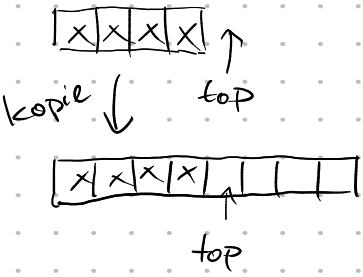
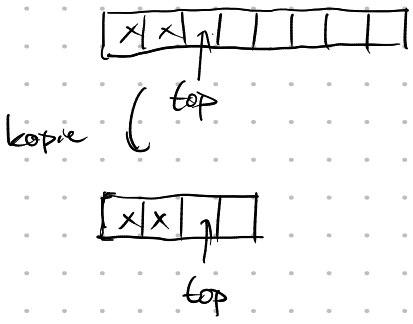
Dynamické pole:





1/4 (1/2)

1/2

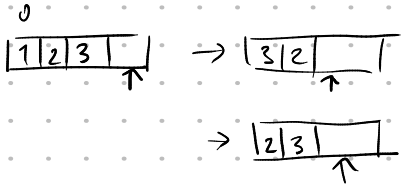
Vkládání:

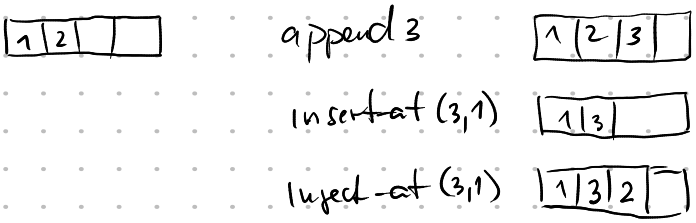
append (na konec)
insert-at (na index) posunutí
inject-at (na index posun)

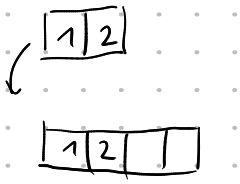
Odebírání:

unordered remove

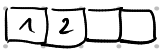
ordered_remove



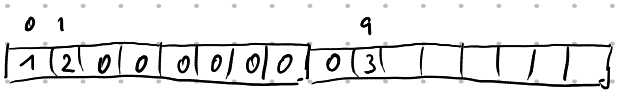




Co když v insert-at dám index větší než kapacita

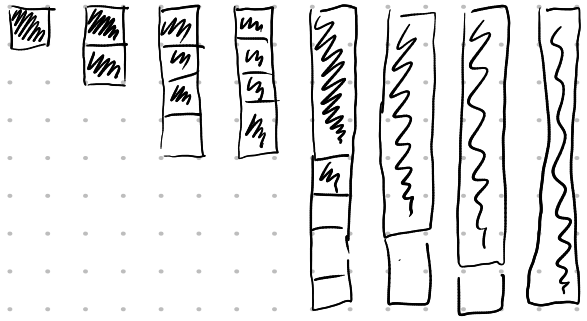


kapacita 4
insert-at (3, 9) → nová kapacita, 2*4=8, 8*2=16



posloupnost operací append

i	1	2	3	4	5	6	7	8	9	10–16	17	18–32	33	34–64	65	...
$c(i)$	1	2	3	1	5	1	1	1	9	1	17	1	33	1	65	

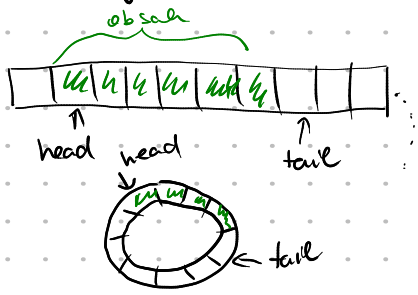


$$c(i) = \begin{cases} 1 & \text{jinak} \\ i & \text{pokud } i \text{ je } i-1 \text{ mocnina } 2 \end{cases} \qquad 0,\ i-1 \text{ (mocnina 2)}$$

Celkem operací: $\sum_{i=1}^{n} c(i) \leq n + \sum_{j=0}^{\lg n} 2^j \leq n + 2n = 3n$

$\frac{3n}{n} = 3$ operace na jeden append. Amortizovaně $3 = O(1)$ dle věty 3

Cyklický buffer



write — na index tail, tail zvětší o 1 (mod velikost)

read — z indexu head, zvětší head o 1 (mod velikost)

problém? co když dojedu na konec pole

pole délky 4

index :	0	1	2	3	4	5	6	7
mod 4 :	0	1	2	3	0	1	2	3

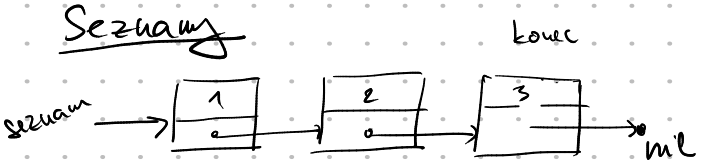
Empty: head = tail (mod velikost)
Full: tail = head − 1 (mod velikost)

