Kolekce objektů
- vyhledávání: klíč
  chceme vrátit objekt
- min, max, projít všechy
- update: insert
  remove

Abstraktní: př. ZÁSOBNÍK (stack)
- push = vložení
- pop = odebrání, LIFO

př. van Emde Boas strom
→ klíče: {0, 1, ..... n-1}, kde $n = 2^k$
→ search, $\lg \lg n$

Objekty = klíč

klíče = lineárně uspořádaná množina

pro $\forall x, y$ platí: $x = y$
$x < y$
$y < x$

V příkladech klíče = čísla (N, Q)

kolekce = podmnožina klíčů

Složitost operací:

$T : \mathbb{N} \to \mathbb{N}$

velikost vstupu — počet operací

počet klíčů ve struktuře — $n$ počet operací s klíči (asymptotické velikosti klíčů)

$\Theta(n)$ - $O(n)$
- $\Theta(n)$
-

# Model paměti + pseudokód (Pointer machine)

atomické: key ; Int ;

složené: Pole
Struktury

Pole:

- kolekce fixního počtu prvků stejného typu
- velikost = počet prvků
- přístup: index ∈ {0, 1, ... velikost - 1}

  přístup k prvku na indexu: čtení + zápis
  $\Theta(1)$

```
A ← [6] key
A[2] ← 5
A[3] ← A[2] * 10
```

Struktura

- kolekce prvků různého typu, přístup k prvkům podle jména
  $\Theta(1)$

```
struct ID
    jméno : typ
    jméno : typ
```

```
struct Rational
    e: Int
    d: Int
```

```
A ← ID()
```

```
A ← Rational()
A.e ← 1
A.d ← 3

B ← Rational()
B.e ← A.e
B.d ← A.d + 3
```

# Referenční sémantika

$A \leftarrow$ [6] key

$B \leftarrow A$

$A[5] \leftarrow 10$
$B[5] \leftarrow 15$

— $A[5] = B[5] = 15$

Reference: unikátní id v paměti

A B
| |
ref1 ref1

[6] key ... ref1

C D
| |
5 6

$C \leftarrow D$ ; $D \leftarrow 10$

C D
| |
6 10

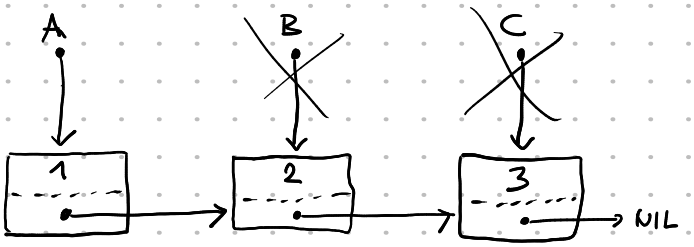
---

## Spojový seznam

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

$A \leftarrow$ Node()
$B \leftarrow$ Node()
$C \leftarrow$ Node()

A.key ← 1
A.next ← B
B.key ← 2
B.next ← C
C.key ← 3
C.next ← nil




A.next
A.next.next

graf

## Řízení běhu:

```
if (podm) then příkaz
if podm
   příkaz1
   příkaz2

else příkaz
else
   příkaz1
   příkaz2
```

```
while (podm) do příkaz
while podm
      příkaz1
      příkaz2
```

```
For (i in start..<end) do příkaz

      i = start, start+1, .... end-1
```

Př.

```
a ← 10
b ← 20

if (a<b) then c←a
else c←b
```

## procedury

HLAVIČKA

```
[ Jméno_procedy
  ← arg:typ  — komentář
  ← arg:typ  — komentář
  → typ
  → typ
```

```
[ TĚLO
  return výsledek1, výsledek2
```

```
[ divrem
  ← a : Int
  ← b : Int
  → Int
  → Int
```

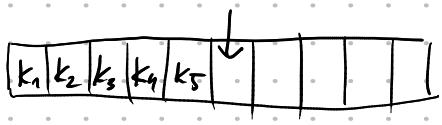
```
[ return a/b, a mod b
```

```
[ x, y ← divrem(10,3)
```

```
Fill
← arr : []key
← k : key
for (i in 0..<len(arr)) do arr[i] ← k
```

```
A ← [10]key
Fill(A, 13)
```

## Pole jako DS pro množinu




$\{k_1, k_2, k_3 .. k_5\}$

kolik jich mám: $m$ ... na indexech $0, \dots, m-1$

```
struct Array-Dict
    data: []key
    top : Int
```

```
insert
← ad : Array-Dict
← k : key
1. if ad.top ≥ len(ad.data) then return
2. ad.data[ad.top] ← k
3. ad.top ← ad.top + 1
```

