}
{\scriptstyle 1\ 1\ 1\ \ \ \ \ }\\
1\ 9\ 7\ 3\ 4\\
+5\ 4\ 2\ 8\ 3\\
\hline
7\ 4\ 0\ 1\ 7\\
\hline\hline
\end{array}
$$

Pseudokode: hvad og hvorfor?

- Overskuelig og præcis beskrivelse af algoritmer.
- Skrevet til mennesker, ikke til computere.
- Ligner rigtig kode, men uden besværet.
- Formål: at beskrive og formidle algoritmer mest effektivt.

Eksempler på simpel pseudokode

```
Example1(n)
    a = 5
    b = a + n
    return b
```

Eksempler på simpel pseudokode

```
Example1(n)
    a = 5
    b = a + n
    return b
```

Kald Example1(3).

Eksempler på simpel pseudokode

```
Example1(n)
    a = 5
    b = a + n
    return b
```

Kald Example1(3).

Kald Example1(7).

Eksempler på simpel pseudokode

```
Example1(n)
    a = 5
    b = a + n
    return b
```

```
Example2()
    x = 7
    x = x + 1
    y = x + 2
    return y
```

Kald Example1(3).

Kald Example1(7).

Eksempler på simpel pseudokode

```
Example1(n)
    a = 5
    b = a + n
    return b
```

Kald Example1(3).

Kald Example1(7).

```
Example2()
    x = 7
    x = x + 1
    y = x + 2
    return y
```

Kald Example2().

Eksempler på simpel pseudokode

```
Example1(n)
    a = 5
    b = a + n
    return b
```

Kald Example1(3).

Kald Example1(7).

```
Example2()
    x = 7
    x = x + 1
    y = x + 2
    return y
```

Kald Example2().

```
Example3(n)
    return Example1(2n)+Example2()
```

Eksempler på simpel pseudokode

```
Example1(n)
    a = 5
    b = a + n
    return b
```

Kald Example1(3).

Kald Example1(7).

```
Example2()
    x = 7
    x = x + 1
    y = x + 2
    return y
```

Kald Example2().

```
Example3(n)
    return Example1(2n)+Example2()
```

Kald Example3(4).

Eksempler (if-else)

```
Example1(n)
    a = 5
    if n ≥ 4
        return a
    else
        return n
```

Eksempler (if-else)

```
Example1(n)
    a = 5
    if n ≥ 4
        return a
    else
        return n
```

Kald Example1(3).

Eksempler (if-else)

```
Example1(n)
    a = 5
    if n ≥ 4
        return a
    else
        return n
```

Kald Example1(3).

Kald Example1(7).

Eksempler (if-else)

```
Example1(n)
    a = 5
    if n ≥ 4
        return a
    else
        return n
```

```
Example2()
    x = 7
    x = x + 1
    if x is even
        x = x + 1
        return x
    else
        return x
```

Kald Example1(3).

Kald Example1(7).

Eksempler (if-else)

```
Example1(n)
    a = 5
    if n ≥ 4
        return a
    else
        return n
```

Kald Example1(3).

Kald Example1(7).

```
Example2()
    x = 7
    x = x + 1
    if x is even
        x = x + 1
        return x
    else
        return x
```

Kald Example2().

Eksempler (if-else)

```
Example1(n)
    a = 5
    if n ≥ 4
        return a
    else
        return n
```

Kald Example1(3).

Kald Example1(7).

```
Example2()
    x = 7
    x = x + 1
    if x is even
        x = x + 1
        return x
    else
        return x
```

Kald Example2().

```
Example3(n)
    return Example1(2n)+Example2()
```

Eksempler (if-else)

```
Example1(n)
    a = 5
    if n ≥ 4
        return a
    else
        return n
```

Kald Example1(3).

Kald Example1(7).

```
Example2()
    x = 7
    x = x + 1
    if x is even
        x = x + 1
        return x
    else
        return x
```

Kald Example2().

```
Example3(n)
    return Example1(2n)+Example2()
```

Kald Example3(4).

Quiz

```
Quiz(n)
    a = 5
    if 2n ≤ n^2
        return a · n^2
    else
        return a^2/n
```

Quiz

```
Quiz(n)
    a = 5
    if 2n ≤ n^2
        return a · n^2
    else
        return a^2/n
```

Hvad returneres når vi kalder Quiz(3)?

