Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Polja
(tabele)

Jurij Mihelič, UniLj, FRI

Polje

- Kapaciteta polja
  - fizična velikost polja (java: `a.length`)
  - največje št. elementov v polju
- Velikost polja
  - logična velikost polja
  - dejansko št. elementov v polju
- Izkoriščenost polja
  - velikost / kapaciteta
  - učinkovito hranjenje podatkov v pomnilniku

Polje

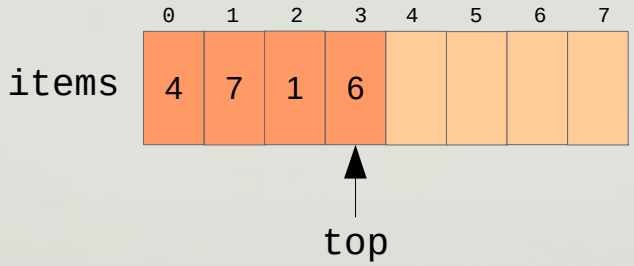
- Zaporedno hranjenje elementov
  - elementi v pomnilniku zasedajo zaporedne lokacije
  - naključni dostop
    - dostop do poljubnega elementa je hitra operacija
  - vstavljanje in brisanje elementov
    - na koncu polja hitro
    - na poljubno lokacijo počasno
  - izkoriščenost predpomnilnika
    - če hranimo vrednosti
    - če hranimo *reference* potem manjša lokalnost

Polje

- Statično polje
  - kapaciteta se ne spreminja
  - lahko dinamično alocirano
- Dinamično polje
  - kapaciteto je moč spreminjati
  - v ozadju delovanja je statično polje
  - operacija `resize(new_capacity)`
    - rezervira nov prostor in
    - vanj skopira ustrezne stare podatke

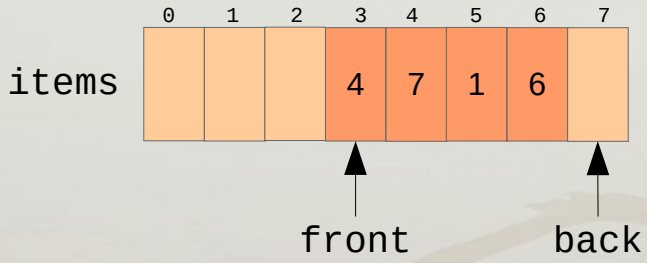
Polje

- Polje kot sklad
  - pozicija `top`
  - statično ali dinamično
  - podliv / preliv (*underflow* / *overflow*)
  - preprečevanje postopanja (*loitering*)



Polje

- Polje kot vrsta (in vrsta z dvema koncema)
  - poziciji `front` in `back`
  - statično ali dinamično
  - podliv / preliv
  - detekcija prazne in polne vrste
  - preprečevanje postopanja

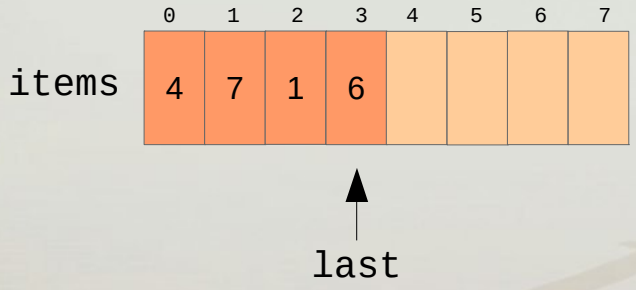


Polje

- Polje kot zaporedje
  - pozicija `last`
  - statično ali dinamično
  - podliv / preliv
  - preprečevanje postopanja

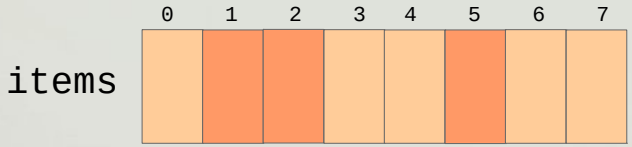
Polje

- Polje kot vreča in množica (1. način)
  - pozicija `last`
  - statično ali dinamično
  - podliv / preliv
  - preprečevanje postopanja



