

Introduktion til Programmering og Problemløsning (PoP)

Lists

Jon Sparring
Department of Computer Science
2021/10/07

UNIVERSITY OF COPENHAGEN



Konstruktion af Lister

Lister

En liste er en sekvens af **elementer af samme type**, men hvor antallet af elementer ikke nødvendigvis er kendt på forhånd.

Ligesom for antallet af tegn i en streng "abc" og i modsætning til antallet af elementer i et tuple ('a','b','c').

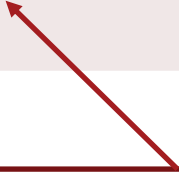
Eksempler

Udtryk : Type	Udtryk : Type
[3; 4; 5]	int list
['h'; 'e'; 'l'; 'l'; 'o']	char list
["hello"; "world"]	string list
[true]	bool list
[(1,2); (2,3); (3,4)]	(int*int) list
[]	'a list

Lister af tal

```
[1..3] = [1; 2; 3]
```

```
[10..-2..0] = [10; 8; 6; 4; 2; 0]
```



Uden parentes: `int*int list = int*(int list)`

Indbygget notation til manipulation af lister

Indicering som strenge:

```
let lst = ['a'; 'e'; 'i'; 'o'; 'u'; 'y']  
lst[2] = 'i'  
lst[2..4] = ['i'; 'o'; 'u']  
lst[2..14] = error message - index out of bound
```

Sammensætte lister med konkatenering-operator @

```
['a'; 'e'] @ ['i'; 'o'] = ['a'; 'e'; 'i'; 'o']  
[] @ [] = []
```

Tilføje et nyt element forest med cons (::)

```
1 :: [2;3]=[1;2;3]  
false :: [] = [false]  
1.2 :: 2.3 :: [] = [1.2; 2.3]  
[] :: [] = [[]]
```

Dekonstruktion af lister og rekursion

```
> match [1..5] with
-   [] -> printfn "Tom"
-   | elm::rst -> printfn "%A::%A" elm rst;;
1::[2; 3; 4; 5]
```

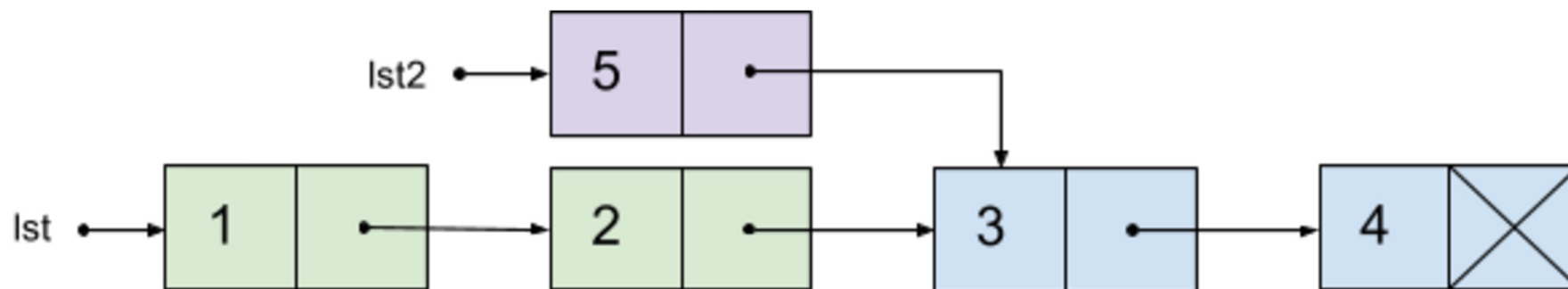
```
> let rec print lst =
-   match lst with
-   | [] -> printfn ""
-   | elm::rst ->
-       printf "%A " elm
-       print rst
-
- print ["hello"; "world"];;
"hello" "world"
val print: lst: 'a list -> unit
val it: unit = ()
```

Repræsentationen af lister som kæder (linked-lists)

Eksempel:

```
let lst = [1;2;3;4]  
let lst2 = 5 :: lst[2..]
```

Lagerrepræsentation:



Konsekvens:

- Det er nemt at hægte et ekstra element på starten af en liste (::).
- Det er **IKKE** nemt (læs: hurtigt) at tilgå det sidste element i en liste.
- Lister er *immutable*, dvs elementer kan ikke opdateres.

Liste properties

```
> let lst = [0..4];;  
val lst: int list = [0; 1; 2; 3; 4]
```

```
> lst.Length;;  
val it: int = 5
```

```
> lst.IsEmpty;;  
val it: bool = false
```

```
> lst.Head;;  
val it: int = 0
```

```
> lst.Tail;;  
val it: int list = [1; 2; 3; 4]
```

Resumé

I denne video har du hørt om:

- Oprettelse af **lister**
- Konsekvenser for at lister er repræsenteret som kæder (**linked lists**)
- Den indbyggede notation til **indicering**, **prepending** og **konkatenering**
- **Out-of-bound** fejlen
- Listers properties **Length**, **Head**, **Tail**, **IsEmpty**