Datastrukturer och algoritmer

Föreläsning 7

Stack och kö

Innehåll

- Stack, Kö
  - Tillämpningar
  - Konstruktion
  - Dubbeländad kö

Stack

- Modell
  - Papperstrave
- Specialisering av Lista
  - Begränsningar på operationer
- Linjärt ordnad struktur
- Före / efter relation

Stack

- LIFO – Last In First Out
- Insättning, borttagning, avläsning i toppen av stacken

Informell specifikation till Stack

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

Formell specifikation

- Uppsättning axiom
- Beskriver relationer mellan typens olika operationer
- Axiom kan användas för att göra formella härledningar i datatypen

Formell specifikation till Stack

Ax 1: Isempty(Empty)
Ax 2: ¬Isempty(Push(v,s))
Ax 3: Pop(Push(v,s)) = s
Ax 4: Top(Push(v,s)) = v
Ax 5: ¬Isempty(s) => Push(Top(s), Pop(s)) = s

Gränsyta till 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

- Konstruktion
  - Lista
    - Toppen av stacken = början av listan
    - Lista som fält
      - Komplexitet

Stack

- Konstruktion
  - Fält
    - Botten i slutet
      - Stacken läggs i slutet av fältet
        - » Push och Pop O(1)
        - » Inga stora dataomflyttningar
    - Botten i början
    - Möjlighet att ha två stackar i samma fält

Stack

- Konstruktion
  - Länkad Lista
    - Riktad lista med 1-celler
      - Uteslutningar och specialiseringar av operationer
    - O(1) – tidskomplexitet

Stack

- Tillämpningar
  - Avbryter bearbetning som senare kanske återupptas
  - Återspårning (backtracking)
    - Till senaste gjorda valet

Stack

- Tillämpningar
  - Rekursion
  - Fakultet n!
    - factorial(n)
      if (n<=1) then return 1;
      else return n*factorial(n-1);
  - Evaluering av uttryck

Kö

- Modell
  - Kö
- Specialisering av Lista
  - Begränsningar på operationer
- Linjärt ordnad struktur

Kö

- FIFO – First In First Out
- Insättning i slutet av kön
- Borttagning i början av kön

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
```


Kö

- Konstruktion
  - Lista
    - Fronten på kön = början av listan
    - Listan som Fält
      - Komplexiteten
    - Riktad lista med 1-celler
      - Första köelementet länkar till det andra osv

Kö

- Konstruktion
  - Cirkulär Lista med 1-celler
    - Länken i slutet av kön pekar på fronten
  - Cirkulär vektor
    - + Inga omflyttningar
    - - Maximal storlek
    - - Outnyttjat utrymme
    - Problem skilja tom kö ifrån en full

Kö

- Tillämpningar
  - Buffert
  - Skrivarkö
  - Bredden-först-traversering

Dubbeländad kö

- Tillåter insättning och borttagning i båda ändarna (Deque)
- `insertFirst(e)`: Sätter in *e* i början av deque
- `insertLast(e)`: Sätter in *e* i slutet av deque

Dubbeländad kö

- `removeFirst()`: Tar bort och returnerar 1:a elementet
- `removeLast()`: Tar bort och returnerar sista elementet
- `first(), last(), size(), isEmpty()`

Dubbeländad kö

- Konstruktion
  - Dubbellänkad Lista
    - Insättning och borttagning i ändarna komplexitet O(1)
    - Huvud och svans nod
- Kan användas för att konstruera Stack och Kö

Modell för datatypens byggande

- Specifikatören – utformar specifikationen
- Implementatören – implementerar specen
- Användaren

Felhantering

- Specifikationen bestämmer implementationens spelrum
- Implementatören ska uppfylla spec.
- Användaren bör inte konstruera algoritmer som beror på ospecade förhållanden
- Att tänka på:
  - Robusthet
  - Effektivitet