# F03 - Testning, stack och kö

## 5DV149 Datastrukturer och algoritmer
## Kapitel 7–8

Niclas Börlin
niclas.borlin@cs.umu.se

2020-01-27 Mon

# Innehåll

- Testning.
- Datatyperna *Stack* och *Kö*.
  - Specifikation och gränssnitt.
  - Konstruktion.
  - Tillämpningar.

# Testning

- Testning är jätteviktigt!
  - Mordechai (1999), The Bug That Destroyed a Rocket", *Journal of Computer Science Education*, vol. 13, no. 2, pp. 15–16.[1]
- Test går att göra under många olika faser i utvecklingen:
  - Problembeskrivning.
  - Systemdesign.
  - Mjukvarukomponenter — *enhetstester*.
  - Mjukvarulösning — *systemtest*.
  - Post-release.
- På denna kurs kommer vi ge en introduktion till testning av mindre mjukvarukomponenter — enhetstestning.

[1] https://www.youtube.com/watch?v=gp_D8r-2hwk

# Syftet med testning

- Syftet är att hitta och identifiera fel **så tidigt som möjligt** i utvecklingsprocessen.
- Mycket dyrare att åtgärda fel som vi hittar i ett senare skede.[2]

[2] https://en.wikipedia.org/wiki/Software_testing

# Utmaningar

- Testning är framför allt en utmaning av **fantasin**.
  1. A QA engineer[3] walks into a bar.
     - Orders a beer.
     - Orders 0 beers.
     - Orders 999999999 beers.
     - Orders a lizard.
     - Orders -1 beers.
     - Orders a sfdeljknesv.

     The bar's first real customer walks in.
     - Asks where the bathroom is.
     - The bar bursts into flame killing everyone.
  2. Star Trek IV: The Voyage Home
     https://www.youtube.com/watch?v=LkqiDu1BQXY
  3. Mideval Help desk
     https://www.youtube.com/watch?v=N5mLK4V5P30
  4. PBJ sandwich
     https://www.youtube.com/watch?v=cDA3_5982h8&t=7s

[3] *Quality Assurance Engineer* — ung. "testare".

# Enhetstestning

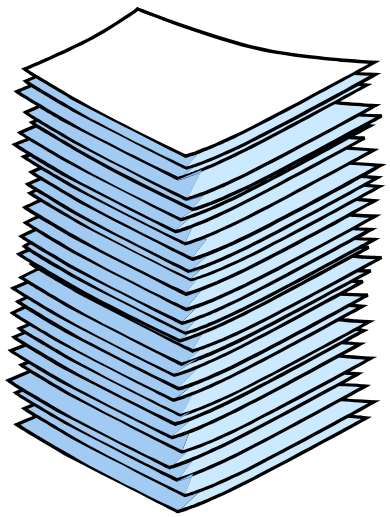
- Målet med enhetstestning är att testa en mjukvarukomponent (t.ex. en funktion eller datatyp) *isolerat* från resten av programmet.
- Man kan på så vis övertyga sig om att komponenten fungerar korrekt innan man använder sig av den i ett större program.
- Enhetstestet fungerar också som ett *kontrakt* som koden måste följa.
- Ändrar man i implementationen så måste den nya koden också klara enhetstestet.
- Görs ofta av programmeraren som skriver/ska skriva koden.
- I utvecklingsmodellen *Test-driven Development (TDD)* [4] så skrivs testet *först*, sedan koden som ska implementera ny funktionalitet.

[4] https://en.wikipedia.org/wiki/Test-driven_development

# Testning av en datatyp

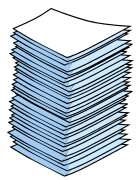
- Vad ska vi testa?
  - Börja med de enklaste testerna!
  - Testa gränsfall.
  - Fundera över vad som kan ställa till problem.
  - Viktigt att täcka in alla operationer...
  - ... och att täcka in de olika fall som finns i operationerna.
- Genom att vi noga tänker efter vilka tester vi gör så kan vi minska antalet tester.
- Testa *så lite som möjligt* i varje test — hjälper till att identifiera vad som går fel.
- Skriv varje test i en egen funktion.

# Abstrakta datatyper — *Stack* och *Kö*



# Stack och Kö

- Modell: Papperstrave.



- Exempel:
  - Web-läsare — *back*-kommandot.
  - *Undo*-kommandot i texteditorer.

- Modell: Kö.

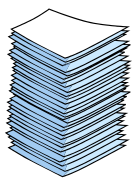
- Exempel:
  - Buffert.
  - Telefonkö.

# Stack och Kö, likheter

- Är *sammansatta* datatyper — lagrar element. Kallas ibland *container*-typer på engelska.
- Är *generiska* datatyper (polytyper):
  - Man kan definiera Stack av heltal, Kö av strängar, osv.
- Är *homogena* datatyper — alla element har samma typ.
- Kan ses som specialisering av datatypen *Lista*.
- Har elementen som är ordnade enligt en före/efter-relation.
  - Linjärt ordnade.

# Stack och Kö, skillnader, insättning, avläsning

- Stack:
  - Insättning sker i *toppen* av stacken.
  - Borttagning, avläsning sker i *toppen* av stacken.
  - Objekt som läggs i stacken tas ut enligt principen *LIFO — Last In First Out*.



- Kö
  - Insättning sker i *slutet* på kön.
  - Borttagning och avläsning sker i *början* av kön.
  - Objekt som läggs på stacken tas ut enligt principen *FIFO — First In First Out*.

# Informell specifikation till Stack

- Empty() — konstruerar en tom stack.
- Push(v,s) — lägger (ett element med värdet) v överst på stacken.
- Top(s) — returnerar värdet av det översta elementet på stacken (förutsatt att stacken inte är tom).
- Pop(s) — avlägsnar det översta elementet från stacken (förutsatt att stacken inte är tom).
- Isempty(s) — testar om stacken är tom.

# Formell definition

- En uppsättning *axiom*.
- Beskriver relationer mellan typens olika operationer.
- Axiom kan användas för att göra formella härledningar för datatypen (och tester!).

# Formell specifikation till Stack

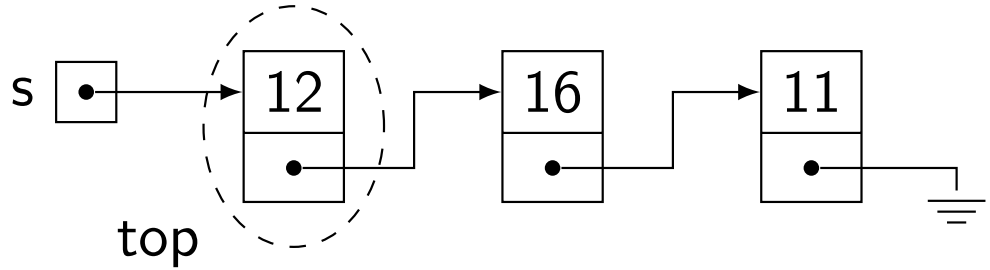
| | | |
|---|---|---|
| Ax 1 | Isempty(Empty()) | En nyskapad stack är tom. |
| Ax 2 | ¬Isempty(Push(v,s)) | En stack som man lagt ett element på är inte tom. |
| Ax 3 | Pop(Push(v,s)) = s | Om vi lägger ett värde på en stack och sen tar bort värdet så ser stacken ut som innan. |
| Ax 4 | Top(Push(v,s)) = v | Om vi lägger ett värde på stacken så ligger värdet överst på stacken. |
| Ax 5 | ¬Isempty(s) ⇒ Push(Top(s),Pop(s)) = s | *Förutsättning: Stacken är inte tom.* Om vi tar bort översta elementet och sen lägger tillbaka det så ser stacken ut som innan. |

# Gränsyta för Stack

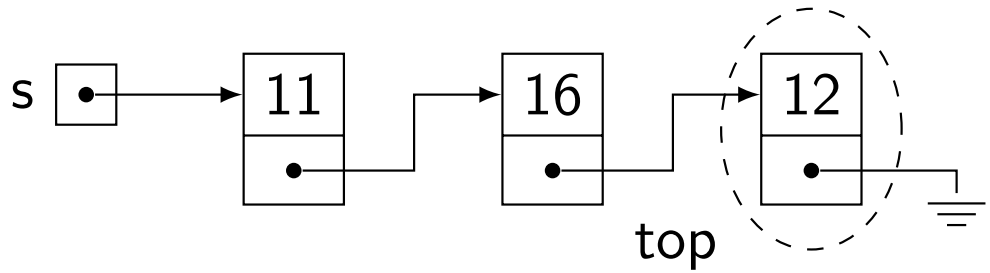
```
abstract datatype  Stack(val)
  Empty() → Stack(val)
  Push(v: val, s: Stack(val)) → Stack(val)
  Top (s: Stack(val)) → val
  Pop (s: Stack(val)) → Stack(val)
  Isempty(s: Stack(val)) → Bool
```

# Stack konstruerad som Lista

- Lista 1: Toppen av stacken = början av listan.

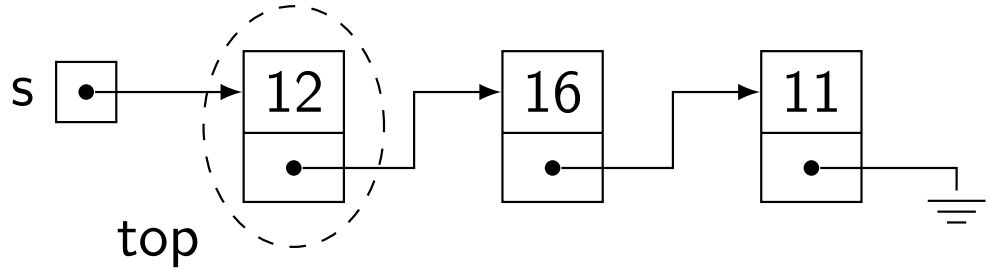



- Lista 2: Toppen av stacken = slutet av listan.

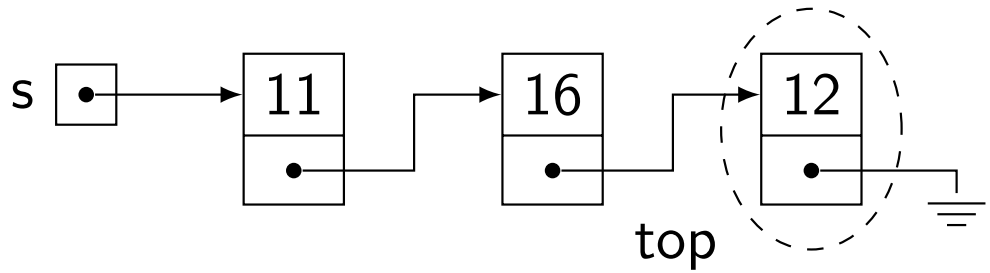


# Stack konstruerad som Lista

- Lista 1: Toppen av stacken = början av listan.



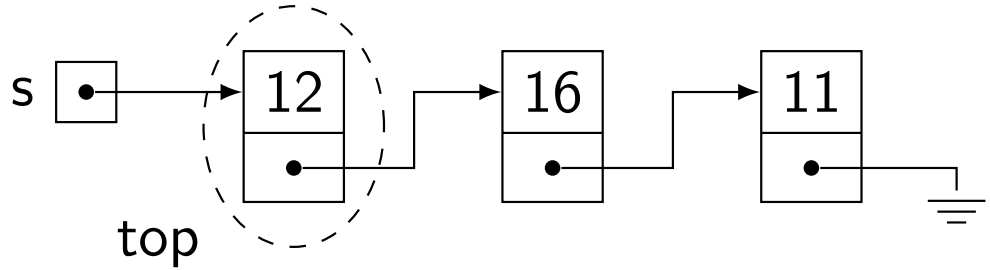

- Lista 2: Toppen av stacken = slutet av listan.



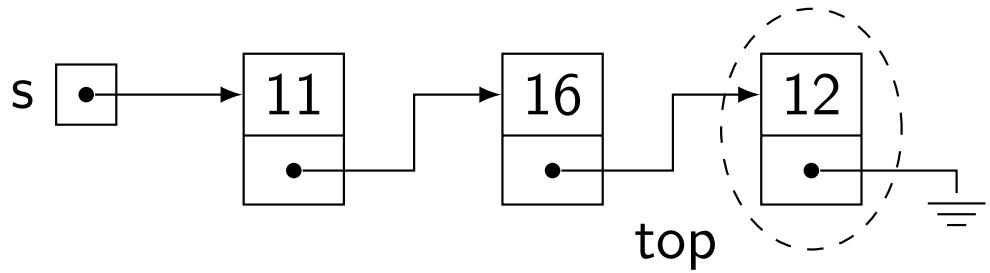

- Vad har grundoperationerna för Stack som Lista för komplexitet?

# Stack konstruerad som Lista

- Lista 1: Toppen av stacken = början av listan.



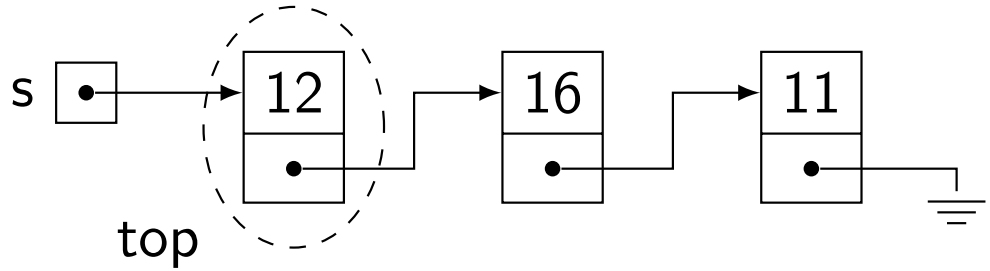

- Lista 2: Toppen av stacken = slutet av listan.



