AALBORG UNIVERSITET

Elemental Datastructures, Heaps and Priority Queues

Algorithms and Datastructures, F25, Lecture 3

Andreas Holck Høeg-Petersen

Department of Computer Science
Aalborg University

February 18, 2025

Opdateringer

- Løsninger på exercises kommer på et eller andet tidspunkt
- Fra evaluering:
  - Grupper?
  - Andet?

Outline

Outline

Datastrukturer

Hvad og hvorfor?

En datastruktur er i bund og grund blot en struktureret samling af data.

Datastrukturer

Hvad og hvorfor?

En datastruktur er i bund og grund blot en struktureret samling af data.

- I kender allerede arrays, som er en sekvens af data-elementer af en bestemt type startende fra index 0 (eller 1, hvis man er CLRS-bogen...)

Datastrukturer

Hvad og hvorfor?

En datastruktur er i bund og grund blot en struktureret samling af data.

- I kender allerede arrays, som er en sekvens af data-elementer af en bestemt type startende fra index 0 (eller 1, hvis man er CLRS-bogen...)
- Vi benytter arrays som den fundamentale byggesten til at konstruere dynamiske mængder ('sets')

Datastrukturer

Hvad og hvorfor?

En datastruktur er i bund og grund blot en struktureret samling af data.

- I kender allerede arrays, som er en sekvens af data-elementer af en bestemt type startende fra index 0 (eller 1, hvis man er CLRS-bogen...)
- Vi benytter arrays som den fundamentale byggesten til at konstruere dynamiske mængder ('sets')
- Dynamiske mængder er noget, vi kan manipulere, f.eks. via Insert, Delete, Search eller lignende operationer

Datastrukturer

Hvad og hvorfor?

En datastruktur er i bund og grund blot en struktureret samling af data.

- I kender allerede arrays, som er en sekvens af data-elementer af en bestemt type startende fra index 0 (eller 1, hvis man er CLRS-bogen...)
- Vi benytter arrays som den fundamentale byggesten til at konstruere dynamiske mængder ('sets')
- Dynamiske mængder er noget, vi kan manipulere, f.eks. via Insert, Delete, Search eller lignende operationer
- Vi starter med at kigge på stacks og queues, som følger to forskellige principper for Insert og Delete

Stacks

Last in, first out

En helt fundamental datastruktur er stakken (en 'stack'). Den kan bedst sammenlignes med en stak tallerkener og følger LIFO-princippet: 'Last In, First Out'.

- Insertion og deletion kaldes henholdsvis Push og Pop
- Vi implementerer en stack med plads til n elementer med et array $S[1 \ldots n]$
- Vi definerer en attribut $S.top$, der peger på det index i S, hvor det seneste indsatte element befinder sig
- $S.size$ fortæller os hvor stor stacken er (dvs. n)

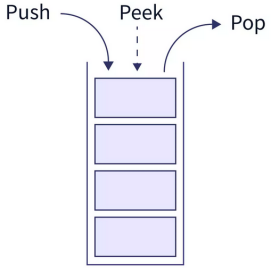


Figure: Source: `https://www.scaler.in/stack-operations/`

Stacks

Operationer

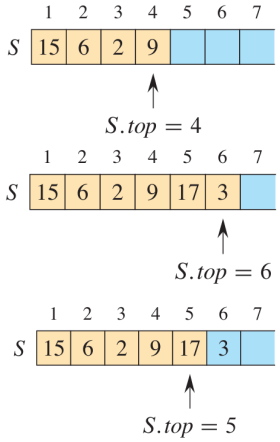
```
Stack-Empty(S)
1  if S.top == 0
2      return True
3  else return False
```

```
Push(S, x)
1  if S.top == S.size
2      error "overflow"
3  else
4      S.top = S.top + 1
5      S[S.top] = x
```

```
Pop(S)
1  if Stack-Empty(S)
2      error "underflow"
3  else
4      S.top = S.top − 1
5      return S[S.top + 1]
```

Eksempel:

- Vi har stacken S med $S.top == 4$
- Vi kalder Push$(S, 17)$ og Push$(S, 3)$
- Nu har vi $S.top == 6$
- Vi kalder Pop(S)
- Kaldet returnerer 3 og sætter $S.top = 5$
- Bemærk at elementet stadig er i arrayet!



Outline

Outline

Dagens temaer

Opsummering

- Vi har mødt vores første sorteringsalgoritme — Insertion-Sort!
  - Simpel at implementere og forstå
  - God til næsten sorterede sekvenser
  - Den asymptotiske worst case køretid er kvadratisk
- Loop invarianter og korrekthed
  - Initialization, maintenance og termination
- Asymptotisk analyse og notation
  - O, Ω, Θ

Tak for i dag!

Flere exercises..

Den bedste måde ikke at snyde sig selv på er lave exercises!