používáme-li
head, tail > velikost

Empty: head = tail
full: tail − head = velikost

$m = 2^k$ $x \bmod m$ je číslo, které je k spodních bitů x

Seznamy

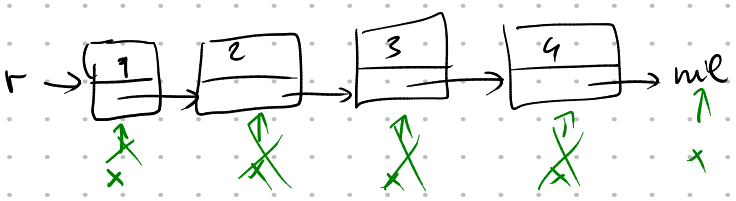



```
struct Node
    key: Key
    next: Node
```

```
a ∈ Node()
a.key ∈ 1
b ∈ Node()
b.key ∈ 2
c ∈ Node()
c.key ∈ 3

a.next ∈ b
b.next ∈ c
c.next ∈ nil

seznam ∈ a
```

```
list-length
← r: Node
→ Int

x ∈ r
m ∈ 0
while x ≠ nil
    m ∈ m+1
    x ∈ x.next
return m
```

idiom procházení seznamu



m: ||||

```
list-length-rec
← r: Node
→ Int

if r = nil then return 0
return 1 + list-length-rec(r.next)
```

```
list-search
← r: Node
← k: Key
→ Node (nebo nil)

x ∈ r
while x ≠ nil and x.key ≠ k
    x ∈ next
return x
```

nil

vrchol s klíčem

Složitost je $\Theta(n)$, kde n je délka seznamu = počet klíčů v seznamu

element-at
← r: Node
← i: Index
→ Node (nebo nil)

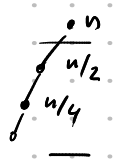
Složitost je $\Theta(i)$

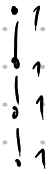
```
x ← r
m ← 0
while x ≠ nil and m < i
    m ← m + 1
    x ← x.next
return x
```

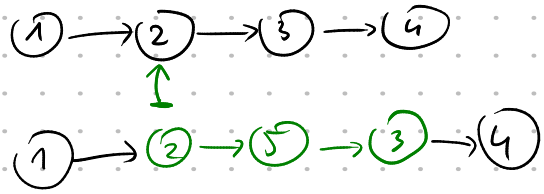
Půlení intervalů pro sčítání:

node_s ← element-at(l, n/2) # délka l je n $\Theta(n/2)$ $\Theta(n)$

$T(n) = T(n/2) + \Theta(n)$
$T(1) = 1$
⇒







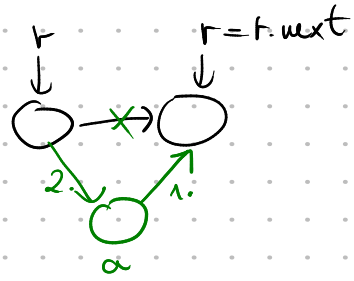
1 2 3 4
1 2 5 3 4

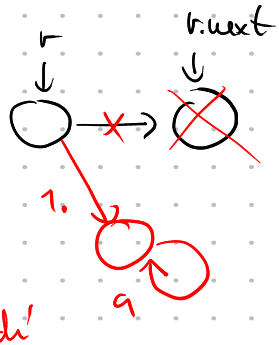
$\Theta(4)$

$\Theta(1)$

insert-after
← r: Node
← a: Node

a.next ← r.next
r.next ← a

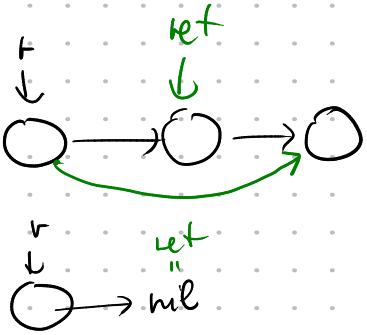




záleží na pořadí

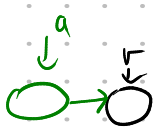
remove-after
← r: Node
→ Node (nebo nil)

```
ret ← r.next
if r.next ≠ nil then r.next ← r.next.next
return ret
```



Složitost je $\Theta(1)$.

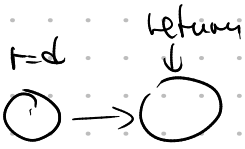
```
insert-at
  ← r: Node
  ← a: Node
  ← i: Int        ... index
  → Node          (začátek seznamu)

if i=0
    a.next ← r
    return a

x ← element-at(r, i-1)
if (x ≠ nil) then insert-after(x, a)
return r
```

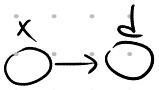


$\Theta(i)$

```
remove
  ← r: Node
  ← d: Node
  → Node

if r=d return r.next

x ← r
while x.next ≠ d and x.next ≠ nil
    x ← x.next
if (x.next≠nil) remove-after(x)
return r
```

$\Theta(a)$, kde a je délka seznamu






```
concatenate
  ← a: Node
  ← b: Node
  → Node

if (a=nil) then return b
if (b=nil) then return a
x ← a
while x.next ≠ nil
    x ← x.next
x.next ← b
return a
```

$\Theta(\text{délka } a)$