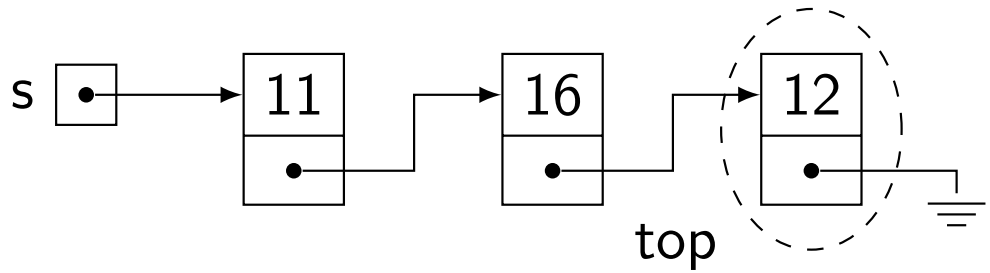

- Vad har grundoperationerna för Stack som Lista för komplexitet?
  - Förutsättningar:
    - Att skapa en lista tar $O(1)$ operationer.
    - Att testa om listan är tom tar $O(1)$ operationer.
    - Att traversera till början av listan tar $O(1)$ operationer.
    - Att traversera till slutet av listan tar $O(n)$ operationer.
    - Att läsa av ett listelement tar $O(1)$ operationer.
    - Att lägga in ett listelement tar $O(1)$ operationer.

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

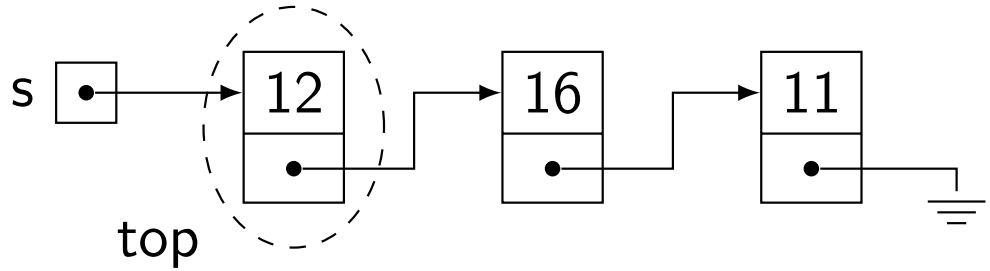



- Komplexitet för grundoperationerna:

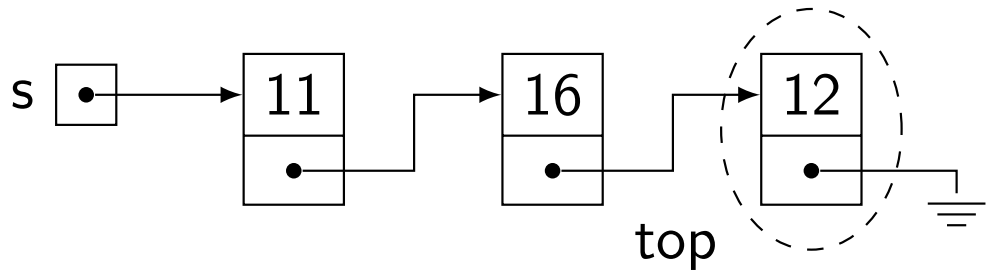
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | | |
| Push | | |
| Top | | |
| Pop | | |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

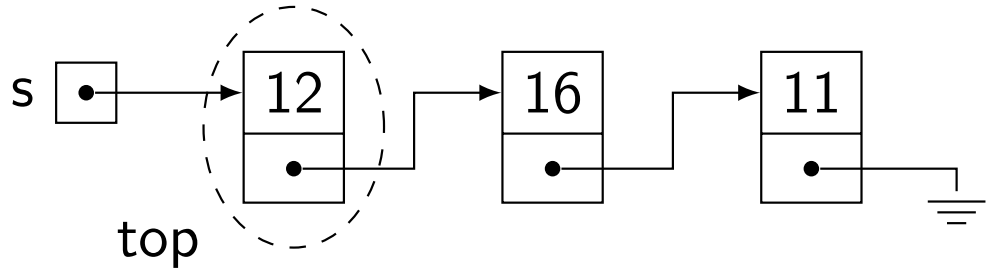



- Komplexitet för grundoperationerna:

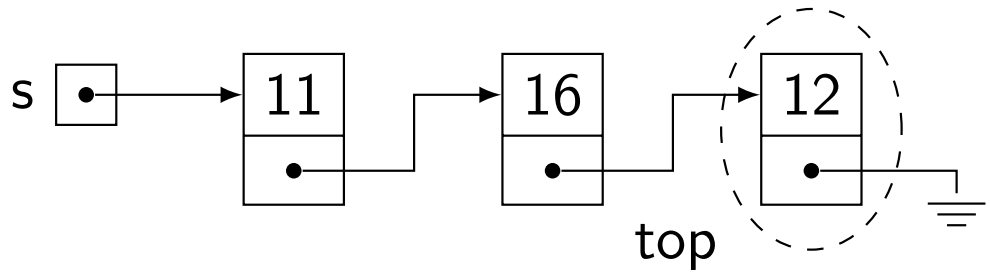
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | |
| Push | | |
| Top | | |
| Pop | | |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

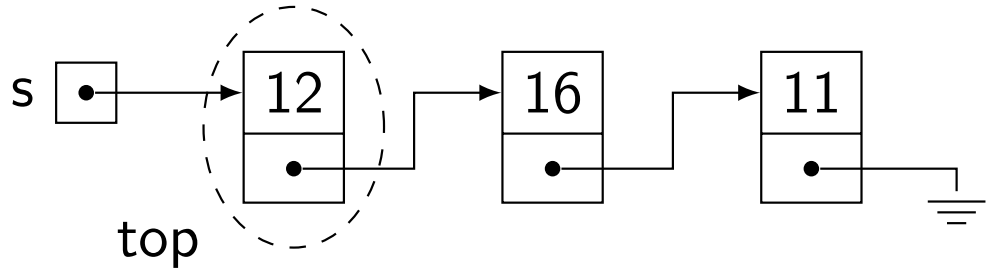



- Komplexitet för grundoperationerna:

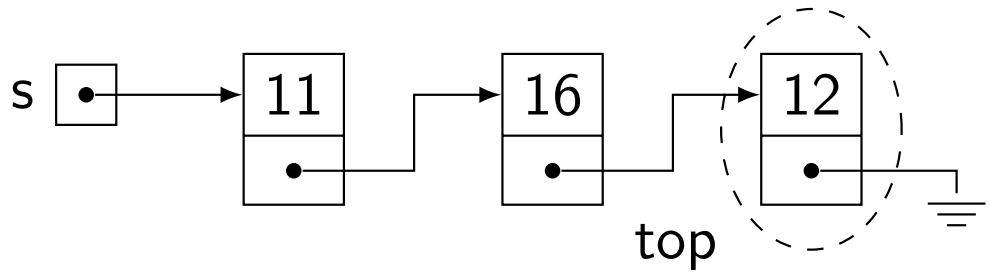
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | | |
| Top | | |
| Pop | | |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

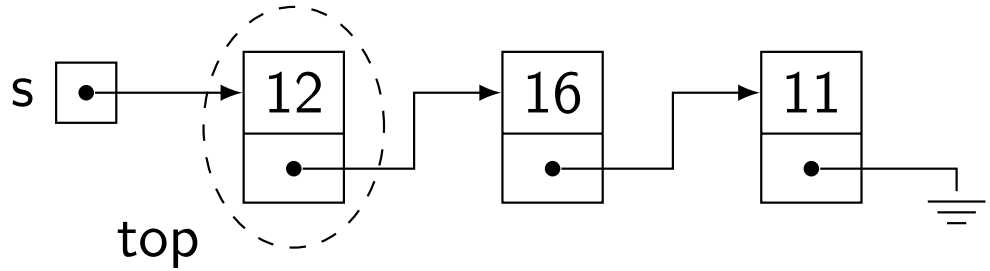



- Komplexitet för grundoperationerna:

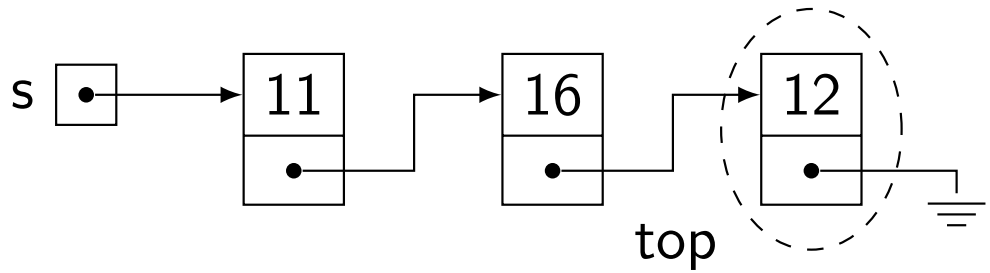
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | |
| Top | | |
| Pop | | |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

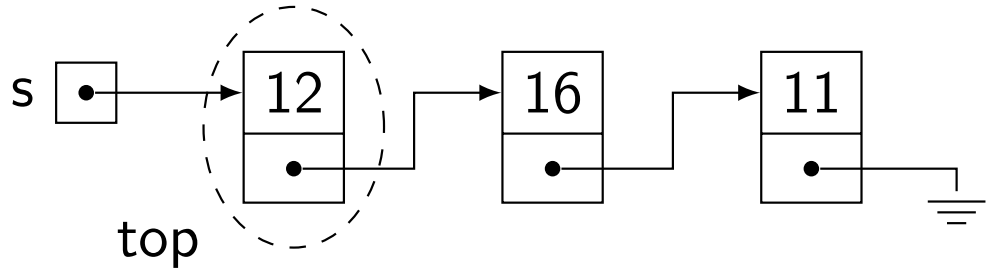



- Komplexitet för grundoperationerna:

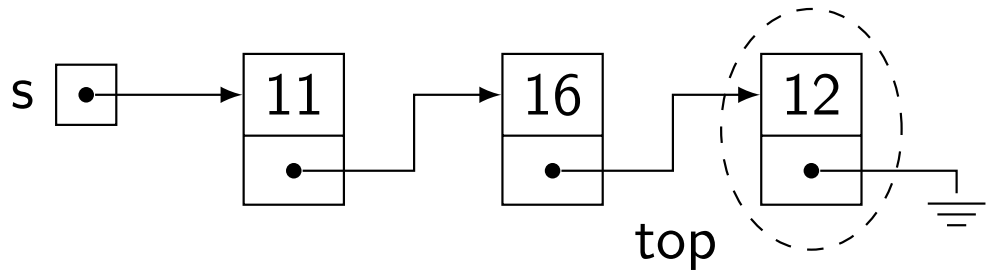
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(n)$ |
| Top | | |
| Pop | | |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

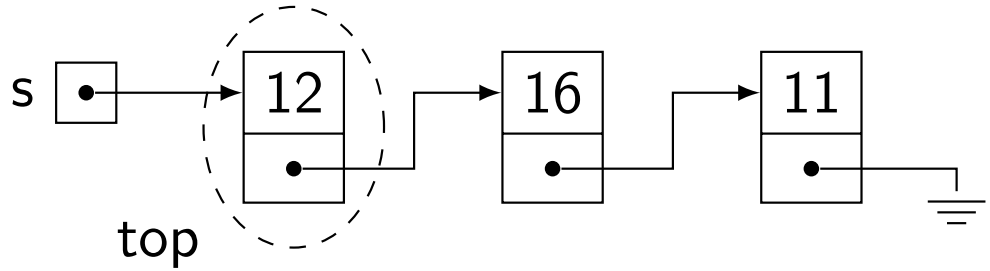



- Komplexitet för grundoperationerna:

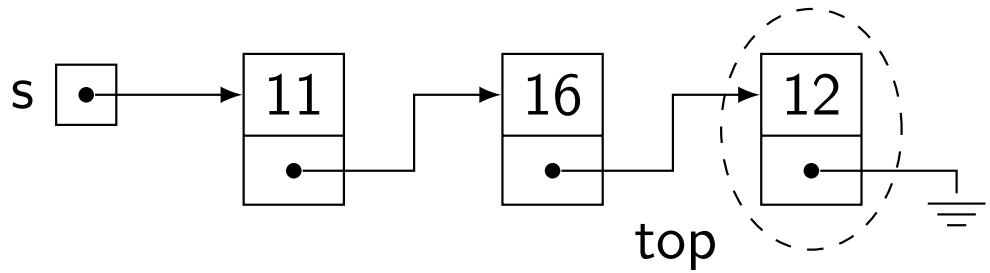
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(n)$ |
| Top | $O(1)$ | |
| Pop | | |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