Př. data: [3][4][5][6] top: ~~0 1 2 3~~ 4

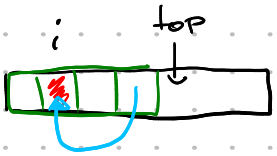
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)

složitost: $\Theta(1)$

Chceme odebrat klíč k (pokud tam je)

1. Najít index, kde je k (search)
2. Smazat prvek na indexu. (delete)

```
delete
← ad: Array-Dict
← i : Int
ad.data[i] ← ad.data[ad.top - 1]
ad.top ← ad.top - 1
```




záleží na pořadí: pro prvky na indexech $j > i$, musíme udělat posun o 1 doleva $\Theta(n)$

nezáleží na pořadí: $\Theta(1)$. Prvek na indexu top-1 dám na index i.

## Vyhledávání v poli:

Je dáno pole $A$, klíč $k$, najdi index $i$ tak, že $A[i] = k$; nebo řekni, že $k$ není

1) průchodem všech prvků

A: 

Složitost $\Theta(n)$

```
for (i in 0..<len(A))
    if (A[i] = k) then return i
return -1
```

Složitost v průměrném případě: $(n = len(A))$

představme si posloupnost $n$ operací vyhledávání (úspěšného)

$F_i = \frac{\text{počet operací s argumentem } A[i]}{n}$

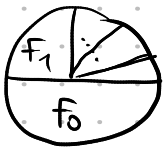
$\sum_{i=0}^{n-1} F_i = 1$

průměrná složitost: $f_0 \cdot 1 + f_1 \cdot 2 + \dots f_{n-1} \cdot n = \sum_{i=0}^{n-1} F_i \cdot (i+1)$

situace: $F_i = 1/n$ $\quad \frac{n+1}{2}$

situace: $F_0 = 1$, $F_{n-1} = 1$ $\quad 1$, $n$

situace: $F_0 = \frac{1}{2}$, $F_1 = \frac{1}{4}$, $\dots$ $F_{n-2} = \frac{1}{2^{n-1}}$, $F_{n-1} = \frac{1}{2^{n-1}}$




$$C(n) = \left( \sum_{i=0}^{n-2} \frac{1}{2^{(i+1)}} \cdot (i+1) \right) + \frac{1}{2^{n-1}} \cdot n$$

$$C(n) = \frac{1}{2} + \frac{1}{4} \cdot 2 + \dots + \frac{1}{2^{n-1}} \cdot (n-1) + \frac{1}{2^{n-1}} \cdot n$$

$$C(n-1) = \frac{1}{2} + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \dots + \frac{1}{2^{n-2}} (n-2) + \frac{1}{2^{n-2}} \cdot (n-1)$$

$$\frac{C(n-1)}{2} = \frac{1}{4} + \frac{1}{8} \cdot 2 + \dots + \frac{1}{2^{n-1}} (n-2) + \frac{1}{2^{n-1}} \cdot (n-1)$$

$$C(n) - \frac{C(n-1)}{2} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots + \frac{1}{2^{n-1}} + \frac{1}{2^{n-1}}$$

$$= 1$$

$$C(n) = \frac{C(n-1)}{2} + 1$$

$$c(1) = 1$$

$c(n) < 2$

pro $c(n-1) < 2$

$$\frac{c(n-1)}{2} < 1$$

$$C(n) < 2$$

KONEC PŘÍKLADU.

Pokud máme zafixováno $f_0 \dots f_{n-1}$.

Průměrná složitost je nejmenší (pouze změnou pořadí prvků)

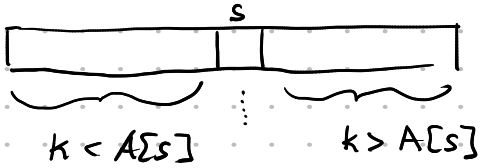
pokud $f_0 \geq f_1 \geq f_2 \dots$

---

Pole je setříděno: $i<j$ pak $A[i] < A[j]$

Pozorování: s je index, k je klíč v poli

porovnáme A[s] a k




1) $A[s] = k$ ..... HURÁ

2) $k < A[s]$

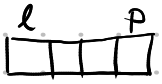
Volba s v "půlce" vede k půlení intervalů (binárnímu vyhledávání)

```
binary-search
∈ A : []Key
∈ k : Key
→ Int
---
l ← 0
p ← len(A) - 1
while l ≤ p
    s ← ⌊(l+p)/2⌋
    if (A[s] = k) then return s
    if (A[s] > k) then p ← s - 1
    else l ← s + 1
return -1
```