Polje

- Polje kot množica in vreča (2. način)
  - karakteristični (bitni) vektor
  - omejitev
    - elementi množice so števila
    - števila so v omejenem intervalu



Dinamično polje

- Dinamična zbirka
  - vrsta, dvrsta, množica, zaporedje, …
  - dodajanje elementa
    - `push(x)`, `enqueue(x)`, `add(x)`, `insert(i, x)`, …
    - lahko zmanjka prostora
    - torej povečamo kapaciteto
  - odstranjevanje elementa
    - `pop()`, `dequeue()`, `remove(x)`, `delete(i)`, …
    - velikost zbirke postane majhna v primerjavi z njeno kapaciteto
    - torej zmanjšamo kapaciteto

Dinamično polje

- Sprememba kapacitete – kdaj?
  - različne strategije
  - `push(x)`
    - **if** velikost == kapaciteta **then** resize()
    - dodajanje elementa
  - `pop()`
    - odstranjevanje elementa
    - **if** velikost <= kapaciteta / 3 **then** resize()

Dinamično polje

- Sprememba kapacitete – kako?
  - `resize()`
    - ustvarimo novo polje
    - kopiramo elemente iz starega polja v novo
  - velikost novega polja
    - različne strategije
    - nova kapaciteta = 2 * velikost
  - zahtevnost
    - `resize()`: $O(n)$
    - torej tudi `push(x)` in `pop()`: $O(n)$
      - čeprav brez `resize()` le $O(1)$

Amortizirana zahtevnost

- Zaporedje operacij
  - operaciji `push(x)` in `pop()`
  - m = št. teh operacij

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

- `resize()` upoštevamo posebej
  - r = št. operacij `resize()`
  - $r = O(m)$

o o * o o o o o * o o o o o o o o o * o o o o o o o o * o o o o o o o o o * o o o o o o o

Amortizirana zahtevnost

- Izhodišče
  - večina operacij stane malo
  - manjšina pa je zelo dragih
- Zahtevnost zaporedja operacij
  - analiza najslabšega primera upošteva manjšino
- Amortizacija
  - porazdelitev velikih stroškov skozi daljše obdobje
  - **skupno zahtevnost** celotnega zaporedja operacij **porazdelimo na** (amortiziramo) **posamezno operacijo**
  - povprečna zahtevnost brez uporabe verjetnosti

Amortizirana zahtevnost

- Dinamični sklad
  - zaporedje m operacij od tega r `resize()`

o o ∗ o o o o o ∗ o o o o o o o o o ∗ o o o o o o o o ∗ o o o o o o o o o ∗ o o o o o o o

  - skupna zahtevnost vseh r klicev `resize()`
  - $O(m)$

zaporedje *m* operacij *add*/*remove* in od tega *r* operacij *resize*

1	**2**	**…**	***i*-1**	***i***	**…**	***r***		zaporedni resize
o_1	o_2	…	o_{i-1}	o_i	…	o_r	o_{r+1}	št. operacij
n_1	n_2		n_{i-1}	n_i		n_r		velikost polja
c_1	c_2		c_{i-1}	c_i		c_r		kapaciteta polja

Povzetek

	operacija	polje
sklad, vrsta, dvrsta	enqueue(x), push(x)	$O(1)$
	dequeue(), pop()	$O(1)$
	enqueueFront(x), push(x)	$O(1)$
	dequeueBack(), pop()	$O(1)$
zaporedje	get(i)	$O(1)$
	set(i, x)	$O(1)$
	find(x)	$O(n)$
	insert(i, x)	$O(n)$
	delete(i)	$O(n)$
vreča, množica	remove(x)	$O(n)$
	add(x) – vreča	$O(1)$
	addUnique(x) – množica	$O(n)$