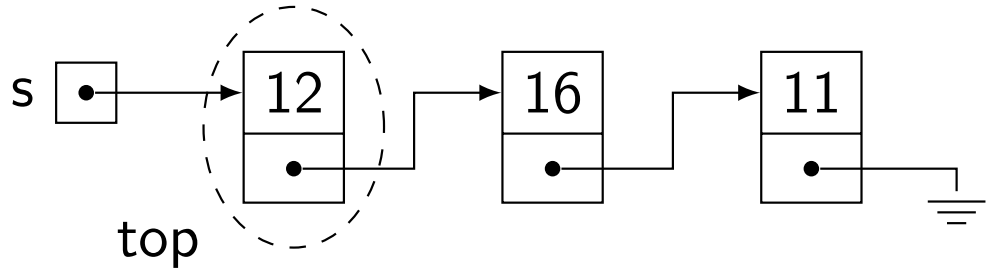



- Komplexitet för grundoperationerna:

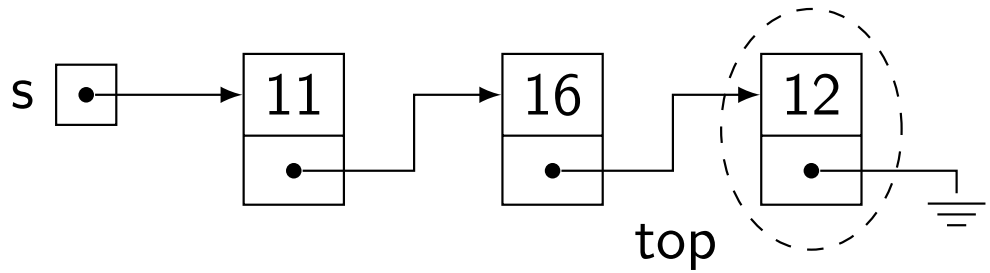
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(n)$ |
| Top | $O(1)$ | $O(n)$ |
| Pop | | |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

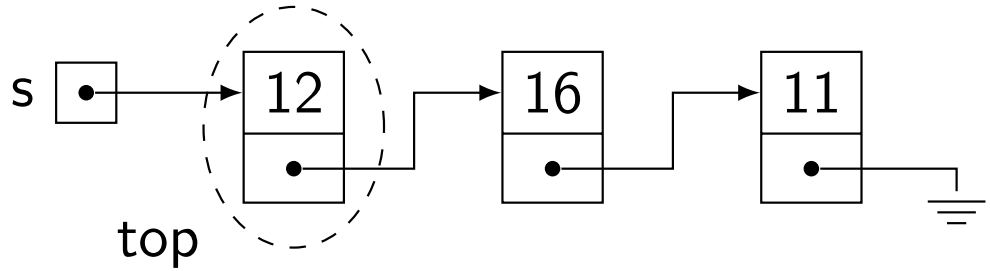



- Komplexitet för grundoperationerna:

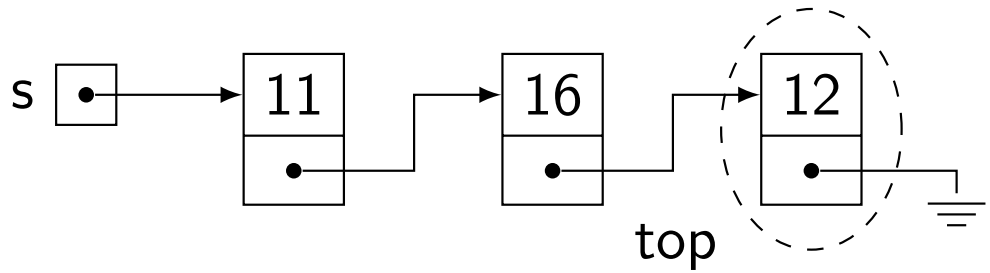
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(n)$ |
| Top | $O(1)$ | $O(n)$ |
| Pop | $O(1)$ | |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

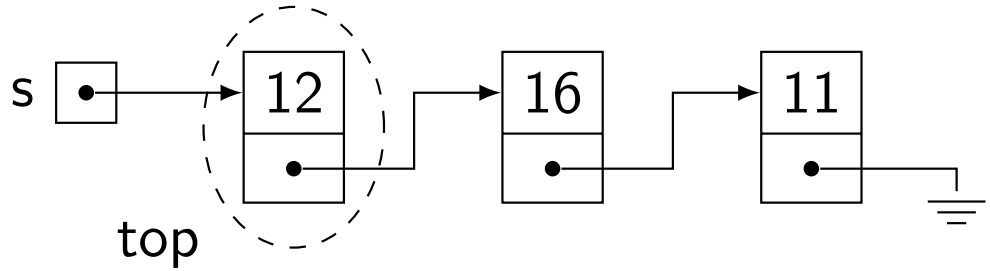



- Komplexitet för grundoperationerna:

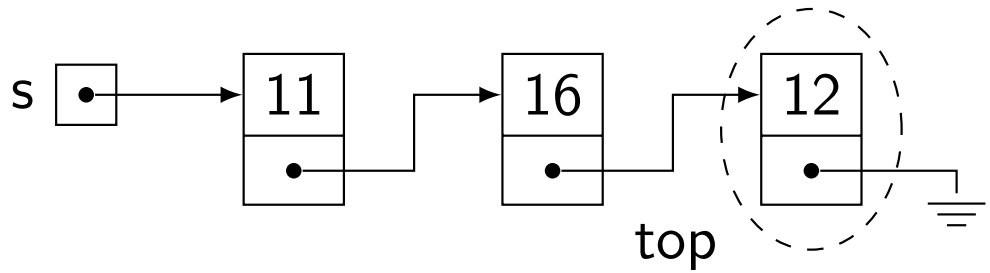
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(n)$ |
| Top | $O(1)$ | $O(n)$ |
| Pop | $O(1)$ | $O(n)$ |
| Isempty | | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

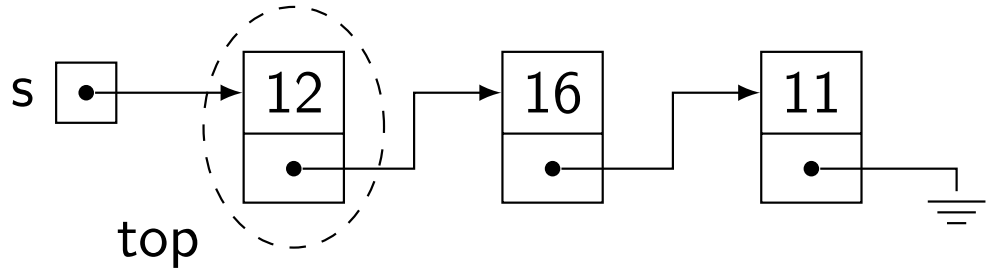



- Komplexitet för grundoperationerna:

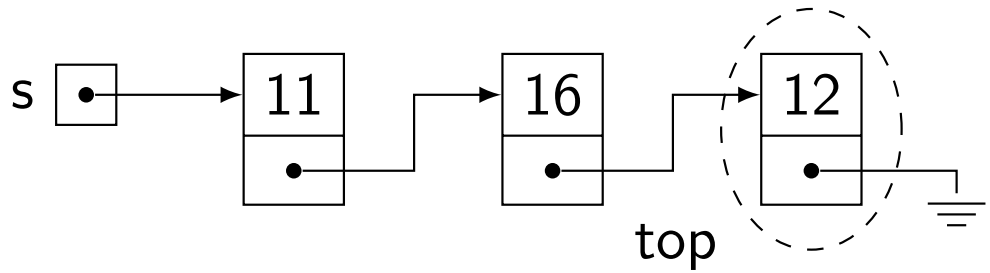
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(n)$ |
| Top | $O(1)$ | $O(n)$ |
| Pop | $O(1)$ | $O(n)$ |
| Isempty | $O(1)$ | |

# Stack konstruerad som Lista, komplexitet

- Lista 1: Toppen av stacken = början av listan.




- Lista 2: Toppen av stacken = slutet av listan.

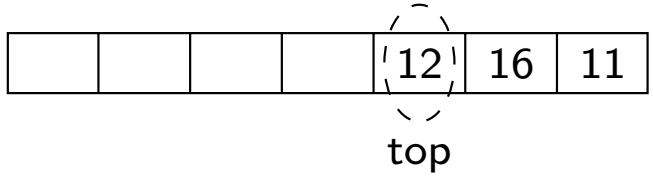



- Komplexitet för grundoperationerna:

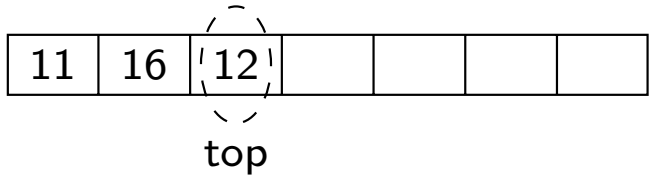
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(n)$ |
| Top | $O(1)$ | $O(n)$ |
| Pop | $O(1)$ | $O(n)$ |
| Isempty | $O(1)$ | $O(1)$ |

# Stack konstruerad som Fält

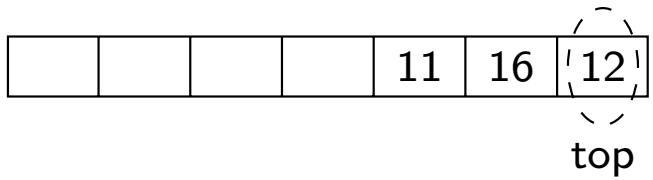
- Fält 1: Botten av stacken i slutet av fältet.



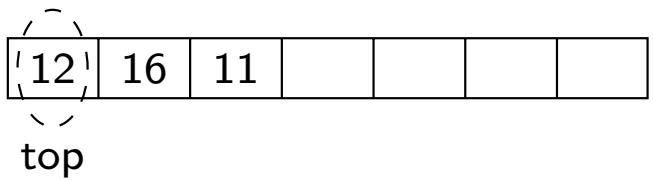

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.

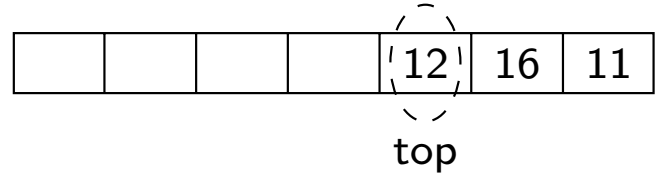



- Fält 4: Toppen av stacken i början av fältet.

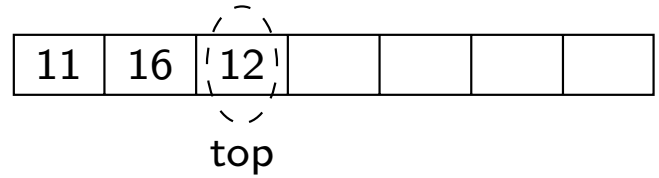


# Stack konstruerad som Fält

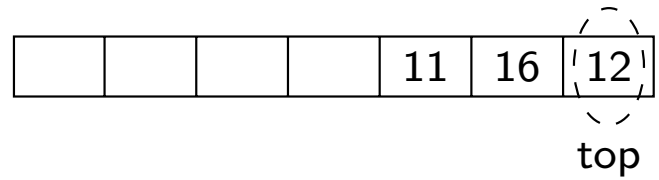
- Fält 1: Botten av stacken i slutet av fältet.



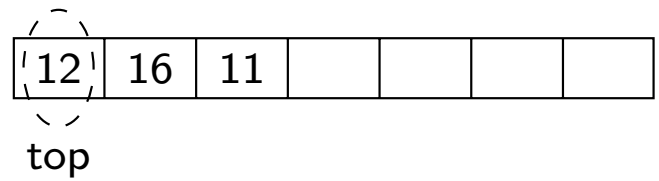

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.



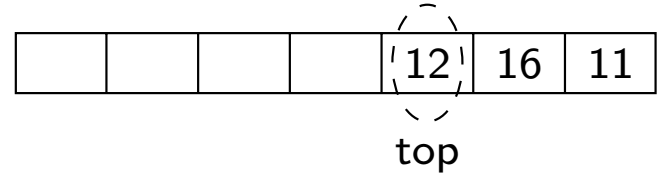

- Fält 4: Toppen av stacken i början av fältet.



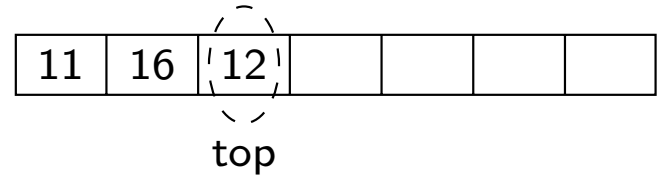

- Vad har grundoperationerna för Stack som Fält för komplexitet?

# Stack konstruerad som Fält

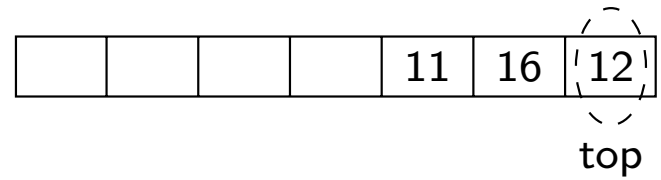
- Fält 1: Botten av stacken i slutet av fältet.



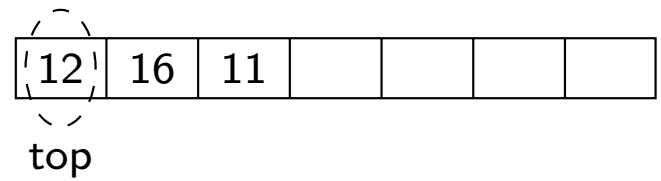

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.



