Gruppövning 1 — Algoritmkonstruktion och pseudokod

5DV149

2024 LP3

Contents

1 Uppgift 1 — Enkel algoritm

Skatteverket vill ha din hjälp med att skriva en applikation för en person som vill räkna ut sin nettolön. För att beräkna nettolönen dras först skatten bort och sedan eventuell fackföreningsavgift.

Om personen tjänar minst 30000 kr är skatten 45%, i intervallet 20000-–29999 kr är skatten 40%, i intervallet 15000–19999 är skatten 35% och under 15000 är den 30%.

Om personen är fackföreningsansluten ska sedan fackföreningsavgiften på 250 kronor dras bort.

Skriv en algoritm i pseudokod som förklarar hur man går till väga för att räkna ut nettolönen (lönen efter skatt och fackföreningsavgift).

2 Uppgift 2 — Fältalgoritmer

Skriv en algoritm i pseduokod som utformas med hjälp av fältspecifikationen (*array*) i kapitel 5.1 (samma som vi gick igenom på föreläsningen) som hittar **medelvärdet** av talen i ett fält. Ni kan anta att fältet är endimensionellt och har ett tal på varje plats. Ni får bara använda de fältoperationer som ingår i gränsytan till Fält.

```
abstract datatype Array(val, index)
  Create(lo, hi: index) → Array(val, index)
  Low(a: Array(val, index)) → index
  High(a: Array(val, index)) → index
  Set-value(i: index, v: val, a: Array(val, index))
                                  → Array(val, index)
  Has-value(i: index, a: Array(val, index)) → Bool
  Inspect-value(i: index, a: Array(val, index)) → val
  Kill(a: Array(val, index)) → ()
```

3 Uppgift 3 — Listalgoritmer

Dessa uppgifter använder definitionerna av datatyperna Lista (`List`) och Riktad lista (`DList`). Gränsytan för dessa är:

```
abstract datatype List(val)
auxiliary pos
  Empty() → List(val)
  Isempty(l: List(val)) → Bool
  First(l: List(val)) → pos
  End(l: List(val)) → pos
  Next(p: pos, l: List(val)) → pos
  Previous(p: pos, l: List(val)) → pos
  Pos-isequal(p1, p2: pos, l: List(val)) → Bool
  Inspect(p: pos, l: List(val)) → val
  Insert(v: val, p: pos, l: List(val))
                                → (List(val), pos)
  Remove(p: pos, l: List(val)) → (List(val), pos)
  Kill(l: List(val)) → ()

abstract datatype DList(val)
auxiliary pos
  Empty() → DList(val)
  Isempty(l: DList(val)) → Bool
  First(l: DList(val)) → pos
  Next(p: pos, l: DList(val)) → pos
  Isend(p: pos, l: DList(val)) → Bool
  Inspect(p: pos, l: DList(val)) → val
  Insert(v: val, p: pos, l: DList(val))
                                → (DList(val), pos)
  Remove(p: pos, l: DList(val)) → (DList(val), pos)
  Kill(l: DList(val)) → ()
```

3.1 Uppgift 3a — Max av Lista

1. Skriv en algoritm i pseudokod som utformas med hjälp av listspecifikationen i kapitel 3.2 (samma som vi gick igenom på föreläsningen) som hittar det största talet i en lista av tal. Ni får bara använda de listoperationer som ingår i gränsytan till lista. Om listan är tom ska värdet Not-A-Number returneras [1]. Ni får anta att operationen *mindre-än* (`Is-less-than(a, b)`) är definierad för värdetypen.

2. Vad har lösningen för tidskomplexitet?

3.2 Uppgift 3b — Kopiera listor

1. Skriv en algoritm som skapar en *rak* kopia av en Lista l, dvs. en lista där ordningen på elementvärdena bevaras. Du får bara använda funktioner i gränsytan till Lista.

2. Skriv en algoritm som skapar en *omvänd* kopia av en Lista l, dvs. där elementvärdena kommer i motsatt ordning (”baklänges”). Du får bara använda funktioner i gränsytan till Lista.

[1] Not-A-Number finns definierad i flyttalsstandarden IEEE 754 som är implementerad i alla moderna flyttalsprocessorer och förekommer i Matlab som funktionen/konstanten `NaN`.