vymezení čás. pole

l ... ind. první pr

p ... ind. posledního prvku




p - l + 1 ... počet prvků

l > p 0

□

p = l

```
binary-search
∈  A : []key
∈  k : key
→  Int

l ← 0
p ← len(A) - 1
while  l ≤ p
    s ← ⌊(l+p)/2⌋
    if(A[s] = k) then return s
    if(A[s] > k) then p ← s-1
    else  l ← s+1

return -1
```

Příklad

1 2 3 4 5 6 7 , $k = 2$

Iterace 1: $l = 0$, $p = 6$
$s \leftarrow 3$
$A[3] = 4 > 2 \Rightarrow p \leftarrow 2$

Iterace 2: $l = 0$, $p = 2$
$s \leftarrow 1$
$A[1] = 2 \Rightarrow$ return 1

$k = 9$

iterace 1: $l = 0$, $p = 6$
$s \leftarrow 3$
$A[3] \leq 4 < 9 \Rightarrow l \leftarrow 4$

Iterace 2: $l = 4$, $p = 6$
$s \leftarrow 5$
$A[5] = 6 < 9 \Rightarrow l \leftarrow 6$

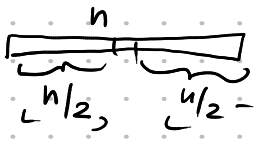
Iterace 3: $l = 6$, $p = 6$
$s \leftarrow 6$
$A[6] = 7 < 9 \Rightarrow l = 7$

Iterace 4: $l = 7$, $p = 6$ $l \not\leq p \rightarrow$ konec cyklu

Analýza:

1) Algoritmus vždy skončí.
= cyklus vždy skončí.
jedna iterace zmenší $p - l + 1$ alespoň o 1

2) pokud je $k$ v poli, tak ho najdeme.
Invariant cyklu:
$A[l] \leq k \leq A[p]$
[hlavní myšlenka]




Složitost

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

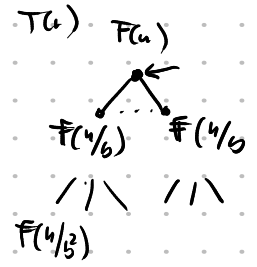
Master theorem: $T(n) = a \cdot T(n/b) + f(n)$

$a = 1$
$b = 2$
$f(n) = \Theta(1)$ ?
$\log_2 1 = 0$
$n^0 = 1$

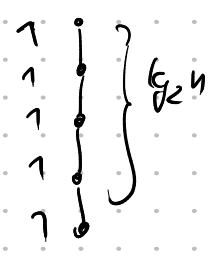
→ $f(n) = O(n^{\log_b a - \varepsilon}) \Rightarrow \Theta(n^{\log_b a})$
→ $f(n) = \Theta(n^{\log_b a}) \Rightarrow \Theta(n^{\log_b a} \cdot \log n)$
→ $f(n) = \Omega(n^{\log_b a + \varepsilon}) \Rightarrow \Theta(f(n))$

$\Theta(\lg n)$

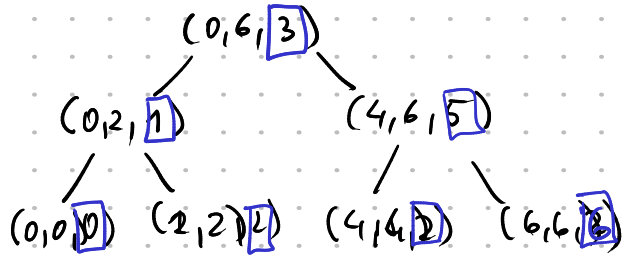
a.

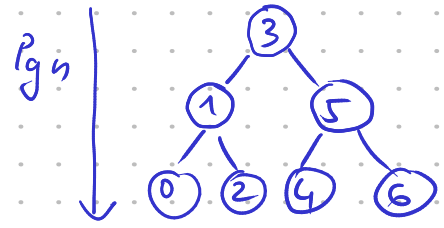



výška $\log_b n$
listů: $a^{\log_b n} = n^{\log_b a}$
$f(n)$



pole delky 7 ) ... (l, p, s) ... (l, p, $\lfloor \frac{l+p}{2} \rfloor$ )




"Motivace pro bin. vyh. strom"