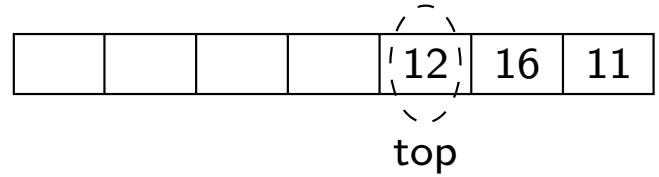

- Fält 4: Toppen av stacken i början av fältet.



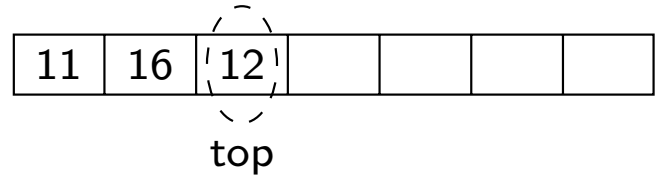

- Vad har grundoperationerna för Stack som Fält för komplexitet?
  - Förutsättningar:
    - Att skapa ett fält tar $O(1)$ operationer.
    - Att testa om fältet är tomt tar $O(1)$ operationer.
    - Att traversera till ett element tar $O(1)$ operationer.
    - Att läsa av ett element tar $O(1)$ operationer.
    - Att lägga in ett värde på en ledig plats tar $O(1)$ operationer.
    - Att flytta ett element tar $O(n)$ operationer.

# Stack konstruerad som Fält, komplexitet

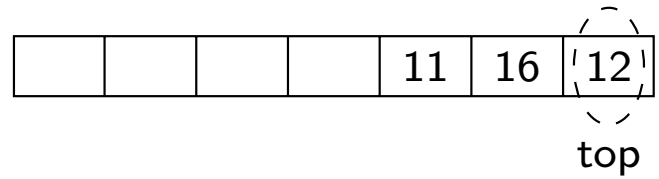
- Fält 1: Botten av stacken i slutet av fältet.



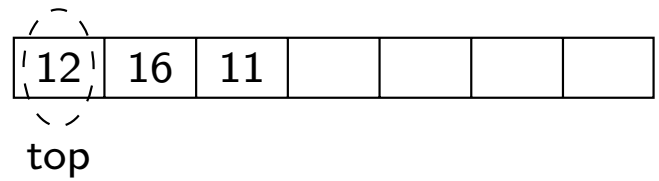

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

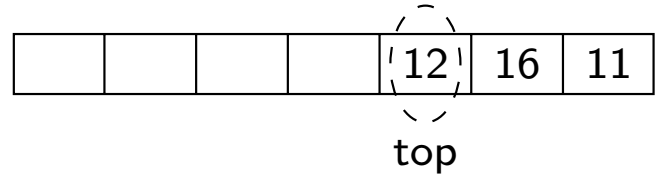



- Komplexitet för grundoperationerna:

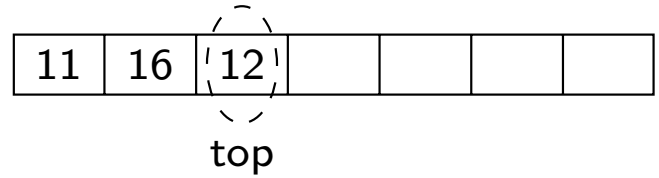
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | | | | |
| Push | | | | |
| Top | | | | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

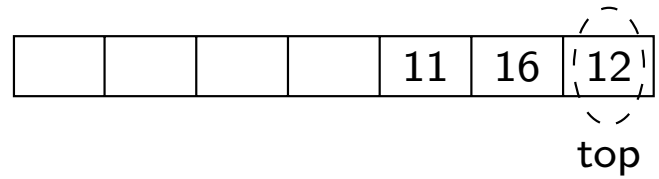
- Fält 1: Botten av stacken i slutet av fältet.



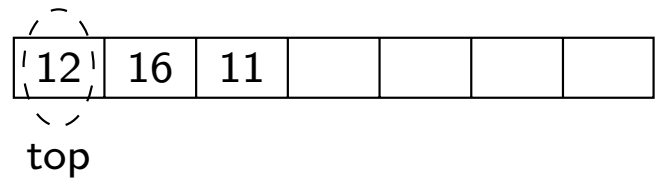

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

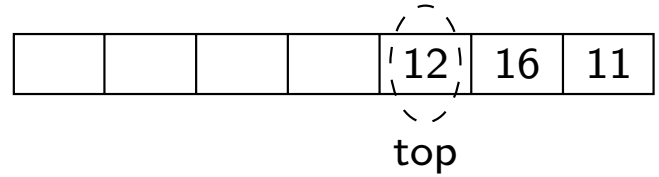



- Komplexitet för grundoperationerna:

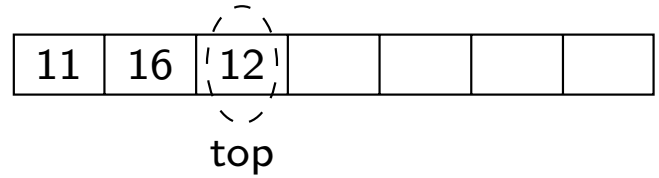
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | | | | |
| Top | | | | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

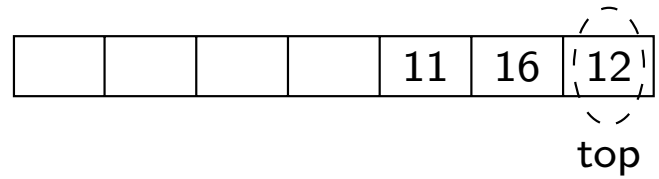
- Fält 1: Botten av stacken i slutet av fältet.



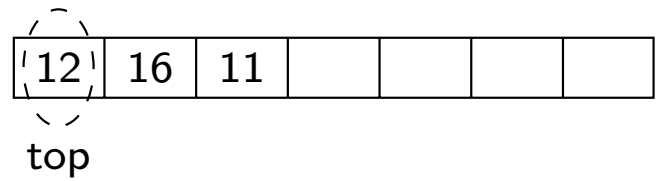

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

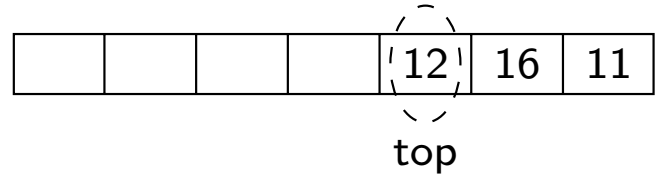



- Komplexitet för grundoperationerna:

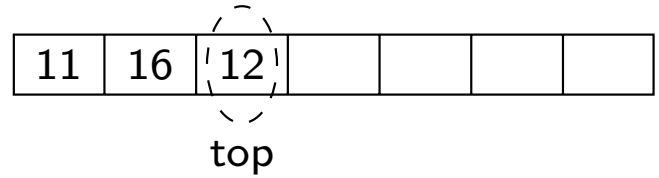
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | | | |
| Top | | | | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

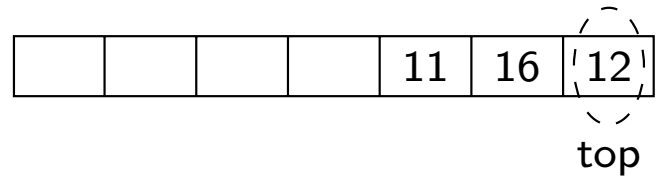
- Fält 1: Botten av stacken i slutet av fältet.



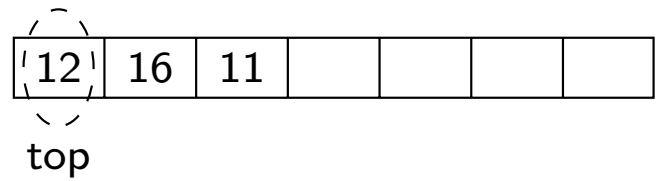

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

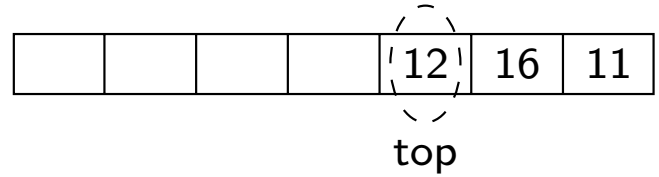



- Komplexitet för grundoperationerna:

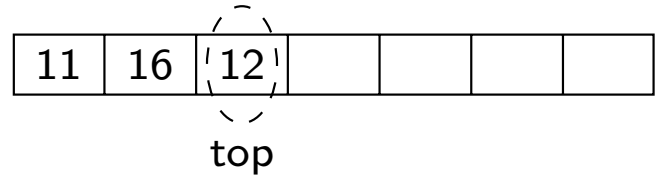
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | | |
| Top | | | | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

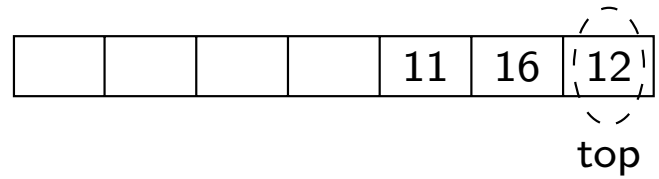
- Fält 1: Botten av stacken i slutet av fältet.



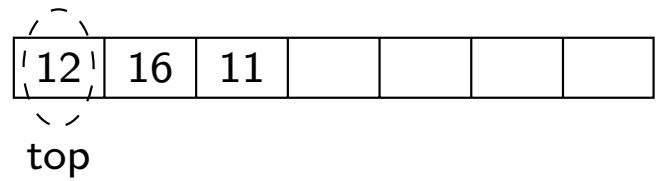

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

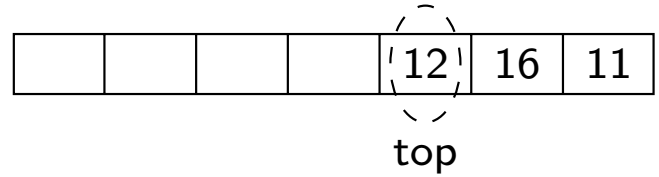



- Komplexitet för grundoperationerna:

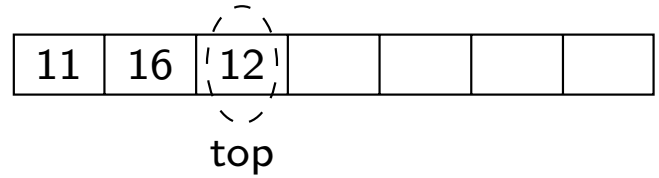
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | |
| Top | | | | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

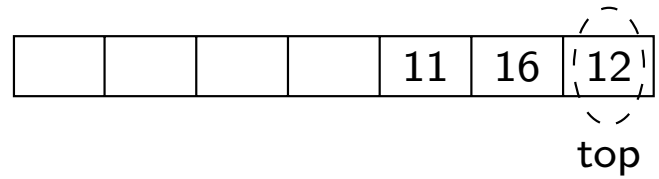
- Fält 1: Botten av stacken i slutet av fältet.



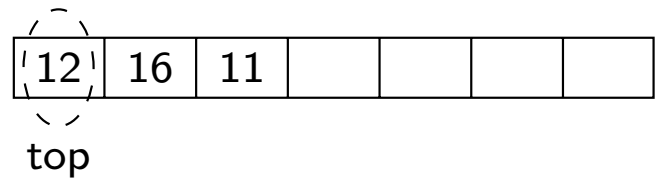

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

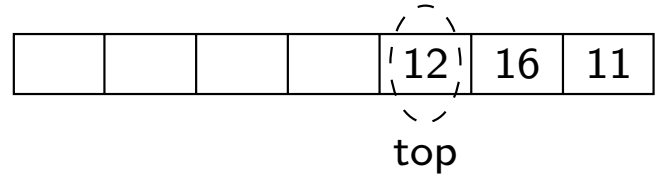



- Komplexitet för grundoperationerna:

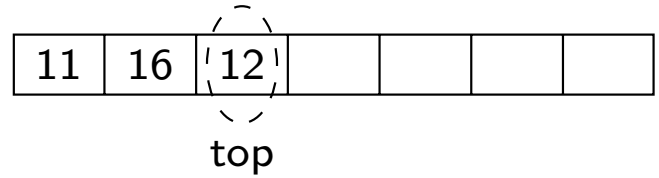
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | | | | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

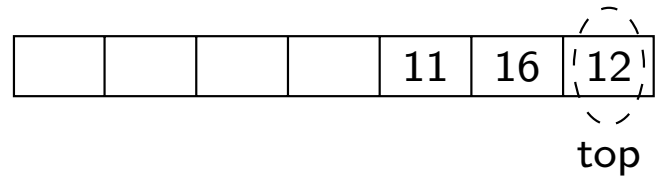
- Fält 1: Botten av stacken i slutet av fältet.