3. Skriv en algoritm som skapar en *omvänd* kopia av en Riktad lista l. Du får bara använda funktioner i gränsytan till Riktad lista.

4. Skriv en algoritm som skapar en *rak* kopia av en Riktad lista l. Du får bara använda funktioner i gränsytan till Riktad lista. Not: Denna uppgift är lite svårare än de tidigare.

3.3 Uppgift 3c — Snittmängd (tentauppgift 2018-03-08)

Snittmängden s av två mängder s_1 och s_2 är en mängd som innehåller alla elementvärden e som ingår både i s_1 och i s_2.

1. Ge pseudokod för en algoritm som beräknar snittmängden där datatypen mängd är konstruerad som en (osorterad) Riktad lista. Ni får anta att operationen likhet `Is-equal(a, b)` är definierad för värdetypen.

2. Ge pseudokod för en algoritm som beräknar snittmängden där datatypen mängd är konstruerad som en Riktad lista där elementvärdena är **sorterade** i stigande ordning. Tänk på att s ska vara sorterad på samma sätt. För fulla poäng ska algoritmen ha optimal komplexitet. Ni får anta att operationerna *likhet* (`Is-equal(a, b)`) och *mindre-än* (`Is-less-than(a, b)`) är definierade för värdetypen.

3. Vad har algoritmerna för relativ förenklad tidskomplexitet om du antar att mängderna innehåller ungefär lika många element n?

4 Uppgift 4 — Länkade listor

Rita figurer som visar vad som händer vid **insättning** och **borttagning** ur en

1. dubbellänkad lista med huvud,

2. enkellänkad lista.

Det är speciellt viktigt att illustrera hur länkar mellan element sätts så att listans integritet bibehålls.

5 Uppgift 5 — Konstruera Stack av Lista

Beskriv hur man skulle konstruera en Stack i Lista på så sätt att **sista** elementet i listan är Stackens **topp**. Formulera alla operationerna i gränsytan till Stack med hjälp av gränsytan till Lista. Ange den relativa förenklade komplexiteten för operationerna.

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

6 Uppgift 6 — Stack- och Kö-operationer

Hur ser stacken `s` och kön `q` ut efter att följande operationer utförst?

```
q ← Queue-empty()
s ← Stack-empty()
q ← Enqueue(1, q)
q ← Enqueue(2, q)
q ← Enqueue(3, q)
s ← Push(Front(q), s)
q ← Dequeue(q)
s ← Push(Front(q), s)
q ← Dequeue(q)
s ← Push(Front(q), s)
q ← Dequeue(q)
q ← Enqueue(Top(s))
s ← Pop(s)
q ← Enqueue(Top(s))
s ← Pop(s)
q ← Enqueue(Top(s))
```

7 Uppgift 7 — Kö med 1-celler

Antag att du ska konstruera en kö med hjälp av 1-celler. Hur ska riktningen mellan cellerna organiseras, dvs ska fronten på kön på först eller sist i kedjan av celler? Varför?

8 Uppgift 8 — Längd på en Kö

Skriv en algoritm som tar en kö `q` som argument och returnerar **längden** på kön `q`, dvs. antalet element som finns i `q`. Du får bara använda funktioner i gränsytan till Kö. Argumentet `q` skickas *by reference* så när algoritmen är klar så ska `q` se likadan ut som när algoritmen startade.

```
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
  Kill(q: Queue(val)) → ()
```

9 Uppgift 9 — Djupet på en Stack

Skriv en algoritm som tar en stack `s` som argument och returnerar **djupet** på stacken `s`, dvs. antalet element som ligger i `s`. Du får bara använda funktioner i gränsytan till Stack. Argumentet `s` skickas *by reference* så när algoritmen är klar så ska `s` se likadan ut som när algoritmen startade.

10 Uppgift 10 — Kopiering av en Kö

Skriv en algoritm som tar en kö `q` som argument och returnerar en **kopia** på `q`. Du får bara använda funktioner i gränsytan till Kö. Argumentet `q` skickas *by reference* så när algoritmen är klar så ska `q` se likadan ut som när algoritmen startade.

11 Uppgift 11 — Kopiering av en Stack

Skriv en algoritm som tar en Stack `s` som argument och returnerar en **kopia** på `s`. Argumentet `s` skickas *by reference* så när algoritmen är klar så ska `s` se likadan ut som när algoritmen startade.