(a) $25/3$

(b) 5

(c) 75

(d) 45

(e) $9/5$

socrative.com $\rightarrow$ Student login.

Room name: ABRAHAMSEN3464

Løkker

```
Loop1(n)
    for i from 1 to n
        print(i)
    print(15)
```

Løkker

```
Loop1(n)
    for i from 1 to n
        print(i)
    print(15)
```

Kald Loop1(5)

Løkker

```
Loop1(n)
    for i from 1 to n
        print(i)
    print(15)
```

Kald Loop1(5)

Output: 1 2 3 4 5 15

Løkker

```
Loop1(n)
    for i from 1 to n
        print(i)
    print(15)
```

Kald Loop1(5)

Output: 1 2 3 4 5 15

```
Loop2(n)
    i = 1
    while i ≤ n
        print(i)
        i = i + 1
    print(15)
```

Løkker

```
Loop1(n)
    for i from 1 to n
        print(i)
    print(15)
```

Kald Loop1(5)

Output: 1 2 3 4 5 15

```
Loop2(n)
    i = 1
    while i ≤ n
        print(i)
        i = i + 1
    print(15)
```

Kald Loop2(5)

Output: 1 2 3 4 5 15

Løkker

```
Loop1(n)
    for i from 1 to n
        print(i)
    print(15)
```

Kald Loop1(5)

Output: 1 2 3 4 5 15

```
Loop2(n)
    i = 1
    while i ≤ n
        print(i)
        i = i + 1
    print(15)
```

Kald Loop2(5)

Output: 1 2 3 4 5 15

```
Loop2(n):
    i ← 1;
    while (i ≤ n):
        print(i);
        i ← i + 1;
    print(15);
```

Løkker

```
Loop1(n)
    for i from 1 to n
        print(i)
    print(15)
```

Kald Loop1(5)

Output: 1 2 3 4 5 15

```
Loop2(n)
    i = 1
    while i ≤ n
        print(i)
        i = i + 1
    print(15)
```

Kald Loop2(5)

Output: 1 2 3 4 5 15

```
PrimeSum(n)
    s = 0
    for i from 1 to n
        if i is a prime number
            s = s + i
    return(s)
```

```
Loop2(n):
    i ← 1;
    while (i ≤ n):
        print(i);
        i ← i + 1;
    print(15);
```

Løkker

```
Loop1(n)
    for i from 1 to n
        print(i)
    print(15)
```

Kald Loop1(5)

Output: 1 2 3 4 5 15

```
Loop2(n)
    i = 1
    while i ≤ n
        print(i)
        i = i + 1
    print(15)
```

Kald Loop2(5)

Output: 1 2 3 4 5 15

Pas på! Hvordan udregnes det?

```
PrimeSum(n)
    s = 0
    for i from 1 to n
        if i is a prime number
            s = s + i
    return(s)
```

```
Loop2(n):
    i ← 1;
    while (i ≤ n):
        print(i);
        i ← i + 1;
    print(15);
```

Quiz

```
Loop4(n)
    s = 0
    for i from 1 to n
        s = s + i
        print(s)
```

Quiz

```
Loop4(n)
    s = 0
    for i from 1 to n
        s = s + i
        print(s)
```

Hvad udskriver funktionen når man kalder Loop4(5)?
socrative.com $\rightarrow$ Student login.
Room name: ABRAHAMSEN3464

(a) 5 10 15 20 25
(b) 1 3 6 10 15
(c) 1 2 3 4 5
(d) 0 1 3 6 10
(e) 1 1 2 3 5

Pas på!

```
E(n)
    return F(n)
```

```
F(n)
    return E(n)
```

Hvad sker der hvis vi kalder E(7)?

Pas på!

```
E(n)
    return F(n)
```

```
F(n)
    return E(n)
```

Hvad sker der hvis vi kalder E(7)?

```
L(n)
    i = 5
    while i ≠ n
        print(i)
        i = i + 1
```

Hvad sker der hvis vi kalder L(3)?

Pas på!

```
E(n)
    return F(n)
```

```
F(n)
    return E(n)
```

Hvad sker der hvis vi kalder E(7)?

```
L(n)
    i = 5
    while i ≠ n
        print(i)
        i = i + 1
```

Hvad sker der hvis vi kalder L(3)?

5 6 7 8 9 10 11 12 ...

Uendelig løkke