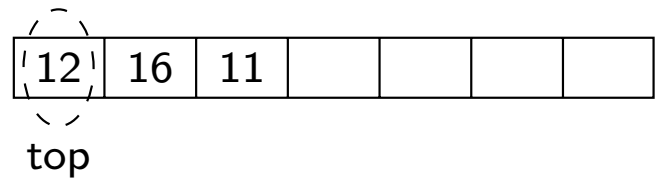

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

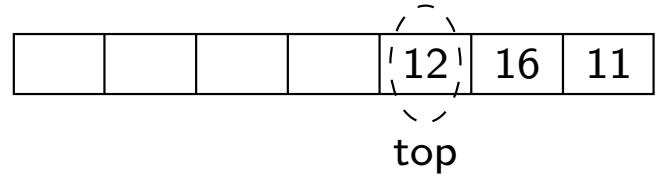



- Komplexitet för grundoperationerna:

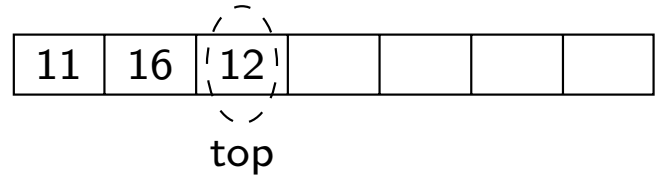
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | | | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

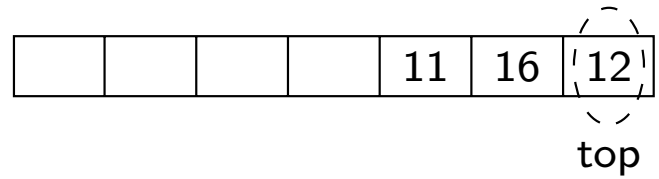
- Fält 1: Botten av stacken i slutet av fältet.



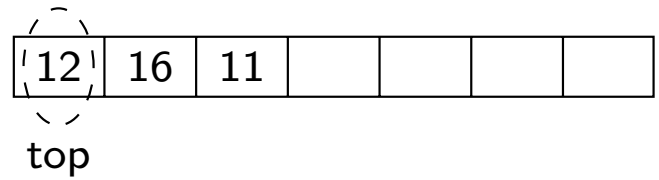

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

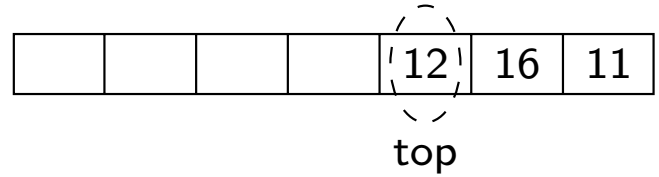



- Komplexitet för grundoperationerna:

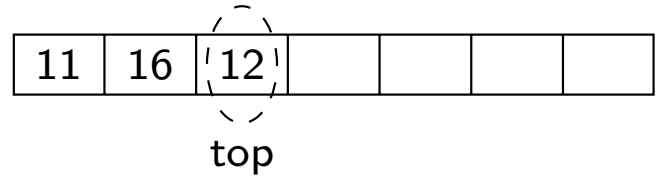
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | $O(1)$ | | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

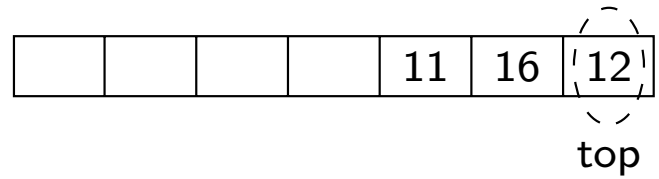
- Fält 1: Botten av stacken i slutet av fältet.



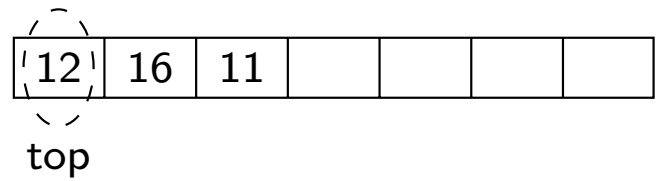

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

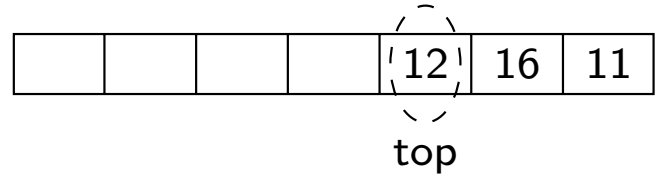



- Komplexitet för grundoperationerna:

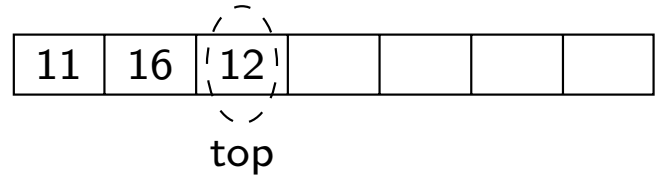
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | $O(1)$ | $O(1)$ | |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

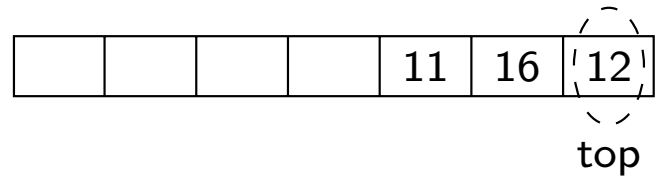
- Fält 1: Botten av stacken i slutet av fältet.



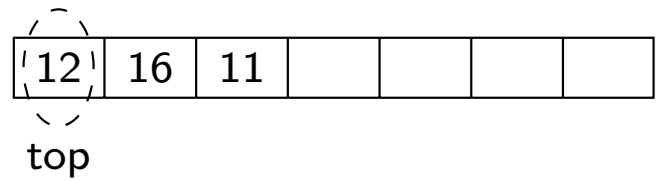

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

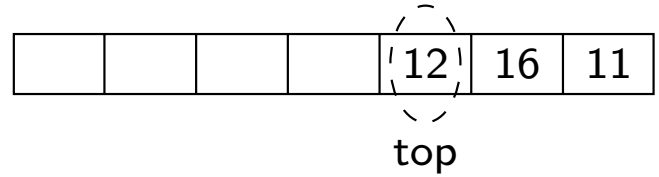



- Komplexitet för grundoperationerna:

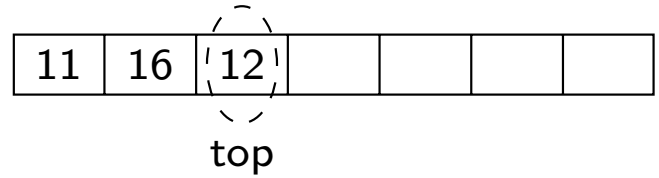
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Pop | | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

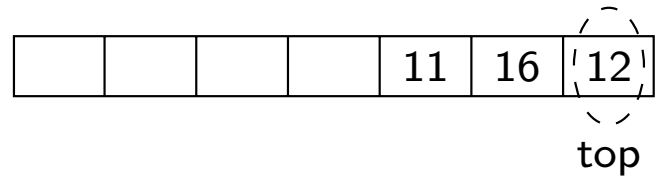
- Fält 1: Botten av stacken i slutet av fältet.



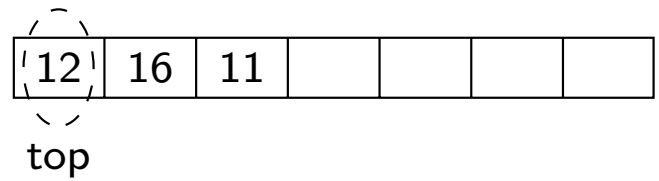

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

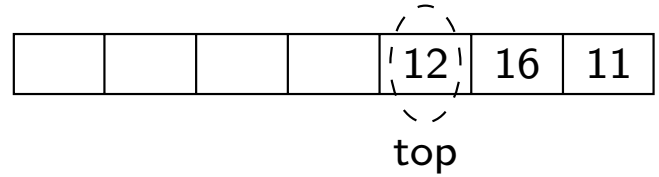



- Komplexitet för grundoperationerna:

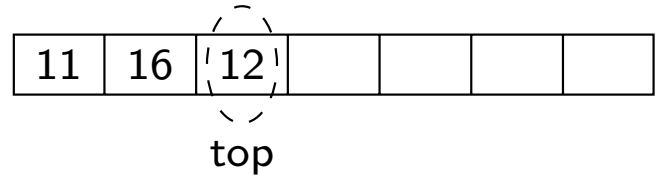
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Pop | $O(1)$ | | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

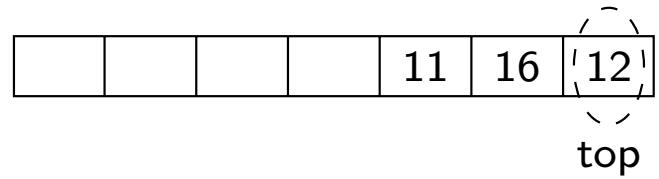
- Fält 1: Botten av stacken i slutet av fältet.



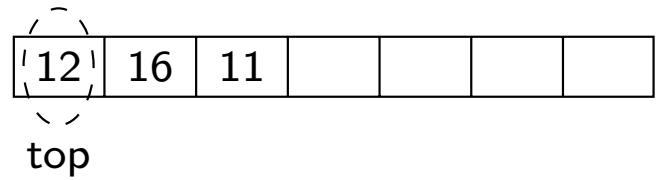

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

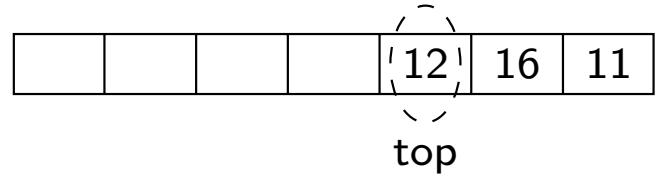



- Komplexitet för grundoperationerna:

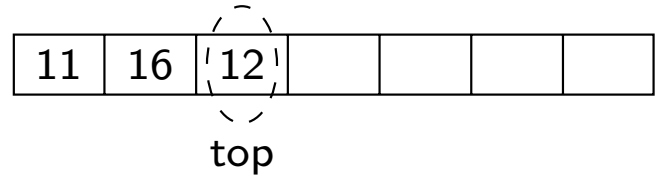
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Pop | $O(1)$ | $O(1)$ | | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

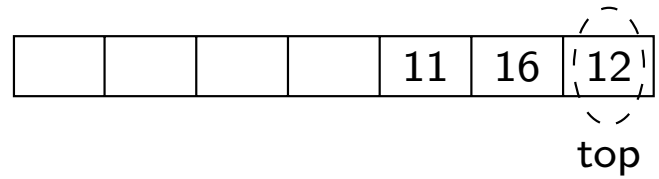
- Fält 1: Botten av stacken i slutet av fältet.



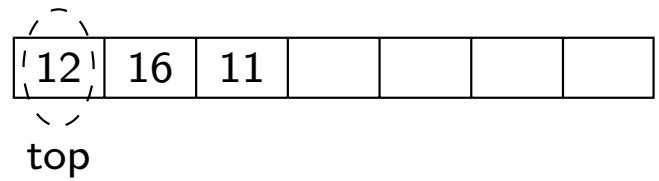

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

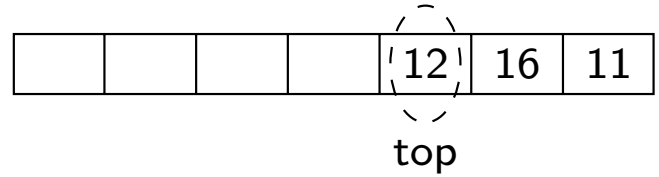



- Komplexitet för grundoperationerna:

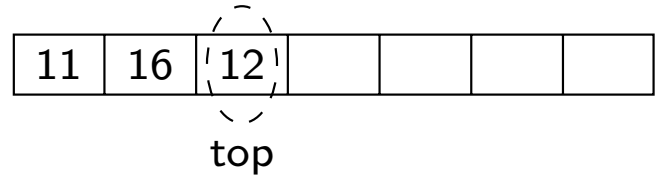
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Pop | $O(1)$ | $O(1)$ | $O(n)$ | |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

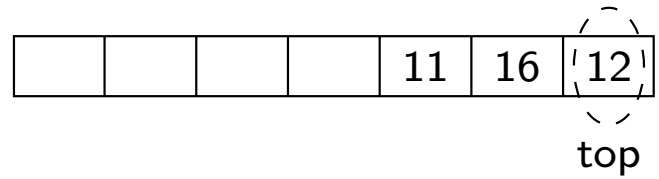
- Fält 1: Botten av stacken i slutet av fältet.



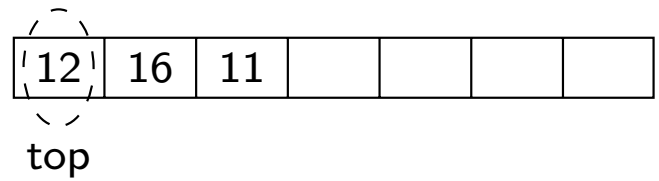

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.

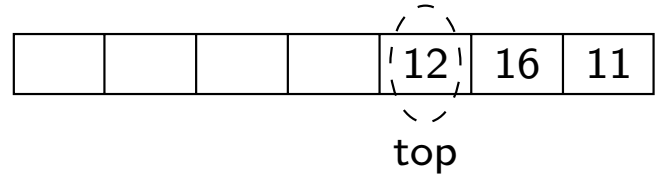



- Komplexitet för grundoperationerna:

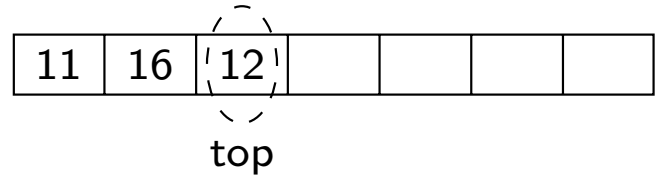
| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Pop | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Isempty | | | | |

# Stack konstruerad som Fält, komplexitet

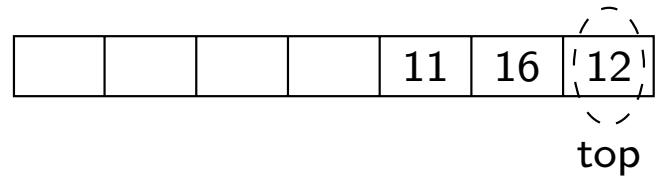
- Fält 1: Botten av stacken i slutet av fältet.



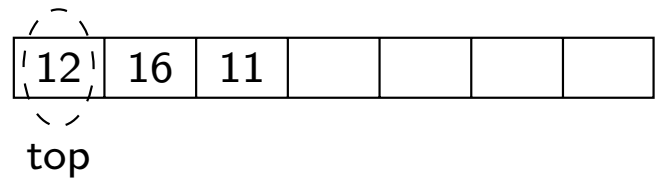

- Fält 2: Botten av stacken i början av fältet.




- Fält 3: Toppen av stacken i slutet av fältet.




- Fält 4: Toppen av stacken i början av fältet.




- Komplexitet för grundoperationerna:

| Operation | Fält 1 | Fält 2 | Fält 3 | Fält 4 |
|---|---|---|---|---|
| Empty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Push | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Top | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Pop | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ |
| Isempty | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |

# Relativ och absolut komplexitet

- Relativ komplexitet:
  - Tittar bara på "ytan", dvs. hur många list-/fält-operationer som behövs per stack-operation.
  - Ex. antalet listoperationer är inte beroende av antalet element i stacken.
- Absolut komplexitet:
  - Multiplicerar alla relativa komplexiteter ned till fysiska datatyper.
  - Dvs. tittar även på hur listan/fältet är konstruerad.

# Stack, tillämpningar

- Avbryter bearbetning som senare kanske återupptas.
- Återspårning (*backtracking*):
  - Till senaste gjorda valet.
- Traversera i andra datatyper (grafer och träd).
- Rekursion — Factorial:
  - Factorial(n)
  - if (n<=1) then return 1;
  - else return n*factorial(n-1);
- Används vid s.k. djupet-först-sökning (*depth-first search*).
- Evaluering av uttryck: `( 2 * 6 ) + ( ( 1 + 3 ) * ( 4 * ( 5 - 3 ) ) )`

# Gränsyta för Kö

```
abstract datatype  Queue(val)
  Empty() → Queue(val)
  Enqueue(v : val, q : Queue(val)) → Queue(val)
  Front (q : Queue(val)) → val
  Dequeue (q : Queue(val)) → Queue(val)
  Isempty (q : Queue(val)) → Bool
```

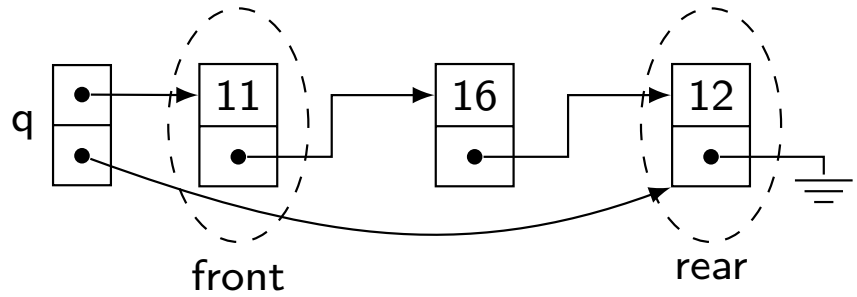
# Gränsyta för Kö och Stack

```
abstract datatype  Stack(val)
  Empty() → Stack(val)
  Push(v : val, s : Stack(val))
                        → Stack(val)
  Top (s : Stack(val)) → val
  Pop (s : Stack(val)) → Stack(val)
  Isempty(s : Stack(val)) → Bool
```

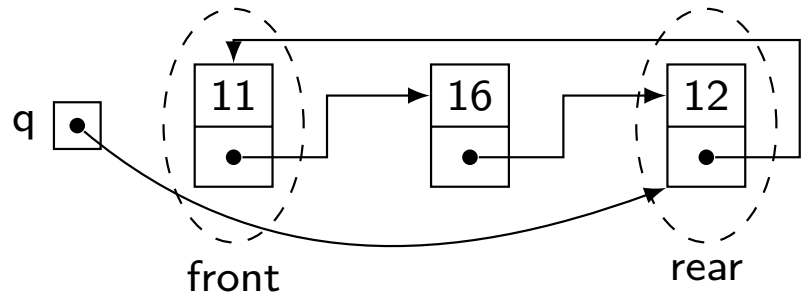
```
abstract datatype  Queue(val)
  Empty() → Queue(val)
  Enqueue(v : val, q : Queue(val))
                          → Queue(val)
  Front (q : Queue(val)) → val
  Dequeue (q : Queue(val)) → Queue(val)
  Isempty (q : Queue(val)) → Bool
```

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

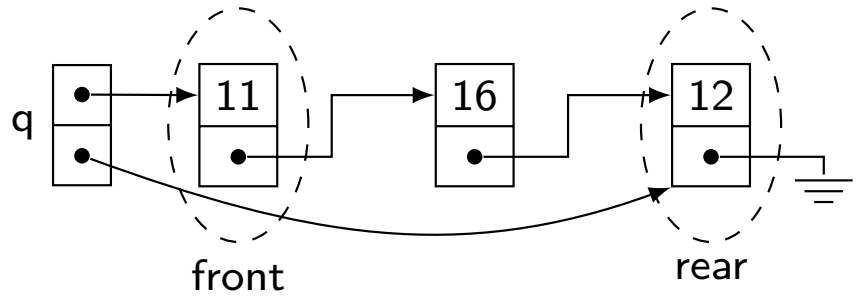



- Komplexitet för grundoperationerna:

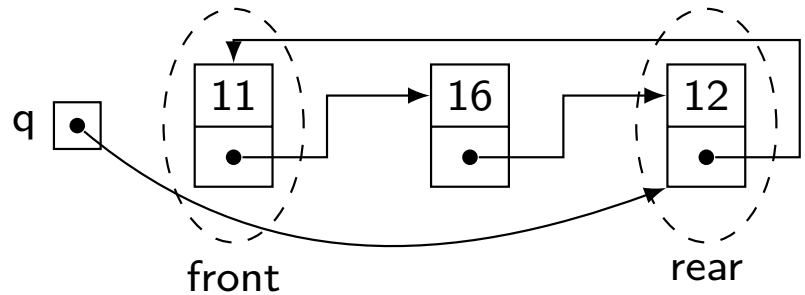
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | | |
| Enqueue | | |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

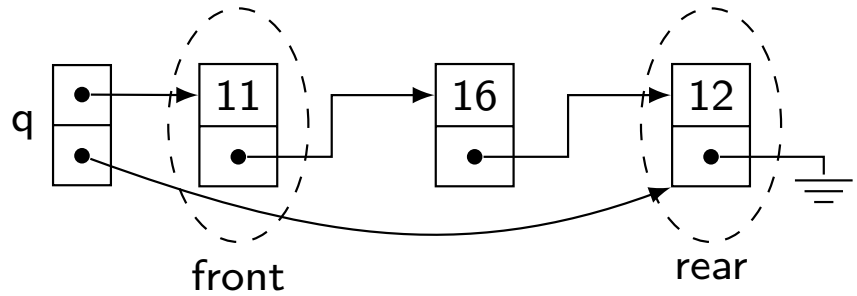



- Komplexitet för grundoperationerna:

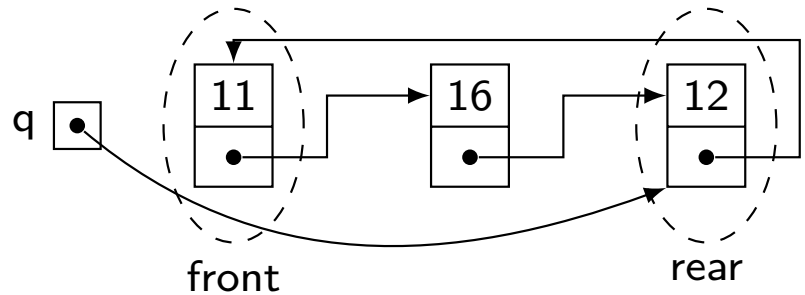
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | |
| Enqueue | | |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

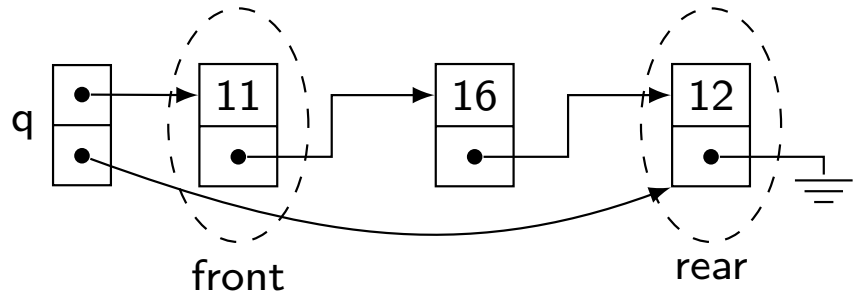



- Komplexitet för grundoperationerna:

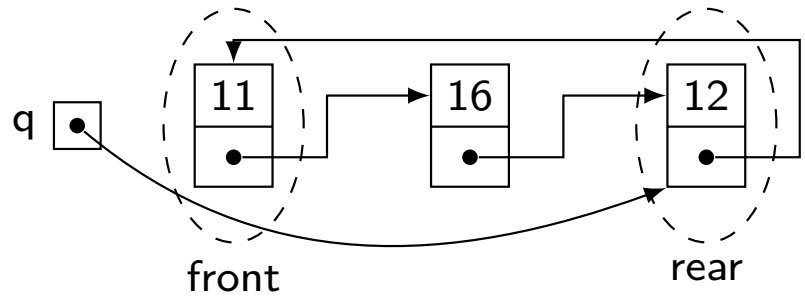
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | | |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

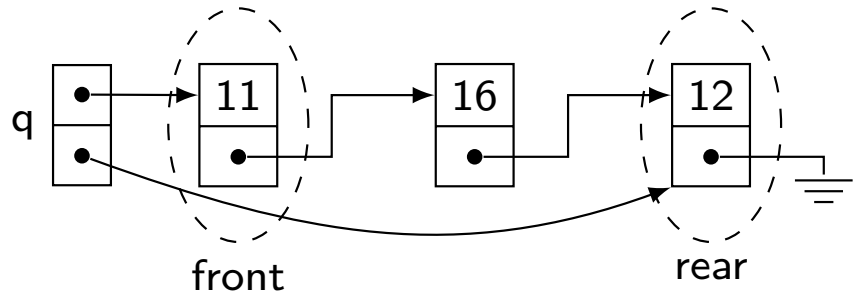



- Komplexitet för grundoperationerna:

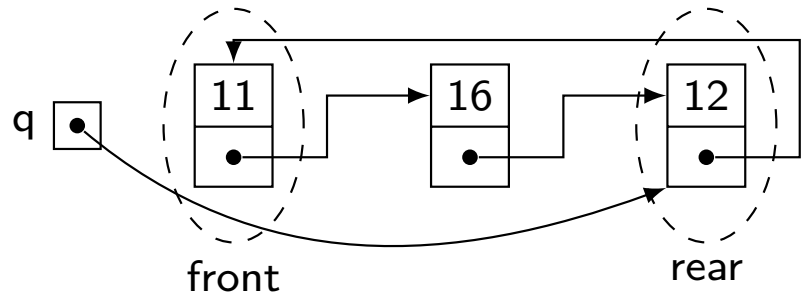
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(1)$ | |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

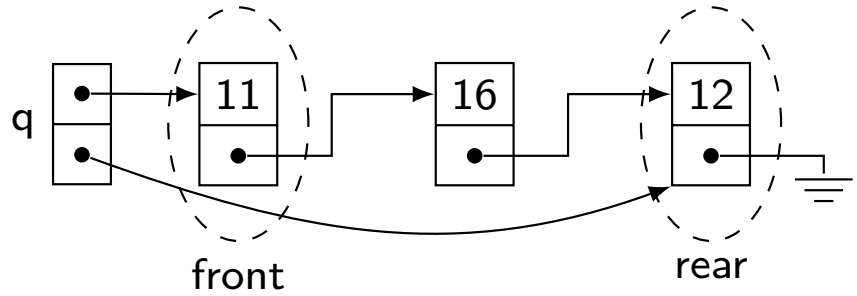



- Komplexitet för grundoperationerna:

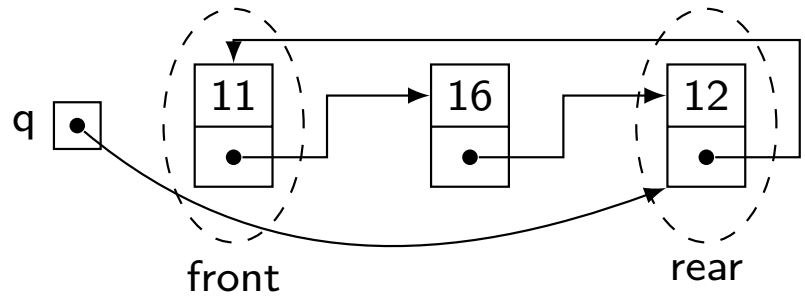
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(1)$ | $O(1)$ |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

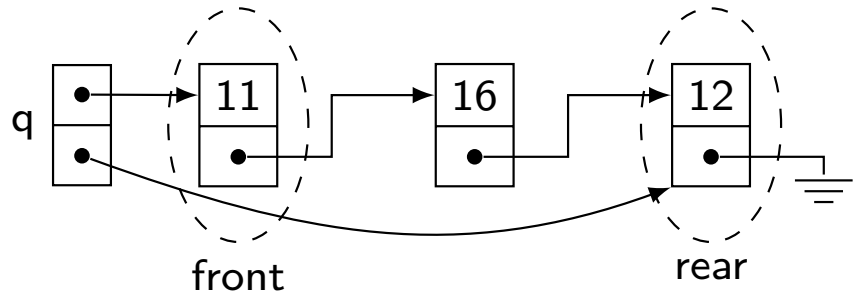



- Komplexitet för grundoperationerna:

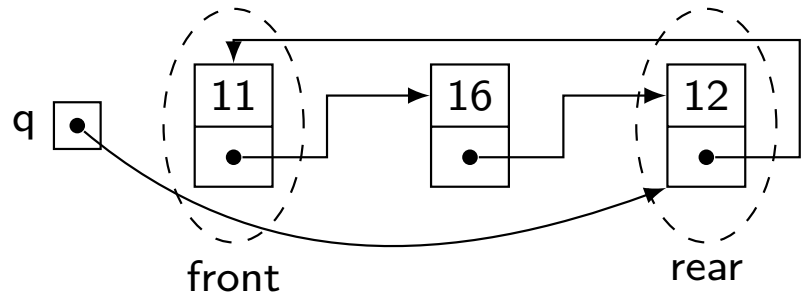
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(1)$ | $O(1)$ |
| Front | $O(1)$ | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

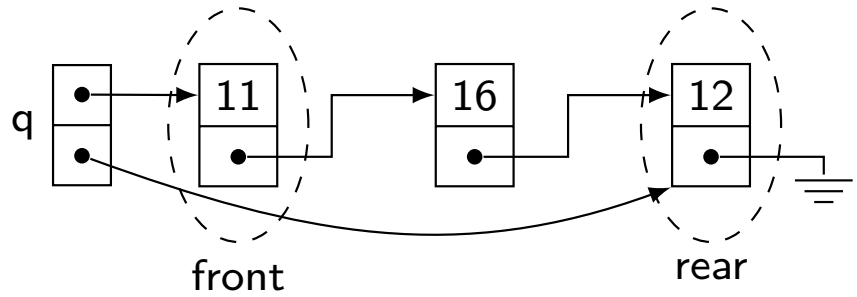



- Komplexitet för grundoperationerna:

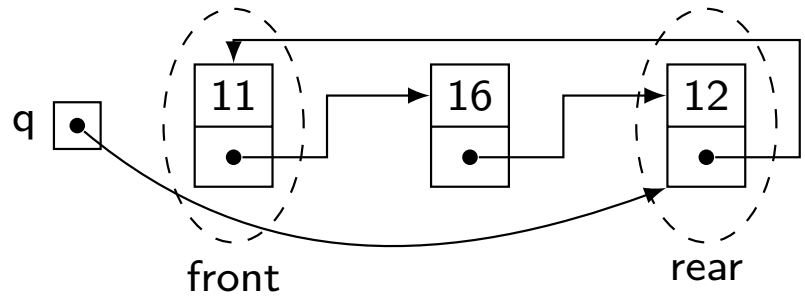
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(1)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

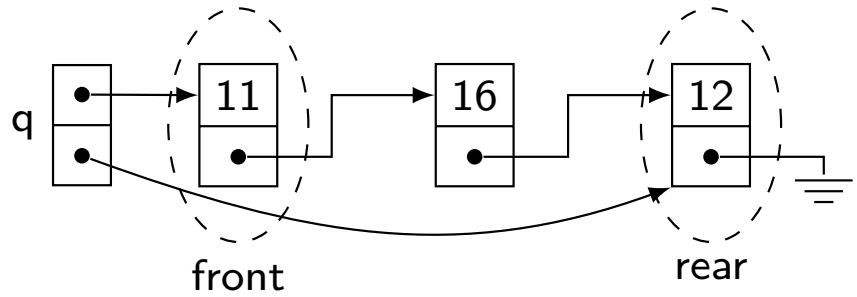



- Komplexitet för grundoperationerna:

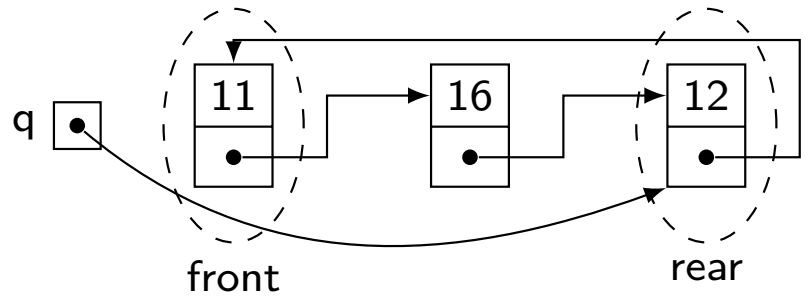
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(1)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

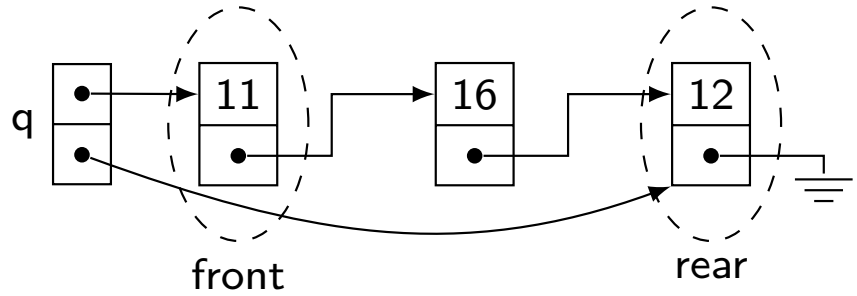



- Komplexitet för grundoperationerna:

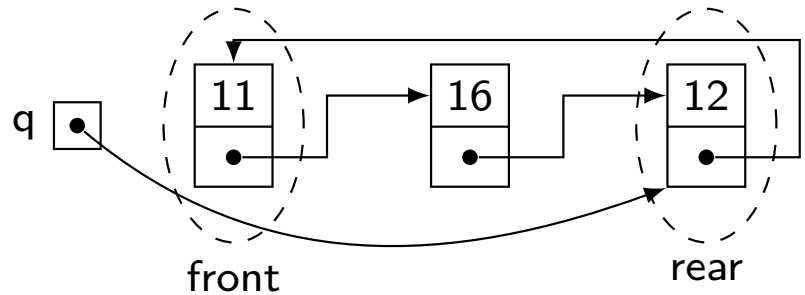
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(1)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | $O(1)$ |
| Isempty | | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

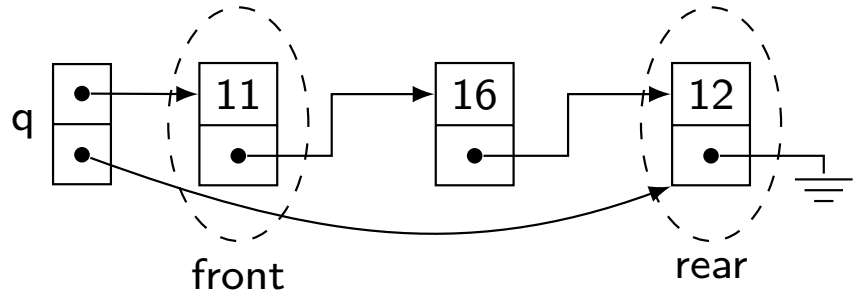



- Komplexitet för grundoperationerna:

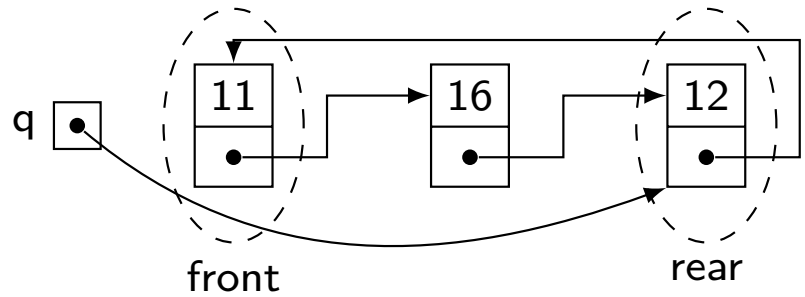
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(1)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | $O(1)$ |
| Isempty | $O(1)$ | |

# Kö konstruerad som Lista av 1-celler

- Lista 1: Fronten på kön = början av listan:




- Lista 2: Cirkulär lista: Länken i slutet av listan pekar på början av kön.

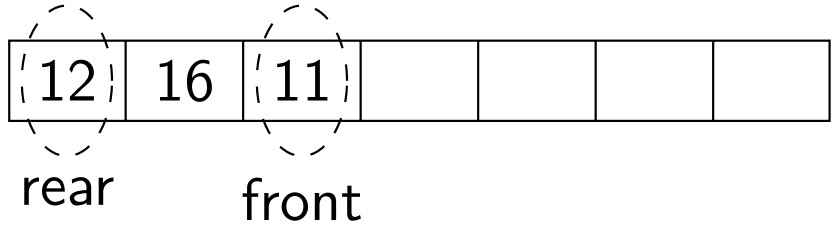



- Komplexitet för grundoperationerna:

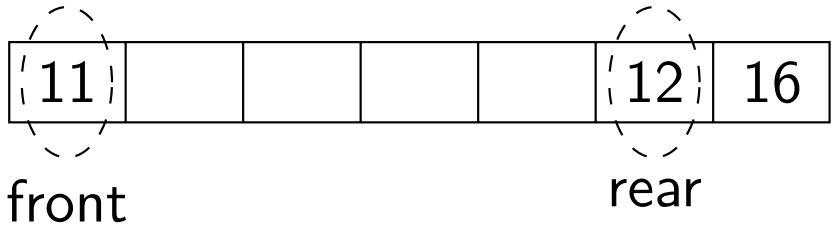
| Operation | Lista 1 | Lista 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(1)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | $O(1)$ |
| Isempty | $O(1)$ | $O(1)$ |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

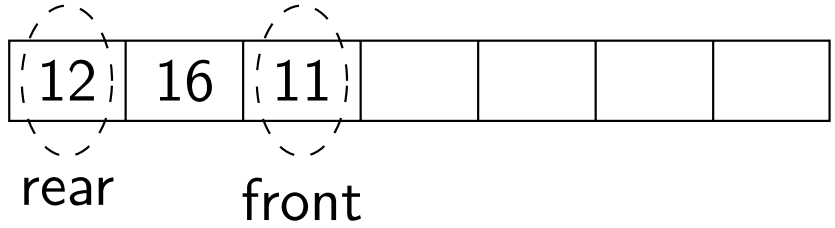



- Komplexitet för grundoperationerna:

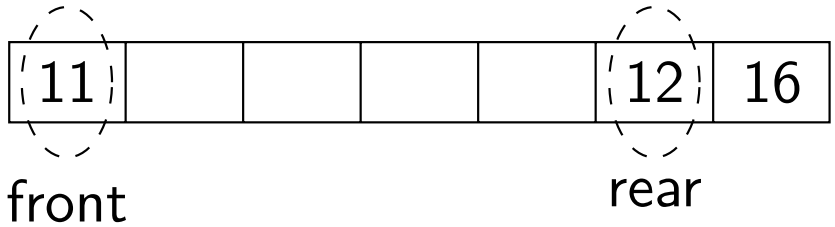
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | | |
| Enqueue | | |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

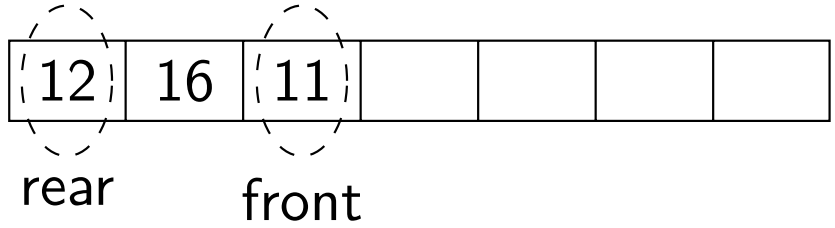



- Komplexitet för grundoperationerna:

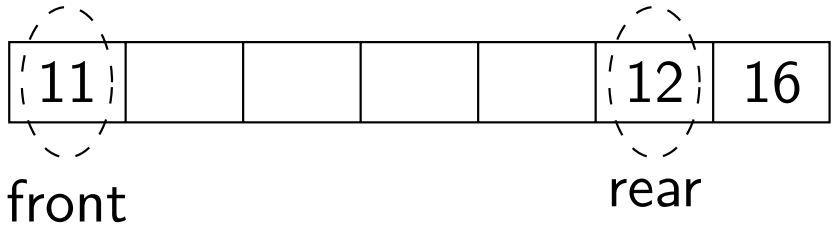
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | |
| Enqueue | | |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

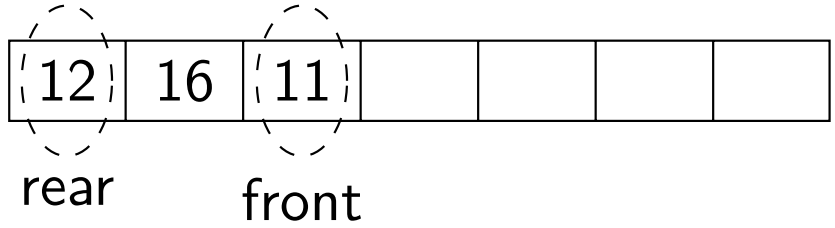



- Komplexitet för grundoperationerna:

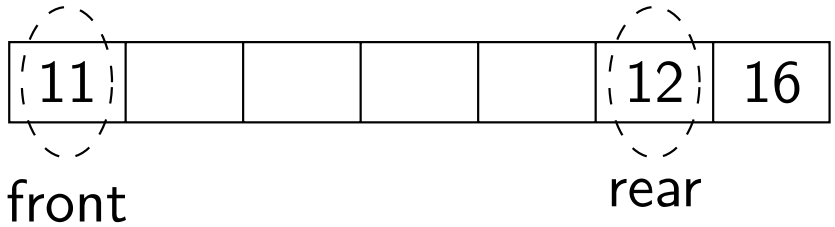
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | | |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

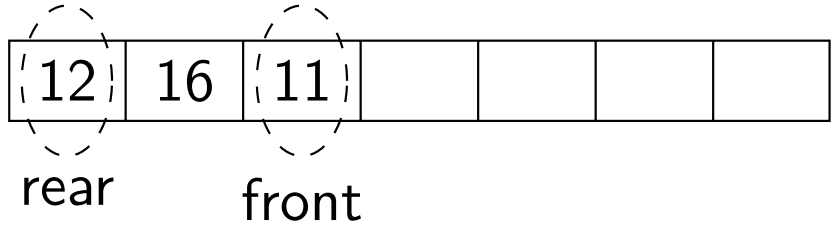



- Komplexitet för grundoperationerna:

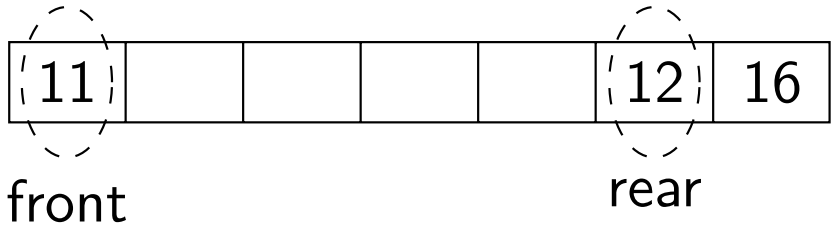
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

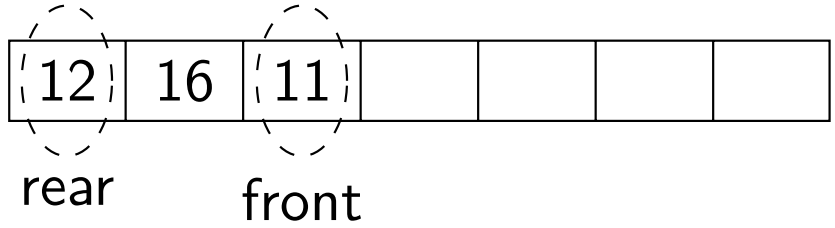



- Komplexitet för grundoperationerna:

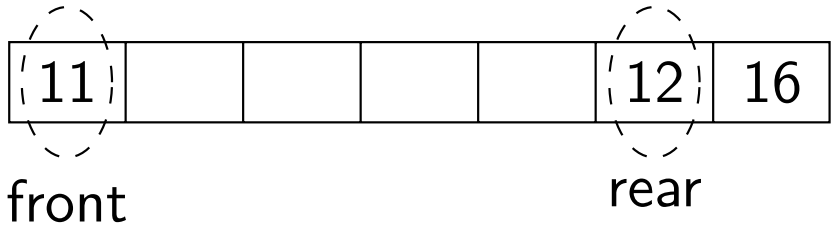
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | $O(1)$ |
| Front | | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

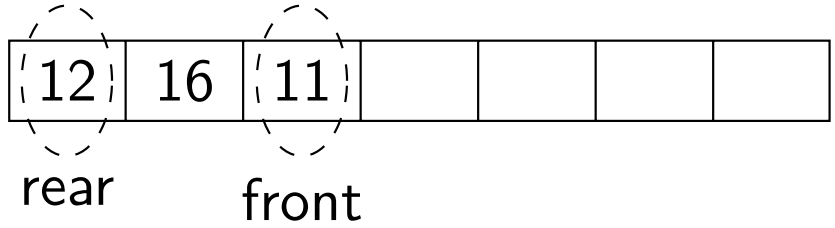



- Komplexitet för grundoperationerna:

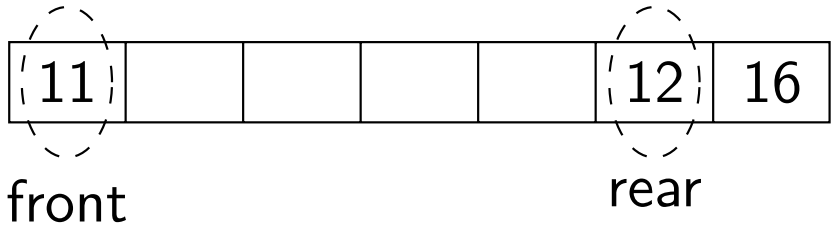
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | $O(1)$ |
| Front | $O(1)$ | |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

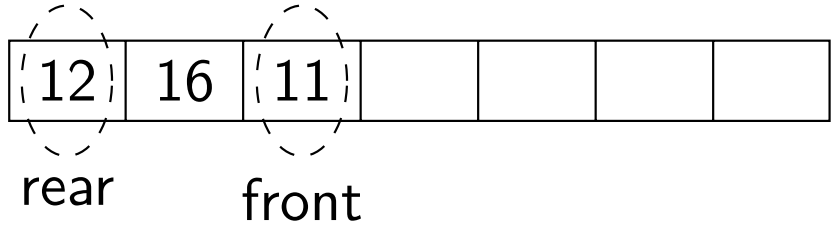



- Komplexitet för grundoperationerna:

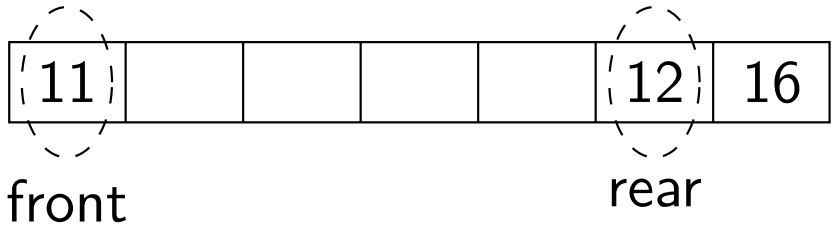
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | | |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

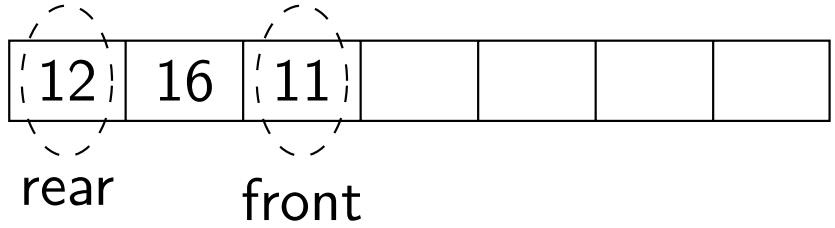



- Komplexitet för grundoperationerna:

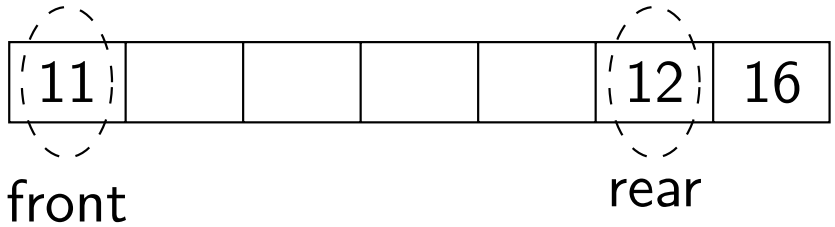
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

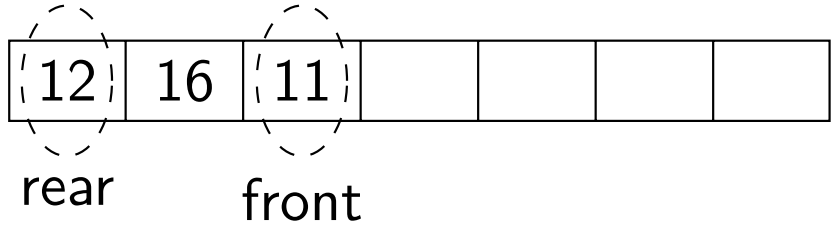



- Komplexitet för grundoperationerna:

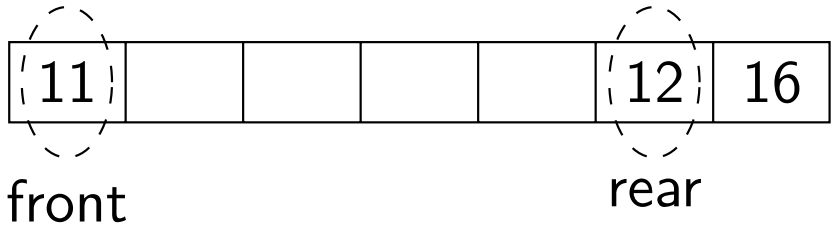
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | $O(1)$ |
| Isempty | | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

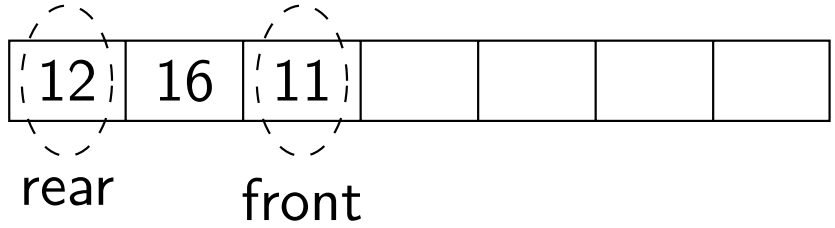



- Komplexitet för grundoperationerna:

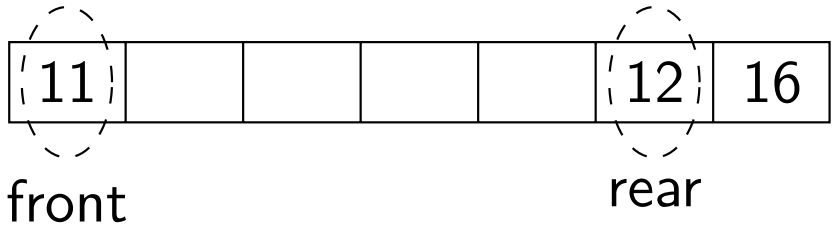
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | $O(1)$ |
| Isempty | $O(1)$ | |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:

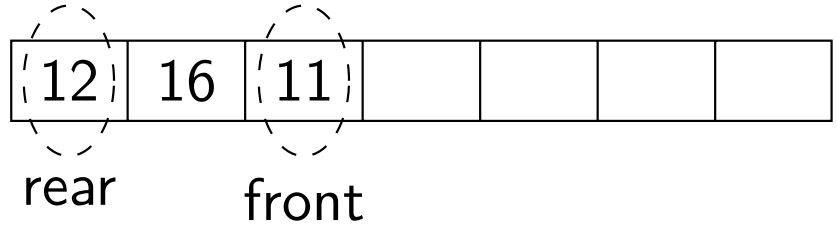



- Komplexitet för grundoperationerna:

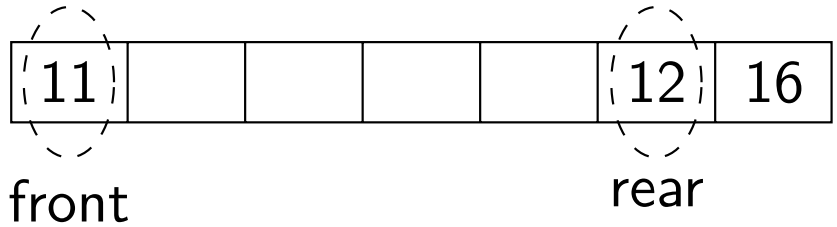
| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | $O(1)$ |
| Isempty | $O(1)$ | $O(1)$ |

# Kö konstruerad som Fält

- Fält 1: Rak vektor:




- Fält 2: Cirkulär vektor:




- Komplexitet för grundoperationerna:

| Operation | Fält 1 | Fält 2 |
|---|---|---|
| Empty | $O(1)$ | $O(1)$ |
| Enqueue | $O(n)$ | $O(1)$ |
| Front | $O(1)$ | $O(1)$ |
| Dequeue | $O(1)$ | $O(1)$ |
| Isempty | $O(1)$ | $O(1)$ |

- Nackdelar:
  - Maximal storlek.
  - Outnyttjat utrymme.
  - Fält 1: Enqueue är dyrt pga. omflyttningar.
  - Fält 2: Svårt att skilja tom från full kö.

# Kö, tillämpningar

- Buffert:
  - Routrar.
  - Skrivarkö.
- Bredden-först-traversering.